

The main body of the document consists of a grid of 10 columns and 10 rows of small, illegible diagnostic test data tables. Each table appears to be a structured list of data points, possibly representing test results for different components or parameters. The text is too small to be read, but the layout is consistent across the grid.

4

11

ECF:DRP88501

00010000

780223 IDENTIFICATION P10 411

HDR1DRLPCASEQ

00010000

780223
SEQ 0001

PRODUCT CODE: MAINDEC-11-DRLPC-A-D
PRODUCT NAME: LPA/ARI1 DIAGNOSTIC TEST I
DATE: JANUARY 1978
MAINTAINER: DIAGNOSTIC GROUP

COPYRIGHT (C) 1978 DIGITAL EQUIPMENT CORP., 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 ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE AR11 LAB SYSTEM OPTION. MOST FUNCTIONS OF THE OPTION WILL BE TESTED. DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR MAY BE REQUIRED TO SUPPLY OPTION CHARACTERISTICS.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZARA-(REV)". IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE AR11 OPTION WHEN IT IS ON THE LPA-11XX I/O BUS. NO RECALING IS NEEDED. SOME TESTS DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TESTS, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN "MD-11-DZARA". YOU SHOULD RUN "MD-11-DRLPA" BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 FAMILY COMPUTER WITH 16K WORDS OF MEMORY
AR11 HEX OPTION MODULE INSTALLED
LPA-11X SYSTEM
TELETYPE (OR EQUIVALENT)

2.2 STORAGE

THIS PROGRAM USES LESS THAN 16K OF MEMORY.

3. LOADING PROCEDURE

PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

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

STANDARD PDP-11 FORMAT

SW 15 = 1	HALT ON ERROR
SW 14 = 1	LOOP ON TEST
SW 13 = 1	INHIBIT ERROR TYPINGS
SW 12 = 1	STORAGE SCOPE CONNECTED
SW 11 = 1	INHIBIT INTERACTIONS
SW 10 = 1	ALL 1'S A TO D TEST JUMPER INSTALLED
SW 09 = 1	LOOP ON ERROR
SW 08 = 1	LOOP ON TEST IN SWR <7:0>

REFER TO 9. FOR SOFTWARE SWITCH REGISTER CONTROL

4.2 STARTING ADDRESS OR ADDRESSES

200 IS THE STARTING ADDRESS OF THE LOGIC TEST.
 204 IS THE RESTART ADDRESS OF THE LOGIC TEST.

5. OPERATING PROCEDURE

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

6. ERRORS

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

7. RESTRICTIONS

 EXTERNAL A TO D START INPUT MUST NOT BE CONNECTED.

8. MISCELLANEOUS

8.1 EXECUTION TIME

 THE LOGIC TEST WILL TAKE APPROXIMATELY 4 MINUTES FOR COMPLETION AND WILL TYPE 'END PASS'.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS

LOCATION \$BASE CONTAINS THE AR11 BASE DEVICE ADDRESS <170400>
 LOCATION \$VECT1 CONTAINS THE AR11 BASE INTERRUPT VECTOR <340>
 LOCATION \$DBRL CONTAINS THE AR11 A TO D BR LEVEL <300><6>
 LOCATION \$CKBL CONTAINS THE AR11 CLOCK BR LEVEL <300><6>
 LOCATION \$VCRBL CONTAINS THE AR11 SCOPE BR LEVEL <200><4>
 LOCATION \$FILLS CONTAINS THE TTY FILLER CHARACTER COUNT
 LOCATION \$NUIL CONTAINS THE TTY FILLER CHARACTER

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

8.3 MULTIPLE AR11 TESTING

A PROVISION IS MADE FOR TESTING SEQUENTIAL AR11. STARTING AT BUS ADDRESS/VECTOR DEFINED BY \$BASE AND \$VECT1. THE HEADER TYPEOUT WILL INFORM THE OPERATOR OF THE NUMBER OF AR11'S FOUND.

8.4 XXDP/ACT/APT NOTES

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

8.5 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:

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

EO1

E OR D
DATA=

"D"
"DATA TO BE DEPOSITED"

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

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

9. SOFTWARE SWITCH REGISTER OPERATION

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

10. 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. MB254 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 MB254 FALLS INTO THE GROUP "B" CATEGORY.

G01

THE LPA-11KX DIAGNOSTIC KIT WILL INCLUDE:

SEQ 0006

<u>OPTION</u>	<u>GROUP</u>	<u>DIAG. #</u>	<u>DIAG. TITLE</u>
LPA11-KX	LEVEL 2	MD-11-DRLPA	LPA11-K SYSTEM DIAG.
MB254	"B"	MD-11-DRLPN	MB254 (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.
MB200-YC	B	MD-11-DZLPL	LPA/DMC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/DMC JMP+ROM READ TEST

TABLE OF CONTENTS

64	BASIC DEFINITIONS
174	OPERATIONAL SWITCH SETTINGS
189	TRAP CATCHER
198	STARTING ADDRESS(ES)
202	ACT11 HOOKS
213	APT PARAMETER BLOCK
235	COMMON TAGS
282	APT MAILBOX-ETABLE
350	ERROR POINTER TABLE
551	INITIALIZE THE COMMON TAGS
638	TYPE PROGRAM NAME
645	GET VALUE FOR SOFTWARE SWITCH REGISTER
687	T1 TEST EACH BUS ADDRESS FOR TIMEOUT
712	T2 TEST THAT THE PRESET BUFFER CAN HOLD #0
726	T3 TEST THE COUNTER PRESET BUFFER CAN HOLD #377
740	T4 TEST THAT PRESET BUFFER CAN HOLD #125
754	T5 TEST THAT PRESET BUFFER CAN HOLD #252
768	T6 TEST THAT PRESET BUFFER CAN HOLD A COUNT PATTERN
787	T7 TEST INIT TO CLEAR COUNT PRESET BUFFER WHEN IT IS =-1
804	T10 TEST THAT THE COUNTER CAN HOLD #0
818	T11 TEST THE COUNTER CAN HOLD #377
833	T12 TEST THAT COUNTER CAN HOLD #125
847	T13 TEST THAT COUNTER CAN HOLD #252
861	T14 TEST THAT COUNTER CAN HOLD A COUNT PATTERN
880	T15 TEST INIT TO CLEAR COUNTER WHEN IT IS =-1
897	T16 TEST ENABLE COUNTER (BIT 0) CAN BE SET
911	T17 TEST RATE SELECT (BIT 1) MAY BE SET
925	T20 TEST THAT RATE SELECT (BIT 2) MAY BE SET
939	T21 TEST THAT RATE SELECT (BIT 3) MAY BE SET
953	T22 TEST CLOCK INTERRUPT ENABLE (BIT 6) CAN BE SET
967	T23 TEST MODE (BIT 8) CAN BE SET
981	T24 TEST EXT INTERRUPT ENABLE (BIT 14) CAN BE SET
995	T25 TEST THAT CLK DONE (BIT 7) CAN BE SET
1008	T26 TEST THAT CLK EXT INPUT (BIT 15) CAN BE SET
1021	T27 TEST THAT THE EXT FLAG DOES NOT SET FROM THE OUTSIDE SOURCE
1039	T30 MAINT. COUNT THE COUNTER REGISTER AT RATE 1MHZ
1083	T31 TEST THAT OVERFLOW SET CLK DONE (BIT 7)
1113	T32 MAINT. COUNT THE COUNTER REGISTER AT RATE #100KHZ
1154	T33 MAINT. COUNT THE COUNTER REGISTER AT RATE #10KHZ
1193	T34 MAINT. COUNT THE COUNTER REGISTER AT RATE #1KHZ
1230	T35 MAINT. COUNT THE COUNTER REGISTER AT RATE #100HZ
1267	T36 TEST THAT RESET CLEARS RATE SELECT AND MODE BITS
1284	T37 TEST THAT RESET CLEARS CLK INTERRUPT ENABLE
1301	T40 TEST THAT RESET CLEARS CLK FLAGS
1318	T41 TEST THAT RESET CLEARS COUNTER ENABLE
1339	T42 TEST CLOCK TO COUNT UP AT 1 MHZ
1361	T43 TEST CLOCK TO COUNT UP AT 100KHZ
1383	T44 TEST CLOCK TO COUNT UP AT 10 KHZ
1399	T45 TEST CLOCK TO COUNT UP AT 1KHZ
1414	T46 TEST CLOCK TO COUNT UP AT 100HZ
1429	T47 TEST THAT CLOCK ENABLE DOES NOT CLEAR ON DONE (MODE 1) 1 KHZ
1453	T50 TEST THAT CLOCK ENABLE DOES NOT CLEAR DONE (MODE 1 100HZ)
1476	T51 TEST THAT RESET SETS VC READY BIT
1489	T52 TEST THAT VC MODE BIT 2 CAN BE SET AND CLEARED

1513	T53	TEST THAT VC MODE BIT 3 CAN BE SET
1528	T54	TEST THAT VC INTERRUPT ENABLE (BIT 6) CAN BE SET
1543	T55	TEST THAT CHANNEL (BIT 9) CAN BE SET
1558	T56	TEST THAT STORE (BIT 10) CAN BE SET
1572	T57	TEST THAT WRITE THRU (BIT 11) CAN BE SET
1586	T60	TEST THAT ERASE (BIT 12) CAN BE SET
1603	T61	TEST THAT THE X REGISTER CAN BE CLEARED
1617	T62	TEST THAT THE X REGISTER CAN BE LOADED WITH #1777
1631	T63	TEST THAT THE X REGISTER CAN BE LOADED WITH #525
1645	T64	TEST THAT THE X REGISTER CAN BE LOADED WITH #1252
1659	T65	TEST THAT THE X REGISTER CAN HOLD A COUNT PATTERN
1679	T66	TEST THAT THE Y REGISTER CAN BE CLEARED
1693	T67	TEST THAT THE Y REGISTER CAN BE LOADED WITH #1777
1707	T70	TEST THAT THE Y REGISTER CAN BE LOADED WITH #525
1721	T71	TEST THAT THE Y REGISTER CAN BE LOADED WITH #1252
1735	T72	TEST THAT THE Y REGISTER CAN HOLD A COUNT PATTERN
1755	T73	TEST THAT THE X-Y REGISTERS CAN HOLD DIFFERENT DATA
1779	T74	TEST THAT WHEN INTENSIFY BIT IS SET THAT THE VC READY BIT RESETS
1802	T75	TEST THAT VC MODE 1 (INTENSIFY ON X) RESETS THE READY FLAG
1831	T76	TEST THAT VC MODE 2 (INTENSIFY ON Y) RESETS THE READY FLAG
1860	T77	TEST WHEN ERASE IS SET, VC READY BIT RESETS
1889	T100	TEST THAT RESET CLEARS VC MODE BITS
1906	T101	TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU
1923	T102	TEST THAT RESET CLEARS X REGISTER
1940	T103	TEST THAT RESET CLEARS Y REGISTER
1957	T104	DOES EXTERNAL ENABLE (BIT 4) SET
1971	T105	DOES CLOCK OVERFLOW ENABLE (BIT 5) SET
1986	T106	DOES AD INTERRUPT ENABLE (BIT 6) SET
2001	T107	DOES MUX CHANNEL (BIT 8) SET
2016	T110	DOES MUX CHANNEL (BIT 9) SET
2030	T111	DOES MUX CHANNEL (BIT 10) SET
2045	T112	DOES MUX CHANNEL (BIT 11) SET
2060	T113	DOES UNIPOLAR/BIPOLAR (BIT 13) SET
2077	T114	DOES AD DONE (BIT 7) SET AND CLEAR
2120	T115	TEST THAT THE CONVERTED NUMBER = 1777 (SW BIT 10=1)
2149	T116	TEST THAT NO EXTERNAL CONVERSIONS INPUT
2176	T117	TEST THAT CLOCK CAN START A CONVERSION
2204	T120	TEST THAT RESET CLEARS MUX AND UNIPOLAR BITS
2221	T121	TEST THAT RESET CLEARS EXT AND INTERRUPT ENABLE BITS
2237	T122	TEST THAT RESET CLEARS AD DONE
2256	T123	TEST THAT RESET CLEARS AD BUFFER REG
2277	T124	LOAD DIFFERENT NUMBERS INTO DIFFERENT REG.
2327	T125	DETERMINE IF MORE AR11'S ARE TO BE TESTED
2344		END OF PASS ROUTINE
2409		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2707		SCOPE HANDLER ROUTINE
2773		ERROR HANDLER ROUTINE
2823		TTY INPUT ROUTINE
2962		READ AN OCTAL NUMBER FROM THE TTY
3001		ERROR MESSAGE TIMEOUT ROUTINE
3049		POWER DOWN AND UP ROUTINES
3100		BINARY TO OCTAL (ASCII) AND TYPE
3177		TYPE ROUTINE
3256		APT COMMUNICATIONS ROUTINE

J01

MAINDEC-11-DRLPC-A MACY11 27(654) 15-DEC-77 10:54
DRLPC.P11 TABLE OF CONTENTS

SEQ 0009

3313 TRAP DECODER
3336 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
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
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

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

.REM !

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

TEST FOR PROPER SELECTION OF THE LOW BYTE OPERATION
TEST FOR PROPER SELECTION OF HIGH BYTE OPERATION
A TO D PRE-INTERRUPT SETUP
TEST THAT A TO D INTERRUPTS AT LEVEL INDICATED
TEST THAT A TO D DOES NOT INTERRUPT AT LEVEL INDICATED
CLOCK PRE-INTERRUPT SETUP
SCOPE PRE-INTERRUPT SETUP
TEST THAT THE CLOCK INTERRUPTS AT LEVEL INDICATED -1
TEST THAT THE CLOCK DOES NOT INTERRUPT AT LEVEL INDICATED
TEST THAT THE DISPLAY DOES INTERRUPT AT LEVEL INDICATED
TEST THAT THE DISPLAY DOES NOT INTERRUPT AT LEVEL INDICATED
TEST 1MHZ REPEATABILITY
TEST 100KHZ REPEATABILITY
TEST 10KHZ REPEATABILITY
TEST 1KHZ REPEATABILITY
TEST AD GO (BIT 0) CAN BE SET AND CLEARED

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

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

;*MISCELLANEOUS DEFINITIONS
000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
000012 LF= 12 ;;CODE FOR LINE FEED
000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
75 177776 PS= 177776 ;;PROCESSOR STATUS WORD
76 .EQUIV PS,PSW
77 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
78 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
79 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
80 177570 DDI:P= 177570 ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
81 000000 RO= %0 ;;GENERAL REGISTER

84	000001	R1=	%1	::	GENERAL REGISTER
85	000002	R2=	%2	::	GENERAL REGISTER
86	000003	R3=	%3	::	GENERAL REGISTER
87	000004	R4=	%4	::	GENERAL REGISTER
88	000005	R5=	%5	::	GENERAL REGISTER
89	000006	R6=	%6	::	GENERAL REGISTER
90	000007	R7=	%7	::	GENERAL REGISTER
91	000006	SP=	%6	::	STACK POINTER
92	000007	PC=	%7	::	PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

94		PR0=	0	::	PRIORITY LEVEL 0
95	000000	PR1=	40	::	PRIORITY LEVEL 1
96	000040	PR2=	100	::	PRIORITY LEVEL 2
97	000100	PR3=	140	::	PRIORITY LEVEL 3
98	000140	PR4=	200	::	PRIORITY LEVEL 4
99	000200	PR5=	240	::	PRIORITY LEVEL 5
100	000240	PR6=	300	::	PRIORITY LEVEL 6
101	000300	PR7=	340	::	PRIORITY LEVEL 7
102	000340				

.*"SWITCH REGISTER" SWITCH DEFINITIONS

104		SW15=	100000		
105	100000	SW14=	40000		
106	040000	SW13=	20000		
107	020000	SW12=	10000		
108	010000	SW11=	4000		
109	004000	SW10=	2000		
110	002000	SW09=	1000		
111	001000	SW08=	400		
112	000400	SW07=	200		
113	000200	SW06=	100		
114	000100	SW05=	40		
115	000040	SW04=	20		
116	000020	SW03=	10		
117	000010	SW02=	4		
118	000004	SW01=	2		
119	000002	SW00=	1		
120	000001				
121		.EQUIV	SW09, SW9		
122		.EQUIV	SW08, SW8		
123		.EQUIV	SW07, SW7		
124		.EQUIV	SW06, SW6		
125		.EQUIV	SW05, SW5		
126		.EQUIV	SW04, SW4		
127		.EQUIV	SW03, SW3		
128		.EQUIV	SW02, SW2		
129		.EQUIV	SW01, SW1		
130		.EQUIV	SW00, SW0		

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

132		BIT15=	100000		
133	100000	BIT14=	40000		
134	040000	BIT13=	20000		
135	020000	BIT12=	10000		
136	010000	BIT11=	4000		
137	004000				


```

138      002000      BIT10= 2000
139      001000      BIT09= 1000
140      000400      BIT08= 400
141      000200      BIT07= 200
142      000100      BIT06= 100
143      000040      BIT05= 40
144      000020      BIT04= 20
145      000010      BIT03= 10
146      000004      BIT02= 4
147      000002      BIT01= 2
148      000001      BIT00= 1
149      .EQUIV      BIT09,BIT9
150      .EQUIV      BIT08,BIT8
151      .EQUIV      BIT07,BIT7
152      .EQUIV      BIT06,BIT6
153      .EQUIV      BIT05,BIT5
154      .EQUIV      BIT04,BIT4
155      .EQUIV      BIT03,BIT3
156      .EQUIV      BIT02,BIT2
157      .EQUIV      BIT01,BIT1
158      .EQUIV      BIT00,BIT0
159
160      .*BASIC "CPU" TRAP VECTOR ADDRESSES
161      000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
162      000010      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
163      000014      TBITVEC=14        ;; "T" BIT
164      000014      TRTVEC= 14         ;; TRACE TRAP
165      000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
166      000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
167      000024      PWRVEC= 24         ;; POWER FAIL
168      000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
169      000034      TRAPVEC=34        ;; "TRAP" TRAP
170      000060      TKVEC= 60          ;; TTY KEYBOARD VECTOR
171      000064      TPVEC= 64          ;; TTY PRINTER VECTOR
172      000240      PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
173      170400      ABASE=170400
174      000340      AVECT1=340
175      000200      APRIOR=200
176      .SBTTL      OPERATIONAL SWITCH SETTINGS
177      .*
178      .*          SWITCH          USE
179      .*          -----
180      .*          15             HALT ON ERROR
181      .*          14             LOOP ON TEST
182      .*          13             INHIBIT ERROR TYPEOUTS
183      .*          12             STORAGE SCOPE CONNECTED
184      .*          11             INHIBIT ITERATIONS
185      .*          10             ALL 1'S JUMPER IS INSTALLED
186      .*          9              LOOP ON ERROR
187      .*          8              LOOP ON TEST IN SWR<7:0>
188      .SBTTL      TRAP CATCHER
189
190      000000      .=0
191      .*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
    
```

```

192      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
193      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
194      . =174
195      000174 000000  DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
196      000176 000000  SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
197      .SBTTL  STARTING ADDRESS(ES)
198      000200 000137 001672  JMP      @#BEGIN ;; JUMP TO STARTING ADDRESS OF PROGRAM
199      000204 000137 001676  JMP      @#BEGIN1 ;; JUMP TO RESTART ADDRESS
200
201      .SBTTL  ACT11 HOOKS
202
203      ;*****
204      ;HOOKS REQUIRED BY ACT11
205      $SVPC=.          ;SAVE PC
206      . =46
207      000046 013746  $ENDAD          ;; 1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
208      . =52
209      000052 000000  .WORD 0        ;; 2)SET LOC.52 TO ZERO
210      . = $SVPC      ;; RESTORE PC
211      . =1000
212      .SBTTL  APT PARAMETER BLOCK
213
214      ;*****
215      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
216      ;*****
217      . $X=.          ;SAVE CURRENT LOCATION
218      . =24          ;SET POWER FAIL TO POINT TO START OF PROGRAM
219      000024 000200  200           ;FOR APT START UP
220      . =44          ;POINT TO APT INDIRECT ADDRESS PNTR.
221      000044 001000  $APTHDR      ;POINT TO APT HEADER BLOCK
222      . = $X          ;RESET LOCATION COUNTER
223      ;*****
224      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
225      ;INTERFACE SPEC.
226
227      001000  $APTHD:
228      001000 000000  $HIBTS: .WORD 0          ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
229      001002 001176  $MBADR: .WORD $MAIL      ;; ADDRESS OF AP: MAILBOX (BITS 0-15)
230      001004 000010  $STMT:  .WORD 10        ;; RUN TIM OF LONGEST TEST
231      001006 000010  $PASTM: .WORD 10        ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
232      001010 000200  $UNITM: .WORD 200       ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
233      001012 000052  .WORD $ETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)
    
```

234
235
236
237
238
239
240
241 001100
242 001100 000000
243 001102 000
244 001103 000
245 001104 000000
246 001106 000000
247 001110 000000
248 001112 000000
249 001114 000
250 001115 001
251 001116 000000
252 001120 000000
253 001122 000000
254 001124 000000
255 001126 000000
256 001130 000000
257 001132 000000
258 001134 000
259 001135 000
260 001136 000000
261 001140 177570
262 001142 177570
263 001144 177560
264 001146 177562
265 001150 177564
266 001152 177566
267 001154 000
268 001155 002
269 001156 012
270 001157 000
271 001160 000000
272
273 001162 000000
274 001164 000000
275 001166 000000
276 001170 000000
277 001172 077
278 001173 015
279 001174 000012
280
281
282
283
284
285 001176
286 001176 000000
287 001200 000000

.SBTTL COMMON TAGS

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

SCMTAG: . =1100

;; START OF COMMON TAGS
\$STNM: .WORD 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERITL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
\$DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$REGAD: .WORD 0
\$REGO: .WORD 0
\$REG1: .WORD 0
\$TIMES: 0
\$ESCAPE: 0
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL BYTE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED
;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR
;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
;; CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
;; CONTAINS ((\$REGAD)+0)
;; CONTAINS ((\$REGAD)+2)
;; MAX. NUMBER OF ITERATIONS
;; ESCAPE ON ERROR ADDRESS
;; QUESTION MARK
;; CARRIAGE RETURN
;; LINE FEED

.SBTTL APT MAILBOX-ETABLE

\$MAIL: .WORD AMSTY ;; APT MAILBOX
\$MSGTY: .WORD AFATAL ;; MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;; FATAL ERROR NUMBER

342 001314 000000
343 001316 000000
344 001320 000000
345
346
347 001322
348

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

SETEND:

349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363 001322
 364
 365
 366 001322 014620
 367 001324 015414
 368 001326 016220
 369 001330 000000
 370
 371
 372 001332 014660
 373 001334 015454
 374 001336 016232
 375 001340 000000
 376
 377
 378 001342 014707
 379 001344 015472
 380 001346 016240
 381 001350 000000
 382
 383
 384 001352 014746
 385 001354 015532
 386 001356 016252
 387 001360 000000
 388
 389
 390 001362 015005
 391 001364 015572
 392 001366 016264
 393 001370 000000
 394
 395
 396 001372 015033
 397 001374 015607
 398 001376 016272
 399 001400 000000
 400
 401
 402 001402 015073

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

;ITEM 1 ;:A TO D STATUS REGISTER IN ERROR
 EM1 ;:ERRPC ADCS AOSTAT EXPECTED
 DH1 ;:\$ERRPC ADCS \$BDDAT \$GDDAT
 DT1
 0

 ;ITEM 2 ;:A TO d INTERRUPT ERROR
 EM2 ;:ERRPC ADCS
 DH2 ;:\$ERRPC ADCS
 DT2
 0

 ;ITEM 3 ;:CLOCK STATUS REGISTER IN ERROR
 EM3 ;:ERRPC CSR CKSTAT EXPECTED
 DH3 ;:\$ERRPC CSR \$BDDAT \$GDDAT
 DT3
 0

 ;ITEM 4 ;:CLOCK PRESET REGISTER IN ERROR
 EM4 ;:ERRPC CSB CKBUFF EXPECTED
 DH4 ;:\$ERRPC CSB \$BDDAT \$GDDAT
 DT4
 0

 ;ITEM 5 ;:CLOCK INTERRUPT ERROR
 EM5 ;:ERRPC CSR
 DH5 ;:\$ERRPC CSR
 DT5
 0

 ;ITEM 6 ;:CLOCK COUNTER REGISTER IN ERROR
 EM6 ;:ERRPC CSC CKCNTR EXPECTED
 DH6 ;:\$ERRPC CSC \$BDDAT \$GDDAT
 DT6
 0

 ;ITEM 7 ;:CLOCK COUNTED IN ERROR
 EM7

403	001404	015607	DH6	:ERRPC	CSC	CKCNTR	EXPECTED
404	001406	016272	DT6	;\$ERRPC	CSC	\$BDDAT	\$GDDAT
405	001410	000000	0				
406							
407			; ITEM	10			
408	001412	015122	EM10	:CLOCK REPEATABILITY FAILED			
409	001414	015647	DH10	:ERRPC	CSC	TIME1	TIME2
410	001416	016272	DT6	;\$ERRPC	CSC	\$BDDAT	\$GDDAT
411	001420	000000	0				
412							
413			; ITEM	11			
414	001422	015155	EM11	:VC STATUS REGISTER IN ERROR			
415	001424	015705	DH11	:ERRPC	VCADR	VCSTAT	EXPECTED
416	001426	016304	DT11	;\$ERRPC	VCSTAT	\$BDDAT	\$GDDAT
417	001430	000000	0				
418							
419			; ITEM	12			
420	001432	015211	EM12	:X REGISTER IN ERROR			
421	001434	015745	DH12	:ERRPC	VCXPOS	X POS.	EXPECTED
422	001436	016316	DT12	;\$ERRPC	VCXPOS	\$BDDAT	\$GDDAT
423	001440	000000	0				
424							
425			; ITEM	13			
426	001442	015235	EM13	:Y REGISTER IN ERROR			
427	001444	016006	DH13	:ERRPC	VCYPOS	Y POS.	EXPECTED
428	001446	016330	DT13	;\$ERRPC	VCYPOS	\$BDDAT	\$GDDAT
429	001450	000000	0				
430							
431			; ITEM	14			
432	001452	015261	EM14	:SCOPE INTERRUPT ERROR			
433	001454	016047	DH14	:ERRPC	VCADR		
434	001456	016342	DT14	;\$ERRPC	VCSTAT		
435	001460	000000	0				
436							
437			; ITEM	15			
438	001462	015307	EM15	:DEVICE BUS ERROR			
439	001464	016066	DH15	:ERRPC	BASE ADDRESS	ACTUAL ADDRESS	
440	001466	016350	DT15	;\$ERRPC	ARBADD	\$BDDAT	
441	001470	000000	0				
442							
443			; ITEM	16			
444	001472	015330	EM16	:INCORRECT A/D BUFFER DATA			
445	001474	016116	DH16	:ERRPC	ADDR	BUFFER	EXPECTED
446	001476	016360	DT16	;\$ERRPC	ADDR	\$BDDAT	\$GDDAT
447	001500	000000	0				
448							
449			; ITEM	17			
450	001502	015362	EM17	:DUAL REGISTER ADDRESSING			
451	001504	016157	DH17	:ERRPC	BUFADR	READ	EXPECTED
452	001506	016372	DT17	;\$ERRPC	BUFADR	\$BDDAT	\$GDDAT
453	001510	000000	0				
454							
455							
456							

; ADDRESS OF KMC-11 OF LPA-11 THE ADDR FOR KMADD MAY BE

CHANGED BY THE USER TO REFLECT
A DIFFERENT KMC-II ADDR. THE
REST OF THE ADDRESSES WILL
BE CHANGED BY THE PROGRAM.

457			:		
458			:		
459			:		
460			:		
461			:		
462			:		
463	001512		LPCI:		
464	001512	170460	KMAD0:	.WORD 170460	;BASE KMC ADDR. MAY BE PATCHED BY USER.
465					
466	001514		LPMR:		
467	001514	170461	KMAD1:	.WORD 170460+1	; >DO NOT < ;KMC-CSR ADDR
468	001516		LPC0:		
469	001516	170462	KMAD2:	.WORD 170460+2	; >PATCH < ;
470	001520		LPS0:		
471	001520	170463	KMAD3:	.WORD 170460+3	; >THIS AREA <
472	001522		LPADL:		
473	001522	170464	KMAD4:	.WORD 170460+4	;
474	001524		LPADH:		
475	001524	170465	KMAD5:	.WORD 170460+5	; >DO NOT <
476	001526		LPMS1:		
477	001526	170466	KMAD6:	.WORD 170460+6	; >PATCH <
478	001530		LPMS2:		
479	001530	170467	KMAD7:	.WORD 170460+7	; >THIS AREA <
480					
481	001532	000340	VECTOR:	.WORD AVECT1&777	;BASE VECTOR OF KMC
482	001534	000344	VECTPS:	.WORD 4+AVECT1&777	;VECTR ADDR.+2
483					
484	001536	000004	VERSN:	.WORD 4	;CURRENT VERSION NUMBER OF MICROCODE.
485					
486	001540	000000	.DVLS:	.WORD 0	; /DEVICE LIST OF I/O ADDR. DEFINED
487	001542	000020		.BLKW 16.	; /BY INIT.
488					
489	001602	000300	ADBRL:	300	;A TO D BR LEVEL
490	001604	000300	CKBRL:	300	;CLOCK BR LEVEL
491	001606	000200	VCBRL:	200	;SCOPE BR LEVEL
492					
493					
494					
495	001610	170400	ADCS:	170400	;A TO D STATUS REG
496	001612	170402	ADDBR:	170402	;A TO D BUFFER
497					
498	001614	170404	CSR:	170404	;CLOCK STATUS REGISTER
499	001616	170406	CSB:	170406	;CLOCK PRESET BUFFER
500					
501	001620	170410	VCSTAT:	170410	;VC STATUS REGISTER
502	001622	170412	VCXREG:	170412	;VC X AXIS REGISTER
503	001624	170414	VCYREG:	170414	;VC Y AXIS REGISTER
504					
505					
506	001626	170416	CSC:	170416	;CLOCK COUNTER REGISTER
507					
508	001630	000340	ADINT:	340	;A TO D INTERRUPT VECTOR
509	001632	000342	ADINT1:	342	
510					

;PROGRAM WILL CHANGE THE NEXT 16 LOCATIONS

```

511 001634 000344 KWIV: 344 ;CLOCK INTERRUPT VECTOR
512 001636 000346 KWIVS: 346
513
514 001640 000350 VCIV: 350 ;SCOPE INTERRUPT VECTOR
515 001642 000352 VCIVS: 352
516
517 001644 170400 ARBADD: 170400 ;CURRENT DEVICE ADDRESS
518 001646 000340 ARBVCT: 340 ;CURRENT DEVICE VECTOR
519 001650 000000 NMBEXT: 0 ;NUMBER OF ADDITIONAL AR11'S
520 001652 000000 NBEXT: 0
521 001654 000000 COUNT: 0
522 001656 000000 DELAY: 0
523 001660 000000 TEMP: 0
524 001662 000000 SWITCH: 0
525 001664 000000 BRLEV1: 0
526 001666 000000 BRLEV2: 0
527 001670 000000 $TMDAT: 0
528
529
530 001672 005000 BEGIN: CLR RO ;CLEAR RO
531 001674 000402 BR RBEG
532 001676 012700 177777 BEGIN1: MOV #-1,RO ;LOAD RO
533 001702 RBEG:
534
535 ; THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
536 ;
537
538 001702 010046 MOV RO, -(SP)
539 001704 010146 MOV R1, -(SP)
540 001706 013700 001512 MOV KMADD, RO ;GET KMC-11 ADDRESS.
541 001712 012701 001514 MOV #KMAD1, R1 ;GET ADDR. OF ADDR. LIST.
542
543 001716 005200 64$: INC RO ;UPDATE ADDR.
544 001720 010021 MOV RO, (1)+ ;WRITE ADDR.
545 001722 020127 001532 CMP R1, #KMAD7+2 ;DONE ALL ADDRESSES?
546 001726 001373 BNE 64$ ;NO - DO NEXT ADDR.
547 001730 005037 001540 CLR .DVLS ;CLR ADDR. LIST.
548 001734 012601 MOV (SP)+, R1
549 001736 012600 MOV (SP)+, RO
550 .SBTTL INITIALIZE THE COMMON TAGS
551 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
552 001740 012706 001100 MOV #SCMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
553 001744 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
554 001746 022706 001140 CMP #SWR, R6 ;;DONE?
555 001752 001374 BNE -6 ;;LOOP BACK IF NO
556 001754 012706 001100 MOV #STACK, SP ;;SETUP THE STACK POINTER
557 ;;INITIALIZE A FEW VECTORS
558 001760 012737 016404 000020 MOV #SCOPE, #IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
559 001766 012737 000340 000022 MOV #340, #IOTVEC+2 ;;LEVEL 7
560 001774 012737 016666 000030 MOV #ERROR, #EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
561 002002 012737 000340 000032 MOV #340, #EMTVEC+2 ;;LEVEL 7
562 002010 012737 021262 000034 MOV #TRAP, #TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
563 002016 012737 000340 000036 MOV #340, #TRAPVEC+2 ;;LEVEL 7
564 002024 012737 020064 000024 MOV #PWRDN, #PWRVEC ;;POWER FAILURE VECTOR

```

```

565 002032 012737 000340 000026      MOV      #340, @#PWRVEC+2    ;; LEVEL 7
566 002040 013737 013714 013706      MOV      $ENDCT, $EOPCT    ;; SETUP END-OF-PROGRAM COUNTER
567 002046 005037 001166                CLR      $TIMES            ;; INITIALIZE NUMBER OF ITERATIONS
568 002052 005037 001170                CLR      $ESCAPE          ;; CLEAR THE ESCAPE ON ERROR ADDRESS
569 002056 112737 000001 001115      MOV      #1, $ERMAX        ;; ALLOW ONE ERROR PER TEST
570 002064 012737 002064 001106      MOV      #., $LPADR        ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
571 002072 012737 002072 001110      MOV      #., $LPERR        ;; SETUP THE ERROR LOOP ADDRESS
572                                     ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
573                                     ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
574 002100 013746 000004                MOV      @#ERRVEC -(SP)    ;; SAVE ERROR VECTOR
575 002104 012737 002140 000004      MOV      #65$, @#ERRVEC    ;; SET UP ERROR VECTOR
576 002112 012737 177570 001140      MOV      #DSWR, SWR        ;; SETUP FOR A HARDWARE SWICH REGISTER
577 002120 012737 177570 001142      MOV      #DDISP, DISPLAY   ;; AND A HARDWARE DISPLAY REGISTER
578 002126 022777 177777 177004      CMP      #-1, @SWR         ;; TRY TO REFERENCE HARDWARE SWR
579 002134 001012                BNE      67$              ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
580                                     ;; AND THE HARDWARE SWR IS NOT = -1
581 002136 000403                TR      66$              ;; BRANCH IF NO TIMEOUT
582 002140 012716 002146 66$:          MOV      #66$, (SP)        ;; SET UP FOR TRAP RETURN
583 002144 000002                RTI
584 002146 012737 000176 001140 66$:   MOV      #SWREG, SWR       ;; POINT TO SOFTWARE SWR
585 002154 012737 000174 001142      MOV      #DISPRG, DISPLAY  ;;
586 002162 012637 000004 67$:          MOV      (SP)+, @#ERRVEC   ;; RESTORE ERROR VECTOR
587
588 002166 005037 001204                CLR      $PASS            ;; CLEAR PASS COUNT
589 002172 132737 000200 001217      BITB    #APTSIZE, $ENVM    ;; TEST USER SIZE UNDER APT
590 002200 001403                BEQ     68$              ;; YES, USE NON-APT SWITCH
591 002202 012737 001220 001140      MOV     #SSWREG, SWR      ;; NO, USE APT SWITCH REGISTER
592 002210 68$:
593 002210 012706 001100                MOV     #STACK, SP        ;; LOAD STACK
594 002214 012737 002262 000004      MOV     #1$, @#4         ;; LOAD BUS ERROR
595 002222 013702 001252                MOV     $BASE, R2        ;; LOAD STARTING ADDRESS
596 002226 005003                CLR     R3                ;; CLEAR COUNT
597 002230 010237 001670 2$:          MOV     R2, $TMDAT
598
599                                     ; *
600 002244 005737 022222                MOV     @STMDAT, $GDDAT   ; /READ DEVICE REG $TMDAT, PUT DATA IN $GDDAT.
601 002250 001004                TST    $AERR
602 002252 062702 000020                BNE    1$
603 002256 005203                ADD    #20, R2           ;; EXIST, UPDATE TEST ADDRESS
604 002260 000763                INC    R3                ;; UPDATE # OF AR11'S
605 002262 005703                BR     2$
606 002264 001001                TST    R3                ;; TEST IF FIRST DOES EXIST
607 002266 000000                BR     3$
608                                     ;; FIRST AR11 DOES NOT EXIST
609 002270 005303                BNE    3$                ;; CHECK THE PROGRAM DEVICE ADDRESS
610 002272 010337 001650 3$:          DEC    R3                ;; ADJUST R3
611 002276 012737 000006 000004      MOV     R3, NMBEXT        ;; SAVE THE NUMBER OF ADDITIONAL AR11
612 002304 005037 000006                MOV     #6, @#4          ;; RESET BUS ERROR
613 002310 013737 001252 001644      CLR     @#6
614 002316 013737 001246 001646      MOV     $BASE, ARBADD     ;; LOAD FIRST ADDRESS
615 002324 013737 001650 001652      MOV     $VECT1, ARBVCT   ;; LOAD FIRST VECTOR
616 002332 000240                MOV     NMBEXT, NBEXT    ;; LOAD NUMBER OF AR11'S
617 002334 RBEG1: NOP
618 002334 RBEG2: MOV      #232, R2    ;; LOAD R2

```



```

619 002340 012701 000230      MOV      #230,R1      ;LOAD R1
620 002344 010221      5$: MOV      R2,(R1)+  ;LOAD .+2
621 002346 005021      CLR      (R1)+      ;LOAD HALT
622 002350 010102      MOV      R1,R2      ;LOAD R2
623 002352 005722      TST      (R2)+      ;BUMP R2
624 002354 020227 001002      CMP      R2,#1002   ;TEST FOR LFST
625 002360 001371      BNE      5$         ;BR UNTIL DCNE
626 002362 005700      TST      R0         ;TEST R0
627 002364 001402      BEQ      2$         ;BR IF CLEARED
628 002366 000137 002642      JMP      4$         ;INHIBIT TYPOUT
629 002372 005737 000042      2$: TST      @#42     ;TEST ACT-11 OR DOP
630 002376 001402      BEQ      3$         ;BR IF CLEARED
631 002400 000137 002642      JMP      4$         ;INHIBIT TYPOUT
632 002404
633 002404 104401 002412      3$: TYPE     65$      ;;TYPE ASCIZ STRING
634 002410 000415      BR      64$        ;;GET OVER THE ASCIZ
635
636 002444      ;;65$: .ASCIZ <15><12>/AR-11 DIAGNOSTIC TEST I/
637
638      64$: .SBTTL TYPE PROGRAM NAME
639      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
640 002444 005227 177777      INC      #-1        ;;FIRST TIME?
641 002450 001044      BNE      66$        ;;BRANCH IF NO
642 002452 022737 013746 000042      CMP      #SENDAD,@#42 ;ACT-11?
643 002460 001440      BEQ      66$        ;;BRANCH IF YES
644 002462 104401 002530      TYPE     67$        ;TYPE ASCIZ STRING
645 002466 005737 000042      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
646 002472 001012      TST      @#42     ;ARE WE RUNNING UNDER XXDP/ACT?
647 002474 123727 001216 000001      BNE      68$        ;BRANCH IF YES
648 002502 001406      CMPB    $ENV,#1    ;ARE WE RUNNING UNDER APT?
649 002504 023727 001140 000176      BEQ      68$        ;BRANCH IF YES
650 002512 001005      CMP      SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
651 002514 104406      BNE      69$        ;BRANCH IF NO
652 002516 000403      GTSWR
653 002520 112737 000001 001134      BR      69$        ;GET SOFT-SWR SETTINGS
654 002526      68$: MOVVB   #1,$AUTOB ;SET AUTO-MODE INDICATOR
655 002526 000415      69$: BR      66$        ;GET OVER THE ASCIZ
656
657 002562      ;;67$: .ASCIZ <CRLF><15><12>/MAINDEC-11-DRLPC-A/<15><12><CRLF>
658 002562 013746 001650      66$: MOV      NMBEXT,-(SP) ;PUSH ON STACK
659 002566 104403      TYPOS
660 002570 000002      .WORD   2          ;TYPE OCTAL
661 002572 104401 002600      TYPE     71$        ;TYPE ASCIZ STRING
662 002576 000421      BR      70$        ;GET OVER THE ASCIZ
663
664 002642      ;;71$: .ASCIZ / (8) ADDITIONAL AR11'S CONNECTED/<15><12>
665      70$:
666 002642 000240      4$: NOP
667 002644 012700 001610      MOV      #ADCS,R0   ;LOAD POINTER
668 002650 013720 001644      10$: MOV      ARBADD,(R0)+
669 002654 022700 001630      CMP      #ADINT,R0  ;TEST FOR END
670 002660 001373      BNE
671 002662 013720 001646      11$: MOV      ARBVCT,(R0)+ ;LOAD VECTOR
672 002666 022700 001644      CMP      #ARBADD,R0

```

```

673 002672 001373          BNE      11$
674 002674 012700 001612    MOV     #AODBR,R0
675 002700 012701 000002    MOV     #2,R1
676 002704 060120          12$:   ADD     R1,(R0)+
677 002706 005721          TST     (R1)+
678 002710 022701 000020    CMP     #20,R1
679 002714 001373          BNE     12$
680 002716 005720          TST     (R0)+
681 002720 012701 000002    MOV     #2,R1
682 002724 060120          13$:   ADD     R1,(R0)+
683 002726 005721          TST     (R1)+
684 002730 022701 000014    CMP     #14,R1
685 002734 001373          BNE     13$
686                                     ;*****
687                                     ;*TEST 1      TEST EACH BUS ADDRESS FOR TIMEOUT
688                                     ;*****
689 002736 000004          †ST1:  SCOPE
690
691 002740 005037 001540    CLR     .DVL$
692 002744 013737 001610 001126  2$:   MOV     ADCS,$BDDAT      ;LOAD WITH BUS ADDRESS TO BE TESTED
693 002752
694
695                                     ;*
696                                     MOV     $GDDAT,$SBDDAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG $BDDAT
697 002762 005737 022222          TST     $AERR           ;TEST ADDRESS
698 002766 001011          BNE     1$
699 002770 023737 001126 001626    CMP     $BDDAT,CSC     ;TEST FOR LAST
700 002776 001404          BEQ     3$             ;BR IF DONE
701 003000 062737 000002 001126    ADD     #2,$BDDAT      ;MAKE NEXT ADDRESS
702 003006 000761          BR      2$            ;TRY MORE
703 003010
704 003010 000403          3$:   BR      TST2         ;;BR TO NEXT TEST
705
706 003012 104015 013572    1$:   ERROR  15           ;DEVICE BUS ERROR
707 003014 000137          JMP     BYPASS         ;DONT TEST ANY MORE
708
709
710                                     ;*****
711                                     ;*TEST 2      TEST THAT THE PRESET BUFFER CAN HOLD #0
712                                     ;*****
713 003020 000004          †ST2:  SCOPE
714 003022 012737 000000 001124    MOV     #0,$GDDAT      ;LOAD EXPECTED
715
716                                     ;*
717                                     MOV     $GDDAT,$CSB     ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
718                                     ;*
719 003050 023737 001124 001126    MOV     $CSB,$BDDAT    ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
720 003056 001401          CMP     $GDDAT,$BDDAT ;COMPARE
721 003060 104004          BEQ     TST3          ;;BR IF EQUAL
722                                     ERROR  4               ;ERROR, COUNTER PRESET FAILED TO CLEAR
723
724                                     ;*****
725                                     ;*TEST 3      TEST THE COUNTER PRESET BUFFER CAN HOLD #377
726                                     ;*****
726 003062 000004          †ST3:  SCOPE

```

```

727 003064 012737 000377 001124      MOV      #377,$GDDAT          ;LOAD EXPECTED
728
729          ;*      MOV      $GDDAT,2CSB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
730
731          ;*      MOV      2CSB,$BDDAT      ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
732 003112 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
733 003120 001401          BEQ      TST4              ;;BR IF EQUAL
734 003122 104004          ERROR    4                ;;ERROR, COUNTER PRESET FAILED TO LOAD
735
736          ;;*****
737          ;;TEST 4      TEST THAT PRESET BUFFER CAN HOLD #125
738          ;;*****
739 003124 000004      TST4:  SCOPE
740 003126 012737 000125 001124      MOV      #125,$GDDAT        ;LOAD EXPECTED
741
742          ;*      MOV      $GDDAT,2CSB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
743
744          ;*      MOV      2CSB,$BDDAT      ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
745 003154 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
746 003162 001401          BEQ      TST5              ;;BR IF EQUAL
747 003164 104004          ERROR    4                ;;ERROR, COUNTER PRESET FAILED TO LOAD
748
749          ;;*****
750          ;;TEST 5      TEST THAT PRESET BUFFER CAN HOLD #252
751          ;;*****
752 003166 000004      TST5:  SCOPE
753 003170 012737 000252 001124      MOV      #252,$GDDAT        ;LOAD EXPECTED
754
755          ;*      MOV      $GDDAT,2CSB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
756
757          ;*      MOV      2CSB,$BDDAT      ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
758 003216 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
759 003224 001401          BEQ      TST6              ;;BR IF EQUAL
760 003226 104004          ERROR    4                ;;ERROR, COUNTER PRESET FAILED TO LOAD
761
762          ;;*****
763          ;;TEST 6      TEST THAT PRESET BUFFER CAN HOLD A COUNT PATTERN
764          ;;*****
765 003230 000004      TST6:  SCOPE
766 003232 012737 003244 001110      MOV      #1$,2LPERR        ;LOAD ERROR SCOPE RETURN
767 003240 005037 001124          CLR      $GDDAT            ;CLEAR PATTERN
768 003244          1$:
769
770          ;*      MOV      $GDDAT,2CSB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
771
772          ;*      MOV      2CSB,$BDDAT      ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
773 003264 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
774 003272 001401          BEQ      2$                ;;BR IF EQUAL
775 003274 104004          ERROR    4                ;;PRESET BUFFER FAILED TO HOLD A COUNT PATTERN
776
777 003276 105237 001124      2$:   INCB    $GDDAT          ;UPDATE PATTERN
778 003302 001360          BNE     1$                ;BR UNTIL DONE
779
780          ;;*****

```

```

781 ;*TEST 7 TEST INIT TO CLEAR COUNT PRESET BUFFER WHEN IT IS =-1
782 ;*****
783 003304 000004 ;*ST7: SCOPE
784 003306 012737 000020 001166 MOV #20,$TIMES ;;DO 20 ITERATIONS
785 003314 012737 000000 001124 MOV #0,$GDDAT ;LOAD EXPECTED
786 003322 012737 177777 001670 MOV #-1,$TMDAT
787
788 ;* MOV $TMDAT,$CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
789 003340 004737 023174 JSR PC,$RESET
790
791 ;* MOV $CSB,$BDDAT ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
792 003354 005737 001126 TST $BDDAT
793 003360 001401 BEQ TST10 ;;BR IF EQUAL
794 003362 104004 ERROR 4 ;ERROR, INIT FAILED TO CLEAR CSB
795
796 ;*****
797 ;*TEST 10 TEST THAT THE COUNTER CAN HOLD #0
798 ;*****
799 003364 000004 ;*ST10: SCOPE
800 003366 012737 000000 001124 MOV #0,$GDDAT ;LOAD EXPECTED
801
802 ;* MOV $GDDAT,$CSB ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
803
804 ;* MOV $CSB,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
805 003414 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
806 003422 001401 BEQ TST11 ;;BR IF EQUAL
807 003424 104006 ERROR 6 ;ERROR, COUNTER FAILED TO CLEAR
808
809 ;*****
810 ;*TEST 11 TEST THE COUNTER CAN HOLD #377
811 ;*****
812 003426 000004 ;*ST11: SCOPE
813 003430 012737 000377 001124 MOV #377,$GDDAT ;LOAD EXPECTED
814
815 ;* MOV $GDDAT,$CSB ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
816
817 ;* MOV $CSB,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
818 003456 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
819 003464 001401 BEQ TST12 ;;BR IF EQUAL
820 003466 104006 ERROR 6 ;ERROR, COUNTER FAILED TO LOAD
821
822 ;*****
823 ;*TEST 12 TEST THAT COUNTER CAN HOLD #125
824 ;*****
825 003470 000004 ;*ST12: SCOPE
826 003472 012737 000125 001124 MOV #125,$GDDAT ;LOAD EXPECTED
827
828 ;* MOV $GDDAT,$CSB ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSB
829
830 ;* MOV $CSB,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
831 003520 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
832 003526 001401 BEQ TST13 ;;BR IF EQUAL
833 003530 104006 ERROR 6 ;ERROR, COUNTER FAILED TO LOAD
834

```

```

835
836
837
838
839 003532 000004
840 003534 012737 000252 001124
841
842
843
844
845 003562 023737 001124 001126
846 003570 001401
847 003572 104006
848
849
850
851
852 003574 000004
853 003576 012737 003610 001110
854 003604 005037 001124
855 003610
856
857
858
859
860 003630 023737 001124 001126
861 003636 001401
862 003640 104006
863
864 003642 105237 001124
865 003646 001360
866
867
868
869
870 003650 000004
871 003652 012737 000000 001124
872 003660 012737 177777 001670
873
874
875 003676 004737 023174
876
877
878 003712 005737 001126
879 003716 001401
880 003720 104006
881
882
883
884
885
886 003722 000004
887 003724 012737 000021 001124
888

```

```

*****
*TEST 13 TEST THAT COUNTER CAN HOLD #252
*****
↑ST13: SCOPE
MOV #252,$GDDAT ;LOAD EXPECTED
;* MOV $GDDAT,$CSB ; PUT DATA FROM $GDDAT TO DEVICE REG CSB
;* MOV $CSB,$BDDAT ; READ DEVICE REG CSC,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST14 ; BR IF EQUAL
ERROR 6 ;ERROR, COUNTER FAILED TO LOAD

*****
*TEST 14 TEST THAT COUNTER CAN HOLD A COUNT PATTERN
*****
↑ST14: SCOPE
MOV #1,$LPERR ;LOAD ERROR SCOPE RETURN
CLR $GDDAT ;CLEAR PATTERN
1$:
;* MOV $GDDAT,$CSB ; PUT DATA FROM $GDDAT TO DEVICE REG CSB
;* MOV $CSB,$BDDAT ; READ DEVICE REG CSC,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ; BR IF EQUAL
ERROR 6 ;COUNTER FAILED TO HOLD A COUNT PATTERN
2$: INCB $GDDAT ;UPDATE PATTERN
BNE 1$ ;BR UNTIL DONE

*****
*TEST 15 TEST INIT TO CLEAR COUNTER WHEN IT IS =-1
*****
↑ST15: SCOPE
MOV #0,$GDDAT ;LOAD EXPECTED
MOV #-1,$TMDAT
;* MOV $TMDAT,$CSB ; PUT DATA FROM $TMDAT TO DEVICE REG CSB
JSR PC,$RESET
;* MOV $CSB,$BDDAT ; READ DEVICE REG CSC,PUT DATA IN $BDDAT.
TST $BDDAT
BEQ TST16 ; BR IF EQUAL
ERROR 6 ;ERROR, INIT FAILED TO CLEAR COUNTER

*****
*TEST 16 TEST ENABLE COUNTER (BIT 0) CAN BE SET
*****
↑ST16: SCOPE
MOV #BIT4!BIT0,$GDDAT ;LOAD EXPECTED

```



```

889 ;* MOV $GDDAT,DCSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
890
891 ;* MOV DCSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
892 003752 023737 001124 001126 ;* CMP $GDDAT,$BDDAT ;COMPARE
893 003760 001401 BEQ TST17 ;;BR IF EQUAL
894 003762 104003 ERROR 3 ;ERROR COUNTER ENABLE FAILED TO SET
895
896 ;*****
897 ;*TEST 17 TEST RATE SELECT (BIT 1) MAY BE SET
898 ;*****
899 003764 000004 TST17: SCOPE
900 003766 012737 000022 001124 MOV #BIT4!BIT1,$GDDAT ;LOAD EXPECTED
901
902 ;* MOV $GDDAT,DCSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
903
904 ;* MOV DCSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
905 004014 023737 001124 001126 ;* CMP $GDDAT,$BDDAT ;COMPARE
906 004022 001401 BEQ TST20 ;;BR IF EQUAL
907 004024 104003 ERROR 3 ;ERROR, RATE BIT 1 FAILED TO SET
908
909 ;*****
910 ;*TEST 20 TEST THAT RATE SELECT (BIT 2) MAY BE SET
911 ;*****
912 004026 000004 TST20: SCOPE
913 004030 012737 000024 001124 MOV #BIT4!BIT2,$GDDAT ;LOAD EXPECTED
914
915 ;* MOV $GDDAT,DCSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
916
917 ;* MOV DCSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
918 004056 023737 001124 001126 ;* CMP $GDDAT,$BDDAT ;COMPARE
919 004064 001401 BEQ TST21 ;;BR IF EQUAL
920 004066 104003 ERROR 3 ;ERROR, RATE BIT 2 FAILED TO SET
921
922 ;*****
923 ;*TEST 21 TEST THAT RATE SELECT (BIT 3) MAY BE SET
924 ;*****
925 004070 000004 TST21: SCOPE
926 004072 012737 000030 001124 MOV #BIT4!BIT3,$GDDAT ;LOAD EXPECTED
927
928 ;* MOV $GDDAT,DCSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
929
930 ;* MOV DCSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
931 004120 023737 001124 001126 ;* CMP $GDDAT,$BDDAT ;COMPARE
932 004126 001401 BEQ TST22 ;;BR IF EQUAL
933 004130 104003 ERROR 3 ;ERROR, RATE BIT 3 FAILED TO SET
934
935 ;*****
936 ;*TEST 22 TEST CLOCK INTERRUPT ENABLE (BIT 6) CAN BE SET
937 ;*****
938 004132 000004 TST22: SCOPE
939 004134 012737 000120 001124 MOV #BIT6!BIT4,$GDDAT ;LOAD EXPECTED
940
941 ;* MOV $GDDAT,DCSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
942

```

```

943      ;*      MOV      QCSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
944 004162 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
945 004170 001401      BEQ      TST23      ;;BR IF EQUAL
946 004172 104003      ERROR      3      ;ERROR, CLOCK INTERRUPT ENABLE FAILED TO SET
947
948      ;*****
949      ;*TEST 23      TEST MODE (BIT 8) CAN BE SET
950      ;*****
951 004174 000004      TST23: SCOPE
952 004176 012737 000420 001124      MOV      #BIT8!BIT4,$GDDAT      ;LOAD EXPECTED
953
954      ;*      MOV      $GDDAT,QCSR      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
955
956      ;*      MOV      QCSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
957 004224 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
958 004232 001401      BEQ      TST24      ;;BR IF EQUAL
959 004234 104003      ERROR      3      ;ERROR, CSR NOT = 420
960
961      ;*****
962      ;*TEST 24      TEST EXT INTERRUPT ENABLE (BIT 14) CAN BE SET
963      ;*****
964 004236 000004      TST24: SCOPE
965 004240 012737 040020 001124      MOV      #BIT14!BIT4,$GDDAT      ;LOAD EXPECTED
966
967      ;*      MOV      $GDDAT,QCSR      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
968
969      ;*      MOV      QCSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
970 004266 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
971 004274 001401      BEQ      TST25      ;;BR IF EQUAL
972 004276 104003      ERROR      3      ;ERROR, EXT INTERRUPT ENABLE FAILED TO SET
973
974      ;*****
975      ;*TEST 25      TEST THAT CLK DONE (BIT 7) CAN BE SET
976      ;*****
977 004300 000004      TST25: SCOPE
978 004302 012737 000220 001124      MOV      #BIT7!BIT4,$GDDAT      ;LOAD EXPECTED
979
980      ;*      MOV      $GDDAT,QCSR      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
981
982      ;*      MOV      QCSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
983 004330 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
984 004336 001401      BEQ      TST26      ;;BR IF EQUAL
985 004340 104003      ERROR      3      ;CLOCK DONE FAILED TO SET
986
987      ;*****
988      ;*TEST 26      TEST THAT CLK EXT INPUT (BIT 15) CAN BE SET
989      ;*****
989 004342 000004      TST26: SCOPE
990 004344 012737 100020 001124      MOV      #BIT15!BIT4,$GDDAT      ;LOAD EXPECTED
991
992      ;*      MOV      $GDDAT,QCSR      ;/ PUT DATA FROM $GDDAT TO DEVICE REG CSR
993
994      ;*      MOV      QCSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
995 004372 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
996 004400 001401      BEQ      TST27      ;;BR IF EQUAL

```

E03

MAI NDEC-11-DRLPC-A
DRLPC.P11 T26

MACY11 27(654) 15-DEC-77 10:54 PAGE 21
TEST THAT CLK EXT INPUT (BIT 15) CAN BE SET

SEQ 0030

```

997 004402 104003          ERROR 3 ;CLOCK EXT INPUT FAILED TO SET
998 ;*****
999 ;*TEST 27 TEST THAT THE EXT FLAG DOES NOT SET FROM THE OUTSIDE SOURCE
1000 ;*****
1001 004404 000004          TST27: SCOPE
1002 004406 012737 000000 001670 MOV #0,$TMDAT
1003
1004 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1005 004424 012737 000020 001124 MOV #BIT4,$GDDAT ;LOAD EXPECTED
1006 004432 005037 001660 CLR TEMP
1007 004436 1$:
1008
1009 ;* MOV $CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1010 004446 005737 001126 TST $BDDAT ;TEST BIT15
1011 004452 100001 BPL 2$ ;;BR IF CLEARED
1012 004454 104003 ERROR 3 ;ERROR EXT FLG SET IN ERROR
1013 004456 005237 001660 2$: INC TEMP
1014 004462 001365 BNE 1$
1015 ;*****
1016 ;*TEST 30 MAINT. COUNT THE COUNTER REGISTER AT RATE 1MHZ
1017 ;*****
1018 004464 000004          TST30: SCOPE
1019 004466 012737 000004 001166 MOV #4,$TIMES ;;DO 4 ITERATIONS
1020 004474 005037 001670 CLR $TMDAT
1021
1022 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1023
1024 ;* MOV $TMDAT,$CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1025 004520 012737 000002 001670 MOV #2,$TMDAT ;SET UP 1MHZ RATE
1026
1027 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1028 004536 005037 001660 CLR TEMP ;CLEAR PATTERN
1029 004542 013737 001660 1$: MOV TEMP,$GDDAT ;LOAD EXPECTED
1030
1031 ;* MOV $CSC,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1032 004560 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMP
1033 004566 001401 BEQ 2$ ;;BR IF EQUAL
1034 004570 104007 ERROR 7 ;ERROR, CLOCK COUNTER
1035 ; ;BUFFER COUNTED IN ERROR
1036 004572 2$:
1037
1038 ;* MOV $CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1039 004602 105737 001126 TSTB $BDDAT
1040 004606 100005 BPL 3$ ;;BR IF CLEAR
1041 004610 012737 000022 001124 MOV #22,$GDDAT ;LOAD EXPECTED STATUS
1042 004616 104003 ERROR 3 ;CLK DONE SET IN ERROR
1043 004620 000431 BR TST31 ;;BR TO $COPE
1044 004622 3$:
1045
1046 ;* MOV $CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1047 004632 052737 004000 001670 BIS #BIT11,$TMDAT
1048
1049 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1050

```

```

1051 ;* MOV @CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1052 004660 042737 004000 001670 ;* BIC #BIT11,$TMDAT
1053
1054 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1055 004676 105237 00.660 ;* INCB TEMP
1056 004702 001317 ;* BNE 1$ ;BRANCH IF NOT FULL COUNT
1057
1058 ;:*****
1059 ;*TEST 31 TEST THAT OVERFLOW SET CLK DONE (BIT 7)
1060 ;:*****
1061 004704 000004 ;*ST31: SCOPE
1062 004706 012737 000000 001670 ;* MOV #0,$TMDAT ;CLEAR STATUS
1063
1064 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1065 004724 012737 177777 001670 ;* MOV #-1,$TMDAT ;LOAD PRESET
1066
1067 ;* MOV $TMDAT,@CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1068 004742 012737 000002 001670 ;* MOV #BIT1,$TMDAT ;LOAD 1 MHZ. RATE
1069
1070 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1071 004760 012737 000222 001124 ;* MOV #BIT7!BIT4!BIT1,$GDDAT ;LOAD EXPECTED
1072
1073 ;* MOV @CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1074 004776 052737 004000 001670 ;* BIS #BIT11,$TMDAT ;MAINT COUNT
1075
1076 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1077
1078 ;* MOV @CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1079 005024 042737 004000 001670 ;* BIC #BIT11,$TMDAT ;CLEAR BIT
1080
1081 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1082
1083 ;* MOV @CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1084 005052 105737 001126 ;* TSTB $BDDAT ;TEST BIT 7
1085 005056 100401 ;* BMI TST32 ;;BR IF MINUS
1086 005060 104003 ;* ERROR 3 ;ERROR, OVERFLOW FAILED TO SET BIT 7
1087 ;:*****
1088 ;*TEST 32 MAINT. COUNT THE COUNTER REGISTER AT RATE #100KHZ
1089 ;:*****
1090 005062 000004 ;*ST32: SCOPE
1091 005064 012737 000004 001166 ;* MOV #4,$TIMES ;;DO 4 ITERATIONS
1092 005072 012737 000000 001670 ;* MOV #0,$TMDAT ;CLEAR CLOCK STATUS
1093
1094 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1095
1096 ;* MOV $TMDAT,@CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1097 005120 005037 001124 ;* CLR $GDDAT
1098 005124 012737 000004 001670 ;* MOV #4,$TMDAT ;LOAD STATUS, 100KHZ RATE
1099
1100 ;* MOV $TMDAT,@CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1101 005142 005037 001124 ;* CLR $GDDAT
1102 005146 ;*
1103
1104 ;* MOV @CSC,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.

```

G03

MAINDEC-11-DRLPC-A
DRLPC.P11

MACY11 27(654) 15-DEC-77 10:54 PAGE 23
MAINT. COUNT THE COUNTER REGISTER AT RATE #100KHZ

SEQ 0032

```

1105 005156 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1106 005164 001401                      BEQ      2$                ;;BR IF EQUAL
1107 005166 104007                      ERROR    7                ;ERROR, CLOCK COUNTER BUFFER
1108                                     ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1109                                     ;CLOCK UP COUNT OR RATE SELECTION LOGIC
1110 005170 012737 000012 001654 2$:    MOV      #10.,COUNT      ;LOAD COUNT
1111 005176                                     3$:
1112
1113                                     ;*    MOV      @CSR,$TMDAT    ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1114 005206 052737 004000 001670      BIS      @BIT11,$TMDAT
1115
1116                                     ;*    MOV      $TMDAT,@CSR    ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1117
1118                                     ;*    MOV      @CSR,$TMDAT    ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1119 005234 042737 004000 001670      BIC      @BIT11,$TMDAT
1120
1121                                     ;*    MOV      $TMDAT,@CSR    ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1122 005252 005337 001654                      DEC      COUNT            ;DONE
1123 005256 001347                      BNE      3$                ;BR IF NOT
1124 005260 105237 001124                      INCB    $GDDAT            ;INCREMENT EXPECTED VALUE
1125 005264 001330                      BNE      1$
1126
1127                                     ;*****
1128                                     ;*TEST 33      MAINT. COUNT THE COUNTER REGISTER AT RATE #10KHZ
1129                                     ;*****
1130 005266 000004                      *ST33:  SCOPE
1131 005270 012737 000004 001166      MOV      #4,$TIMES        ;;DO 4 ITERATIONS
1132 005276 012737 000000 001670      MOV      #0,$TMDAT        ;CLEAR CLOCK STATUS
1133
1134                                     ;*    MOV      $TMDAT,@CSR    ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1135
1136                                     ;*    MOV      $TMDAT,@CSB    ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1137 005324 012737 000006 001670      MOV      @6,$TMDAT        ;LOAD STATUS. 10KHZ RATE
1138
1139                                     ;*    MOV      $TMDAT,@CSR    ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1140 005342 005037 001124                      CLR      $GDDAT
1141 005346                                     1$:
1142
1143                                     ;*    MOV      @CSC,$BDDAT    ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1144 005356 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE
1145 005364 001401                      BEQ      2$                ;;BR IF EQUAL
1146 005366 104007                      ERROR    7                ;ERROR, CLOCK COUNTER BUFFER
1147                                     ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1148                                     ;CLOCK UP COUNT OR RATE SELECTION LOGIC
1149 005370 012737 000144 001654 2$:    MOV      #100.,COUNT     ;LOAD COUNT
1150 005376                                     3$:
1151
1152                                     ;*    MOV      @CSR,$TMDAT    ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1153 005406 052737 004000 001670      BIS      @BIT11,$TMDAT
1154
1155                                     ;*    MOV      $TMDAT,@CSR    ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1156
1157                                     ;*    MOV      @CSR,$TMDAT    ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1158 005434 042737 004000 001670      BIC      @BIT11,$TMDAT

```

```

1159
1160 ;*      MOV      $TMDAT, @CSR      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1161 005452 005337 001654      DEC      COUNT                    ; DONE
1162 005456 001347      BNE      3$                      ; BR IF NOT
1163 005460 105237 001124      INCB     $GDDAT                  ; UPDATE PATTERN
1164 005464 001330      BNE      1$                      ; BR IF NOT
1165
1166 ;:*****
1167 ;*TEST 34      MAINT. COUNT THE COUNTER REGISTER AT RATE #1KHZ
1168 ;:*****
1168 005466 000004      †ST34:  SCOPE
1169 005470 012737 000004 001166      MOV      #4, $TIMES              ;; DO 4 ITERATIONS
1170 005476 012737 000000 001670      MOV      #0, $TMDAT              ; CLEAR CLOCK STATUS
1171
1172 ;*      MOV      $TMDAT, @CSR      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1173
1174 ;*      MOV      $TMDAT, @CSB      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1175 005524 012737 000001 001124      MOV      #1, $GDDAT              ; LOAD EXPECTED
1176 005532 012737 000010 001670      MOV      #10, $TMDAT             ; LOAD STATUS, 1 KHZ RATE
1177
1178 ;*      MOV      $TMDAT, @CSR      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1179 005550 012737 001750 001654      MOV      #1000., COUNT           ; SET UP A COUNTER
1180 005556
1181
1182 ;*      MOV      @CSR, $TMDAT      ;/ READ DEVICE REG CSR, PUT DATA IN $TMDAT.
1183 005566 052737 004000 001670      BIS      #BIT11, $TMDAT
1184
1185 ;*      MOV      $TMDAT, @CSR      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1186
1187 ;*      MOV      @CSR, $TMDAT      ;/ READ DEVICE REG CSR, PUT DATA IN $TMDAT.
1188 005614 042737 004000 001670      BIC      #BIT11, $TMDAT
1189
1190 ;*      MOV      $TMDAT, @CSR      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1191 005632 005337 001654      DEC      COUNT                    ; BR
1192 005636 001347
1193
1194 ;*      MOV      @CSC, $BDDAT      ;/ READ DEVICE REG CSC, PUT DATA IN $BDDAT.
1195 005650 023737 001124 001126      CMP      $GDDAT, $BDDAT          ; COMPARE
1196 005656 001401      BEQ     TST35                    ;; BR IF EQUAL
1197 005660 104007      ERROR  7                          ; ERROR, CLOCK COUNTER BUFFER
1198 ;:COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1199 ;:CLOCK UP COUNT OR RATE SELECTION LOGIC
1200
1201 ;:*****
1202 ;*TEST 35      MAINT. COUNT THE COUNTER REGISTER AT RATE #100HZ
1203 ;:*****
1204 005662 000004      †ST35:  SCOPE
1205 005664 012737 000004 001166      MOV      #4, $TIMES              ;; DO 4 ITERATIONS
1206 005672 012737 000000 001670      MOV      #0, $TMDAT              ; CLEAR CLOCK STATUS, 100HZ RATE
1207
1208 ;*      MOV      $TMDAT, @CSR      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1209
1210 ;*      MOV      $TMDAT, @CSB      ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1211 005720 012737 000001 001124      MOV      #1, $GDDAT              ; LOAD EXPECTED
1212 005726 012737 000012 001670      MOV      #12, $TMDAT             ; LOAD STATUS, 100 HZ

```

```

1213
1214
1215 005744 012737 023420 001654 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1216 005752 ;* MOV #10000.,COUNT ;SET UP A COUNTER
1217
1218
1219 005762 052737 004000 001670 ;* MOV $CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1220 ;* BIS #BIT11,$TMDAT
1221
1222
1223
1224 006010 042737 004000 001670 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1225 ;* MOV $CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1226 ;* BIC #BIT11,$TMDAT
1227 006026 005337 001654 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1228 006032 001347 ;* DEC COUNT ;
1229 ;* BNE 15 ;BR
1230
1231 006044 023737 001124 001126 ;* MOV $CSC,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1232 006052 001401 ;* CMP $GDDAT,$BDDAT ;COMPARE
1233 006054 104007 ;* BEQ TST36 ;.BR IF EQUAL
1234 ;* ERROR 7 ;ERROR, CLOCK COUNTER BUFFER
1235 ;* ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1236 ;* ;CLOCK UP COUNT OR RATE SELECTION LOGIC
1237
1238 ;* *****
1239 ;* *TEST 36 TEST THAT RESET CLEARS RATE SELECT AND MODE BITS
1240 ;* *****
1241 006056 000004 †TST36: SCOPE
1242 006060 012737 000002 001166 MOV #2,$TIMES ;DO 2 ITERATIONS
1243 006066 012737 000416 001670 MOV #BIT8!BIT3!BIT2!BIT1,$TMDAT ;SET MODE BITS
1244
1245 006104 012737 000020 001124 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1246 006112 004737 023174 ;* MOV #20,$GDDAT
1247 ;* JSR PC,$RESET
1248
1249 006126 023737 001124 001126 ;* MOV $CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1250 006134 001401 ;* CMP $GDDAT,$BDDAT ;COMPARE
1251 006136 104003 ;* BEQ TST37 ;.BR IF EQUAL
1252 ;* ERROR 3 ;ERROR, RESET FAILED TO CLEAR RATE OR MODE BITS
1253
1254 ;* *****
1255 ;* *TEST 37 TEST THAT RESET CLEARS CLK INTERRUPT ENABLE
1256 ;* *****
1257 006140 000004 †TST37: SCOPE
1258 006142 012737 000002 001166 MOV #2,$TIMES ;DO 2 ITERATIONS
1259 006150 012737 040100 001670 MOV #BIT6!BIT14,$TMDAT ;SET CLK INT ENABLE
1260
1261 006166 012737 000020 001124 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1262 006174 004737 023174 ;* MOV #20,$GDDAT
1263 ;* JSR PC,$RESET
1264
1265 006210 023737 001124 001126 ;* MOV $CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1266 006216 001401 ;* CMP $GDDAT,$BDDAT ;COMPARE
1267 ;* BEQ TST40 ;.BR IF EQUAL

```

```

1267 006220 104003          ERROR 3          ;ERROR, RESET FAILED TO CLEAR CLK INT ENABLE
1268
1269
1270 ;*****
1271 ;*TEST 40          TEST THAT RESET CLEARS CLK FLAGS
1272 ;*****
1273 006222 000004          †ST40: SCOPE
1274 006224 012737 000002 001166          MOV #2,$TIMES          ;;DO 2 ITERATIONS
1275 006232 012737 100200 001670          MOV #BIT7!BIT15,$TMDAT ;SET CLK FLAGS
1276
1277 ;*          MOV $TMDAT,$CSR          ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1278 006250 012737 000020 001124          MOV #20,$GDDAT
1279 006256 004737 023174          JSR PC,$RESET
1280
1281 ;*          MOV $CSR,$BDDAT          ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1282 006272 023737 001124 001126          CMP $GDDAT,$BDDAT
1283 006300 001401          BEQ TST41          ;;BR IF EQUAL
1284 006302 104003          ERROR 3          ;ERROR, RESET FAILED TO CLEAR CLK FLAGS
1285
1286 ;*****
1287 ;*TEST 41          TEST THAT RESET CLEARS COUNTER ENABLE
1288 ;*****
1289 006304 000004          †ST41: SCOPE
1290 006306 012737 000002 001166          MOV #2,$TIMES          ;;DO 2 ITERATIONS
1291 006314 012737 000000 001670          MOV #0,$TMDAT          ;CLEAR COUNTER
1292
1293 ;*          MOV $TMDAT,$CSB          ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1294 006332 012737 000001 001670          MOV #BIT0,$TMDAT          ;LOAD COUNTER ENABLE
1295
1296 ;*          MOV $TMDAT,$CSR          ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1297 006350 012737 000020 001126          MOV #20,$BDDAT
1298 006356 004737 023174          JSR PC,$RESET
1299
1300 ;*          MOV $CSR,$BDDAT          ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1301 006372 023737 001124 001126          CMP $GDDAT,$BDDAT
1302 006400 001401          BEQ TST42          ;;BR IF EQUAL
1303 006402 104003          ERROR 3          ;ERROR, RESET FAILED TO CLEAR COUNTER ENABLE
1304
1305 ;*****
1306 ;*TEST 42          TEST CLOCK TO COUNT UP AT 1 MHZ
1307 ;*****
1308 006404 000004          †ST42: SCOPE
1309 006406 012737 000020 001166          MOV #20,$TIMES          ;;DO 20 ITERATIONS
1310 006414 012737 000003 014610          MOV #3,RATE          ;SELECT MODE 0, 1MHZ.. GO
1311 006422 012737 000222 001124          MOV #BIT7!BIT4!BIT1,$GDDAT
1312 006430 004737 014002          JSR PC,UPCNT
1313
1314 ;*          MOV $CSR,$BDDAT          ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1315 006444 023737 001124 001126          CMP $GDDAT,$BDDAT          ;COMPARE
1316 006452 001401          BEQ 1$          ;;BR IF EQUAL
1317 006454 104003          ERROR 3          ;ERROR, 1MHZ RATE FAILED TO SET DONE
1318 006456 012737 000000 001124 1$: MOV #0,$GDDAT
1319
1320 ;*          MOV $CSC,$BDDAT          ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.

```



```

1321 006474 023737 001124 001126      CMP      $GDDAT,$BDDAT
1322 006502 001401      BEQ      TST43          ;;BR IF EQUAL
1323 006504 104006      ERROR   6              ;IN MODE 0 CLOCK COUNTER WAS
                               ;LOADED ON CLK OVERFLOW
1324
1325
1326 ;:*****
1327 ;*TEST 43      TEST CLOCK TO COUNT UP AT 100KHZ
1328 ;:*****
1329 006506 000004      †ST43:  SCOPE
1330 006510 012737 000020 001166      MOV      #20,$TIMES      ;;DO 20 ITERATIONS
1331 006516 012737 000005 014610      MOV      #5,RATE        ;SELECT MODE 0, 100KHZ., GO
1332 006524 012737 000224 001124      MOV      #BIT7!BIT4!BIT2,$GDDAT
1333 006532 004737 014002      JSR      PC,UPCNT
1334
1335 ;*      MOV      @CSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1336 006546 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1337 006554 001401      BEQ      1$                ;;BR IF EQUAL
1338 006556 104003      ERROR   3                ;ERROR, 100KHZ. RATE FAILED TO SET DONE
1339 006560 012737 000000 001124 1$:      MOV      #0,$GDDAT
1340
1341 ;*      MOV      @CSC,$BDDAT      ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1342 006576 023737 001124 001126      CMP      $GDDAT,$BDDAT
1343 006604 001401      BEQ      TST44          ;;BR IF EQUAL
1344 006606 104006      ERROR   6              ;IN MODE 0 CLOCK COUNTER WAS
                               ;LOADED ON CLK OVERFLOW
1345
1346 ;:*****
1347 ;*TEST 44      TEST CLOCK TO COUNT UP AT 10 KHZ
1348 ;:*****
1349 006610 000004      †ST44:  SCOPE
1350 006612 012737 000020 001166      MOV      #20,$TIMES      ;;DO 20 ITERATIONS
1351 006620 012737 000007 014610      MOV      #7,RATE        ;SELECT MODE 0, 10KHZ., GO
1352 006626 012737 000226 001124      MOV      #BIT7!BIT4!BIT2!BIT1,$GDDAT
1353 006634 004737 014002      JSR      PC,UPCNT
1354
1355 ;*      MOV      @CSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1356 006650 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1357 006656 001401      BEQ      TST45          ;;BR IF EQUAL
1358 006660 104007      ERROR   7                ;ERROR, 10 KHZ. FAILED TO SET DONE
1359
1360 ;:*****
1361 ;*TEST 45      TEST CLOCK TO COUNT UP AT 1KHZ
1362 ;:*****
1363 006662 000004      †ST45:  SCOPE
1364 006664 012737 000005 001166      MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1365 006672 012737 000011 014610      MOV      #11,RATE       ;SELECT MODE 0, 1 KHZ, GO
1366 006700 012737 000230 001124      MOV      #BIT7!BIT4!BIT3,$GDDAT
1367 006706 004737 014002      JSR      PC,UPCNT
1368
1369 ;*      MOV      @CSR,$BDDAT      ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1370 006722 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1371 006730 001401      BEQ      TST46          ;;BR IF EQUAL
1372 006732 104007      ERROR   7                ;ERROR, 1KHZ FAILED TO SET DONE

```

```

1375
1376
1377
1378
1379 006734 000004
1380 006736 012737 000005 001166
1381 006744 012737 000013 014610
1382 006752 012737 000232 001124
1383 006760 004737 014002
1384
1385
1386 006774 023737 001124 001126
1387 007002 001401
1388 007004 104007
1389
1390
1391
1392
1393 007006 000004
1394 007010 012737 000005 001166
1395 007016 012737 000411 014610
1396 007024 012737 000631 001124
1397 007032 004737 014002
1398
1399
1400 007046 023737 001124 001126
1401 007054 001401
1402 007056 104007
1403
1404 007060 012737 000376 001124
1405
1406
1407 007076 023737 001124 001126
1408 007104 001401
1409 007106 104006
1410
1411
1412
1413
1414
1415
1416 007110 000004
1417 007112 012737 000020 001166
1418 007120 012737 000413 014610
1419 007126 012737 000633 001124
1420 007134 004737 014002
1421
1422
1423 007150 023737 001124 001126
1424 007156 001401
1425 007160 104007
1426
1427 007162 012737 000376 001124
1428

```

```

*****
; *TEST 46 TEST CLOCK TO COUNT UP AT 100HZ
*****
†ST46: SCOPE
MOV #5,STIMES ;;DO 5 ITERATIONS
MOV #13,RATE ;SELECT MODE 0, 100 HZ, GO
MOV #BIT7!BIT4!BIT3!BIT1,$GDDAT ;LOAD EXPECTED
JSR PC,UPCNT
; * MOV @CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST47 ;;BR IF EQUAL
ERROR 7 ;ERROR, 100HZ FAILED TO SET DONE
*****
; *TEST 47 TEST THAT CLOCK ENABLE DOES NOT CLEAR ON DONE (MODE 1) 1 KHZ
*****
†ST47: SCOPE
MOV #5,STIMES ;;DO 5 ITERATIONS
MOV #411,RATE ;SELECT MODE 1, 1KHZ., GO
MOV #631,$GDDAT ;LOAD EXP
JSR PC,UPCNT
; * MOV @CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF EQUAL
ERROR 7 ;ERROR, 1 KHZ. FAILED TO OVERFLOW
; AND CONTINUE COUNTING
1$: MOV #376,$GDDAT
; * MOV @CSC,$BDDAT ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST50 ;;BR IF EQUAL
ERROR 6 ;IN MODE 1, CLOCK COUNTER FAILED
; TO BE RE-LOADED ON CLOCK OVERFLOW
*****
; *TEST 50 TEST THAT CLOCK ENABLE DOES NOT CLEAR DONE (MODE 1 100HZ)
*****
†ST50: SCOPE
MOV #20,STIMES ;;DO 20 ITERATIONS
MOV #413,RATE ;MODE 1 100HZ GO
MOV #633,$GDDAT ;LOAD EXPECTED
JSR PC,UPCNT
; * MOV @CSR,$BDDAT ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF EQUAL
ERROR 7 ;ERROR, 100HZ FAILED TO
; OVERFLOW AND CONTINUE COUNTING
1$: MOV #376,$GDDAT

```

M03

MAINDEC-11-DRLPC-A
DRLPC.P11 T50

MACY11 27(654) 15-DEC-77 10:54 PAGE 29
TEST THAT CLOCK ENABLE DOES NOT CLEAR DONE (MODE 1 100HZ)

SEQ 0038

```

1429
1430 007200 023737 001124 001126 ;*      MOV      @CSC,$BDDAT      ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1431 007206 001401          CMP      $GDDAT,$BDDAT      ;COMPARE
1432 007210 104006          BEQ      TST51              ;;BR IF EQUAL
1433                                     ;IN MODE 1, CLOCK COUNTER FAILED
1434                                     ;TO BE RE-LOADED ON CLOCK OVERFLOW
1435
1436 ;*****
1437 ;*TEST 51      TEST THAT RESET SETS VC READY BIT
1438 ;*****
1439 007212 000004          †TST51: SCOPE
1440 007214 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
1441 007222 004737 023174          JSR      PC,$RESET
1442
1443 ;*      MOV      @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1444 007236 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1445 007244 001401          BEQ      TST52              ;;BR IF SET
1446 007246 104011          ERROR   11                ;RESET FAILED TO SET READY
1447
1448 ;*****
1449 ;*TEST 52      TEST THAT VC MODE BIT 2 CAN BE SET AND CLEARED
1450 ;*****
1451 007250 000004          †TST52: SCOPE
1452 007252 012737 000004 001670      MOV      #BIT2,$TMDAT      ;LOAD DISPLAY STATUS
1453
1454 ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1455 007270 012737 000204 001124      MOV      #BIT7!BIT2,$GDDAT ;LOAD EXPECTED
1456
1457 ;*      MOV      @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1458 007306 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1459 007314 001401          BEQ      15                ;;BR IF EQUAL
1460 007316 104011          ERROR   11                ;ERROR, VC STATUS NOT = 204
1461 007320 012737 000000 001670 15:  MOV      #0,$TMDAT          ;CLEAR STATUS
1462
1463 ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1464 007336 012737 000200 001124      MOV      #BIT7,$GDDAT      ;LOAD EXPECTED
1465
1466 ;*      MOV      @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1467 007354 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1468 007362 001401          BEQ      TST53              ;;BR IF CLEARED
1469 007364 104011          ERROR   11                ;MODE FAILED TO CLEAR
1470
1471 ;*****
1472 ;*TEST 53      TEST THAT VC MODE BIT 3 CAN BE SET
1473 ;*****
1474 007366 000004          †TST53: SCOPE
1475 007370 012737 000010 001670      MOV      #BIT3,$TMDAT      ;LOAD
1476
1477 ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1478 007406 012737 000210 001124      MOV      #BIT7!BIT3,$GDDAT ;LOAD EXPECTED
1479
1480 ;*      MOV      @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1481 007424 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
1482 007432 001401          BEQ      TST54              ;;BR IF EQUAL
1483 007434 104011          ERROR   11                ;ERROR, VC STATUS NOT = 210

```

N03

MAINDEC-11-DRLPC-A
DRLPC.P11 T53

MACY11 27(654) 15-DEC-77 10:54 PAGE 30
TEST THAT VC MODE BIT 3 CAN BE SET

SEQ 0039

```

1483
1484
1485 ;:*****
1486 ;*TEST 54 TEST THAT VC INTERRUPT ENABLE (BIT 6) CAN BE SET
1487 ;:*****
1487 007436 000004 †ST54: SCOPE
1488 007440 012737 000100 001670 MOV #BIT6,$TMDAT ;LOAD DISPLAY STATUS
1489
1490 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1491 007456 012737 000300 001124 MOV #BIT7!BIT6,$GDDAT ;LOAD EXPECTED
1492
1493 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1494 007474 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1495 007502 001401 BEQ TST55 ;;BR IF EQUAL
1496 007504 104011 ERROR 11 ;ERROR, VC STATUS NOT = 300
1497
1498 ;:*****
1499 ;*TEST 55 TEST THAT CHANNEL (BIT 9) CAN BE SET
1500 ;:*****
1501 007506 000004 †ST55: SCOPE
1502 007510 012737 001000 001670 MOV #BIT9,$TMDAT ;LOAD DISPLAY STATUS
1503
1504 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1505 007526 012737 001200 001124 MOV #BIT9!BIT7,$GDDAT ;LOAD EXPECTED
1506
1507 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1508 007544 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1509 007552 001401 BEQ TST56 ;;BR IF EQUAL
1510 007554 104011 ERROR 11 ;ERROR, VC STATUS NOT = 1200
1511
1512 ;:*****
1513 ;*TEST 56 TEST THAT STORE (BIT 10) CAN BE SET
1514 ;:*****
1515 007556 000004 †ST56: SCOPE
1516 007560 012737 002000 001670 MOV #BIT10,$TMDAT ;LOAD DISPLAY STATUS
1517
1518 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1519 007576 012737 002200 001124 MOV #BIT10!BIT7,$GDDAT ;LOAD EXPECTED
1520
1521 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1522 007614 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1523 007622 001401 BEQ TST57 ;;BR IF EQUAL
1524 007624 104011 ERROR 11 ;ERROR, VC STATUS NOT = 2200
1525
1526 ;:*****
1527 ;*TEST 57 TEST THAT WRITE THRU (BIT 11) CAN BE SET
1528 ;:*****
1528 007626 000004 †ST57: SCOPE
1529 007630 012737 004000 001670 MOV #BIT11,$TMDAT ;LOAD DISPLAY STATUS
1530
1531 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1532 007646 012737 004200 001124 MOV #BIT11!BIT7,$GDDAT ;LOAD EXPECTED
1533
1534 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1535 007664 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1536 007672 001401 BEQ TST60 ;;BR IF EQUAL

```

```

1537 007674 104011          ERROR 11 ;ERROR, VC STATUS NOT = 4200
1538 ;*****
1539 ;*TEST 60 TEST THAT ERASE (BIT 12) CAN BE SET
1540 ;*****
1541 007676 000004          †ST60: SCOPE
1542 007700 012737 010000 001670      MOV #BIT12,$TMDAT ;SET ERASE BIT
1543
1544 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1545 007716 012737 010000 001124      MOV #BIT12,$GDDAT ;LOAD EXPECTED
1546
1547 ;* MOV @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1548 007734 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
1549 007742 001401          BEQ 15 ;;BR IF SET
1550 007744 104011          ERROR 11 ;ERROR, ERASE BIT FAILED TO SET
1551 007746 005037 001670      15: CLR $TMDAT
1552
1553 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1554 ;*****
1555 ;*TEST 61 TEST THAT THE X REGISTER CAN BE CLEARED
1556 ;*****
1557 007762 000004          †ST61: SCOPE
1558 007764 012737 000000 001124      MOV #0,$GDDAT ;LOAD EXPECTED
1559
1560 ;* MOV $GDDAT,@VCXREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1561
1562 ;* MOV @VCXREG,$BDDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1563 010012 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
1564 010020 001401          BEQ TST62 ;;BR IF EQUAL
1565 010022 104012          ERROR 12 ;ERROR, VC XREGISTER NOT = 0
1566
1567 ;*****
1568 ;*TEST 62 TEST THAT THE X REGISTER CAN BE LOADED WITH #1777
1569 ;*****
1570 010024 0007J4          †ST62: SCOPE
1571 010026 012737 001777 001124      MOV #1777,$GDDAT ;LOAD EXPECTED
1572
1573 ;* MOV $GDDAT,@VCXREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1574
1575 ;* MOV @VCXREG,$BDDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1576 010054 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
1577 010062 001401          BEQ TST63 ;;BR IF EQUAL
1578 010064 104012          ERROR 12 ;ERROR, VC X REGISTER NOT = 1777
1579
1580 ;*****
1581 ;*TEST 63 TEST THAT THE X REGISTER CAN BE LOADED WITH #525
1582 ;*****
1583 010066 000004          †ST63: SCOPE
1584 010070 012737 000525 001124      MOV #525,$GDDAT ;LOAD EXPECTED
1585
1586 ;* MOV $GDDAT,@VCXREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1587
1588 ;* MOV @VCXREG,$BDDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1589 010116 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE
1590 010124 001401          BEQ TST64 ;;BR IF EQUAL

```

```

1591 010126 104012          ERROR 12          ;ERROR, VC X REGISTER NOT = 525
1592
1593
1594          ;*****
1595          ;*TEST 64          TEST THAT THE X REGISTER CAN BE LOADED WITH #1252
1596          ;*****
1596 010130 000004          †ST64: SCOPE
1597 010132 012737 001252 001124          MOV #1252,$GDDAT          ;LOAD EXPECTED
1598
1599          ;* MOV $GDDAT,@VCXREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1600
1601          ;* MOV @VCXREG,$BDDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1602 010160 023737 001124 001126          CMP $GDDAT,$BDDAT          ;COMPARE
1603 010166 001401          BEQ TST65          ;;BR IF EQUAL
1604 010170 104012          ERROR 12          ;ERROR, VC X REGISTER NOT = 1252
1605
1606          ;*****
1607          ;*TEST 65          TEST THAT THE X REGISTER CAN HOLD A COUNT PATTERN
1608          ;*****
1609 010172 000004          †ST65: SCOPE
1610 010174 012737 010206 001110          MOV #15,$LPERR          ;LOAD SCOPE ERROR RETURN
1611 010202 005037 001124          CLR $GDDAT          ;CLEAR EXPECTED
1612
1613          15:
1614          ;* MOV $GDDAT,@VCXREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1615
1616          ;* MOV @VCXREG,$BDDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1617 010226 023737 001124 001126          CMP $GDDAT,$BDDAT          ;COMPARE
1618 010234 001401          BEQ 25          ;;BR IF EQUAL
1619 010236 104012          ERROR 12          ;VC X REG FAILED TO HOLD A COUNT PATTERN
1620
1621 010240 005237 001124          25: INC $GDDAT          ;UPDATE PATTERN
1622 010244 022737 002000 001124          CMP #2000,$GDDAT          ;FINISHED?
1623 010252 001355          BNE 15          ;BR IF NOT
1624
1625          ;*****
1626          ;*TEST 66          TEST THAT THE Y REGISTER CAN BE CLEARED
1627          ;*****
1628 010254 000004          †ST66: SCOPE
1629 010256 012737 000000 001124          MOV #0,$GDDAT          ;LOAD EXPECTED
1630
1631          ;* MOV $GDDAT,@VCYREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1632
1633          ;* MOV @VCYREG,$BDDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1634 010304 023737 001124 001126          CMP $GDDAT,$BDDAT          ;COMPARE
1635 010312 001401          BEQ TST67          ;;BR IF EQUAL
1636 010314 104013          ERROR 13          ;ERROR, VC Y REGISTER NOT = 0
1637
1638          ;*****
1639          ;*TEST 67          TEST THAT THE Y REGISTER CAN BE LOADED WITH #1777
1640          ;*****
1641 010316 000004          †S 7: SCOPE
1642 010320 012737 001777 001124          MOV #1777,$GDDAT          ;LOAD EXPECTED
1643
1644          ;* MOV $GDDAT,@VCYREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG

```

```

1645
1646
1647 010346 023737 001124 001126 ;*   MOV   @VCYREG,$BDDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1648 010354 001401           CMP   $GDDAT,$BDDAT ;COMPARE
1649 010356 104013           BEQ   TST70           ;;BR IF EQUAL
1650                                     ERROR 13           ;ERROR, VC Y REGISTER NOT = 1777
1651
1652 ;:*****
1653 ;*TEST 70 TEST THAT THE Y REGISTER CAN BE LOADED WITH #525
1654 ;:*****
1655 010360 000004 TST70: SCOPE
1656 010362 012737 000525 001124 MOV   #525,$GDDAT ;LOAD EXPECTED
1657
1658 ;*   MOV   $GDDAT,@VCYREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1659
1660 ;*   MOV   @VCYREG,$BDDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1661 010410 023737 001124 001126 CMP   $GDDAT,$BDDAT ;COMPARE
1662 010416 001401           BEQ   TST71           ;;BR IF EQUAL
1663 010420 104013           ERROR 13           ;ERROR, VC Y REGISTER NOT = 525
1664
1665 ;:*****
1666 ;*TEST 71 TEST THAT THE Y REGISTER CAN BE LOADED WITH #1252
1667 ;:*****
1668 010422 000004 TST71: SCOPE
1669 010424 012737 001252 001124 MOV   #1252,$GDDAT ;LOAD EXPECTED
1670
1671 ;*   MOV   $GDDAT,@VCYREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1672
1673 ;*   MOV   @VCYREG,$BDDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1674 010452 023737 001124 001126 CMP   $GDDAT,$BDDAT ;COMPARE
1675 010460 001401           BEQ   TST72           ;;BR IF EQUAL
1676 010462 104013           ERROR 13           ;ERROR, VC Y REGISTER NOT = 1252
1677
1678 ;:*****
1679 ;*TEST 72 TEST THAT THE Y REGISTER CAN HOLD A COUNT PATTERN
1680 ;:*****
1681 010464 000004 TST72: SCOPE
1682 010466 012737 010500 001110 MOV   #15,$LPERR ;LOAD SCOPE ERROR RETURN
1683 010474 005037 001124           CLR   $GDDAT ;CLEAR EXPECTED
1684
1685 ;*   MOV   $GDDAT,@VCYREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1686
1687 ;*   MOV   @VCYREG,$BDDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1688 010520 023737 001124 001126 CMP   $GDDAT,$BDDAT ;COMPARE
1689 010526 001401           BEQ   25           ;;BR IF EQUAL
1690 010530 104013           ERROR 13           ;VC Y REG FAILED TO HOLD A COUNT PATTERN
1691
1692 25:   INC   $GDDAT ;UPDATE PATTERN
1693 010532 005237 001124           CMP   #2000,$GDDAT ;FINISHED?
1694 010536 022737 002000 001124 BNE   15           ;BR IF NOT
1695 010544 001355
1696
1697 ;:*****
1698 ;*TEST 73 TEST THAT THE X-Y REGISTERS CAN HOLD DIFFERENT DATA
1699 ;:*****

```

E04

MAINDEC-11-DRLPC-A
DRLPC.P11 T73

MACY11 27(654) 15-DEC-77 10:54 PAGE 34
TEST THAT THE X-Y REGISTERS CAN HOLD DIFFERENT DATA

SEQ 0043

```

1699 010546 000004 TST73: SCOPE
1700 010550 012737 001252 001670 MOV #1252,$TMDAT ;LOAD X REGISTER
1701
1702 ;* MOV $TMDAT,$VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1703 010566 012737 000525 001124 MOV #525,$GDDAT ;LOAD EXPECTED
1704
1705 ;* MOV $GDDAT,$VCYREG ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1706
1707 ;* MOV $VCYREG,$BDDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1708 010614 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1709 010622 001401 BEQ 15 ;;BR IF EQUAL
1710 010624 104013 ERROR 13 ;ERROR, SELECTED Y REGISTER INCORRECTLY
1711
1712 010626 012737 001252 001124 15: MOV #1252,$GDDAT ;LOAD EXPECTED
1713
1714 ;* MOV $VCXREG,$BDDAT ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1715 010644 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1716 010652 001401 BEQ TST74 ;;BR IF EQUAL
1717 010654 104012 ERROR 12 ;ERROR, SELECTED X REGISTER INCORRECTLY
1718
1719 ;*****
1720 ;*TEST 74 TEST THAT WHEN INTENSIFY BIT IS SET THAT THE VC READY BIT RESETS
1721 ;*****
1722 010656 000004 †ST74: SCOPE
1723 ; AND THEN SETS AFTER A DELAY
1724 010660 012700 001000 MOV #1000,R0
1725 010664 012737 000000 001124 MOV #0,$GDDAT ;LOAD EXPECTED
1726 010672 012737 000001 001670 MOV #BIT0,$TMDAT ;INTENSIFY
1727
1728 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1729 010710 15:
1730
1731 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1732 010720 105737 001126 TSTB $BDDAT
1733 010724 100412 BMI TST75 ;;NEXT TEST
1734 010726 005300 DEC R0 ;DELAY
1735 010730 001367 BNE 15 ;LOAD EXPECTED
1736 010732 012737 000200 001124 MOV #BIT7,$GDDAT
1737
1738 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1739 010750 104011 ERROR 11 ;READY FAILED TO SET AFTER A DELAY
1740
1741 ;*****
1742 ;*TEST 75 TEST THAT VC MODE 1 (INTENSIFY ON X) RESETS THE READY FLAG
1743 ;*****
1744 010752 000004 †ST75: SCOPE
1745 ; AND THEN SETS IT
1746 010754 012700 001000 MOV #1000,R0 ;SET UP DELAY
1747 010760 012737 000204 001124 MOV #BIT7!BIT2,$GDDAT ;LOAD EXPECTED
1748 010766 012737 000004 001670 MOV #BIT2,$TMDAT ;LOAD MODE 1
1749
1750 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1751
1752 011004 012737 000000 001670 25: MOV #0,$TMDAT ;ADDRESS X AXIS

```



```

1753
1754 011012          1$:
1755
1756
1757 011022 105737 001126      :*      MOV      @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1758 011026 100416          TSTB     $BDDAT
1759 011030 005300          BMI      TST76          ;;NEXT TEST
1760 011032 001367          DEC      RD              ;DELAY
1761 011034 012737 000204 001124  BNE     1$              ;TEST READY AGAIN
1762
1763
1764 011052 023737 001124 001126  :*      MOV      @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1765 011060 001401          CMP      $GDDAT,$BDDAT
1766 011062 104011          BEQ     TST76          ;;BR IF EQUAL
1767
1768
1769
1770
1771
1772 011064 000004          ;*****
1773
1774 011066 012700 001000          ;*      MOV      @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1775 011072 012737 000210 001124  TSTB     $BDDAT
1776 011100 012737 000010 001670  BMI      TST76          ;;NEXT TEST
1777
1778
1779 011116 012737 000000 001670  2$:      MOV      $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1780
1781
1782 011134          1$:      MOV      $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1783
1784
1785 011144 105737 001126      :*      MOV      @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1786 011150 100416          TSTB     $BDDAT
1787 011152 005300          BMI      TST77          ;;NEXT TEST
1788 011154 001367          DEC      RD              ;DELAY
1789 011156 012737 000210 001124  BNE     1$              ;TEST READY AGAIN
1790
1791
1792 011174 023737 001124 001126  :*      MOV      @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1793 011202 001401          CMP      $GDDAT,$BDDAT
1794 011204 104011          BEQ     TST77          ;;BR IF EQUAL
1795
1796
1797
1798
1799
1800 011206 000004          ;*****
1801 011210 032777 010000 167722  ;*      MOV      @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1802 011216 001443          TSTB     $BDDAT
1803 011220 012700 000010          BMI      TST77          ;;NEXT TEST
1804 011224 005037 001660          DEC      RD              ;DELAY
1805 011230 012737 002000 001670  BNE     1$              ;TEST READY AGAIN
1806

```

```

1807 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1808
1809 ;* MOV $VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
1810 011256 052737 010000 001670 BIS #BIT12,$TMDAT
1811
1812 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1813
1814 011274 1$:
1815
1816 ;* MOV $VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
1817 011304 105737 001670 TSTB $TMDAT
1818 011310 100406 BMI TST100 ;;BR IF SET
1819 011312 005337 001660 DEC TEMP ;DELAY
1820 011316 001366 BNE 1$ ;;BR IF NOT READY
1821 011320 005300 DEC RO ;DECREMENT COUNTER
1822 011322 001364 BNE 1$ ;;BR IF NOT DONE
1823 011324 104011 ERROR 11 ;ERROR, ERASE CLEARED READY AND FAILED
1824 ;TO SET READY AFTER A DELAY
1825 ;:*****
1826 ;*TEST 100 TEST THAT RESET CLEARS VC MODE BITS
1827 ;:*****
1828 011326 000004 †ST100: SCOPE
1829 011330 012737 000002 001166 MOV #2,$TIMES ;;DO 2 ITERATIONS
1830 011336 012737 000014 001670 MOV #BIT3!BIT2,$TMDAT
1831
1832 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1833 011354 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD EXPECTED
1834 011362 004737 023174 JSR PC,$RESET
1835
1836 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1837 011376 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1838 011404 001401 BEQ TST101 ;;BR IF EQUAL
1839 011406 104011 ERROR 11 ;ERROR, RESET FAILED TO CLEAR VC STATUS REG
1840
1841 ;:*****
1842 ;*TEST 101 TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU
1843 ;:*****
1844 011410 000004 †ST101: SCOPE
1845 011412 012737 000002 001166 MOV #2,$TIMES ;;DO 2 ITERATIONS
1846 011420 012737 007100 001670 MOV #BIT11!BIT10!BIT9!BIT6,$TMDAT
1847
1848 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1849 011436 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD EXPECTED
1850 011444 004737 023174 JSR PC,$RESET
1851
1852 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1853 011460 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1854 011466 001401 BEQ TST102 ;;BR IF EQUAL
1855 011470 104011 ERROR 11 ;ERROR, RESET FAILED TO CLEAR VC STATUS
1856
1857 ;:*****
1858 ;*TEST 102 TEST THAT RESET CLEARS X REGISTER
1859 ;:*****
1860 011472 000004 †ST102: SCOPE

```

```

1861 011474 012737 000002 001166      MOV      #2,$TIMES      ;;DO 2 ITERATIONS
1862 011502 012737 177777 001670      MOV      #-1,$TMDAT
1863
1864      ;*      MOV      $TMDAT,$VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1865 011520 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED
1866 011526 004737 023174      JSR      PC,$RESET
1867
1868      ;*      MOV      $VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1869 011542 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE
1870 011550 001401      BEQ      TST103          ;;BR IF EQUAL
1871 011552 104011      ERROR    1              ;ERROR, RESET FAILED TO CLEAR VC X REGISTER
1872
1873      ;*****
1874      ;*TEST 103      TEST THAT RESET CLEARS Y REGISTER
1875      ;*****
1876 011554 000004      †TST103: SCOPE
1877 011556 012737 000002 001166      MOV      #2,$TIMES      ;;DO 2 ITERATIONS
1878 011564 012737 177777 001670      MOV      #-1,$TMDAT
1879
1880      ;*      MOV      $TMDAT,$VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1881 011602 012737 000000 001124      MOV      #0,$GDDAT      ;LOAD EXPECTED
1882 011610 004737 023174      JSR      PC,$RESET
1883
1884      ;*      MOV      $VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1885 011624 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE
1886 011632 001401      BEQ      TST104          ;;BR IF EQUAL
1887 011634 104011      ERROR    1              ;ERROR, RESET FAILED TO CLEAR VC Y REGISTER
1888
1889      ;*****
1890      ;*TEST 104      DOES EXTERNAL ENABLE (BIT 4) SET
1891      ;*****
1892 011636 000004      †TST104: SCOPE
1893 011640 012737 000020 001124      MOV      #BIT4,$GDDAT      ;LOAD EXPECTED
1894
1895      ;*      MOV      $GDDAT,$ADCS    ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1896
1897      ;*      MOV      $ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1898 011666 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE
1899 011674 001401      BEQ      TST105          ;;BR IF SET
1900 011676 104001      ERROR    1              ;EXT ENABLE BIT 4 FAILED TO SET
1901
1902      ;*****
1903      ;*TEST 105      DOES CLOCK OVERFLOW ENABLE (BIT 5) SET
1904      ;*****
1905 011700 000004      †TST105: SCOPE
1906      ;SET BIT 5
1907 011702 012737 000040 001124      MOV      #BITS,$GDDAT      ;LOAD EXPECTED
1908
1909      ;*      MOV      $GDDAT,$ADCS    ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1910
1911      ;*      MOV      $ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1912 011730 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE
1913 011736 001401      BEQ      TST106          ;;BR IF SET
1914 011740 104001      ERROR    1              ;CLOCK OVERFLOW ENABLE FAILED TO SET

```

```

1915
1916 ;:*****
1917 ;*TEST 106 DOES AD INTERRUPT ENABLE (BIT 6) SET
1918 ;:*****
1919 011742 000004 †ST106: SCOPE
1920 ; SET BIT 6
1921 011744 012737 000100 001124 MOV #BIT6,$GDDAT ;LOAD EXPECTED
1922 ;*
1923 ;* MOV $GDDAT,$ADCS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1924 ;*
1925 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1926 011772 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1927 012000 001401 BEQ TST107 ;;BR IF SET
1928 012002 104001 ERROR 1 ;AD INTERRUPT ENABLE FAILED TO SET
1929
1930 ;:*****
1931 ;*TEST 107 DOES MUX CHANNEL (BIT 8) SET
1932 ;:*****
1933 012004 000004 †ST107: SCOPE
1934 ; SET BIT 8
1935 012006 012737 000400 001124 MOV #BIT8,$GDDAT ;LOAD EXPECTED
1936 ;*
1937 ;* MOV $GDDAT,$ADCS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1938 ;*
1939 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1940 012034 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1941 012042 001401 BEQ TST110 ;;BR IF SET
1942 012044 104001 ERROR 1 ;MUX BIT 8 FAILED TO SET
1943
1944 ;:*****
1945 ;*TEST 110 DOES MUX CHANNEL (BIT 9) SET
1946 ;:*****
1947 012046 000004 †ST110: SCOPE
1948 ; SET BIT 9
1949 012050 012737 001000 001124 MOV #BIT9,$GDDAT ;LOAD EXPECTED
1950 ;*
1951 ;* MOV $GDDAT,$ADCS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1952 ;*
1953 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1954 012076 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1955 012104 001401 BEQ TST111 ;;BR IF EQUAL
1956 012106 104001 ERROR 1 ;MUX BIT 9 FAILED TO SET
1957
1958 ;:*****
1959 ;*TEST 111 DOES MUX CHANNEL (BIT 10) SET
1960 ;:*****
1961 012110 000004 †ST111: SCOPE
1962 012112 012737 002000 001124 MOV #BIT10,$GDDAT ;SET BIT 10
1963 ;* ;LOAD EXPECTED
1964 ;* MOV $GDDAT,$ADCS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1965 ;*
1966 ;* MOV $ADCS,$BDDAT ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1967 012140 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1968 012146 001401 BEQ TST112 ;;BR IF EQUAL

```

```

1969 012150 104001          ERROR 1          ;MUX BIT 10 FAILED TO SET
1970
1971          ;*****
1972          ;*TEST 112      DOES MUX CHANNEL (BIT 11) SET
1973          ;*****
1974 012152 000004          †ST112: SCOPE
1975          ;SET BIT 11
1976 012154 012737 004000 001124          MOV      #BIT11,$GDDAT          ;LOAD EXPECTED
1977
1978          ;*      MOV      $GDDAT,$ADCS          ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1979
1980          ;*      MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1981 012202 023737 001124 001126          CMP      $GDDAT,$BDDAT          ;COMPARE
1982 012210 001401          BEQ      TST113          ;;BR IF EQUAL
1983 012212 104001          ERROR 1          ;MUX BIT 11 FAILED TO SET
1984
1985          ;*****
1986          ;*TEST 113      DOES UNIPOLAR/BIPOLAR (BIT 13) SET
1987          ;*****
1988 012214 000004          †ST113: SCOPE
1989          ;SET BIT 13
1990 012216 012737 020000 001124          MOV      #BIT13,$GDDAT          ;LOAD EXPECTED
1991
1992          ;*      MOV      $GDDAT,$ADCS          ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1993
1994          ;*      MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1995 012244 023737 001124 001126          CMP      $GDDAT,$BDDAT          ;COMPARE
1996 012252 001401          BEQ      TST114          ;;BR IF EQUAL
1997 012254 104001          ERROR 1          ;UNIPOLAR /BIPOLAR BIT FAILED TO SET
1998
1999
2000
2001          ;*****
2002          ;*TEST 114      DOES AD DONE (BIT 7) SET AND CLEAR
2003          ;*****
2004 012256 000004          †ST114: SCOPE
2005
2006          ;*      MOV      $ADDBR,$TMDAT          ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2007 012270 013700 001670          MOV      $TMDAT,R0
2008 012274 012700 000000          MOV      #0,R0
2009 012300 012737 000001 001670          MOV      #BIT0,$TMDAT
2010
2011          ;*      MOV      $TMDAT,$ADCS          ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2012 012316 012737 000200 001124          MOV      #BIT7,$GDDAT          ;LOAD EXPECTED
2013 012324
2014          ;*      MOV      $ADCS,$TMDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2015          ;TSTB      $TMDAT
2016 012334 105737 001670          BMI      2$
2017 012340 100407          INC      R0
2018 012342 005200          BNE      1$
2019 012344 001367
2020
2021          ;*      MOV      $ADCS,$BDDAT          ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2022 012356 104001          ERROR 1          ;ERROR, A TO D DONE FAILED TO SET

```

```

2023 012360          2$:
2024
2025                ;*   MOV   QADDBR,$TMDAT  ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2026 012370 013700 001670          MOV   $TMDAT,R0
2027 012374 012737 000000 001124  MOV   #0,$GDDAT          ;CLEAR EXPECTED
2028
2029                ;*   MOV   QADCS,$BDDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2030 012412 105737 001126          TSTB  $BDDAT
2031 012416 100001          BPL   #3$
2032 012420 104001          ERROR  1          ;;BR IF CLEARED
2033                                     ;ERROR, DON& FAILED TO CLEAR UPON
2034 012422          3$:                                     ;READING CONVERTED VALUE
2035
2036                ;*   MOV   QADCS,$BDDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2037 012432 005037 001124          CLR   $GDDAT          ;CLEAR EXPECTED
2038 012436 032737 000001 001126  BIT   #BIT0,$BDDAT
2039 012444 001401          BEQ   TST115
2040 012446 104001          ERROR  1          ;;BR IF CLEARED
2041                                     ;ERROR, AD GO FAILED TO CLEAR
2042
2043                ;*****
2044                ;*TEST 115      TEST THAT THE CONVERTED NUMBER = 1777 (SW BIT 10=1)
2045                ;*****
2046 012450 000004          †ST115: SCOPE
2047 012452 012737 000000 001670  MOV   #0,$TMDAT          ;INIT
2048
2049                ;*   MOV   $TMDAT,QADCS  ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2050
2051                ;*   MOV   QADCS,$TMDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2052 012500 005237 001670          INC   $TMDAT
2053
2054                ;*   MOV   $TMDAT,QADCS  ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2055 012514          1$:
2056
2057                ;*   MOV   QADCS,$TMDAT  ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2058 012524 105737 001670          TSTB  $TMDAT
2059 012530 100371          BPL   1$
2060 012532 012737 001777 001124  MOV   #1777,$GDDAT          ;LOAD EXPECTED
2061
2062                ;*   MOV   QADDBR,$BDDAT  ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
2063 012550 032777 002000 166362  BIT   #BIT10,QSWR          ;TEST BIT 10
2064 012556 001405          BEQ   TST116          ;;BR IF CLEARED
2065 012560 023737 001124 001126  CMP   $GDDAT,$BDDAT          ;COMPARE
2066 012566 001401          BEQ   TST116          ;;BR IF EQUAL
2067 012570 104016          ERROR  16          ;CONVERTED VALUE NOT = 1777
2068                                     ;OPERATOR INFORMED THE PROGRAM THAT THE
2069                                     ; ALL 1'S JUMPER HAD BEEN INSTALLED
2070
2071                ;*****
2072                ;*TEST 116      TEST THAT NO EXTERNAL CONVERSIONS INPUT
2073                ;*****
2074 012572 000004          †ST116: SCOPE
2075 012574 012737 000004 001166  MOV   #4,$TIMES          ;;DO 4 ITERATIONS
2076 012602 012737 000000 001670  MOV   #0,$TMDAT

```

```

2077
2078 012620 012737 000020 001124 ;* MOV $TMDAT, @ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2079 ;* MOV #BIT4, $GDDAT ;LOAD EXPECTED
2080
2081 012636 013700 001670 ;* MOV @ADDBR, $TMDAT ;/READ DEVICE REG ADDBR, PUT DATA IN $TMDAT.
2082 ;* MOV $TMDAT, RO
2083
2084 012652 052737 000020 001670 ;* MOV @ADCS, $TMDAT ;/READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
2085 ;* BIS #BIT4, $TMDAT
2086
2087 012670 012700 000000 ;* MOV $TMDAT, @ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2088 ;* MOV #0, RO
2089 1$: INC RO
2090 012674 005200 ;* BNE 1$
2091 012676 001376 ;*
2092 ;* MOV @ADCS, $BDDAT ;/READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
2093 ;* TSTB $BDDAT
2094 ;* BPL TST117 ;;BR IF PLUS
2095 ;* ERROR 1 ;ERROR EXTERNAL CONVERSION
2096
2097 ;:*****
2098 ;*TEST 117 TEST THAT CLOCK CAN START A CONVERSION
2099 ;:*****
2100 012720 000004 ;* TST117: SCOPE
2101 012722 012737 000004 001166 ;* MOV #4, $TIMES ;;DO 4 ITERATIONS
2102 012730 012737 000000 001670 ;* MOV #0, $TMDAT ;
2103
2104 ;* MOV $TMDAT, @ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2105 ;*
2106 ;* MOV $TMDAT, @CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2107 012756 012737 000240 001124 ;* MOV #BIT7!BITS, $GDDAT ;LOAD EXPECTED DATA
2108 012764 012737 000003 014610 ;* MOV #3, RATE ;LOAD CLOCK RATE
2109
2110 ;* MOV @ADCS, $TMDAT ;/READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
2111 013002 052737 000040 001670 ;* BIS #BITS, $TMDAT
2112 ;*
2113 ;* MOV $TMDAT, @ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2114 013020 004737 014002 ;* JSR PC, UPCNT ;ENABLE THE CLOCK
2115 013024 012700 000040 ;* MOV #BITS, RO ;SETUP DELAY
2116 1$: DEC RO ;DELAY
2117 013032 001376 ;* BNE 1$
2118
2119 ;* MOV @ADCS, $BDDAT ;/READ DEVICE REG ADCS, PUT DATA IN $BDDAT.
2120 013044 023737 001124 001126 ;* CMP $GDDAT, $BDDAT ;COMPARE
2121 013052 001401 ;* BEQ TST120 ;;BR IF EQUAL
2122 013054 104001 ;* ERROR 1 ;CLOCK OVERFLOW FAILED TO START A CONVERSION
2123
2124 ;:*****
2125 ;*TEST 120 TEST THAT RESET CLEARS MUX AND UNIPOLAR BITS
2126 ;:*****
2127 013056 000004 ;* TST120: SCOPE
2128 013060 012737 000002 001166 ;* MOV #2, $TIMES ;;DO 2 ITERATIONS
2129 013066 005037 001124 ;* CLR $GDDAT
2130 013072 012737 027400 001670 ;* MOV #BIT13!BIT11!BIT10!BIT9!BIT8, $TMDAT

```

```

2131
2132          ;*      MOV      $TMDAT,ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2133 013110 004737 023174          JSR      PC,$RESET
2134
2135          ;*      MOV      ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2136 013124 005737 001126          TST      $BDDAT
2137 013130 001401          BEQ      TST121      ;;BR IF CLEARED
2138 013132 104000          ERROR
2139
2140          ;*****
2141          ;*TEST 121      TEST THAT RESET CLEARS EXT AND INTERRUPT ENABLE BITS
2142          ;*****
2143          ;ST121: SCOPE
2144 013134 000004          MOV      #2,$TIMES      ;;DO 2 ITERATIONS
2145 013136 012737 000002 001166          MOV      #BIT6:BIT4,$TMDAT
2146 013144 012737 000120 001670
2147
2148          ;*      MOV      $TMDAT,ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2149 013162 005037 001124          CLR      $GDDAT      ;CLEAR EXPECTED
2150 013166 004737 023174          JSR      PC,$RESET
2151
2152          ;*      MOV      ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2153 013202 001401          BEQ      TST122      ;;BR IF CLEARED
2154 013204 104001          ERROR      1      ;ERROR, RESET FAILED TO CLEAR ENABLES
2155
2156          ;*****
2157          ;*TEST 122      TEST THAT RESET CLEARS AD DONE
2158          ;*****
2159          ;ST122: SCOPE
2160 013206 000004          MOV      #2,$TIMES      ;;DO 2 ITERATIONS
2161 013210 012737 000002 001166          MOV      #BIT0,$TMDAT
2162 013216 012737 000001 001670
2163
2164          ;*      MOV      $TMDAT,ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2165 013234 005037 001124          CLR      $GDDAT
2166
2167          ;*      MOV      ADCS,$TMDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2168 013250 105737 001670          TSTB    $TMDAT
2169 013254 100375          BPL      -4
2170 013256 004737 023174          JSR      PC,$RESET
2171
2172          ;*      MOV      ADCS,$BDDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2173 013272 001401          BEQ      TST123      ;;BR IF CLEARED
2174 013274 104001          ERROR      1      ;ERROR, RESET FAILED TO CLEAR DONE FLAG
2175
2176          ;*****
2177          ;*TEST 123      TEST THAT RESET CLEARS AD BUFFER REG
2178          ;*****
2179          ;ST123: SCOPE
2180 013276 000004          MOV      #2,$TIMES      ;;DO 2 ITERATIONS
2181 013300 012737 000002 001166          MOV      #1,$TMDAT      ;CONVERT
2182 013306 012737 000001 001670
2183
2184          ;*      MOV      $TMDAT,ADCS      ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2185          ;S:
2186          ;*      MOV      ADCS,$TMDAT      ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2187 013334 105737 001670          TSTB    $TMDAT      ;WAIT

```



```

2185 013340 100371          BPL      1$
2186 013342 004737 023174  JSR      PC,$RESET
2187
2188          ;*      MOV      @ADDBR,$BDDAT  ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
2189 013356 005037 001124          CLR      $GDDAT          ;LOAD EXPECTED
2190 013362 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE
2191 013370 001401          BEQ      TST124          ;;BR IF CLEARED
2192 013372 104016          ERROR    16          ;RESET FAILED TO CLEAR AD BUFFER REGISTER
2193          ;*****
2194          ;*TEST 124      LOAD DIFFERENT NUMBERS INTO DIFFERENT REG.
2195          ;*****
2196 013374 000004          †ST124: SCOPE
2197 013376 004737 023174  JSR      PC,$RESET
2198 013402 013700 001626  MOV      CSC,RO          ;LOAD STARTING ADDRESS
2199 013406 062700 000002  ADD      #2,RO          ;ADJUST ADDRESS
2200 013412 012701 013566  MOV      #BUFNUM+20,R1  ;LOAD STARTING TABLE ADDRESS
2201 013416 012702 000010  MOV      #8,R2          ;LOAD COUNT
2202 013422 010037 001670  1$:     MOV      RO,$TMDAT
2203 013426 162700 000002  SUB      #2,RO
2204 013432 162737 000002 001670  SUB      #2,$TMDAT
2205 013440 014137 001124  MOV      -(R1),$GDDAT
2206
2207          ;*      MOV      $GDDAT,@$TMDAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG $TMDAT
2208 013454 005302          DEC      R2          ;DONE ALL
2209 013456 001361          BNE     1$          ;BR IF NOT
2210
2211 013460 013700 001610  MOV      ADCS,RO        ;LOAD STARTING POINTER
2212 013464 012701 013546  MOV      #BUFNUM,R1    ;LOAD STARTING POINTER <EXPECTED>
2213 013470 012702 000010  MOV      #8,R2         ;LOAD # OF REG
2214
2215 013474 011137 001124  3$:     MOV      (R1),$GDDAT  ;READ REG
2216 013500 010037 001670  MOV      RO,$TMDAT
2217
2218          ;*      MOV      @$TMDAT,$BDDAT  ;/READ DEVICE REG $TMDAT,PUT DATA IN $BDDAT.
2219 013514 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE
2220 013522 001403          BEQ     4$          ;;BR IF EQUAL
2221 013524 010037 013566  MOV      RO,BUFADR     ;SAVE BUS ADDRESS
2222 013530 104017          ERROR    17          ;INCORRECT DATA, REG. WAS CHANGED IN ERROR
2223 013532 005721 4$:     TST      (R1)+          ;UPDATE POINTERS
2224 013534 062700 000002  ADD      #2,RO
2225 013540 005302          DEC      R2          ;DONE ALL REG
2226 013542 001354          BNE     3$          ;BR IF NOT
2227 013544 000411          BR      TST125        ;;NEXT TEST
2228
2229          ;# TO BE LOADED INTO DIFFERENT REG
2230
2231 013546 027560          BUFNUM: 27560          ;A TO D STATUS
2232 013550 000000          0          ;A TO D BUFFER
2233 013552 140736          140736          ;CLOCK STATUS
2234 013554 000377          377          ;CLOCK PRESET
2235 013556 007214          7214          ;VC STATUS
2236 013560 001252          1252          ;VC X POS
2237 013562 000525          525          ;VC Y POS
2238 013564 000377          377          ;CLOCK COUNTER

```

```

2239
2240 013566 170400          BUFAOR: 170400          ;BUS ADDRESS OF REG IN ERROR
2241
2242
2243
2244
2245 013570 000004          *TEST 125          DETERMINE IF MORE AR11'S ARE TO BE TESTED
2246 013572 005737 001652          *****
2247 013576 001411          *ST125: SCOPE
2248 013600 062737 000020 001644          BYPASS: TST          NBEXT          ;TEST IF ANY
2249 013606 062737 000020 001646          BEQ          1$          ;BR IF NONE
2250 013614 005337 001652          ADD          #20,ARBADD          ;UPDATE DEVICE ADDRESS
2251 013620 000413          ADD          #20,ARBVCT          ;UPDATE DEVICE VECTOR
2252 013622 013737 001252 001644          DEC          NBEXT          ;ANOTHER ONE ?
2253 013630 013737 001246 001646          BR          BYPASS1          ;BR IF ANOTHER
2254 013636 013737 001650 001652          1$: MOV          $BASE,ARBADD          ;RELOAD ADDRESS
2255 013644 000137 013660          MOV          $VECT1,ARBVCT          ;RELOAD VECTOR
2256 013650 012700 177777          MOV          $MBEXT,NBEXT          ;RELOAD NUMBER
2257 013654 000137 002334          JMP          $EOP          ;DONE
2258          BYPASS1: MOV          #-1,RO          ;TEST ANOTHER UNIT
2259          JMP          RBEG2
2260          .SBTTL END OF PASS ROUTINE
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292

```

*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 BYPASS1

```

$EOP:
        SCOPE
        CLR          $STNM          ;; ZERO THE TEST NUMBER
        CLR          $TIMES          ;; ZERO THE NUMBER OF ITERATIONS
        INC          $PASS          ;; INCREMENT THE PASS NUMBER
        BIC          #100000,$PASS  ;; DON'T ALLOW A NEG. NUMBER
        DEC          (PC)+          ;; LOOP?
$EOPCT: .WORD          1
        BGT          $DOAGN          ;; YES
        MOV          (PC)+,@(PC)+  ;; RESTORE COUNTER
$ENDCT: .WORD          1
        TYPE          $SENDMG          ;; TYPE "END PASS #"
        MOV          $PASS,-(SP)    ;; SAVE $PASS FOR TYPEOUT
        TYPDS          $ENULL          ;; GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE          #42,RO        ;; TYPE A NULL CHARACTER
        MOV          $DOAGN          ;; GET MONITOR ADDRESS
        BEQ          $DOAGN          ;; BRANCH IF NO MONITOR
        RESET          ;; CLEAR THE WORLD
        JSR          PC,(RO)        ;; GO TO MONITOR
        NOP          ;; SAVE ROOM
        NOP          ;; FOR
        NOP          ;; ACT11
$DOAGN: JMP          @(PC)+          ;; RETURN
$RTNAD: .WORD          BYPASS1

```

```

2293 013762 377 377 000
2294 013765 015 042412 042116
2295 013772 050040 051501 020123
2296 014000 000043
2297
2298
2299
2300
2301
2302
2303 014002 012737 000000 001670
2304
2305 014020 012737 177776 001670
2306
2307
2308 014036 013737 014610 001670
2309
2310
2311 014054 012737 000000 001656
2312 014062
2313
2314
2315 014072 105737 001670
2316 014076 100407
2317 014100 022727 000000 000000
2318 014106 062737 000001 001656
2319 014114 001362
2320 014116 000207
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335 014120
2336 014120 010046
2337 014122 010146
2338 014124 010246
2339 014126 010346
2340 014130 010546
2341 014132 012746 020200
2342 014136 016605 000020
2343 014142 100004
2344 014144 005405
2345 014146 112766 000055 000001
2346 014154 005000

```

```

$ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
$ENDMG: .ASCIZ <15><12>/END PASS #/

;SUBROUTINE TO LOAD -2 INTO THE CLOCK PRESET REGISTER
; AND LOAD CLOCK RATE INTO CLOCK STATUS REGISTER
; AND START THE CLOCK. WAIT A PERIOD OF TIME THEN EXIT
UPCNT: MOV #0,$TMDAT ;CLEAR CLOCK STATUS
;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
MOV #-2,$TMDAT ;MOVE -2 INT PRESET
;* MOV $TMDAT,$CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
MOV RATE,$TMDAT ;LOAD RATE AND ENABLE CLOCK
;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
MOV #0,DELAY
UPCNTA:
;* MOV $CSR,$TMDAT ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
TSTB $TMDAT
BMI UPCNTB ;
CMP #0,#0 ;
ADD #1,DELAY ;
BNE UPCNTA ;
UPCNTB: RTS 7 ;EXIT

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

$TYPDS: MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R2,-(SP) ;:PUSH R2 ON STACK
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R5,-(SP) ;:PUSH R5 ON STACK
MOV #20200,-(SP) ;:SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 ;:GET THE INPUT NUMBER
BPL 1$ ;:BR IF INPUT IS POS.
R5 ;:MAKE THE BINARY NUMBER POS.
NEG ;:MAKE THE BINARY NUMBER NEG.
MOVB #'-,1(SP) ;:MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ;:ZERO THE CONSTANTS INDEX

```

```

2347 014156 012703 014334      MOV      #SDBLK,R3      ;; SETUP THE OUTPUT POINTER
2348 014162 112723 000040      MOVVB   #' ,(R3)+     ;; SET THE FIRST CHARACTER TO A BLANK
2349 014166 005002      CLR      R2           ;; CLEAR THE BCD NUMBER
2350 014170 016001 014324      MOV      $DTBL(R0),R1  ;; GET THE CONSTANT
2351 014174 160105      SUB      R1,R5        ;; FORM THIS BCD DIGIT
2352 014176 002402      BLT     4$           ;; BR IF DONE
2353 014200 005202      INC     R2           ;; INCREASE THE BCD DIGIT BY 1
2354 014202 000774      BR      3$
2355 014204 060105      ADD     R1,R5        ;; ADD BACK THE CONSTANT
2356 014206 005702      TST     R2           ;; CHECK IF BCD DIGIT=0
2357 014210 001002      BNE     5$          ;; FALL THROUGH IF 0
2358 014212 105716      TSTB   (SP)         ;; STILL DOING LEADING 0'S?
2359 014214 100407      BMI     7$          ;; BR IF YES
2360 014216 106316      ASLB   (SP)         ;; MSD?
2361 014220 103003      BCC     6$          ;; BR IF NO
2362 014222 116663 000001 177777      MOVVB   1(SP) -1(R3)  ;; YES--SET THE SIGN
2363 014230 052702 000060      BIS     #'0,R2       ;; MAKE THE BCD DIGIT ASCII
2364 014234 052702 000040      BIS     #' ,R2       ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
2365 014240 110223      MOVVB   R2,(R3)+     ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
2366 014242 005720      TST     (R0)+        ;; JUST INCREMENTING
2367 014244 020027 000010      CMP     R0,#10       ;; CHECK THE TABLE INDEX
2368 014250 002746      BLT     2$          ;; GO DO THE NEXT DIGIT
2369 014252 003002      BGT     8$          ;; GO TO EXIT
2370 014254 010502      MOV     R5,R2        ;; GET THE LSD
2371 014256 000764      BR      6$          ;; GO CHANGE TO ASCII
2372 014260 105726      TSTB   (SP)+        ;; WAS THE LSD THE FIRST NON-ZERO?
2373 014262 100003      BPL     9$          ;; BR IF NO
2374 014264 116663 177777 177776      MOVVB   -1(SP) ,-2(R3) ;; YES--SET THE SIGN FOR TYPING
2375 014272 105013      CLRB   (R3)         ;; SET THE TERMINATOR
2376 014274 012605      MOV     (SP)+,R5     ;; POP STACK INTO R5
2377 014276 012603      MOV     (SP)+,R3     ;; POP STACK INTO R3
2378 014300 012602      MOV     (SP)+,R2     ;; POP STACK INTO R2
2379 014302 012601      MOV     (SP)+,R1     ;; POP STACK INTO R1
2380 014304 012600      MOV     (SP)+,R0     ;; POP STACK INTO R0
2381 014306 104401 014334      TYPE   $SDBLK        ;; NOW TYPE THE NUMBER
2382 014312 016666 000002 000004      MOV     2(SP),4(SP)  ;; ADJUST THE STACK
2383 014320 012616      MOV     (SP)+,(SP)
2384 014322 000002      RTI
2385 014324 023420      $DTBL: 10000.
2386 014326 001750      1000.
2387 014330 000144      100.
2388 014332 000012      10.
2389 014334 000004      $SDBLK: .BLKW 4
2390      ;; SUBROUTINE TO TEST THE CLOCK REPEATABILITY
2391      ;; FIRST CLEAR CLOCK STATUS AND PRESET BUFFER
2392      ;; THEN ENABLE THE CLOCK TO COUNT AT A RATE.
2393      ;; DECREMENT R0 FOR SOME PERIOD OF TIME, WHEN R0 = 0
2394      ;; SAVE THE COUNTER VALUE AND REPEAT THIS OPERATION AGAIN
2395      ;; THEN COMPARE THE FIRST TIMED VALUE TO THE SECOND TIMED VALUE
2396      ;; <MACHINE AND MEMORY TIMING NOT IMPORTANT>
2397      ;; TO BE WITHIN THE VALUE SPECIFIED BY LOCATION CNTDEV
2398      ;; IF GREATER THAN EXPECTED IT IS AN ERROR.
2399      ;; ALSO TEST THAT THE COUNTER HAS REACHED A MIN. COUNT
2400

```

```

2401 014344 012737 000000 001670 REPEAT: MOV #0,$TMDAT ;STOP THE CLOCK
2402 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2403 ;* MOV $TMDAT,$CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
2404 ;* MOV (PS)+,RATE ;SET UP RATE
2405 014372 012537 014610 MOV (RS)+,CNTDEV ;SET UP CNT. DEV
2406 014376 012537 014612 MOV (RS)+,MINCNT ;SET UP MIN COUNT
2407 014402 012537 014614 MOV (S)+,CKDLY ;SAVE DELAY
2408 014406 012537 014616 JSR PC,10$ ;DUMMY TO CHARGE THE "CACHE"
2409 014412 004737 014502 JSR PC,10$ ;ENABLE THE CLOCK
2410 014416 004737 014502 MOV RO,$GDDAT ;SAVE FIRST TIME
2411 014422 010037 001124 JSR PC,10$ ;ENABLE THE CLOCK
2412 014426 004737 014502 MOV RO,$BDDAT ;SAVE SECOND TIME
2413 014432 010037 001126 MOV $GDDAT,RO ;GET FIRST TIME AGAIN
2414 014436 013700 001124 SUB $BDDAT,RO ;SUBTRACT SECOND TIME
2415 014442 163700 001126 BPL 3$
2416 014446 100001 NEG RO ;MAGNITUDE OF DIFFERENCE IN RO
2417 014450 005400 CMP $BDDAT,MINCNT ;COMPARE TO MIN. COUNT
2418 014452 023737 001126 014614 3$: BGE 4$ ;BRANCH IF GREATER
2419 014460 002004 MOV MINCNT,$GDDAT ;LOAD $GDDAT FOR TYPE-OUT
2420 014462 013737 014614 001124 RTS R5
2421 014470 000205 RTS R5
2422 014472 005725 4$: TST (RS)+ ;UPDATE THE STACK
2423 014474 020037 014612 CMP RO,CNTDEV ;COMPARE TO DEVIATION
2424 014500 000205 RTS R5
2425
2426
2427 014502 013700 014616 001670 10$: MOV CKDLY,RO ;GET DELAY
2428 014506 013737 014610 MOV RATE,$TMDAT ;LOAD RATE
2429
2430 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2431 014524 005237 001670 INC $TMDAT ;ENABLE CLOCK
2432
2433 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2434 014540 005300 1$: DEC RO ;DELAY
2435 014542 001376 BNE 1$
2436
2437 ;* MOV $CSC,$TMDAT ;/READ DEVICE REG CSC,PUT DATA IN $TMDAT.
2438 014554 013700 001670 MOV $TMDAT,RO
2439 014560 012737 000000 001670 MOV #0,$TMDAT ;STOP CLOCK
2440
2441 ;* MOV $TMDAT,$CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2442
2443 ;* MOV $TMDAT,$CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
2444 014606 000207 RTS PC ;EXIT
2445
2446 014610 000000 RATE: 0 ;CLOCK RATE
2447 014612 000000 CNTDEV: 0 ;CLOCK DEV.
2448 014614 000000 MINCNT: 0 ;MIN. COUNT
2449 014616 000000 CKDLY: 0
2450 014620 020101 047524 042040 EM1: .ASCIZ /A TO D STATUS REGISTER IN ERROR/
2451 014626 051440 040524 052524
2452 014634 020123 042522 044507
2453 014642 052123 051105 044440
2454 014650 020116 051105 047522

```

2455	014656	000122				
2456	014660	020101	047524	042040	EM2:	.ASCIZ /A TO D INTERRUPT ERROR/
2457	014666	044440	052116	051105		
2458	014674	052522	052120	042440		
2459	014702	051122	051117	000		
2460	014707	103	047514	045503	EM3:	.ASCIZ /CLOCK STATUS REGISTER IN ERROR/
2461	014714	051440	040524	052524		
2462	014722	020123	042522	044507		
2463	014730	052123	051105	044440		
2464	014736	020116	051105	047522		
2465	014744	000122				
2466	014746	046103	041517	020113	EM4:	.ASCIZ /CLOCK PRESET REGISTER IN ERROR/
2467	014754	051120	051505	052105		
2468	014762	051040	043505	051511		
2469	014770	042524	020122	047111		
2470	014776	042440	051122	051117		
2471	015004	000				
2472	015005	103	047514	045503	EM5:	.ASCIZ /CLOCK INTERRUPT ERROR/
2473	015012	044440	052116	051105		
2474	015020	052522	052120	042440		
2475	015026	051122	051117	000		
2476	015033	103	047514	045503	EM6:	.ASCIZ /CLOCK COUNTER REGISTER IN ERROR/
2477	015040	041440	052517	052116		
2478	015046	051105	051040	043505		
2479	015054	051511	042524	020122		
2480	015062	047111	042440	051122		
2481	015070	051117	000			
2482	015073	103	047514	045503	EM7:	.ASCIZ /CLOCK COUNTED IN ERROR/
2483	015100	041440	052517	052116		
2484	015106	042105	044440	020116		
2485	015114	051105	047522	000122		
2486	015122	046103	041517	020113	EM10:	.ASCIZ /CLOCK REPEATABILITY FAILED/
2487	015130	042522	042520	052101		
2488	015136	041101	046111	052111		
2489	015144	020131	040506	046111		
2490	015152	042105	000			
2491						
2492	015155	126	020103	052123	EM11:	.ASCIZ /VC STATUS REGISTER IN ERROR/
2493	015162	052101	051525	051040		
2494	015170	043505	051511	042524		
2495	015176	020122	047111	042440		
2496	015204	051122	051117	000		
2497	015211	130	051040	043505	EM12:	.ASCIZ /X REGISTER IN ERROR/
2498	015216	051511	042524	020122		
2499	015224	047111	042440	051122		
2500	015232	051117	000			
2501	015235	131	051040	043505	EM13:	.ASCIZ /Y REGISTER IN ERROR/
2502	015242	051511	042524	020122		
2503	015250	047111	042440	051122		
2504	015256	051117	000			
2505	015261	123	047503	042520	EM14:	.ASCIZ /SCOPE INTERRUPT ERROR/
2506	015266	044440	052116	051105		
2507	015274	052522	052120	042440		
2508	015302	051122	051117	000		

2509	015307	104	053105	041511	EM15:	.ASCIZ	/DEVICE BUS ERROR/
2510	015314	020105	052502	020123			
2511	015322	051105	047522	000122			
2512							
2513	015330	047111	047503	051122	EM16:	.ASCIZ	"INCORRECT A/D BUFFER DATA"
2514	015336	041505	020124	027501			
2515	015344	020104	052502	043106			
2516	015352	051105	042040	052101			
2517	015360	000101					
2518	015362	044103	047101	042507	EM17:	.ASCIZ	/CHANGED REGISTER IN ERROR/
2519	015370	020104	042522	044507			
2520	015376	052123	051105	044440			
2521	015404	020116	051105	047522			
2522	015412	000122					
2523	015414	051105	050122	020103	DH1:	.ASCIZ	/ERRPC ADCS ADSTAT EXPECTED/
2524	015422	020040	040440	041504			
2525	015430	020123	020040	042101			
2526	015436	052123	052101	042440			
2527	015444	050130	041505	042524			
2528	015452	000104					
2529	015454	051105	050122	020103	DH2:	.ASCIZ	/ERRPC ADCS/
2530	015462	020040	040440	041504			
2531	015470	000123					
2532	015472	051105	050122	020103	DH3:	.ASCIZ	/ERRPC CSR CKSTAT EXPECTED/
2533	015500	020040	041440	051123			
2534	015506	020040	020040	045503			
2535	015514	052123	052101	042440			
2536	015522	050130	041505	042524			
2537	015530	000104					
2538	015532	051105	050122	020103	DH4:	.ASCIZ	/ERRPC CSB CKBUFF EXPECTED/
2539	015540	020040	041440	041123			
2540	015546	020040	020040	045503			
2541	015554	052502	043106	042440			
2542	015562	050130	041505	042524			
2543	015570	000104					
2544	015572	051105	050122	020103	DH5:	.ASCIZ	/ERRPC CSR/
2545	015600	020040	041440	051123			
2546	015606	000					
2547	015607	105	051122	041520	DH6:	.ASCIZ	/ERRPC CSC CKCNTR EXPECTED/
2548	015614	020040	020040	051503			
2549	015622	020103	020040	041440			
2550	015630	041513	052116	020122			
2551	015636	054105	042520	052103			
2552	015644	042105	000				
2553	015647	105	051122	041520	DH10:	.ASCIZ	/ERRPC CSC TIME1 TIME2/
2554	015654	020040	020040	051503			
2555	015662	020103	020040	052040			
2556	015670	046511	030505	020040			
2557	015676	052040	046511	031105			
2558	015704	000					
2559	015705	105	051122	041520	DH11:	.ASCIZ	/ERRPC VCADR VCSTAT EXPECTED/
2560	015712	020040	053040	040503			
2561	015720	051104	020040	053040			
2562	015726	051503	040524	020124			

2563	015734	054105	042520	052103					
2564	015742	042105	000						
2565	015745	105	051122	041520	DH12:	.ASCIZ	/ERRPC	VCXREG	X POS. EXPECTED/
2566	015752	020040	053040	054103					
2567	015760	042522	020107	054040					
2568	015766	050040	051517	020056					
2569	015774	042440	050130	041505					
2570	016002	042524	000104						
2571	016006	051105	050122	020103	DH13:	.ASCIZ	/ERRPC	VCYREG	Y POS. EXPECTED/
2572	016014	020040	041526	051131					
2573	016022	043505	020040	020131					
2574	016030	047520	027123	020040					
2575	016036	054105	042520	052103					
2576	016044	042105	000						
2577	016047	105	051122	041520	DH14:	.ASCIZ	/ERRPC	VCSTAT/	
2578	016054	020040	053040	051503					
2579	016062	040524	000124						
2580	016066	051105	050122	020103	DH15:	.ASCIZ	/ERRPC	BASE	ACTUAL/
2581	016074	020040	041040	051501					
2582	016102	020105	020040	040440					
2583	016110	052103	040525	000114					
2584	016116	051105	050122	020103	DH16:	.ASCIZ	/ERRPC	ADDR	READ EXPECTED/
2585	016124	020040	042101	041104					
2586	016132	020122	020040	051040					
2587	016140	040505	020104	020040					
2588	016146	054105	042520	052103					
2589	016154	042105	000						
2590	016157	105	051122	041520	DH17:	.ASCIZ	/ERRPC	BUFADR	READ EXPECTED/
2591	016164	020040	041040	043125					
2592	016172	042101	020122	020040					
2593	016200	042522	042101	020040					
2594	016206	042440	050130	041505					
2595	016214	042524	000104						
2596						.EVEN			
2597	016220	001116	001610	001126	DT1:	\$ERRPC,ADCS,\$BDDAT,\$GDDAT,0			
2598	016226	001124	000000						
2599	016232	001116	001610	000000	DT2:	\$ERRPC,ADCS,0			
2600	016240	001116	001614	001126	DT3:	\$ERRPC,CSR,\$BDDAT,\$GDDAT,0			
2601	016246	001124	000000						
2602	016252	001116	001616	001126	DT4:	\$ERRPC,CSB,\$BDDAT,\$GDDAT,0			
2603	016260	001124	000000						
2604	016264	001116	001614	000000	DT5:	\$ERRPC,CSR,0			
2605	016272	001116	001626	001126	DT6:	\$ERRPC,CSC,\$BDDAT,\$GDDAT,0			
2606	016300	001124	000000						
2607	016304	001116	001620	001126	DT11:	\$ERRPC,VCSTAT,\$BDDAT,\$GDDAT,0			
2608	016312	001124	000000						
2609	016316	001116	001622	001126	DT12:	\$ERRPC,VCXREG,\$BDDAT,\$GDDAT,0			
2610	016324	001124	000000						
2611	016330	001116	001624	001126	DT13:	\$ERRPC,VCYREG,\$BDDAT,\$GDDAT,0			
2612	016336	001124	000000						
2613	016342	001116	001620	000000	DT14:	\$ERRPC,VCSTAT,0			
2614	016350	001116	001644	001126	DT15:	\$ERRPC,ARBADD,\$BDDAT,0			
2615	016356	000000							
2616	016360	001116	001612	001126	DT16:	\$ERRPC,ADDR,\$BDDAT,\$GDDAT,0			


```

2617 016366 001124 000000
2618 016372 001116 013566 001126 DT17: $ERRPC,BUFADR,$BDDAT,$GDDAT,0
2619 016400 001124 000000
2620
2621 .SBTTL SCOPE HANDLER ROUTINE
2622
2623 ;*****
2624 ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2625 ;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2626 ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2627 ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2628 ;*SW14=1 LOOP ON TEST
2629 ;*SW11=1 INHIBIT ITERATIONS
2630 ;*SW09=1 LOOP ON ERROR
2631 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
2632 ;*CALL
2633 ;* SCOPE ;;SCOPE=IOT
2634
2635 $SCOPE:
2636 016404 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2637 016406 104407 CKSWR
2638 016410 032777 040000 162522 1$: BIT #BIT14,$SWR ;;LOOP ON PRESENT TEST?
2639 016416 001114 BNE $OVER ;;YES IF SW14=1
2640 ;*****START OF CODE FOR THE XOR TESTER*****
2641 016420 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2642 ;THIS INSTRUCTION TO A "NOP" (NOP=240)
2643 016422 013746 000004 MOV @#ERRVEC, -(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2644 016426 012737 016446 000004 MOV #5,$@#ERRVEC ;;SET FOR TIMEOUT
2645 016434 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
2646 016440 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
2647 016444 000463 BR $SVLAD ;;GO TO THE NEXT TEST!
2648 016446 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
2649 016450 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
2650 016454 000423 BR 7$ ;;LOOP ON THE PRESENT TEST!
2651 016456 6$: ;*****END OF CODE FOR THE XOR TESTER*****
2652 016456 032777 000400 162454 BIT #BIT08,$SWR ;;LOOP ON SPEC. TEST?
2653 016464 001404 BEQ 2$ ;;BR IF NO
2654 016466 127737 162446 001102 CMPB $SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
2655 016474 001465 BEQ $OVER ;;BR IF YES
2656 016476 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
2657 016502 001421 BEQ 3$ ;;BR IF NO
2658 016504 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2659 016512 101015 BHI 3$ ;;BR IF NO
2660 016514 032777 001000 162416 BIT #BIT09,$SWR ;;LOOP ON ERROR?
2661 016522 001404 BEQ 4$ ;;BR IF NO
2662 016524 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
2663 016532 000446 BR $OVER
2664 016534 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
2665 016540 005037 001166 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2666 016544 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
2667 016546 032777 004000 162364 3$: BIT #BIT11,$SWR ;;INHIBIT ITERATIONS?
2668 016554 001011 BNE 1$ ;;BR IF YES
2669 016556 005737 001204 TST $PASS ;;IF FIRST PASS OF PROGRAM
2670 016562 001406 BEQ 1$ ;;INHIBIT ITERATIONS

```

```

2671 016564 005237 001104      INC      $ICNT      ;; INCREMENT ITERATION COUNT
2672 016570 023737 001166 001104      CMP      $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
2673 016576 002024      BGE      $OVER      ;; BR IF MORE ITERATION REQUIRED
2674 016600 012737 000001 001104 1$:      MOV      #1,$ICNT    ;; REINITIALIZE THE ITERATION COUNTER
2675 016606 013737 016664 001166      MOV      $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
2676 016614 105237 001102      $SVLAD: INCB     $STNM      ;; COUNT TEST NUMBERS
2677 016620 113737 001102 001202      MOV      $STNM,$STNM  ;; SET TEST NUMBER IN APT MAILBOX
2678 016626 011637 001106      MOV      (SP),$LPADR  ;; SAVE SCOPE LOOP ADDRESS
2679 016632 011637 001110      MOV      (SP),$LPERR  ;; SAVE ERROR LOOP ADDRESS
2680 016636 005037 001170      CLR      $ESCAPE     ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2681 016642 112737 000001 001115      MOV      #1,$ERMAX   ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2682 016650 013777 001102 162264 $OVER:   MOV      $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
2683 016656 013716 001106      MOV      $LPADR,(SP) ;; FUDGE RETURN ADDRESS
2684 016662 000002      RTI                    ;; FIXES PS
2685 016664 000006      $MXCNT: 6.            ;; MAX. NUMBER OF ITERATIONS
2686
2687      .SBTTL  ERROR HANDLER ROUTINE
2688
2689      ;*****
2690      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2691      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2692      ;*AND GO TO $ERRTYP ON ERROR
2693      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2694      ;*$W15=1      HALT ON ERROR
2695      ;*$W13=1      INHIBIT ERROR TYPEOUTS
2696      ;*$W09=1      LOOP ON ERROR
2697      ;*CALL
2698      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2699
2700      $ERROR:
2701 016666 104407      CKSWR
2702 016670 105237 001103      7$:      INCB     $ERFLG    ;; TEST FOR CHANGE IN SOFT-SWR
2703 016674 001775      BEQ      7$          ;; SET THE ERROR FLAG
2704 016676 013777 001102 162236      MOV      $STNM,$DISPLAY ;; DON'T LET THE FLAG GO TO ZERO
2705 016704 005237 001112      INC      $ERTTL     ;; DISPLAY TEST NUMBER AND ERROR FLAG
2706 016710 011637 001116      MOV      (SP),$ERRPC  ;; INC THE ERROR COUNT
2707 016714 162737 000002 001116      SUB      #2,$ERRPC   ;; GET ADDRESS OF ERROR INSTRUCTION
2708 016722 117737 162170 001114      MOV      @($ERRPC,$ITEMB) ;; STRIP AND SAVE THE ERROR ITEM CODE
2709 016730 032777 020000 162202      BIT      #BIT13,$SWR  ;; SKIP TYPEOUT IF SET
2710 016736 001004      BNE      20$        ;; SKIP TYPEOUTS
2711 016740 004737 017730      JSR      PC,$ERRTYP  ;; GO TO USER ERROR ROUTINE
2712 016744 104401 001173      TYPE    , $CRLF
2713 016750
2714 016750 122737 000001 001216      20$:     CMP      #APTENV,$ENV  ;; RUNNING IN APT MODE
2715 016756 001007      BNE      2$          ;; NO SKIP APT ERROR REPORT
2716 016760 113737 001114 016772      MOV      $ITEMB,21$  ;; SET ITEM NUMBER AS ERROR NUMBER
2717 016766 004737 021032      JSR      PC,$ATY4    ;; REPORT FATAL ERROR TO APT
2718 016772 000
2719 016773 000
2720 016774 000777      21$:     .BYTE    0
2721 016776 005777 162136      22$:     .BYTE    0
2722 017002 100002      BR      22$         ;; APT ERROR LOOP
2723 017004 000000      TST     @SWR        ;; HALT ON ERROR
2724 017006 104407 001000 162122      23$:     BPL      3$          ;; SKIP IF CONTINUE
2725 017010 032777      HALT
2726 017010 032777      CKSWR    ;; HALT ON ERROR!
2727 017010 032777      BIT      #BIT09,$SWR ;; TEST FOR CHANGE IN SOFT-SWR
2728 017010 032777      ;; LOOP ON ERROR SWITCH SET?

```

```

2725 017016 001402          BEQ      4$          ;; BR IF NO
2726 017020 013716 001110    MOV      $LPERR, (SP) ;; FUDGE RETURN FOR LOOPING
2727 017024 005737 001170    4$:    TST      $ESCAPE ;; CHECK FOR AN ESCAPE ADDRESS
2728 017030 001402          BEQ      5$          ;; BR IF NONE
2729 017032 013716 001170    MOV      $ESCAPE, (SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
2730 017036
2731 017036 022737 013746 000042 5$:    CMP      #SEI,DAD, @#42 ;; ACT-11 AUTO-ACCEPT?
2732 017044 001001          BNE      6$          ;; BRANCH IF NO
2733 017046 000000          HALT          ;; YES
2734 017050
2735 017050 000002          6$:    RTI          ;; RETURN
2736
2737 .SBTTL  TTY INPUT ROUTINE
2738 ;; *****
2739 .ENABL  LSB
2740
2741 ;; *****
2742 ;; *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2743 ;; *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2744 ;; *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2745 ;; *WHEN OPERATING IN TTY FLAG MODE.
2746 017052 022737 000176 001140 $CKSWR: CMP      #SWREG, SWR ;; IS THE SOFT-SWR SELECTED?
2747 017060 001074          BNE      15$        ;; BRANCH IF NO
2748 017062 105777 162056    TSTB    @STKS      ;; CHAR THERE?
2749 017066 100071          BPL      15$        ;; IF NO, DON'T WAIT AROUND
2750 017070 117746 162052    MOVB    @STKB, -(SP) ;; SAVE THE CHAR
2751 017074 042716 177600    BIC     #1C177, (SP) ;; STRIP-OFF THE ASCII
2752 017100 022726 000007    CMP     #7, (SP)+  ;; IS IT A CONTROL G?
2753 017104 001062          BNE      15$        ;; NO, RETURN TO USER
2754 017106 123727 001134 000001  CMPB    $AUTOB, #1 ;; ARE WE RUNNING IN AUTO-MODE?
2755 017114 001456          BEQ      15$        ;; BRANCH IF YES
2756
2757 017116 104401 017577    $GTSWR: TYPE    , $CNTLG ;; ECHO THE CONTROL-G (↑G)
2758 017122 104401 017604    TYPE    , $MSWR   ;; TYPE CURRENT CONTENTS
2759 017126 013746 000176    MOV     $WREG, -(SP) ;; SAVE SWREG FOR TYPEOUT
2760 017132 104402          TYPOC    ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2761 017134 104401 017615    TYPE    , $MNEW   ;; PROMPT FOR NEW SWR
2762 017140 005046          19$:   CLR     -(SP)    ;; CLEAR COUNTER
2763 017142 005046          CLR     -(SP)    ;; THE NEW SWR
2764 017144 105777 161774    7$:    TSTB    @STKS      ;; CHAR THERE?
2765 017150 100375          BPL     7$        ;; IF NOT TRY AGAIN
2766
2767 017152 117746 161770    MOVB    @STKB, -(SP) ;; PICK UP CHAR
2768 017156 042716 177600    BIC     #1C177, (SP) ;; MAKE IT 7-BIT ASCII
2769
2770
2771
2772 017162 021627 000025    9$:    CMP     (SP), #25 ;; IS IT A CONTROL-U?
2773 017166 001005          BNE     10$        ;; BRANCH IF NOT
2774 017170 104401 017572    TYPE    , $CNTLU  ;; YES, ECHO CONTROL-U (↑U)
2775 017174 062706 000006    20$:   ADD     #6, SP   ;; IGNORE PREVIOUS INPUT
2776 017200 000757          BR     19$        ;; LET'S TRY IT AGAIN
2777
2778
    
```

```

2779 017202 021627 000015 10$: CMP (SP),#15 ;: IS IT A <CR>?
2780 017206 001022 ;: BNE 16$ ;: BRANCH IF NO
2781 017210 005766 000004 ;: TST 4(SP) ;: YES, IS IT THE FIRST CHAR?
2782 017214 001403 ;: BEQ 11$ ;: BRANCH IF YES
2783 017216 016677 000002 161714 ;: MOV 2(SP),@SWR ;: SAVE NEW SWR
2784 017224 062706 000006 11$: ADD #6,SP ;: CLEAR UP STACK
2785 017230 104401 001173 14$: TYPE $CRLF ;: ECHO <CR> AND <LF>
2786 017234 123727 001135 000001 ;: CMPB $INTAG,#1 ;: RE-ENABLE TTY KBD INTERRUPTS?
2787 017242 001003 ;: BNE 15$ ;: BRANCH IF NOT
2788 017244 012777 000100 161672 ;: MOV #100,@STKS ;: RE-ENABLE TTY KBD INTERRUPTS
2789 017252 000002 ;: RTI ;: RETURN
2790 017254 004737 020744 ;: JSR PC,$TYPEC ;: ECHO CHAR
2791 017260 021627 000060 ;: CMP (SP),#60 ;: CHAR < 0?
2792 017264 002420 ;: BLT 18$ ;: BRANCH IF YES
2793 017266 021627 000067 ;: CMP (SP),#67 ;: CHAR > 7?
2794 017272 003015 ;: BGT 18$ ;: BRANCH IF YES
2795 017274 042726 000060 ;: BIC #60,(SP)+ ;: STRIP-OFF ASCII
2796 017300 005766 000002 ;: TST 2(SP) ;: IS THIS THE FIRST CHAR
2797 017304 001403 ;: BEQ 17$ ;: BRANCH IF YES
2798 017306 006316 ;: ASL (SP) ;: NO, SHIFT PRESENT
2799 017310 006316 ;: ASL (SP) ;: CHAR OVER TO MAKE
2800 017312 006316 ;: ASL (SP) ;: ROOM FOR NEW ONE.
2801 017314 005266 000002 17$: INC 2(SP) ;: KEEP COUNT OF CHAR
2802 017320 056616 177776 ;: BIS -2(SP),(SP) ;: SET IN NEW CHAR
2803 017324 000707 ;: BR 7$ ;: GET THE NEXT ONE
2804 017326 104401 001172 18$: TYPE $QUES ;: TYPE ?<CR><LF>
2805 017332 000720 ;: BR 20$ ;: SIMULATE CONTROL-U
2806 ;: .DSABL LSB
2807
2808
2809

```

;;*****

```

2810 ;: *****
2811 ;: *CALLS: ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY-
2812 ;: * RDCCHR ;: INPUT A SINGLE CHARACTER FROM THE TTY
2813 ;: * RETURN HERE ;: CHARACTER IS ON THE STACK
2814 ;: * ;: WITH PARITY BIT STRIPPED OFF
2815 ;: *
2816 ;:
2817 $RDCCHR: MOV (SP),-(SP) ;: PUSH DOWN THE PC
2818 017336 016666 000004 000002 ;: MOV 4(SP),2(SP) ;: SAVE THE PS
2819 017344 105777 161574 ;: 1$: TSTB @STKS ;: WAIT FOR
2820 017350 100375 ;: BPL 1$ ;: A CHARACTER
2821 017352 117766 161570 000004 ;: MOVB @STKB,4(SP) ;: READ THE TTY
2822 017360 042766 177600 000004 ;: BIC #177,4(SP) ;: GET RID OF JUNK IF ANY
2823 017366 026627 000004 000023 ;: CMP 4(SP),#23 ;: IS IT A CONTROL-S?
2824 017374 001013 ;: BNE 3$ ;: BRANCH IF NO
2825 017376 105777 161542 2$: TSTB @STKS ;: WAIT FOR A CHARACTER
2826 017402 100375 ;: BPL 2$ ;: LOOP UNTIL ITS THERE
2827 017404 117746 161536 ;: MOVB @STKB,-(SP) ;: GET CHARACTER
2828 017410 042716 177600 ;: BIC #177,(SP) ;: MAKE IT 7-BIT ASCII
2829 017414 022627 000021 ;: CMP (SP)+,#21 ;: IS IT A CONTROL-Q?
2830 017420 001366 ;: BNE 2$ ;: IF NOT DISCARD IT
2831 017422 000750 ;: BR 1$ ;: YES, RESUME
2832 017424 026627 000004 000140 3$: CMP 4(SP),#140 ;: IS IT UPPER CASE?

```

```

2833 017432 002407          BLT      4$          ;; BRANCH IF YES
2834 017434 026627 000004 000175    CMP      4(SP),#175  ;; IS IT A SPECIAL CHAR?
2835 017442 003003          BGT      4$          ;; BRANCH IF YES
2836 017444 042766 000040 000004    BIC      #40,4(SP)  ;; MAKE IT UPPER CASE
2837 017452 000002          RTI          ;; GO BACK TO USER
2838
2839 *****
2840 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2841 *CALL:
2842 *
2843 *   RDLIN          ;; INPUT A STRING FROM THE TTY
2844 *   RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2845 *               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2846
2845 017454 010346          $RDLIN: MOV      R3, -(SP)  ;; SAVE R3
2846 017456 012703 017562    1$:   MOV      #STTYIN, R3  ;; GET ADDRESS
2847 017462 022703 017572    2$:   CMP      #STTYIN+8., R3  ;; BUFFER FULL?
2848 017466 101405          BLOS     4$          ;; BR IF YES
2849 017470 104410          RDOCHR   ;; GO READ ONE CHARACTER FROM THE TTY
2850 017472 112613          MOVB    (SP)+, (R3)  ;; GET CHARACTER
2851 017474 122713 000177    10$:  CMPB    #177, (R3)  ;; IS IT A RUBOUT
2852 017500 001003          BNE     3$          ;; SKIP IF NOT
2853 017502 104401 001172    4$:   TYPE    $QUES  ;; TYPE A '?'
2854 017506 000763          BR      1$          ;; CLEAR THE BUFFER AND LOOP
2855 017510 111337 017560    3$:   MOVB    (R3), 9$  ;; ECHO THE CHARACTER
2856 017514 104401 017560          TYPE    9$
2857 017520 122723 000015          CMPB    #15, (R3)+  ;; CHECK FOR RETURN
2858 017524 001356          BNE     2$          ;; LOOP IF NOT RETURN
2859 017526 105063 177777          CLRB   -1(R3)      ;; CLEAR RETURN (THE 15)
2860 017532 104401 001174          TYPE    $LF        ;; TYPE A LINE FEED
2861 017536 012603          MOV     (SP)+, R3  ;; RESTORE R3
2862 017540 011646          MOV     (SP), -(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2863 017542 016666 000004 000002    MOV     4(SP), 2(SP) ;; FIRST ASCII CHARACTER ON IT
2864 017550 012766 017562 000004    MOV     #STTYIN, 4(SP)
2865 017556 000002          RTI
2866 017560 000          9$:   .BYTE    0          ;; STORAGE FOR ASCII CHAR. TO TYPE
2867 017561 000          .BYTE    0          ;; TERMINATOR
2868 017562 000010          $TTYIN: .BLKB    8.  ;; RESERVE 8 BYTES FOR TTY INPUT
2869 017572 052536 005015 000    $CNTLU: .ASCIZ  /↑U/<15><12>  ;; CONTROL "U"
2870 017577 136 006507 000012    $CNTLG: .ASCIZ  /↑G/<15><12>  ;; CONTROL "G"
2871 017604 005015 053523 020122    $MSWR:  .ASCIZ  <15><12>/SWR = /
2872 017612 020075 000
2873 017615 040 047040 053505    $MNEW:  .ASCIZ  / NEW = /
2874 017622 036440 000040
2875
2876 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2877
2878 *****
2879 *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2880 *CHANGE IT TO BINARY.
2881 *CALL:
2882 *
2883 *   RDOCT          ;; READ AN OCTAL NUMBER
2884 *   RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
2885 *               ;; HIGH ORDER BITS ARE IN $HIOCT
2886
2885 017626 011646          $RDOCT: MOV     (SP), -(SP)  ;; PROVIDE SPACE FOR THE
2886 017630 016666 000004 000002    MOV     4(SP), 2(SP)  ;; INPUT NUMBER

```

```

2887 017636 010046      MOV      R0,-(SP)      ;; PUSH R0 ON STACK
2888 017640 010146      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
2889 017642 010246      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
2890 017644 104411      1$: RDLIN              ;; READ AN ASCII LINE
2891 017646 012600      MOV      (SP)+,R0      ;; GET ADDRESS OF 1ST CHARACTER
2892 017650 005001      CLR      R1            ;; CLEAR DATA WORD
2893 017652 005002      CLR      R2
2894 017654 112046      2$: MOVB      (R0)+,-(SP) ;; PICKUP THIS CHARACTER
2895 017656 001412      BEQ      3$            ;; IF ZERO GET OUT
2896 017660 006301      ASL      R1            ;; *2
2897 017662 006102      ROL      R2
2898 017664 006301      ASL      R1            ;; *4
2899 017666 006102      ROL      R2
2900 017670 006301      ASL      R1            ;; *8
2901 017672 006102      ROL      R2
2902 017674 042716 177770      BIC      #1C7,(SP)     ;; STRIP THE ASCII JUNK
2903 017700 062601      ADD      (SP)+,R1      ;; ADD IN THIS DIGIT
2904 017702 000764      BR       2$            ;; LOOP
2905 017704 005726      3$: TST      (SP)+      ;; CLEAN TERMINATOR FROM STACK
2906 017706 010166 000012      MOV      R1,12(SP)     ;; SAVE THE RESULT
2907 017712 010237 017726      MOV      R2,$SHIOCT
2908 017716 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
2909 017720 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
2910 017722 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
2911 017724 000002      RTI
2912 017726 000000      $SHIOCT: .WORD      0  ;; HIGH ORDER BITS GO HERE
2913
2914      .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
2915
2916      ;; *****
2917      ;; *THIS ROUTINE USES THE "ITEM CONTROL BY"E" ($ITEMB) TO DETERMINE WHICH
2918      ;; *ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB).
2919      ;; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
2920
2921      $ERRTYP:
2922 017730 104401 001173      TYPE      $SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
2923 017734 010046      MOV      R0,-(SP)      ;; SAVE R0
2924 017736 005000      CLR      R0            ;; PICKUP THE ITEM INDEX
2925 017740 153700 001114      BISB     2,$ITEMB,R0
2926 017744 001004      BNE      1$
2927
2928      ;; IF ITEM NUMBER IS ZERO, JUST
2929      ;; TYPE THE PC OF THE ERROR
2930 017746 013746 001116      MOV      $ERRPC,-(SP)  ;; SAVE $ERRPC FOR TYPEOUT
2931
2932      ;; ERROR ADDRESS
2933      ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2934      ;; GET OUT
2935      ;; ADJUST THE INDEX SO THAT IT WILL
2936      ;; WORK FOR THE ERROR TABLE
2937
2938      1$: TYPOC
2939      BR       6$
2940      DEC      R0
2941      ASL      R0
2942      ASL      R0
2943      ASL      R0
2944      ADD      #,$ERRTB,R0 ;; FORM TABLE POINTER
2945      MOV      (R0)+,2$   ;; PICKUP "ERROR MESSAGE" POINTER
2946      BEQ      3$        ;; SKIP TYPEOUT IF NO POINTER
2947      TYPE     "ERROR MESSAGE"
2948      ;; TYPE THE "ERROR MESSAGE"
2949      2$: .WORD      0  ;; "ERROR MESSAGE" POINTER GOES HERE
2950

```

```

2941 020004 104401 001173          TYPE      $SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
2942 020010 012037 020020          3$: MOV      (RO)+,4$      ;; PICKUP "DATA HEADER" POINTER
2943 020014 001404          BEQ      5$              ;; SKIP TYPEOUT IF 0
2944 020016 104401          TYPE      $SCRLF          ;; TYPE THE "DATA HEADER"
2945 020020 000000          4$: .WORD 0              ;; "DATA HEADER" POINTER GOES HERE
2946 020022 104401 001173          TYPE      $SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
2947 020026 011000          5$: MOV      (RO),RO      ;; PICKUP "DATA TABLE" POINTER
2948 020030 001004          BNE      7$              ;; GO TYPE THE DATA
2949 020032 012600          6$: MOV      (SP)+,RO      ;; RESTORE RO
2950 020034 104401 001173          TYPE      $SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
2951 020040 000207          RTS      PC              ;; RETURN
2952 020042          7$:
2953 020042 013046          MOV      2(RO)+,-(SP)    ;; SAVE 2(RO)+ FOR TYPEOUT
2954 020044 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2955 020046 005710          TST      (RO)           ;; IS THERE ANOTHER NUMBER?
2956 020050 001770          BEQ      6$              ;; BR IF NO
2957 020052 104401 020060          TYPE      8$            ;; TYPE TWO(2) SPACES
2958 020056 000771          BR      7$              ;; LOOP
2959 020060 020040 000          8$: .ASCIZ  / /          ;; TWO(2) SPACES
2960 020064 020064

```

.SBTTL POWER DOWN AND UP ROUTINES

```

2965 020064 012737 020230 000024 $PWRDN: MOV      #SILLUP,2#PWRVEC ;; SET FOR FAST UP
2966 020072 012737 000340 000026      MOV      #340,2#PWRVEC+2 ;; Prio:7
2967 020100 010046          MOV      RO,-(SP)        ;; PUSH RO ON STACK
2968 020102 010146          MOV      R1,-(SP)        ;; PUSH R1 ON STACK
2969 020104 010246          MOV      R2,-(SP)        ;; PUSH R2 ON STACK
2970 020106 010346          MOV      R3,-(SP)        ;; PUSH R3 ON STACK
2971 020110 010446          MOV      R4,-(SP)        ;; PUSH R4 ON STACK
2972 020112 010546          MOV      R5,-(SP)        ;; PUSH R5 ON STACK
2973 020114 017746 161020          MOV      2SWR,-(SP)      ;; PUSH 2SWR ON STACK
2974 020120 010637 020234          MOV      SP,$SAVR6      ;; SAVE SP
2975 020124 012737 020136 000024          MOV      #SPWRUP,2#PWRVEC ;; SET UP VECTOR
2976 020132 000000          HALT
2977 020134 000776          BR      -2              ;; HANG UP
2978

```

```

2981 020136 012737 020230 000024 $PWRUP: MOV      #SILLUP,2#PWRVEC ;; SET FOR FAST DOWN
2982 020144 013706 020234          MOV      $SAVR6,SP      ;; GET SP
2983 020150 005037 020234          CLR      $SAVR6         ;; WAIT LOOP FOR THE TTY
2984 020154 005237 020234          1$: INC      $SAVR6      ;; WAIT FOR THE INC
2985 020160 001375          BNE      1$             ;; OF WORD
2986 020162 012677 160752          MOV      (SP)+,2SWR     ;; POP STACK INTO 2SWR
2987 020166 012605          MOV      (SP)+,R5       ;; POP STACK INTO R5
2988 020170 012604          MOV      (SP)+,R4       ;; POP STACK INTO R4
2989 020172 012603          MOV      (SP)+,R3       ;; POP STACK INTO R3
2990 020174 012602          MOV      (SP)+,R2       ;; POP STACK INTO R2
2991 020176 012601          MOV      (SP)+,R1       ;; POP STACK INTO R1
2992 020200 012600          MOV      (SP)+,RO       ;; POP STACK INTO RO
2993 020202 012737 020064 000024          MOV      #SPWRDN,2#PWRVEC ;; SET UP THE POWER DOWN VECTOR
2994

```


DRLPC.P11

BINARY TO OCTAL (ASCII) AND TYPE

```

3049 020360 113704 020531          MOVB  $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
3050 020364 005404          NEG   R4
3051 020366 062704 000006          ADD   #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3052 020372 110437 020530          MOVB  R4,$OMODE      ;;SAVE IT FOR USE
3053 020376 113704 020527          MOVB  $OFILL,R4     ;;GET THE ZERO FILL SWITCH
3054 020402 016605 000012          MOV   12(SP),R5    ;;PICKUP THE INPUT NUMBER
3055 020406 005003          CLR   R3           ;;CLEAR THE OUTPUT WORD
3056 020410 006105          1$:  ROL   R5           ;;ROTATE MSB INTO "C"
3057 020412 000404          BR   3$           ;;GO DO MSB
3058 020414 006105          2$:  ROL   R5           ;;FORM THIS DIGIT
3059 020416 006105          ROL   R5
3060 020420 006105          ROL   R5
3061 020422 010503          MOV   R5,R3
3062 020424 006103          3$:  ROL   R3           ;;GET LSB OF THIS DIGIT
3063 020426 105337 020530          DECB  $OMODE      ;;TYPE THIS DIGIT?
3064 020432 100016          BPL   7$           ;;BR IF NO
3065 020434 042703 177770          BIC   #177770,R3  ;;GET RID OF JUNK
3066 020440 001002          BNE   4$           ;;TEST FOR 0
3067 020442 005704          TST   R4           ;;SUPPRESS THIS 0?
3068 020444 001403          BEQ   5$           ;;BR IF YES
3069 020446 005204          4$:  INC   R4           ;;DON'T SUPPRESS ANYMORE 0'S
3070 020450 052703 000060          BIS   #'0,R3      ;;MAKE THIS DIGIT ASCII
3071 020454 052703 000040          5$:  BIS   #' ,R3    ;;MAKE ASCII IF NOT ALREADY
3072 020460 110337 020524          MOVB  R3,$$       ;;SAVE FOR TYPING
3073 020464 104401 020524          TYPE  B$           ;;GO TYPE THIS DIGIT
3074 020470 105337 020526          7$:  DECB  $OCNT    ;;COUNT BY 1
3075 020474 003347          BGT   2$           ;;BR IF MORE TO DO
3076 020476 002402          BLT   6$           ;;BR IF DONE
3077 020500 005204          INC   R4           ;;INSURE LAST DIGIT ISN'T A BLANK
3078 020502 000744          BR   2$           ;;GO DO THE LAST DIGIT
3079 020504 012605          6$:  MOV   (SP)+,R5  ;;RESTORE R5
3080 020506 012604          MOV   (SP)+,R4    ;;RESTORE R4
3081 020510 012603          MOV   (SP)+,R3    ;;RESTORE R3
3082 020512 016666 000002 000004  MOV   2(SP),4(SP) ;;SET THE STACK FOR RETURNING
3083 020520 012616          MOV   (SP)+,(SP)
3084 020522 000002          RTI
3085 020524          8$:  .BYTE 0           ;;RETURN
3086 020525          .BYTE 0           ;;STORAGE FOR ASCII DIGIT
3087 020526          .BYTE 0           ;;TERMINATOR FOR TYPE ROUTINE
3088 020527          .BYTE 0           ;;OCTAL DIGIT COUNTER
3089 020530 000000          $OCNT: .BYTE 0    ;;ZERO FILL SWITCH
3090          $OFILL: .BYTE 0  ;;NUMBER OF DIGITS TO TYPE
3091          $OMODE: .WORD 0
3092          .SBTTL TYPE ROUTINE
3093          ;;*****
3094          ;;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3095          ;;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3096          ;;NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3097          ;;NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3098          ;;NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3099          ;;
3100          ;;CALL:
3101          ;;*1) USING A TRAP INSTRUCTION
3102          ;;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCII STRING
3103          ;;*OR

```

```

3103 ;* TYPE
3104 ;* MESADR
3105 ;*
3106
3107 020532 105737 001157 $TYPE: TSTB $TFPLG ;: IS THERE A TERMINAL?
3108 020536 100002 BPL 1$ ;: BR IF YES
3109 020540 000000 HALT ;: HALT HERE IF NO TERMINAL
3110 020542 000430 BR 3$ ;: LEAVE
3111 020544 010046 1$: MOV RD, -(SP) ;: SAVE RD
3112 020546 017600 000002 MOV @2(SP), RD ;: GET ADDRESS OF ASCIZ STRING
3113 020552 122737 000001 001216 CMPB #APTENV, $ENV ;: RUNNING IN APT MODE
3114 020560 001011 BNE 62$ ;: NO GO CHECK FOR APT CONSOLE
3115 020562 132737 000100 001217 BITB #APTSPool, $ENVM ;: SPOOL MESSAGE TO APT
3116 020570 001405 BEQ 62$ ;: NO GO CHECK FOR CONSOLE
3117 020572 010037 020602 MOV RD, 61$ ;: SETUP MESSAGE ADDRESS FOR APT
3118 020576 004737 021022 JSR PC, $ATY3 ;: SPOOL MESSAGE TO APT
3119 020602 000000 61$: .WORD 0 ;: MESSAGE ADDRESS
3120 020604 132737 000040 001217 62$: BITB #APTCSUP, $ENVM ;: APT CONSOLE SUPPRESSED
3121 020612 001003 BNE 60$ ;: YES, SKIP TYPE OUT
3122 020614 112046 2$: MOVB (RD)+, -(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
3123 020616 001005 BNE 4$ ;: BR IF IT ISN'T THE TERMINATOR
3124 020620 005726 TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
3125 020622 012600 60$: MOV (SP)+, RD ;: RESTORE RD
3126 020624 062716 000002 3$: ADD #2, (SP) ;: ADJUST RETURN PC
3127 020630 000002 RTI ;: RETURN
3128 020632 122716 000011 4$: CMPB #HT, (SP) ;: BRANCH IF <HT>
3129 020636 001430 BEQ 8$ ;:
3130 020640 122716 000200 CMPB #CRLF, (SP) ;: BRANCH IF NOT <CRLF>
3131 020644 001006 BNE 5$ ;:
3132 020646 005726 TST (SP)+ ;: POP <CR><LF> EQUIV
3133 020650 104401 TYPE ;: TYPE A CR AND LF
3134 020652 001173 $CRLF
3135 020654 105037 021010 CLRB $CHARCNT ;: CLEAR CHARACTER COUNT
3136 020660 000755 BR 2$ ;: GET NEXT CHARACTER
3137 020662 004737 020744 5$: JSR PC, $TYPEC ;: GO TYPE THIS CHARACTER
3138 020666 123726 001156 6$: CMPB $FILLC, (SP)+ ;: IS IT TIME FOR FILLER CHARS.?
3139 020672 001350 BNE 2$ ;: IF NO GO GET NEXT CHAR.
3140 020674 013746 001154 MOV $NULL, -(SP) ;: GET # OF FILLER CHARS. NEEDED
3141 ;: AND THE NULL CHAR.
3142 020700 105366 000001 7$: DECB 1(SP) ;: DOES A NULL NEED TO BE TYPED?
3143 020704 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
3144 020706 004737 020744 JSR PC, $TYPEC ;: GO TYPE A NULL
3145 020712 105337 021010 DECB $CHARCNT ;: DO NOT COUNT AS A COUNT
3146 020716 000770 BR 7$ ;: LOOP
3147
3148 ;HORIZONTAL TAB PROCESSOR
3149
3150 020720 112716 000040 8$: MOVB #' (SP) ;: REPLACE TAB WITH SPACE
3151 020724 004737 020744 9$: JSR PC, $TYPEC ;: TYPE A SPACE
3152 020730 132737 000007 021010 BITB #7, $CHARCNT ;: BRANCH IF NOT AT
3153 020736 001372 BNE 9$ ;: TAB STOP
3154 020740 005726 TST (SP)+ ;: POP SPACE OFF STACK
3155 020742 000724 BR 2$ ;: GET NEXT CHARACTER
3156 020744 105777 160200 $TYPEC: TSTB @2$TPS ;: WAIT UNTIL PRINTER IS READY

```

```

3157 020750 100375          BPL      $TYPEC
3158 020752 116677 000002 160172  MOVB   2(SP),2$TPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.
3159 020760 122766 000015 000002  CMPB   #CR,2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
3160 020766 001003          BNE    1$              ;; BRANCH IF NO
3161 020770 105037 021010  CLRB   $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
3162 020774 000406          BR     $TYPEX        ;; EXIT
3163 020776 122766 000012 000002 1$:  CMPB   #LF,2(SP)      ;; IS CHARACTER A LINE FEED?
3164 021004 001402          BEQ    $TYPEX        ;; BRANCH IF YES
3165 021006 105227          INCB   (PC)+         ;; COUNT THE CHARACTER
3166 021010 000000          $CHARCNT: .WORD 0    ;; CHARACTER COUNT STORAGE
3167 021012 000207          $TYPEX: RTS         PC
3168
3169          .SBTTL  APT COMMUNICATIONS ROUTINE
3170
3171          .:*****
3172 021014 112737 000001 021260 $ATY1:  MOVB   #1,$FFLG      ;; TO REPORT FATAL ERROR
3173 021022 112737 000001 021256 $ATY3:  MOVB   #1,$MFLG      ;; TO TYPE A MESSAGE
3174 021030 000403          BR     $ATYC
3175 021032 112737 000001 021260 $ATY4:  MOVB   #1,$FFLG      ;; TO ONLY REPORT FATAL ERROR
3176 021040
3177 021040 010046          MOV    R0,-(SP)      ;; PUSH R0 ON STACK
3178 021042 010146          MOV    R1,-(SP)      ;; PUSH R1 ON STACK
3179 021044 105737 021256  TSTB   $MFLG        ;; SHOULD TYPE A MESSAGE?
3180 021050 001450          BEQ    5$           ;; IF NOT: BR
3181 021052 122737 000001 001216  CMPB   #APTENV,$ENV   ;; OPERATING UNDER APT?
3182 021060 001031          BNE    3$           ;; IF NOT: BR
3183 021062 132737 000100 001217  BITB   #APTSPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
3184 021070 001425          BEQ    3$           ;; IF NOT: BR
3185 021072 017600 000004          MOV    24(SP),R0     ;; GET MESSAGE ADDR.
3186 021076 062766 000002 000004  ADD    #2,4(SP)      ;; BUMP RETURN ADDR.
3187 021104 005737 001176 1$:  TST   $MSGTYPE      ;; SEE IF DONE W/ LAST XMISSION?
3188 021110 001375          BNE    1$           ;; IF NOT: WAIT
3189 021112 010037 001212  MOV    R0,$MSGAD     ;; PUT ADDR IN MAILBOX
3190 021116 105720          TSTB   (R0)+        ;; FIND END OF MESSAGE
3191 021120 001376          BNE    2$           ;;
3192 021122 163700 001212  SUB    $MSGAD,R0     ;; SUB START OF MESSAGE
3193 021126 006200          R0     ;; GET MESSAGE LNTH IN WORDS
3194 021130 010037 001214  MOV    R0,$MSGGLT    ;; PUT LENGTH IN MAILBOX
3195 021134 012737 000004 001176  MOV    #4,$MSGTYPE   ;; TELL APT TO TAKE MSG.
3196 021142 000413          BR     5$
3197 021144 017637 000004 021170 3$:  MOV    24(SP),4$     ;; PUT MSG ADDR IN JSR LINKAGE
3198 021152 062766 000002 000004  ADD    #2,4(SP)      ;; BUMP RETURN ADDRESS
3199 021160 013746 177776          MOV    177776-(SP)  ;; PUSH 177776 ON STACK
3200 021164 004737 020532  JSR    PC,$TYPE     ;; CALL TYPE MACRO
3201 021170 000000          4$:  .WORD 0
3202 021172
3203 021172 105737 021260 5$:  TSTB   $FFLG        ;; SHOULD REPORT FATAL ERROR?
3204 021176 001416          BEQ    12$         ;; IF NOT: BR
3205 021200 005737 001216  TST   $ENV          ;; RUNNING UNDER APT?
3206 021204 001413          BEQ    12$         ;; IF NOT: BR
3207 021206 005737 001176 11$:  TST   $MSGTYPE     ;; FINISHED LAST MESSAGE?
3208 021212 001375          BNE    11$        ;; IF NOT: WAIT
3209 021214 017637 000004 001200  MOV    24(SP),$FATAL ;; GET ERROR #
3210 021222 062766 000002 000004  ADD    #2,4(SP)     ;; BUMP RETURN ADDR.

```

```

3211 021230 005237 001176          INC      $MSGTYPE      ;; TELL APT TO TAKE ERROR
3212 021234 105037 021260          12$:    CLR      $FFLG      ;; CLEAR FATAL FLAG
3213 021240 105037 021257          CLR      $LFLG      ;; CLEAR LOG FLAG
3214 021244 105037 021256          CLR      $MFLG      ;; CLEAR MESSAGE FLAG
3215 021250 012601          MOV      (SP)+,R1    ;; POP STACK INTO R1
3216 021252 012600          MOV      (SP)+,R0    ;; POP STACK INTO R0
3217 021254 000207          RTS      PC          ;; RETURN
3218 021256          000          $MFLG: .BYTE 0      ;; MESSG. FLAG
3219 021257          000          $LFLG: .BYTE 0      ;; LOG FLAG
3220 021260          000          $FFLG: .BYTE 0      ;; FATAL FLAG
3221          021262          .EVEN
3222          000200          APTSIZE=200
3223          000001          APTENV=001
3224          000100          APTSPool=100
3225          000040          APTCSUP=040
3226          .SBTTL TRAP DEC0DER
3227
3228          ;; *****
3229          ;; *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3230          ;; *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3231          ;; *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3232          ;; *GO TO THAT ROUTINE.
3233
3234 021262 010046          $TRAP:  MOV      RO, -(SP)      ;; SAVE RO
3235 021264 016600 000002      MOV      2(SP),RO      ;; GET TRAP ADDRESS
3236 021270 005740          TST      -(RO)        ;; BACKUP BY 2
3237 021272 111000          MOV      (RO),RO      ;; GET RIGHT BYTE OF TRAP
3238 021274 006300          ASL      RO          ;; POSITION FOR INDEXING
3239 021276 016000 021316      MOV      $TRPAD(RO),RO ;; INDEX TO TABLE
3240 021302 000200          RTS      RO          ;; GO TO ROUTINE
3241
3242
3243          ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
3244
3245 021304 011646          $TRAP2: MOV      (SP), -(SP)    ;; MOVE THE PC DOWN
3246 021306 016666 000004 000002  MOV      4(SP), 2(SP)    ;; MOVE THE PSW DOWN
3247 021314 000002          RTI          ;; RESTORE THE PSW
3248
3249          .SBTTL TRAP TABLE
3250
3251          ;; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3252          ;; *BY THE "TRAP" INSTRUCTION.
3253
3254          :          ROUTINE
3255          :          -----
3256 021316 021304          $TRPAD: .WORD  $TRAP2      TRAP+1(104401)  TTY TYPEOUT ROUTINE
3257 021320 020532          $TYPE  ;; CALL=TYPE      TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3258 021322 020330          $TYPOC ;; CALL=TYPOC     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3259 021324 020304          $TYPOS ;; CALL=TYPOS     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3260 021326 020344          $TYPON ;; CALL=TYPON      TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3261 021330 014120          $TYPDS ;; CALL=TYPDS
3262
3263 021332 017122          $GTSWR ;; CALL=GTSWR      TRAP+6(104406)  GET SOFT-SWR SETTING
3264

```

DRLPC.P11 TRAP TABLE

```

3265 021334 017052 $CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
3266 021336 017334 $RDCHR ;;CALL=RDCHR TRAP+10(104410) IY TYPEIN CHARACTER ROUTINE
3267 021340 017454 $RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
3268 021342 017626 $RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286 021344 $LPAI:
3287 021344 013746 000004 MOV 4,-(SP)
3288
3289 021350 000413 BR 31$
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302 021352 012737 021376 000004 MOV #30$,4
3303 021360 005237 170000 INC 170000
3304 021364 104401 021372 TYPE ,65$
3305 021370 000401 BR 64$
3306
3307 021374 64$: .ASCIZ <7>##
3308 021374 000401 BR 31$
3309 021376 022626 30$: CMP (SP)+,(SP)+
3310 021400 012637 000004 31$: MOV (SP)+,4
3311 021404 005037 022222 ;ALL THIS JUNK MUST BE REMOVED!!
3312 021410 004537 022224 CLR $AERR
3313 021414 000000G JSR R5,$LOAD ;LOAD MICRO-CODE.
3314 ;FILE "DRLPX2.OBJ"
3315 021416 052777 040000 160066 BIS #BIT14,@KMA00 ;ISSUE KMC+DMC INIT.
3316
3317 021424 1$:
3318 ;"HANGS" HERE THEN KMC-11 ERROR.

```

```

;*
;*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,$LPAI
;*      .WORD   0        ;ADDR. OF DEVICE ADDRESS.
;* ROUTINES REQUIRED: .LOADLP
;* PROGRAMS REQUIRED: DRLPX2
;*
;*
;*      ;RETURNS WITH $AERR=1 IF SLAVE
;*      ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.

```

```

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

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

```

```

;;TYPE ASCIZ STRING
;;GET OVER THE ASCIZ

```

```

3319 021424 010146      MOV      R1,-(SP)
3320 021426 005001      CLR      R1
3321 021430 005201      25:    INC      R1          ;STALL FOR DMC-UP
3322 021432 001376      BNE     25
3323 021434 012777 104000 160050      MOV     #BIT15:BIT11,2KMADO ;SET RUN, AND ENABLE ARBITRATION.
3324 021442 105201      255:   INCB   R1
3325 021444 001376      BNE     255
3326
3327 021446 032777 000040 160036      BIT     #BITS,2KMADO ;SLAVE READY? (READING IPBM SR)
3328 021454 001401      BEQ     35
3329
3330 021456 104000      ERROR
3331
3332 021460 012777 000004 160030      35:    MOV     #4,2KMAD2 ;READ FAST PATH
3333 021466
3334 021466 004537 023134      45:    JSR     R5, $TOuT ;-TOUT-CHECK FOR TIMEOUT
3335
3336 021472 104000      ERROR
3337
3338
3339
3340
3341
3342 021474 000774      BR      45
3343
3344
3345 021476 122777 000377 160012      CMPB   #377,2KMAD2 ;/RETURNS HERE-FROM-TIMED OUT.
3346 021504 001370      BNE     45 ;WAIT TILL KMC DON& COMMAND.
3347 021506 122777 000377 160006      CMPB   #377,2KMAD4 ;IF FAST PATH=377 THEN ERROR.
3348 021514 001001      BNE     355
3349 021516 104000      ERROR
3350
3351
3352 021520 122777 000004 157774      355:   CMPB   #4,2KMAD4 ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
3353 021526 001543      BEQ     55 ;YES-CONTINUE.
3354 021530 005227 177777      INC     #-1
3355 021534 001140      BNE     55
3356 021536 005227 177777      INC     #-1
3357 021542 001135      BNE     55
3358 021544 104401 021552      TYPE   ,67$ ;:TYPE ASCIZ STRING
3359 021550 000440      BR      66$ ;:GET OVER THE ASCIZ
3360
3361 021652
3362 021652 104401 021660      66$:   .ASCIZ <200>"W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3363 021656 000430
3364
3365 021740
3366 021740 104401 021746      69$:   .ASCIZ <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
3367 021744 000434
3368
3369 022036
3370
3371 022036 112737 177777 022170      70$:   .ASCIZ <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
3372 022044 012501      71$:   TYPE   ,71$ ;:TYPE ASCIZ STRING
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
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
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```

3373	022046	021127	000000		6\$:	CMP	(R1),#0		; TERM REACHED?
3374	022052	001444				BEQ	10\$		
3375	022054	105237	022170			INCB	11\$		
3376	022060	113777	022170	157434		MOVB	11\$,QKMA04		; FIFO DATA
3377	022066	004737	022172			JSR	PC,20\$; ISSUE SEND
3378	022072	112177	157424			MOVB	(P1)+,QKMA04		; SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
3379	022076	004737	022172			JSR	PC,20\$; ISSUE SEND
3380	022102	112177	157414			MOVB	(R1)+,QKMA04		; SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
3381	022106	004737	022172			JSR	PC,20\$		
3382									
3383	022112	032777	000002	157372	7\$:	BIT	#BIT1,QKMA00		; WAIT FOR FIFO DATA
3384	022120	001374				BNE	7\$; =1 NO DATA. =0 DATA.
3385	022122	112777	000002	157366		MOVB	#2,QKMA02		; READ FIFO.
3386									
3387	022130				8\$:				
3388	022130	004537	023134			JSR	R5, \$TOUT		; -TOUT-CHECK FOR TIMEOUT
3389									
3390	022134	104000				ERROR			; /TIME-OUT ERROR
3391									; /WE FAILED TO COMPLETE
3392									; /CURRENT OPERATION.
3393									; /CONTINUES IN THIS LOOP
3394									; /WOULD MAKE US "HANG" HERE
3395									
3396	022136	000774				BR	8\$		
3397									
3398									; /RETURNS HERE-FROM-TIMED OUT.
3399	022140	122777	000377	157350		CMPB	#377,QKMA02		; WAIT FOR READ.
3400	022146	001370				BNE	8\$		
3401	022150	105777	157346			TSTB	QKMA04		; WAS A ZERO RETURNED?
3402	022154	001734				BEQ	6\$; YES GET NEXT ADDR.
3403									; SLAVE WILL RETURN CODE 0 IF
3404	022156	005237	022222			INC	\$AERR		; DEV PRESENT. ELSE
3405									; EXIT \$AERR=1 IF SLAVE GIVES ERROR.
3406	022162	005041				CLR	-(1)		; GET RID OF REFERENCE TO BAD ADDR.
3407	022164	012601			10\$:	MOV	(SP)+,R1		
3408	022166	000205				RTS	R5		; RETURN ALL ADDR. CHECKED.
3409									
3410	022170	000000			11\$:	.WORD	0		; HOLDS DAC CODE PLUS OFFSET
3411									; TO SLAVES ADDR. TABLE.
3412									
3413	022172	112777	000003	157316	20\$:	MOVB	#3,QKMA02		; ISSUE FIFO WRITE
3414	022200				21\$:				
3415	022200	004537	023134			JSR	R5, \$TOUT		; -TOUT-CHECK FOR TIMEOUT
3416									
3417	022204	104000				ERROR			; /TIME-OUT ERROR
3418									; /WE FAILED TO COMPLETE
3419									; /CURRENT OPERATION.
3420									; /CONTINUES IN THIS LOOP
3421									; /WOULD MAKE US "HANG" HERE
3422									
3423	022206	000774				BR	21\$		
3424									
3425									; /RETURNS HERE-FROM-TIMED OUT.
3426	022210	122777	000377	157300		CMPB	#377,QKMA02		; KMC CODE WILL RETURN A "377"

```

3427 022216 001370      BNE      21$      ;WHEN DONE COMMAND.
3428 022220 000207      RTS       PC
3429
3430 022222 000000      $AERR:  .WORD   0      ;=0 IF ADDR. LIST OK,=1 IF BAD.
3431
3432      ;*
3433      ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
3434      ;*   CALL = JSR      R5,$LOAD
3435      ;*   .WORD   XX      ;ADDR. OF MICRO CODE.
3436      ;*   ;RETURNS HERE
3437      ;*   NOTE:   MICRO CODE FILE MUST END IN -1 DATA.
3438      ;*
3439
3440 022224 010446      $LOAD:  MOV      R4,-(SP)      ;SAVE R4.
3441 022226 010046      MOV      R0,-(SP)      ;SAVE R0.
3442 022230 012500      1$:     MOV      (5)+,R0      ;GET PROG. ADDR.
3443 022232 005077 157254      CLR      @KMADO        ;CLEAR CSR
3444 022236 005077 157260      CLR      @KMAD4        ;CLEAR CRAM ADDR.
3445 022242 052777 002000 157242 2$:     BIS      #2000,@KMADO    ;SELECT CRAM.
3446 022250 012077 157252      MOV      (0)+,@KMAD6    ;WRITE DATA.
3447 022254 052777 020000 157230      BIS      #20000,@KMADO  ;SET CRAM WRITE
3448 022262 005077 157224      CLR      @KMADO        ;DISABLE CRAM.
3449 022266 005277 157230      INC      @KMAD4        ;UPDATE CRAM ADDR.
3450 022272 021027 177777      CMP      (0),#-1       ;ALL DONE?
3451 022276 001361      BNE      2$           ;NO LOOP.
3452 022300 005077 157216      CLR      @KMAD4        ;CLEAR CRAM ADDR.
3453 022304 016500 177776      MOV      -2(5),R0      ;GET MICRO CODE ADDR.
3454
3455 022310 052777 002000 157174 3$:     BIS      #2000,@KMADO    ;SELECT CRAM
3456 022316 022077 157204      CMP      (R0)+,@KMAD6   ;DATA OK?
3457 022322 001013      BNE      5$           ;NO - REPORT AN ERROR.
3458 022324 021027 177777      CMP      (0),#-1       ;ALL DONE?
3459 022330 001405      BEQ      4$           ;YES - EXIT
3460 022332 005077 157154      CLR      @KMADO        ;NO - DESELECT CRAM.
3461 022336 005277 157160      INC      @KMAD4        ;UPDATE CRAM ADDR.
3462 022342 000762      BR       3$
3463
3464 022344 012600      4$:     MOV      (SP)+,R0      ;RESTORE R0
3465 022346 012604      MOV      (SP)+,R4      ;RESTORE R4
3466 022350 000205      RTS      R5           ;EXIT
3467
3468 022352      5$:     ;COME HERE ON LOAD ERROR
3469 022352 005745      TST      -(5)
3470 022354 105204      INCB    R4           ;UPDATE ERROR COUNTER.
3471 022356 100324      BPL     1$           ;IF NOT TOO MANY, TRY AGAIN.
3472 022360 000000      HALT
3473
3474 022362 000722      BR       1$           ;KMC-11 FAULT. YOU COULD TRY
3475      ;TO PRESS CONTINUE TO GIVE IT
3476      ;ANOTHER CHANCE, BUT I DOUBT
3477      ;THAT THAT WOULD WORK. SINCE I'VE
3478      ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
3479      ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
3480

```



```

3481
3482
3483
3484
3485
3486
3487
3488 022364 010046          $TLKW: MOV    RO, -(SP)          ;SAVE RO
3489 022366 012500          MOV    (S)+,RO          ;GET DEVICE OFFSET
3490 022370 052700 000340  BIS    #340,RO          ;ADD WRITE CODE.
3491 022374 004737 022646  JSR    PC,$LPW          ;WAIT FOR FAST PATH READY
3492 022400 010037 022472  MOV    RO,W1
3493 022404 010077 157112  MOV    RO,@KMA4
3494 022410 112777 000005 157100  MOVB  #5,@KMA2          ;ISSUE FAST PATH WRITE
3495 022416 004737 022646  JSR    PC,$LPW          ;WAIT FOR RDY
3496 022422 011537 022474  MOV    (S),W2
3497 022426 112577 157070  MOVB  (S)+,@KMA4        ;WRITE LOW BYTE DATA.
3498
3499 022432 112777 000005 157056  MOVB  #5,@KMA2          ;FP WRITE
3500 022440 004737 022646  JSR    PC,$LPW
3501 022444 111537 022476  MOVB  (S),W3
3502 022450 112577 157046  MOVB  (S)+,@KMA4        ;WRITE HIGH BYTE
3503 022454 112777 000005 157034  MOVB  #5,@KMA2
3504 022462 004737 022646  JSR    PC,$LPW
3505 022466 012600          MOV    (SP)+,RO
3506 022470 000205          RTS     RS              ;EXIT DONE.
3507 022472 000000          W1:    0
3508 022474 000000          W2:    0
3509 022476 000000          W3:    0
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520 022500 010046          $TLKR: MOV    RO, -(SP)          ;SAVE RO
3521 022502 012500          MOV    (S)+,RO          ;GET OFFSET
3522 022504 052700 000300  BIS    #300,RO          ;ADD READ CODE
3523 022510 004737 022646  JSR    PC,$LPW          ;WAIT TILL READY
3524 022514 110077 157002  MOVB  RO,@KMA4
3525 022520 112777 000005 156770  MOVB  #5,@KMA2          ;ISSUE WRITE FP
3526 022526 004737 022646  JSR    PC,$LPW
3527 022532 010037 022642  MOV    RO,RD1
3528
3529 022536 004537 023134  JSR    RS, $TOUT        ;-TOUT-CHECK FOR TIMEOUT
3530
3531 022542 104000          ERROR
3532
3533
3534
; *THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
; *
; *      CALL = JSR    RS,$TLKW
; *              .WORD 0          ;OFFSET OF DEVICE ADDR.
; *              .WORD 0          ;DATA TO BE WRITTEN
; *
; *THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
; *
; *      CALL = JSR    RS,$TLKR
; *              .WORD 0          ;OFFSET OF DEVICE
; *              .WORD 0          ;RETURNS HERE
; *      *DATA IN WORD $DATR
; *
; /TIME-OUT ERROR
; /WE FAILED TO COMPLETE
; /CURRENT OPERATION.
; /CONTINUES IN THIS LOOP
    
```

```

3535 ;/WOULD MAKE US "HANG" HERE
3536
3537 022544 000774 BR 1$
3538
3539 ;/RETURNS HERE-FROM-TIMED OUT.
3540 022546 032777 000040 156736 BIT #BIT5,AKMADO ;FAST PATH GOT DATA?
3541 022554 001370 BNE 1$
3542 022556 112777 000004 156732 MOVB #4,AKMAD2 ;ISSUE FAST PATH READ
3543 022564 004737 022646 JSR PC,$LPW
3544 022570 117737 156726 022644 MOVB AKMAD4,$DATR ;GET LOW BYTE
3545 2$:
3546 022576 004537 023134 JSR R5,$TOUT ;-TOUT-CHECK FOR TIMEOUT
3547
3548 022602 104000 ERROR ;/TIME-OUT ERROR
3549 ;/WE FAILED TO COMPLETE
3550 ;/CURRENT OPERATION.
3551 ;/CONTINUES IN THIS LOOP
3552 ;/WOULD MAKE US "HANG" HERE
3553
3554 022604 000774 BR 2$
3555
3556 ;/RETURNS HERE-FROM-TIMED OUT.
3557 022606 032777 000040 156676 BIT #BIT5,AKMADO ;FAST PATH READY?
3558 022614 001370 BNE 2$
3559 022616 112777 000004 156672 MOVB #4,AKMAD2 ;ISSUE FAST PATH READ
3560 022624 004737 022646 JSR PC,$LPW
3561 022630 117737 156666 022645 MOVB AKMAD4,$DATR+1 ;SAVE HIGH BYTE
3562 022636 012600 MOV (SP)+,R0
3563 022640 000205 RTS R5
3564 022642 000000 RD1: 0
3565 022644 000000 $DATR: .WORD 0
3566
3567 ; THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
3568 ; AS FAST PATH TO BE READ.
3569
3570 ; CALL = JSR PC,$LPW
3571
3572 ; IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
3573 ; THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
3574
3575
3576 022646 010146 $LPW: MOV R1,-(SP) ;SAVE R1
3577 022650 005001 CLR R1
3578 022652 122777 000377 156636 1$: CMPB #377,AKMAD2 ;FINISHED INSTRUCTION?
3579 022660 001403 BEQ 2$
3580 022662 005201 INC R1 ;TIME OUT?
3581 022664 001372 BNE 1$
3582 022666 000411 BR 10$
3583
3584 022670 032777 000020 156614 2$: BIT #BIT4,AKMADO ;FAST PATH READ?
3585 022676 001403 BEQ 3$
3586 022700 005201 INC R1 ;NO - TIME OUT?
3587 022702 001372 BNE 2$
3588 022704 000402 BR 10$ ;YES - REPORT AN ERROR

```

```

3589
3590 022706 012601          3$:  MOV      (SP)+,R1      ;RESTORE R1
3591 022710 000207          RTS      PC              ;EXIT
3592
3593 022712
3594 022712 104401 022720  10$:  TYPE      65$          ;;TYPE ASCIZ STRING
3595 022716 000407          BR       64$            ;;GET OVER THE ASCIZ
3596
3597 022736          65$:  .ASCIZ  <200>#LPA-11 FAULT#
3598          64$:
3599 022736 000000          11$:  HALT
3600 022740 000776          BR       11$          ;LPA-11 FAULT RUN LPA-11
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615 022742 010046          $OUTLP: MOV      R0,-(SP)      ;SAVE R0
3616 022744 010146          MOV      R1,-(SP)      ;SAVE R1
3617
3618 022746 012700 001540          MOV      #,DVLS,R0      ;PROGRAM DEFINED LIST.
3619 022752 005001          CLR      R1
3620 022754 005710          1$:  TST      (0)          ;TERMINATOR REACHED?
3621 022756 001421          BEQ      10$          ;YES NEXT STEP.
3622 022760 027520 000000          CMP      2(5),(0)+      ;MATCH WITH ADDR IN LIST?
3623 022764 001402          BEQ      2$
3624 022766 005201          INC      R1
3625 022770 000771          BR       1$
3626
3627 022772 010137 023010          2$:  MOV      R1,3$          ;SAVE OFFSET, DEVICE KNOWN.
3628 022776 005725          TST      (5)+
3629 023000 013537 023012          MOV      2(5)+,4$      ;GET DATA TO BE WRITTEN
3630 023004 004537 022364          JSR      R5,$TLKW      ;DO WRITE
3631 023010 000000          3$:  .WORD   0            ;DEVICE OFFSET
3632 023012 000000          4$:  .WORD   0            ;DATA TO BE WRITTEN.
3633 023014 012601          MOV      (SP)+,R1
3634 023016 012600          MOV      (SP)+,R0
3635 023020 000205          RTS      R5
3636 023022 017520 000000          10$: MOV      2(5),(0)+      ;SAVE ADDR.
3637 023026 005010          CLR      (0)
3638 023030 004537 021344          JSR      R5,$LPAI
3639 023034 001540          .WORD   .DVL$
3640 023036 000755          BR       2$
3641
3642

```

```

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

```

```

3643 ; *THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
3644 ; *TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
3645 ; *
3646 ; *FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
3647 ; *USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
3648 ; *WITH THE NEW ADDR.
3649 ; *WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
3650 ; *$TLKR
3651 ; *
3652 ; *      CALL THROUGH      MOVEI   DATA,ADDR.
3653 ; *      WHICH EQUALS:
3654 ; *      JSR      R5,$INLP
3655 ; *      .WORD   XX      ADDR OF DEVICE
3656 ; *      .WORD   YY      ADDR TO $TORE READ DATA.
3657 023040 010046      $INLP:  MOV    R0,-(SP)      ;SAVE R0
3658 023042 010146      MOV    R1,-(SP)      ;SAVE R1
3659
3660 023044 012700 001540      MOV    #.DVLS,R0      ;PROG DEFINED ADDR. LIST.
3661 023050 005001
3662 023052 005710      1$:   CLR    R1
3663 023054 001420      TST   R0              ;EOL REACHED?
3664
3665 023056 027520 000000      BEQ   10$             ;YES - DEFINE NEW ADDR.
3666 023062 001402
3667 023064 005201      CMP   @($),(0)+      ;ADDR. MATCH?
3668 023066 000771      BEQ   2$
3669
3670 023070 010137 023102      2$:   MOV    R1,3$
3671 023074 005725      TST   ($)+
3672 023076 004537 022500      JSR   R5,$TLKR      ;GO READ DEVICE
3673
3674 023102 000000      $OFS=.
3675 3$:   .WORD  0          ;OFFSET OF DEVICE
3676
3676 023104 013735 022644      MOV   $DATA,@($)+   ;STORE DATA.
3677 023110 012601      MOV   (SP)+,R1      ;RESTORE R1
3678 023112 012600      MOV   (SP)+,R0      ;RESTORE R2
3679 023114 000205      RTS   R5            ;EXIT
3680
3681 023116 017520 000000      10$:  MOV   @($),(0)+
3682 023122 005010      CLR   (0)
3683 023124 004537 021344      JSR   R5,$LPAI
3684 023130 001540      .WORD .DVLS
3685 023132 000756      BR   2$
3686
3687 ; *$STOUT ROUTINE USED TO WATCH IF
3688 ; *WE'RE IN A LOOP TOO-LONG
3689 ; *CALL= JSR R5,$STOUT
3690 ; *      ERROR X      ;RETURNS HERE ON TIMEOUT
3691 ; *      BR
3692 ; *      ;RETURNS HERE NO ERROR
3693 ; *
3694
3695 023134 020537 023170      $STOUT: CMP   R5,$$AD      ;SAME ADDR?
3696 023140 001405      BEQ   1$

```

DRLPC.P11 TRAP TABLE

```

3697 023142 010537 023170      MOV    R5,$$AD      ;NO-SAVE THIS ADDR.
3698 023146 005037 023172      CLR    $CNT         ;CLR CNT AT ADDR.
3699 023152 000403                BR     2$          ;
3700 023154 005237 023172      1$:   INC    $CNT         ;OVERFLOW?
3701 023160 100402                BMI    3$          ;YES-ERROR RETURN
3702 023162 062705 000004      2$:   ADD    #4,R5      ;NO-NON ERROR RETURN
3703 023166 000205                3$:   RTS    R5        ;RETURN.
3704
3705 023170 000000      $$AD:  .WORD  0        ;CONTAINS LOOP ADDR.
3706 023172 000000      $CNT:  .WORD  0        ;# OF TIMES AT ADDR.
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717 023174 000005      $RESET: RESET          ;RESET THE WORLD.
3718
3719
3720 023206 005737 022222      ;*   MOV    2$,$1$    ;/READ DEVICE REG 2$,PUT DATA IN 1$.
3721 023212 001004                TST    $AERR        ;IF NO ERROR,LOOP
3722 023214 062737 000002 023230      BNE    10$          ;THERE WAS AN ERROR.
3723
3724
3725
3726
3727 023222 000764                ADD    #2,2$        ;UPDATE DEVICE ADDR.
3728 023224
3729 023224 000207                ;YOU SEE ,WE HAVE TO PROTECT OUR SELF!
3730 023226 000000                ;IF 2$ CONTAINED A VALID ADDR,WE
3731 023230 160000                ;MUST KEEP TRYING UNTIL WE GENERATE
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746 023232
3747 023232 005737 023314      SDELAY: TST    RTCCSR    ;CLOCK PRESENT?
3748 023236 100016                BPL    10$          ;
3749 023240 012737 000002 023304      MOV    #2,TIME      ;
3750 023246 052777 000115 000040      BIS    #15,@RTCCSR  ;START CLOCK

```

```

;* THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
;*USE FOR A RESET. FIRST,WE DO A RESET INSTRUCTION.
;*THEN WE CLR ".DVLST" WHICH FORCES US TO RESET BOTH THE
;*KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.

```

```

CALL=JSR PC,$RESET ;REPLACES "RESET INSTRUCTION
;RETURNS HERE.

```

```

SDELAY- ROUTINE TO GIVE A MINOR DELAY.
IS NOT TIME DEPENDENT CODE SENCE
NOT USED TO GET SPECIFIC TIME BUT
JUST A LITTLE DELAY.

```

```

THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS

```

```

CALL= JSR PC, SDELAY

```

3751	023254	005037	177776		CLR	PS	
3752	023260	005737	023304	1\$:	TST	TIME	
3753	023264	001375			BNE	1\$	
3754	023266	005077	000022		CLR	2RTCCSR	; STOP CLOCK
3755							
3756	023272	000207			RTS PC		
3757	023274	105237	023304	10\$:	INCB	TIME	
3758	023300	001375			BNE	10\$	
3759	023302	000207			RTS	PC	
3760							
3761	023304	000000		TIME:	.WORD	0	
3762							
3763	023306	005337	023304	CLKINT:	DEC	TIME	
3764	023312	000002			RTI		
3765	023314	000000		RTCCSR:	.WORD	0	; CLOCK CSR IF USED.
3766							
3767							
3768							
3769							
3770							
3771							
3772							
3773							
3774							
3775							
3776							
3777	023316			\$UTK:	CLR	.DVLS	
3778	023316	005037	001540				
3779	023322			21\$:			
3780	023322	104401	023330		TYPE	65\$:: TYPE ASCIZ STRING
3781	023326	000405			BR	64\$:: GET OVER THE ASCIZ
3782							
3783	023342			:: 65\$:	.ASCIZ	<200>#E OR D?#	
3784	023342	105777	155576	64\$:			
3785	023346	100375		1\$:	TSTB	2\$TKS	
3786	023350	117737	155572		BPL	1\$	
3787	023356	104401	023472		MOVB	2\$TKB, 20\$; GET INPUT
3788	023362	142737	000240		TYPE	20\$; ECHO, NEXT MESSAGE.
3789	023370	104412			BICB	#240, 20\$; STRIP PARITY, LC
3790	023372	012637	023470		RDOCT		; GET ADDR.
3791	023376	123727	023472		MOV	(SP)+, 14\$	
3792	023404	001411	000104		CMPB	20\$, #D	; DEPOSIT?
3793					BEQ	10\$	
3794	023406	004537	023040				
3795	023412	023470		2\$:	JSI	R5, \$INLP	; GET DATA
3796	023414	023426			.WORD	14\$	
3797					.WORD	5\$	
3798	023416	013746	023426				
3799	023422	104402			MOV	5\$, -(SP)	:: SAVE 5\$ FOR TYPEOUT
3800	023424	000736			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
3801	023426	000000			BR	21\$; LOOP.
3802				5\$:	.WORD	0	
3803	023430			10\$:			
3804	023430	104401	023436		TYPE	.67\$:: TYPE ASCIZ STRING

3805	023434	000404		
3806				
3807	023446			
3808	023446	104412		
3809	023450	012637	023466	
3810				
3811	023454	004537	022742	
3812	023460	023470		
3813	023462	023466		
3814	023464	000716		
3815				
3816	023466	000000		
3817	023470	000000		
3818	023472	100001	042504	044526
3819	023500	042503	040440	042104
3820	023506	036522	000040	
3821				
3822				
3823				
3824				
3825				
3826				
3827				
3828				
3829				
3830				
3831				
3832				
3833				
3834				
3835				
3836				
3837				
3838				
3839				
3840				
3841				
3842				
3843				
3844				
3845	023512	012537	023522	
3846	023516	004537	023040	
3847	023522	000000		
3848	023524	023620		
3849	023526	113777	023102	155772
3850	023534	113777	023102	155766
3851	023542	013737	023522	023562
3852	023550	062737	000002	023562
3853	023556	004537	023040	
3854	023562	000000		
3855	023564	023620		
3856	023566	113777	023102	155724
3857	023574	152777	000340	155724
3858	023602	152777	000300	155720

```

BR 66$ ;:GET OVER THE ASCIZ
;;67$: .ASCIZ <200>#DATA= #
66$: RDOCT
MOV (SP)+,13$
11$: JSR R5,$OUTLP ;:OUTPUT ROUTINE.
12$: .WORD 14$ ;:DEVICE ADDR.
      .WORD 13$ ;:DATA
BR 21$
13$: .WORD 0
14$: .WORD 0
20$: .ASCIZ <1><200>#DEVICE ADDR= #
      .EVEN

```

```

: THIS ROUTINE LOOKS THROUGH CURENT .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 TAKING PURPOSES.
: TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
: A/D CSR IN BSEL 4, AND 5.
: (2) HE MUST CALL THIS ROUTINE:
      JSR R5,$PUTS ;:CALL SET UP ROUTINE.
      .WORD ADCSR ;:ADDR. OF A/D CSR.
      ;:RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
      ;:(UNTILL ONE DOES A RESET)
: (3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
: START CONVERSION CAUTION*DO WITH MOV8 INSTR.!
: (4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
: (5)READ KMC REG 4,5 FOR A/D RESULT.
: (6) TO TAKE MORE SAMPLES, SIMPLY PUT A/D CSR INTO
: BSEL 4,5 AND CODE 6 INTO BSEL 2.

```

```

$PUTS: MOV (5)+,1$ ;:GET ADDR OF ADDR. OF A/D
      JSR R5,$INLP
1$: .WORD 0
      .WORD 10$
      MOV8 $OFS,2KMA06
      MOV8 $OFS,2KMA07
      MOV 1$,2$
      ADD #2,2$
      JSR R5,$INLP
2$: .WORD 0
      .WORD 10$
      MOV8 $OFS,2KMA03
      BIS8 #340,2KMA06
      BIS8 #300,2KMA07

```

3859	023610	152777	000300	155702	BISB	#300,2KMA03
3860	023616	000205			RTS	R5
3861	023620	000000			10\$: .WORD	0
3862						
3863		000001			.END	

CSB	001616	499#	717	719	730	732	743	745	756	758	771	773	789	792
		803	816	830	843	858	875	1025	1068	1097	1137	1175	1211	1293
		2309	2406	2444	2602									
CSC	001626	506#	699	805	818	832	845	860	878	1032	1105	1144	1195	1231
		1321	1342	1407	1430	2198	2438	2605						
CSR	001614	498#	890	892	903	905	916	918	929	931	942	944	955	957
		968	970	981	983	953	995	1005	1010	1023	1028	1039	1047	1050
		1052	1055	1065	1071	1074	1077	1079	1082	1084	1095	1101	1114	1117
		1119	1122	1135	1140	1153	1156	1158	1161	1173	1179	1183	1186	1188
		1191	1209	1215	1219	1222	1224	1227	1245	1249	1261	1265	1277	1281
		1296	1300	1315	1336	1357	1372	1386	1400	1423	2107	2306	2312	2316
		2404	2431	2434	2442	2600	2604							
		80#	262	577										
DDISP =	177570													
DELAY	001656	522#	2312*	2319*										
OH1	015414	367	2523#											
OH10	015647	409	2553#											
OH11	015705	415	2559#											
OH12	015745	421	2565#											
OH13	016006	427	2571#											
OH14	016047	433	2577#											
OH15	016066	439	2580#											
OH16	016116	445	2584#											
OH17	016157	451	2590#											
OH2	015454	373	2529#											
OH3	015472	379	2532#											
OH4	015532	385	2538#											
OH5	015572	391	2544#											
OH6	015607	397	403	2547#										
DISPLA	001142	262#	577*	585*	2682*	2703*								
DISPRE	000174	195#	585											
ORLPX2=	*****	14#	3313											
OSWR =	177570	79#	261	576										
DT1	016220	368	2597#											
DT11	016304	416	2607#											
DT12	016316	422	2609#											
DT13	016330	428	2611#											
DT14	016342	434	2613#											
DT15	016350	440	2614#											
DT16	016360	446	2616#											
DT17	016372	452	2618#											
DT2	016232	374	2599#											
DT3	016240	380	2600#											
DT4	016252	386	2602#											
DT5	016264	392	2604#											
DT6	016272	398	404	410	2605#									
EMTVEC=	000030	168#	560*	561*										
EM1	014620	366	2450#											
EM10	015122	408	2486#											
EM11	015155	414	2492#											
EM12	015211	420	2497#											
EM13	015235	426	2501#											
EM14	015261	432	2505#											
EM15	015307	438	2509#											
EM16	015330	444	2513#											

G

PSW = 177776	76*													
PWRMSG 020236	2997	3004*												
PWRVEC= 000024	167*	564*	565*	2966*	2967*	2976*	2982*	2994*	2995*					
RATE 014610	1310*	1331*	1352*	1367*	1381*	1395*	1418*	2108*	2309	2406*	2428	2446*		
RBEG 001702	531	533*												
RBEG1 002332	616*													
RBEG2 002334	617*	2257												
RDCHR = 104410	2849	3266*												
RDLIN = 104411	2890	3267*												
RDOCT = 104412	3268*	3789	3808											
RD1 022642	3527*	3564*												
REPEAT 014344	2401*													
RESVEC= 000010	162*													
RTCCSR 023314	3747	3750*	3754*	3765*										
RO =%000000	83*	530*	532*	538	540*	543*	544	549*	626	667*	668*	669	671*	
	672	674*	676*	680	682*	1724*	1734*	1746*	1759*	1774*	1787*	1803*	1821*	
	2007*	2008*	2018*	2026*	2082*	2088*	2089*	2115*	2116*	2198*	2199*	2202	2203*	
	2211*	2216	2221	2224*	2256*	2283*	2286	2336	2346*	2350	2366	2367	2380*	
	2412	2414	2415*	2416*	2418*	2424	2427*	2434*	2438*	2887	2891*	2894	2910*	
	2923	2924*	2925*	2932*	2933*	2934*	2935*	2936*	2937	2942	2947*	2949*	2953	
	2955	2968	2993*	3111	3112*	3117	3122	3125*	3177	3185*	3189	3190	3192*	
	3193*	3194	3216*	3234	3235*	3236	3237*	3238*	3239*	3240*	3441	3442*	3453*	
	3456	3464*	3488	3489*	3490*	3492	3493	3505*	3520	3521*	3522*	3524	3527	
	3562*	3615	3618*	3634*	3657	3660*	3678*							
R1 =%000001	84*	539	541*	545	548*	619*	620*	621*	622	675*	676	677	678	
	681*	682	683	684	2200*	2205	2212*	2215	2223	2337	2350*	2351	2355	
	2379*	2888	2892*	2896*	2898*	2900*	2903*	2906	2909*	2969	2992*	3178	3215*	
	3319	3320*	3321*	3324*	3372*	3373	3378	3380	3407*	3576	3577*	3580*	3586*	
	3590*	3616	3619*	3624*	3627	3633*	3658	3661*	3667*	3670	3677*			
R2 =%000002	85*	595*	597	602*	618*	620	622*	623	624	2201*	2208*	2213*	2225*	
	2338	2349*	2353*	2356	2363*	2364*	2365	2370*	2378*	2889	2893*	2897*	2899*	
	2901*	2907	2908*	2970	2991*									
R3 =%000003	86*	596*	603*	605	609*	610	2339	2347*	2348*	2362*	2365*	2374*	2375*	
	2377*	2845	2846*	2847	2850*	2851	2855	2857	2859*	2861*	2971	2990*	3046	
	3055*	3061*	3062*	3065*	3070*	3071*	3072	3081*						
R4 =%000004	87*	2972	2989*	3047	3049*	3050*	3051*	3052	3053*	3067	3069*	3077*	3080*	
	3440	3465*	3470*											
R5 =%000005	88*	600*	696*	717*	719*	730*	732*	743*	745*	756*	758*	771*	773*	
	789*	792*	803*	805*	816*	818*	830*	832*	843*	845*	858*	860*	875*	
	878*	890*	892*	903*	905*	916*	918*	929*	931*	942*	944*	955*	957*	
	968*	970*	981*	983*	993*	995*	1005*	1010*	1023*	1025*	1028*	1032*	1039*	
	1047*	1050*	1052*	1055*	1065*	1068*	1071*	1074*	1077*	1079*	1082*	1084*	1095*	
	1097*	1101*	1105*	1114*	1117*	1119*	1122*	1135*	1137*	1140*	1144*	1153*	1156*	
	1158*	1161*	1173*	1175*	1179*	1183*	1186*	1188*	1191*	1195*	1209*	1211*	1215*	
	1219*	1222*	1224*	1227*	1231*	1245*	1249*	1261*	1265*	1277*	1281*	1293*	1296*	
	1300*	1315*	1321*	1336*	1342*	1357*	1372*	1386*	1400*	1407*	1423*	1430*	1443*	
	1454*	1457*	1463*	1466*	1477*	1480*	1491*	1494*	1505*	1508*	1519*	1522*	1532*	
	1535*	1545*	1548*	1554*	1561*	1563*	1574*	1576*	1587*	1589*	1600*	1602*	1615*	
	1617*	1632*	1634*	1645*	1647*	1658*	1660*	1671*	1673*	1686*	1688*	1703*	1706*	
	1708*	1715*	1729*	1732*	1739*	1751*	1757*	1764*	1779*	1782*	1785*	1792*	1808*	
	1810*	1813*	1817*	1833*	1837*	1849*	1853*	1865*	1869*	1881*	1885*	1896*	1898*	
	1910*	1912*	1924*	1926*	1938*	1940*	1952*	1954*	1965*	1967*	1979*	1981*	1993*	
	1995*	2007*	2012*	2016*	2022*	2026*	2030*	2037*	2050*	2052*	2055*	2058*	2063*	
	2079*	2082*	2085*	2088*	2093*	2105*	2107*	2111*	2114*	2120*	2133*	2136*	2148*	

SW8 =	000400	122#							
SW9 =	001000	121#							
TBITVE =	000014	163#							
TEMP	001660	523#	1006*	1013*	1028*	1029	1055*	1804*	1819*
TIME	023304	3749*	3752	3757*	3761#	3763*			
TKVEC =	000060	170#							
TPVEC =	000064	171#							
TRAPVE =	000034	169#	562*	563*					
TRTVEC =	000014	164#							
TST1	002736	689#							
TST10	003364	793	799#						
TST100	011326	1802	1818	1828#					
TST101	011410	1838	1844#						
TST102	011472	1854	1860#						
TST103	011554	1870	1876#						
TST104	011636	1886	1892#						
TST105	011700	1899	1905#						
TST106	011742	1913	1919#						
TST107	012004	1927	1933#						
TST11	003426	806	812#						
TST110	012046	1941	1947#						
TST111	012110	1955	1960#						
TST112	012152	1968	1974#						
TST113	012214	1982	1988#						
TST114	012256	1996	2004#						
TST115	012450	2039	2046#						
TST116	012572	2064	2066	2074#					
TST117	012720	2094	2100#						
TST12	003470	819	826#						
TST120	013056	2121	2127#						
TST121	013134	2137	2143#						
TST122	013206	2152	2158#						
TST123	013276	2171	2176#						
TST124	013374	2191	2196#						
TST125	013570	2227	2245#						
TST13	003532	833	839#						
TST14	003574	846	852#						
TST15	003650	870#							
TST16	003722	879	886#						
TST17	003764	893	899#						
TST2	003020	704	713#						
TST20	004026	906	912#						
TST21	004070	919	925#						
TST22	004132	932	938#						
TST23	004174	945	951#						
TST24	004236	958	964#						
TST25	004300	971	977#						
TST26	004342	984	989#						
TST27	004404	996	1001#						
TST3	003062	720	726#						
TST30	004464	1018#							
TST31	004704	1043	1061#						
TST32	005062	1085	1090#						
TST33	005266	1130#							

TST34	005466	1168#																		
TST35	005662	1196	1204#																	
TST36	006056	1232	1240#																	
TST37	006140	1250	1256#																	
TST4	003124	733	739#																	
TST40	006222	1266	1272#																	
TST41	006304	1282	1288#																	
TST42	006404	1301	1308#																	
TST43	006506	1322	1329#																	
TST44	006610	1343	1350#																	
TST45	006662	1358	1365#																	
TST46	006734	1373	1379#																	
TST47	007006	1387	1393#																	
TST5	003166	746	752#																	
TST50	007110	1408	1416#																	
TST51	007212	1431	1438#																	
TST52	007250	1444	1450#																	
TST53	007366	1467	1473#																	
TST54	007436	1481	1487#																	
TST55	007506	1495	1501#																	
TST56	007556	1509	1515#																	
TST57	007626	1523	1528#																	
TST6	003230	759	765#																	
TST60	007676	1536	1541#																	
TST61	007762	1557#																		
TST62	010024	1564	1570#																	
TST63	010066	1577	1583#																	
TST64	010130	1590	1596#																	
TST65	010172	1603	1609#																	
TST66	010254	1628#																		
TST67	010316	1635	1641#																	
TST7	003304	783#																		
TST70	010360	1648	1654#																	
TST71	010422	1661	1667#																	
TST72	010464	1674	1680#																	
TST73	010546	1699#																		
TST74	010656	1716	1722#																	
TST75	010752	1733	1744#																	
TST76	011064	1758	1765	1772#																
TST77	011206	1786	1793	1800#																
TYPDS =	104405	2281	3261#																	
TYPE =	104401	633	643	661	2279	2282	2381	2711	2757	2758	2761	2774	2785	2804						
		2853	2856	2860	2922	2939	2941	2944	2946	2950	2957	2996	3073	3133						
		3257#	3304	3358	3362	3366	3594	3780	3787	3804										
		2760	2930	2954	3258#	3799														
		3260#																		
		659	3259#																	
TYPOC =	104402																			
TYPON =	104404																			
TYPOS =	104403																			
UPCNT	014002	1312	1333	1354	1369	1383	1397	1420	2114	2303#										
UPCNTA	014062	2313#	2320																	
UPCNTB	014116	2317	2321#																	
VCBRL	001606	491#																		
VCIV	001640	514#																		
VCIVS	001642	515#																		
VCSTAT	001620	501#	1443	1454	1457	1463	1466	1477	1480	1491	1494	1505	1508	1519						

	1857#	1873#	1889#	1902#	1916#	1930#	1944#	1957#	1971#	1985#	2001#	2043#	2071#
\$OCNT 020526	2097#	2124#	2140#	2155#	2173#	2193#	2242#						
\$OFs = 023102	3045#	3074#	3087#										
\$OMoDE 020530	3673#	3849	3850	3856									
\$OUTLP 022742	3040#	3044#	3049	3052*	3063*	3089#							
	696	717	730	743	756	771	789	803	816	830	843	858	875
	890	903	916	929	942	955	968	981	993	1005	1023	1025	1028
	1050	1055	1065	1068	1071	1077	1082	1095	1097	1101	1117	1122	1135
	1137	1140	1156	1161	1173	1175	1179	1186	1191	1209	1211	1215	1222
	1227	1245	1261	1277	1293	1296	1454	1463	1477	1491	1505	1519	1532
	1545	1554	1561	1574	1587	1600	1615	1632	1645	1658	1671	1686	1703
	1706	1729	1751	1779	1782	1808	1813	1833	1849	1865	1881	1896	1910
	1924	1938	1952	1965	1979	1993	2012	2050	2055	2079	2088	2105	2107
	2114	2133	2148	2163	2181	2208	2306	2309	2312	2404	2406	2431	2434
	2442	2444	3615#	3811									
\$OVER 016650	2639	2655	2663	2673	2682#								
\$PASS 001204	289#	588#	2271*	2272*	2280	2293	2669	2686					
\$PASTM 001006	231#												
\$PUTS 023512	3845#												
\$PWRAD 020224	2999#												
\$PWRDN 020064	564	2966#	2994										
\$PWARMG 020220	2997#												
\$PWRUP 020136	2976	2982#											
\$QUES 001172	277#	2736	2804	2853	2869	3169							
\$ROCHR 017334	2817#	3266											
\$RDOEC = ***** U	3269												
\$RDLIN 017454	2845#	3267											
\$RDOCT 017626	2885#	3268											
\$RDSZ = 000010	2838#												
\$REGAD 001160	271#												
\$REGO 001162	273#												
\$REG1 001164	274#												
\$RESET 023174	789	875	1246	1262	1278	1297	1440	1834	1850	1866	1882	2133	2149
	2168	2186	2197	3717#	3727								
\$RTNAD 013760	2292#												
\$R2A = ***** U	3269												
\$SAD 023170	3695	3697*	3705#										
\$SAVRE = ***** U	3269												
\$SAVR6 020234	2975#	2983	2984*	2985*	3003#								
\$SCOPE 016404	558	2635#											
\$SETUP = 000137	442#	557	558	560	562	564	566	567	568	570	641	644	2269
	2636	2700	2723	2731	2741	2875							
\$STUP = 177777	442#												
\$SVLAD 016614	2647	2676#											
\$SVPC = 000210	205#	210											
\$SWR = 165400	30#	63	180	181	182	193	184	185	186	187	275	276	277
	567	568	570	571	690	714	727	740	753	766	784	800	813
	827	840	853	871	887	900	913	926	939	952	965	978	990
	1002	1019	1062	1091	1131	1169	1205	1241	1257	1273	1289	1309	1330
	1351	1366	1380	1394	1417	1439	1451	1474	1488	1502	1516	1529	1542
	1558	1571	1584	1597	1610	1629	1642	1655	1668	1681	1700	1723	1745
	1773	1801	1829	1845	1861	1877	1893	1906	1920	1934	1948	1961	1975
	1989	2005	2047	2075	2101	2128	2144	2159	2177	2197	2246	2264	2270
	2285	2291	2293	2627	2628	2629	2630	2631	2638	2650	2652	2653	2656

COMMEN	173#														
ENDCOM	173#														
ERROR	67#	706	721	734	747	760	775	794	807	820	834	847	862	880	894
	907	920	933	946	959	972	985	997	1012	1034	1042	1086	1107	1146	1197
	1233	1251	1267	1283	1302	1317	1323	1338	1344	1359	1374	1388	1402	1409	1425
	1432	1445	1459	1468	1482	1496	1510	1524	1537	1550	1565	1578	1591	1604	1619
	1636	1649	1662	1675	1690	1710	1717	1739	1766	1794	1823	1839	1855	1871	1887
	1900	1914	1928	1942	1956	1969	1983	1997	2022	2032	2040	2067	2095	2122	2138
	2153	2172	2192	2222	3330	3336	3349	3390	3417	3531	3548				
ESCAPE	173#														
GETPRI	173#														
GETSWR	173#	644#													
MOVEI	20#	598	717	730	743	756	771	790	803	816	830	843	858	876	890
	903	916	929	942	955	968	981	993	1007	1030	1036	1044	1050	1072	1077
	1082	1102	1111	1117	1141	1150	1156	1180	1186	1193	1216	1222	1229	1247	1263
	1279	1298	1313	1319	1334	1340	1355	1370	1384	1398	1405	1421	1428	1441	1455
	1464	1478	1492	1506	1520	1533	1546	1561	1574	1587	1600	1615	1632	1645	1658
	1671	1686	1706	1713	1729	1737	1754	1762	1782	1790	1808	1814	1835	1851	1867
	1883	1896	1910	1924	1938	1952	1965	1979	1993	2005	2013	2020	2023	2028	2034
	2050	2055	2061	2080	2083	2091	2109	2118	2134	2150	2164	2169	2181	2187	2217
	2313	2436	3718												
MOVEM	19#	693	715	728	741	754	768	787	801	814	828	841	855	873	888
	901	914	927	940	953	966	979	991	1003	1021	1023	1026	1048	1053	1063
	1066	1069	1075	1080	1093	1095	1099	1115	1120	1133	1135	1138	1154	1159	1171
	1173	1177	1184	1189	1207	1209	1213	1220	1225	1243	1259	1275	1291	1294	1452
	1461	1475	1489	1503	1517	1530	1543	1552	1559	1572	1585	1598	1612	1630	1643
	1656	1669	1683	1701	1704	1727	1749	1777	1780	1806	1811	1831	1847	1863	1879
	1894	1908	1922	1936	1950	1963	1977	1991	2010	2048	2053	2077	2086	2103	2105
	2112	2131	2146	2161	2179	2206	2304	2307	2310	2402	2404	2429	2432	2440	2442
MULT	173#														
NEWTST	173#	686	710	723	736	749	762	780	796	809	823	836	849	867	883
	896	909	922	935	948	961	974	986	998	1015	1058	1087	1127	1165	1201
	1237	1253	1269	1285	1305	1326	1347	1362	1376	1390	1413	1435	1447	1470	1484
	1498	1512	1525	1538	1554	1567	1580	1593	1606	1625	1638	1651	1664	1677	1696
	1719	1741	1769	1797	1825	1841	1857	1873	1889	1902	1916	1930	1944	1957	1971
	1985	2001	2043	2071	2097	2124	2140	2155	2173	2193	2242				
POP	173#	2376	2908	2987	2988	3215	3216								
PUSH	173#	2335	2887	2968	2974	3176	3178	3199							
REPORT	173#														
SCOPE	68#	689	713	726	739	752	765	783	799	812	826	839	852	870	886
	899	912	925	938	951	964	977	989	1001	1018	1061	1090	1130	1168	1204
	1240	1256	1272	1288	1308	1329	1350	1365	1379	1393	1416	1438	1450	1473	1487
	1501	1515	1528	1541	1557	1570	1583	1596	1609	1628	1641	1654	1667	1680	1699
	1722	1744	1772	1800	1828	1844	1860	1876	1892	1905	1919	1933	1947	1960	1974
	1988	2004	2046	2074	2100	2127	2143	2158	2176	2196	2245	2268			
SETPRI	173#														
SETTRA	3249#	3258	3259	3260	3261	3263	3265	3266	3267	3268					
SETUP	173#	550													
SKIP	173#	704	720	733	746	759	774	793	806	819	833	846	861	879	893
	906	919	932	945	958	971	984	996	1011	1033	1040	1043	1085	1106	1145
	1196	1232	1250	1266	1282	1301	1316	1322	1337	1343	1358	1373	1387	1401	1408
	1424	1431	1444	1458	1467	1481	1495	1509	1523	1536	1549	1564	1577	1590	1603
	1618	1635	1648	1661	1674	1689	1709	1716	1733	1758	1760	1765	1786	1788	1793
	1802	1818	1820	1822	1838	1854	1870	1886	1899	1913	1927	1941	1955	1968	1982

.RESET	22#	3707
.SETUP	30#	442
.SWRHI	30#	176
.SWRLO	188#	
.UTK	29#	3767
.\$ACT1	30#	201
.\$APT8	30#	281#
.\$APTH	30#	212
.\$APTY	30#	3169
.\$CATC	30#	188
.\$CMTA	30#	234
.\$EOP	30#	2258
.\$ERRC	30#	2686
.\$ERRT	30#	2914
.\$INLP	28#	3642
.\$MMAC	18#	
.\$OUTL	27#	3603
.\$PARM	30#	
.\$POWE	30#	2962
.\$RDOC	30#	2875
.\$READ	30#	2736
.\$SAVE	30#	
.\$SCOP	30#	2621
.\$SPAC	30#	
.\$SWDO	30#	
.\$TLKW	26#	3481
.\$TOUT	489#	3686
.\$TRAP	30#	3226
.\$TYPD	30#	2323
.\$TYPE	30#	3090
.\$TYPO	30#	3013

ADD	602	676	682	701	2199	2224	2248	2249	2319	2355	2775	2784	2903	2936	3041
ASL	3051	3126	3186	3198	3210	3702	3722	3852							
ASLB	2798	2799	2800	2896	2898	2900	2933	2934	2935	3238					
ASR	2360														
BCC	3193														
BEG	2361														
	590	627	630	642	648	700	720	733	746	759	774	793	806	819	833
	846	861	879	893	906	919	932	945	958	971	984	996	1033	1106	1145
	1196	1232	1250	1266	1282	1301	1316	1322	1337	1343	1358	1373	1387	1401	1408
	1424	1431	1444	1458	1467	1481	1495	1509	1523	1536	1549	1564	1577	1590	1603
	1618	1635	1648	1661	1674	1689	1709	1716	1765	1793	1802	1838	1854	1870	1886
	1899	1913	1927	1941	1955	1968	1982	1996	2039	2064	2066	2121	2137	2152	2171
	2191	2220	2247	2284	2653	2655	2657	2661	2670	2702	2725	2728	2755	2782	2797
	2895	2938	2943	2956	3068	3116	3129	3164	3180	3184	3204	3206	3328	3353	3374
	3402	3459	3579	3585	3621	3623	3663	3666	3696	3792					
BGE	2420	2673													
BGT	2275	2369	2794	2835	3075										
BHI	2659														
BIC	1052	1079	1119	1158	1188	1224	2272	2751	2768	2795	2822	2828	2836	2902	3065
BICB	3788														
BIS	1047	1074	1114	1153	1183	1219	1810	2085	2111	2363	2364	2802	3070	3071	3315
	3445	3447	3455	3490	3522	3750									
BISB	2925	3857	3858	3859											
BIT	1801	2038	2063	2638	2652	2660	2667	2708	2724	3327	3383	3540	3557	3584	
BITB	589	3115	3120	3152	3183										
BLOS	2848														
BLT	2352	2368	2792	2833	3076	3143									
BMI	1085	1733	1758	1786	1818	2017	2317	2359	3701						
BNE	546	555	579	601	606	625	640	646	650	670	673	679	685	698	778
	865	1014	1056	1123	1125	1162	1164	1192	1228	1623	1694	1735	1760	1788	1820
	1822	2019	2090	2117	2209	2226	2320	2357	2435	2639	2668	2709	2714	2732	2747
	2753	2773	2780	2787	2824	2830	2852	2858	2926	2948	2986	3066	3114	3121	3123
	3131	3139	3153	3160	3182	3188	3191	3208	3322	3325	3346	3348	3355	3357	3384
	3400	3427	3451	3457	3541	3558	3581	3587	3721	3753	3758	3758	3758	3758	
BPL	1011	1040	2031	2059	2094	2167	2185	2343	2373	2417	2721	2749	2765	2820	2826
	3064	3108	3157	3471	3748	3785									
BR	531	581	604	634	652	655	662	702	704	1043	2227	2251	2354	2371	2641
	2647	2650	2663	2666	2719	2776	2803	2805	2831	2854	2904	2931	2958	2978	3002
	3042	3057	3078	3110	3136	3146	3155	3162	3174	3196	3289	3305	3308	3342	3359
	3363	3367	3396	3423	3462	3474	3537	3554	3582	3588	3595	3600	3625	3640	3668
	3685	3699	3727	3781	3800	3805	3814								
CLR	530	547	553	567	568	588	596	612	621	691	767	854	1006	1020	1028
	1097	1101	1140	1551	1611	1682	1804	2037	2129	2148	2163	2189	2269	2270	2346
	2349	2665	2680	2762	2763	2892	2893	2924	2984	3055	3311	3320	3406	3443	3444
	3448	3452	3460	3577	3619	3637	3661	3682	3698	3751	3754	3778			
CLRB	2375	2664	2859	3135	3161	3212	3213	3214							
CMP	545	554	578	624	641	649	669	672	678	684	699	719	732	745	758
	773	805	818	832	845	860	892	905	918	931	944	957	970	983	995
	1032	1105	1144	1195	1231	1249	1265	1281	1300	1315	1321	1336	1342	1357	1372
	1386	1400	1407	1423	1430	1443	1457	1466	1480	1494	1508	1522	1535	1548	1563
	1576	1589	1602	1617	1622	1634	1647	1660	1673	1688	1693	1708	1715	1764	1792
	1837	1853	1869	1885	1898	1912	1926	1940	1954	1967	1981	1995	2065	2120	2190
	2219	2318	2367	2419	2424	2648	2672	2731	2746	2752	2772	2779	2791	2793	2823
	2829	2832	2834	2847	3309	3373	3450	3456	3458	3622	3665	3695			

CMPB	647	2654	2658	2713	2754	2786	2851	2857	3113	3128	3130	3138	3159	3163	3181
DEC	3345	3347	3352	3399	3426	3578	3791								
DEC	609	1122	1161	1191	1227	1734	1759	1787	1819	1821	2116	2208	2225	2250	2273
DEC	2434	2932	3763												
DEC	3063	3074	3142	3145											
EMT	67														
HALT	194	607	2722	2733	2977	3001	3109	3472	3599						
INC	543	603	639	1013	1621	1692	2018	2052	2089	2271	2353	2431	2671	2704	2801
	2985	3069	3077	3211	3303	3321	3354	3356	3404	3449	3461	3580	3586	3624	3667
	3700														
INCB	777	864	1055	1124	1163	2676	2701	3165	3324	3375	3470	3757			
IOT	68														
JMP	198	199	628	631	707	2255	2257	2291							
JSR	600	696	717	719	730	732	743	745	756	758	771	773	789	792	803
	805	816	818	830	832	843	845	858	860	875	878	890	892	903	905
	916	918	929	931	942	944	955	957	968	970	981	983	993	995	1005
	1010	1023	1025	1028	1032	1039	1047	1050	1052	1055	1065	1068	1071	1074	1077
	1079	1082	1084	1095	1097	1101	1105	1114	1117	1119	1122	1135	1137	1140	1144
	1153	1156	1158	1161	1173	1175	1179	1183	1186	1188	1191	1195	1209	1211	1215
	1219	1222	1224	1227	1231	1245	1246	1249	1261	1262	1265	1277	1278	1281	1293
	1296	1297	1300	1312	1315	1321	1333	1336	1342	1354	1357	1369	1372	1383	1386
	1397	1400	1407	1420	1423	1430	1443	1454	1457	1463	1466	1477	1480	1491	1491
	1494	1505	1508	1519	1522	1532	1535	1545	1548	1554	1561	1563	1574	1576	1587
	1589	1600	1602	1615	1617	1632	1634	1645	1647	1658	1660	1671	1673	1686	1688
	1703	1706	1708	1715	1729	1732	1739	1751	1757	1764	1779	1782	1785	1792	1808
	1810	1813	1817	1833	1834	1837	1849	1850	1853	1865	1866	1869	1881	1882	1885
	1896	1898	1910	1912	1924	1926	1938	1940	1952	1954	1965	1967	1979	1981	1993
	1995	2007	2012	2016	2022	2026	2030	2037	2050	2052	2055	2058	2063	2079	2082
	2085	2088	2093	2105	2107	2111	2114	2120	2133	2136	2148	2149	2152	2163	2166
	2168	2171	2181	2184	2186	2189	2197	2208	2219	2286	2306	2309	2312	2316	2404
	2406	2410	2411	2413	2431	2434	2438	2442	2444	2710	2716	2790	3118	3137	3144
	3151	3200	3312	3334	3377	3379	3381	3388	3415	3491	3495	3500	3504	3523	3526
	3529	3543	3546	3560	3630	3638	3672	3683	3720	3794	3811	3846	3853		
MOV	532	538	539	540	541	544	548	549	552	556	558	559	560	561	562
	563	564	565	566	570	571	574	575	576	577	582	584	585	586	591
	592	594	595	597	610	611	613	614	615	618	619	620	622	658	667
	668	671	674	675	681	692	714	727	740	753	766	784	785	786	800
	813	827	840	853	871	872	887	900	913	926	939	952	965	978	990
	1002	1005	1019	1025	1029	1041	1062	1065	1068	1071	1091	1092	1098	1110	1131
	1132	1137	1149	1169	1170	1175	1176	1179	1205	1206	1211	1212	1215	1241	1242
	1245	1257	1258	1261	1273	1274	1277	1289	1290	1293	1296	1309	1310	1311	1318
	1330	1331	1332	1339	1351	1352	1353	1366	1367	1368	1380	1381	1382	1394	1395
	1396	1404	1417	1418	1419	1427	1439	1451	1454	1460	1463	1474	1477	1488	1491
	1502	1505	1516	1519	1529	1532	1542	1545	1558	1571	1584	1597	1610	1629	1642
	1655	1668	1681	1700	1703	1712	1724	1725	1726	1736	1746	1747	1748	1752	1761
	1774	1775	1776	1779	1789	1803	1805	1829	1830	1833	1845	1846	1849	1861	1862
	1865	1877	1878	1881	1893	1907	1921	1935	1949	1962	1976	1990	2007	2008	2009
	2012	2026	2027	2047	2060	2075	2076	2079	2082	2088	2101	2102	2107	2108	2115
	2128	2130	2144	2145	2159	2160	2177	2178	2198	2200	2201	2202	2205	2211	2212
	2213	2215	2216	2221	2232	2253	2254	2256	2276	2280	2283	2303	2306	2309	2312
	2336	2337	2338	2339	2340	2341	2342	2347	2350	2370	2376	2377	2378	2379	2380
	2382	2383	2401	2406	2407	2408	2409	2412	2414	2415	2421	2427	2428	2438	2439
	2643	2644	2646	2649	2662	2674	2675	2678	2679	2682	2683	2703	2705	2726	2729
	2759	2783	2788	2817	2818	2845	2846	2861	2862	2863	2864	2885	2886	2887	2888

dRLPC.P11

CROSS REFERENCE TABLE

SEQ 0103

MOV B	2889	2891	2906	2907	2908	2909	2910	2923	2928	2937	2942	2947	2949	2953	2966
	2967	2968	2969	2970	2971	2972	2973	2974	2975	2976	2982	2983	2987	2988	2989
	2990	2991	2992	2993	2994	2995	2998	3038	3046	3047	3048	3054	3061	3079	3080
	3081	3082	3083	3111	3112	3117	3125	3140	3177	3178	3185	3189	3194	3195	3197
	3199	3209	3215	3216	3234	3235	3239	3245	3246	3287	3302	3310	3319	3323	3332
	3372	3407	3440	3441	3442	3446	3453	3464	3465	3488	3489	3492	3493	3496	3505
	3520	3521	3527	3562	3576	3590	3615	3616	3618	3627	3629	3633	3634	3636	3657
	3658	3660	3670	3676	3677	3678	3681	3697	3749	3790	3798	3809	3845	3851	
	569	653	2345	2348	2362	2365	2374	2677	2681	2707	2715	2750	2767	2821	2827
	2850	2855	2894	3039	3040	3043	3044	3045	3049	3052	3053	3072	3122	3150	3158
	3172	3173	3175	3237	3371	3376	3378	3380	3385	3413	3494	3497	3499	3501	3502
	3503	3524	3525	3542	3544	3559	3561	3786	3849	3850	3856				
NEG	2344	2418	3050												
NOP	616	666	2287	2288	2289										
RESET	2285	3717													
ROL	2897	2899	2901	3056	3058	3059	3060	3062							
RTI	583	2384	2684	2735	2789	2837	2865	2911	3000	3084	3127	3247	3764		
r TS	2321	2422	2425	2444	2951	3167	2865	2911	3000	3084	3127	3247	3764	3591	3635
	3679	3703	3729	3756	3759	3860	3217	3240	3408	3428	3466	3506	3563		
SUB	2203	2204	2351	2416	2706	3192									
TRAP	3249	3258	3259	3260	3261	3263	3265	3266	3267	3268					
TST	600	605	623	626	629	645	677	680	683	697	792	878	1010	2136	2223
	2246	2356	2366	2423	2645	2669	2720	2727	2781	2796	2905	2955	3067	3124	3132
	3154	3187	3205	3207	3236	3469	3620	3628	3662	3671	3720	3747	3752		
TSTB	1039	1084	1732	1757	1785	1817	2016	2030	2058	2093	2166	2184	2316	2358	2372
	2656	2748	2764	2819	2825	3107	3156	3179	3190	3203	3401	3784			
.ASCII	277	278													
.ASCIIZ	279	636	657	664	2294	2450	2456	2460	2466	2472	2476	2482	2486	2492	2497
	2501	2505	2509	2513	2518	2523	2529	2532	2538	2544	2547	2553	2559	2565	2571
	2577	2580	2584	2590	2869	2870	2871	2873	2875	2879	3004	3307	3361	3365	3597
	3783	3807	3818												
.ASECT	14														
.BLKB	2868														
.BLKW	487	2389													
.BYTE	243	244	249	250	258	259	267	268	269	270	295	296	306	307	314
	315	317	318	320	321	2293	2717	2718	2866	2867	3085	3086	3087	3088	3218
	3219	3220													
.DSABL	2806														
.ENABL	30	2739													
.END	3863														
.ENDC	58	67	159	173	184	186	187	188	199	204	208	210	215	217	224
	237	241	243	271	275	276	277	281	284	306	314	317	320	323	324
	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339
	340	341	342	343	344	345	349	442	484	550	556	557	560	562	564
	566	567	568	570	572	593	636	641	643	649	655	657	664	687	688
	689	690	705	711	712	713	714	721	724	725	726	727	734	737	738
	739	740	747	750	751	752	753	760	763	764	765	766	775	781	782
	783	784	785	794	797	798	799	800	807	810	811	812	813	820	824
	825	826	827	834	837	838	839	840	847	850	851	852	853	862	868
	869	870	871	880	884	885	886	887	894	897	898	899	900	907	910
	911	912	913	920	923	924	925	926	933	936	937	938	939	946	949
	950	951	952	953	962	963	964	965	972	975	976	977	978	985	987
	988	989	990	997	999	1000	1001	1002	1012	1016	1017	1018	1019	1020	1034
	1041	1044	1059	1060	1061	1062	1086	1088	1089	1090	1091	1092	1107	1128	1129

1130	1131	1132	1146	1166	1167	1168	1169	1170	1197	1202	1203	1204	1205	1206	
1233	1238	1239	1240	1241	1242	1251	1254	1255	1256	1257	1258	1267	1270	1271	
1272	1273	1274	1283	1286	1287	1288	1289	1290	1302	1306	1307	1308	1309	1310	
1317	1323	1327	1328	1329	1330	1331	1338	1344	1348	1349	1350	1351	1352	1359	
1363	1364	1365	1366	1367	1374	1377	1378	1379	1380	1381	1388	1391	1392	1393	
1394	1395	1402	1409	1414	1415	1416	1417	1418	1425	1432	1436	1437	1438	1439	
1445	1448	1449	1450	1451	1459	1468	1471	1472	1473	1474	1482	1485	1486	1487	
1488	1496	1499	1500	1501	1502	1510	1513	1514	1515	1516	1524	1526	1527	1528	
1529	1537	1539	1540	1541	1542	1550	1555	1556	1557	1558	1565	1568	1569	1570	
1571	1578	1581	1582	1583	1584	1591	1594	1595	1596	1597	1604	1607	1608	1609	
1610	1613	1626	1627	1628	1629	1636	1639	1640	1641	1642	1649	1652	1653	1654	
1655	1662	1665	1666	1667	1668	1675	1678	1679	1680	1681	1690	1697	1698	1699	
1700	1710	1717	1720	1721	1722	1723	1734	1742	1743	1744	1745	1759	1761	1766	
1770	1771	1772	1773	1787	1789	1794	1798	1799	1800	1801	1803	1819	1821	1823	
1826	1827	1828	1829	1830	1839	1842	1843	1844	1845	1846	1855	1858	1859	1860	
1861	1862	1871	1874	1875	1876	1877	1878	1887	1890	1891	1892	1893	1900	1903	
1904	1905	1906	1914	1917	1918	1919	1920	1928	1931	1932	1933	1934	1942	1945	
1946	1947	1948	1956	1958	1959	1960	1961	1969	1972	1973	1974	1975	1983	1986	
1987	1988	1989	1997	2002	2003	2004	2005	2032	2040	2044	2045	2046	2047	2065	
2067	2072	2073	2074	2075	2076	2095	2098	2099	2100	2101	2102	2122	2125	2126	
2127	2128	2129	2138	2141	2142	2143	2144	2145	2153	2156	2157	2158	2159	2160	
2172	2174	2175	2176	2177	2178	2192	2194	2195	2196	2197	2221	2238	2243	2244	
2245	2246	2261	2263	2264	2266	2269	2275	2278	2279	2283	2285	2291	2293	2294	
2297	2326	2624	2627	2632	2638	2640	2651	2654	2655	2656	2658	2660	2667	2671	
2676	2678	2682	2685	2686	2689	2692	2701	2705	2710	2711	2712	2720	2731	2735	
2736	2739	2740	2742	2770	2806	2810	2838	2839	2846	2848	2851	2853	2869	2875	
2878	2880	2913	2917	2932	2961	2965	2974	2975	2981	2987	2988	2998	3000	3004	
3016	3093	3122	3172	3173	3176	3203	3218	3229	3235	3238	3257	3258	3259	3260	
3261	3262	3263	3264	3265	3266	3267	3268	3269	3307	3337	3344	3361	3365	3369	
3391	3398	3418	3425	3532	3539	3549	3556	3597	3767	3783	3807				
.EQUIV	67	68	76	121	122	123	124	125	126	127	128	129	130	149	150
.EVEN	151	152	153	154	155	156	157	158							
	284	636	657	664	2596	2960	3011	3221	3307	3361	3365	3369	3597	3783	3807
3821															
.GLOBL	14														
.IF	54	65	131	159	183	185	186	187	188	197	203	206	208	214	216
	223	236	240	242	271	275	276	277	280	281	283	306	314	317	320
	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337
	338	339	340	341	342	343	344	345	349	442	481	550	551	556	558
	560	562	564	566	567	568	570	588	635	640	641	642	644	647	656
	663	686	688	690	704	710	712	714	720	723	725	727	733	736	738
	740	746	749	751	753	759	762	764	766	774	780	782	784	785	793
	796	798	800	806	809	811	813	819	823	825	827	833	836	838	840
	846	849	851	853	861	867	869	871	879	883	885	887	893	896	898
	900	906	909	911	913	919	922	924	926	932	935	937	939	945	948
	950	952	958	961	963	965	971	974	976	978	984	986	988	990	996
	998	1000	1002	1011	1015	1017	1019	1020	1033	1040	1043	1058	1060	1062	1085
	1087	1089	1091	1092	1106	1127	1129	1131	1132	1145	1165	1167	1169	1170	1196
	1201	1203	1205	1206	1232	1237	1239	1241	1242	1250	1253	1255	1257	1258	1266
	1269	1271	1273	1274	1282	1285	1287	1289	1290	1301	1305	1307	1309	1310	1316
	1322	1326	1328	1330	1331	1337	1343	1347	1349	1351	1352	1358	1362	1364	1366
	1367	1373	1376	1378	1380	1381	1387	1390	1392	1394	1395	1401	1408	1413	1415
	1417	1418	1424	1431	1435	1437	1439	1444	1447	1449	1451	1458	1467	1470	1472
	1474	1481	1484	1486	1488	1495	1498	1500	1502	1509	1512	1514	1516	1523	1525

1527	1529	1536	1538	1540	1542	1549	1554	1556	1558	1564	1567	1569	1571	1577
1580	1582	1584	1590	1593	1595	1597	1603	1606	1608	1610	1618	1625	1627	1629
1635	1638	1640	1642	1648	1651	1653	1655	1661	1664	1666	1668	1674	1677	1679
1681	1689	1696	1698	1700	1709	1716	1719	1721	1723	1733	1741	1743	1745	1758
1760	1765	1769	1771	1773	1786	1788	1793	1797	1799	1801	1802	1818	1820	1822
1825	1827	1829	1830	1838	1841	1843	1845	1846	1854	1857	1859	1861	1862	1870
1873	1875	1877	1878	1886	1889	1851	1893	1899	1902	1904	1906	1913	1916	1918
1920	1927	1930	1932	1934	1941	1944	1946	1948	1955	1957	1959	1961	1968	1971
1973	1975	1982	1985	1987	1989	1996	2001	2003	2005	2031	2039	2043	2045	2047
2064	2066	2071	2073	2075	2076	2094	2097	2099	2101	2102	2121	2124	2126	2128
2129	2137	2140	2142	2144	2145	2152	2155	2157	2159	2160	2171	2173	2175	2177
2178	2191	2193	2195	2197	2220	2227	2242	2244	2246	2260	2261	2262	2263	2264
2265	2266	2268	2274	2277	2279	2283	2285	2291	2293	2294	2325	2327	2328	2331
22637	22638	22650	22652	22653	22654	22656	22657	22658	22667	22669	22677	22679	22684	22685
22686	22688	22691	2701	2704	2708	2710	2711	2713	2720	2724	2731	2735	2736	2738
22740	22741	22742	2770	2809	2810	2838	2846	2847	2851	2852	2868	2869	2875	2877
22880	22892	22916	22931	22947	22964	22974	22975	22980	22987	22988	22996	22998	3000	3004
3015	3092	3113	3171	3173	3176	3203	3218	3228	3234	3238	3249	3258	3259	3260
3261	3262	3263	3265	3266	3267	3268	3269	3306	3336	3342	3360	3364	3368	3390
3396	3417	3423	3531	3537	3548	3554	3596	3767	3782	3806				
65	183	185	187	188	204	208	210	215	217	224	237	240	243	271
281	284	556	640	642	687	688	689	690	705	711	712	713	714	721
724	725	726	727	734	737	738	739	740	747	750	751	752	753	760
763	764	765	766	774	781	782	783	784	794	797	798	799	800	807
810	811	812	813	820	824	825	826	827	834	837	838	839	840	847
850	851	852	853	861	868	869	870	871	880	884	885	886	887	894
897	898	899	900	907	910	911	912	913	920	923	924	925	926	933
936	937	938	939	946	949	950	951	952	959	962	963	964	965	972
975	976	977	978	985	987	988	989	990	997	999				
1016	1017	1018	1019	1033	1040	1044	1059	1060	1061	1062	1086	1088	1089	1011
1091	1106	1128	1129	1130	1131	1145	1166	1167	1168	1169	1197	1202	1203	1090
1205	1233	1238	1239	1240	1241	1251	1254	1255	1256	1257	1267	1270	1271	1204
1273	1283	1286	1287	1288	1289	1302	1306	1307	1308	1309	1316	1323	1327	1272
1329	1330	1337	1344	1348	1349	1350	1351	1359	1363	1364	1365	1366	1374	1328
1378	1379	1380	1388	1391	1392	1393	1394	1401	1409	1414	1415	1416	1417	1377
1432	1436	1437	1438	1439	1445	1448	1449	1450	1451	1458	1468	1471	1472	1424
1474	1482	1485	1486	1487	1488	1496	1499	1500	1501	1502	1510	1513	1514	1473
1516	1524	1526	1527	1528	1529	1537	1539	1540	1541	1542	1549	1555	1556	1515
1558	1565	1568	1569	1570	1571	1578	1581	1582	1583	1584	1591	1594	1595	1557
1597	1604	1607	1608	1609	1610	1618	1626	1627	1628	1629	1636	1639	1640	1596
1642	1649	1652	1653	1654	1655	1662	1665	1666	1667	1668	1675	1678	1679	1641
1681	1689	1697	1698	1699	1700	1709	1717	1720	1721	1722	1723	1734	1742	1680
1744	1745	1759	1760	1766	1770	1771	1772	1773	1787	1788	1794	1798	1799	1743
1801	1803	1819	1820	1822	1826	1827	1828	1829	1839	1842	1843	1844	1845	1800
1858	1859	1860	1861	1871	1874	1875	1876	1877	1887	1890	1891	1892	1893	1855
1903	1904	1905	1906	1914	1917	1918	1919	1920	1928	1931	1932	1933	1934	1900
1945	1946	1947	1948	1956	1958	1959	1960	1961	1969	1972	1973	1974	1975	1942
1986	1987	1988	1989	1997	2002	2003	2004	2005	2031	2040	2044	2045	2046	1983
2065	2067	2072	2073	2074	2075	2095	2098	2099	2100	2101	2122	2125	2126	2047
2128	2138	2141	2142	2143	2144	2153	2156	2157	2158	2159	2172	2174	2175	2127
2177	2192	2194	2195	2196	2197	2220	2228	2243	2244	2245	2246	2261	2265	2176
2274	2277	2293	2326	2624	2651	2654	2655	2658	2685	2689	2691	2705	2731	2269
2739	2742	2810	2812	2817	2838	2839	2848	2852	2869	2878	2917	2932	2961	2736
2981	2996	3016	3093	3172	3229	3235	3336	3342	3390	3396	3417	3423	3531	2965

.IFF

.IFT	3548	3554	664	2666	2711	2812	2817	2896	2912	2913	3307	3361	3365	3369	3597
	636	657													
.IFTF	3783	3807	664	2664	2710	2757	2810	2813	2892	2896	2912	3307	3361	3365	3369
	636	657													
.IIF	3597	3783	3807												
	53	58	63	180	181	182	184	187	188	194	280	284	557	560	566
	567	568	570	571	641	2263	2269	2270	2281	2293	2297	2627	2628	2629	2630
	2631	2632	2636	2665	2666	2682	2685	2686	2692	2693	2694	2695	2700	2723	2731
	2736	2739	2760	2861	2869	2875	2929	2954	3169	3257	3258	3259	3260	3261	3263
.IRP	3265	3266	3267	3268	3799										
	442	686	710	723	736	749	762	780	796	809	823	836	849	867	883
	896	909	922	935	948	961	974	986	998	1015	1058	1087	1127	1165	1201
	1237	1253	1269	1285	1305	1326	1347	1362	1376	1390	1413	1435	1447	1470	1484
	1498	1512	1525	1538	1554	1567	1580	1593	1606	1625	1638	1651	1664	1677	1696
	1719	1741	1769	1797	1825	1841	1857	1873	1889	1902	1916	1930	1944	1957	1971
	1985	2001	2043	2071	2097	2124	2140	2155	2173	2193	2242	2336	2376	2637	2887
	2908	2968	2974	2987	2988	3177	3178	3199	3215	3216					
.LIST	14	30	173	187	194	271	273	274	275	281	284	442	572	600	636
	641	644	657	664	686	690	696	710	714	717	719	723	727	730	732
	736	740	743	745	749	753	756	758	762	766	771	773	780	784	789
	792	796	800	803	805	809	813	816	818	823	827	830	832	836	840
	843	845	849	853	858	860	867	871	875	878	883	887	890	892	896
	900	903	905	909	913	916	918	922	926	929	931	935	939	942	944
	948	952	955	957	961	965	968	970	974	978	981	983	986	990	993
	995	998	1002	1005	1010	1015	1019	1023	1025	1028	1032	1039	1047	1050	1052
	1055	1058	1062	1065	1068	1071	1074	1077	1079	1082	1084	1087	1091	1095	1097
	1101	1105	1114	1117	1119	1122	1127	1131	1135	1137	1140	1144	1153	1156	1158
	1161	1165	1169	1173	1175	1179	1183	1186	1188	1191	1195	1201	1205	1209	1211
	1215	1219	1222	1224	1227	1231	1237	1241	1245	1249	1253	1257	1261	1265	1269
	1273	1277	1281	1285	1289	1293	1296	1300	1305	1309	1315	1321	1326	1330	1336
	1342	1347	1351	1357	1362	1366	1372	1376	1380	1386	1390	1394	1400	1407	1413
	1417	1423	1430	1435	1439	1443	1447	1451	1454	1457	1463	1466	1470	1474	1477
	1480	1484	1488	1491	1494	1498	1502	1505	1508	1512	1516	1519	1522	1525	1529
	1532	1535	1538	1542	1545	1548	1554	1558	1561	1563	1567	1571	1574	1576	1580
	1584	1587	1589	1593	1597	1600	1602	1606	1610	1615	1617	1625	1629	1632	1634
	1638	1642	1645	1647	1651	1655	1658	1660	1664	1668	1671	1673	1677	1681	1686
	1688	1696	1700	1703	1706	1708	1715	1719	1723	1729	1732	1739	1741	1745	1751
	1757	1764	1769	1773	1779	1782	1785	1792	1797	1801	1808	1810	1813	1817	1825
	1829	1833	1837	1841	1845	1849	1853	1857	1861	1865	1869	1873	1877	1881	1885
	1889	1893	1896	1898	1902	1906	1910	1912	1916	1920	1924	1926	1930	1934	1938
	1940	1944	1948	1952	1954	1957	1961	1965	1967	1971	1975	1979	1981	1985	1989
	1993	1995	2001	2005	2007	2012	2016	2022	2026	2030	2037	2043	2047	2050	2052
	2055	2058	2063	2071	2075	2079	2082	2085	2088	2093	2097	2101	2105	2107	2111
	2114	2120	2124	2128	2133	2136	2140	2144	2148	2152	2155	2159	2163	2166	2171
	2173	2177	2181	2184	2189	2193	2197	2208	2219	2242	2246	2269	2285	2306	2309
	2312	2316	2404	2406	2431	2434	2438	2442	2444	2631	2731	2838	3249	3257	3258
	3259	3260	3261	3262	3263	3264	3265	3266	3267	3268	3269	3307	3361	3365	3369
	3597	3720	3783	3807											
.MACRO	17	18	19	20	22	24	25	26	27	28	29	30	188	234	489
.MCALL	588	3249													
.NLIST	30	173	281	572	644										
	14	30	173	187	194	271	273	274	275	281	284	442	572	600	636
	641	644	657	664	686	690	696	710	714	717	719	723	727	730	732
	736	740	743	745	749	753	756	758	762	766	771	773	780	784	789

	792	796	800	803	805	809	813	816	818	823	827	830	832	836	840
	843	845	849	853	858	860	867	871	875	878	883	887	890	892	896
	900	903	905	909	913	916	918	922	926	929	931	935	939	942	944
	948	952	955	957	961	965	968	970	974	978	981	983	986	990	993
	995	998	1002	1005	1010	1015	1019	1023	1025	1028	1032	1039	1047	1050	1052
	1055	1058	1062	1065	1068	1071	1074	1077	1079	1082	1084	1087	1091	1095	1097
	1101	1105	1114	1117	1119	1122	1127	1131	1135	1137	1140	1144	1153	1156	1158
	1161	1165	1169	1173	1175	1179	1183	1186	1188	1191	1195	1201	1205	1209	1211
	1215	1219	1222	1224	1227	1231	1237	1241	1245	1249	1253	1257	1261	1265	1269
	1273	1277	1281	1285	1289	1293	1296	1300	1305	1309	1315	1321	1326	1330	1336
	1342	1347	1351	1357	1362	1366	1372	1376	1380	1386	1390	1394	1400	1407	1413
	1417	1423	1430	1435	1439	1443	1447	1451	1454	1457	1463	1466	1470	1474	1477
	1480	1484	1488	1491	1494	1498	1502	1505	1508	1512	1516	1519	1522	1525	1529
	1532	1535	1538	1542	1545	1548	1554	1558	1561	1563	1567	1571	1574	1576	1580
	1584	1587	1589	1593	1597	1600	1602	1606	1610	1615	1617	1625	1629	1632	1634
	1638	1642	1645	1647	1651	1655	1658	1660	1664	1668	1671	1673	1677	1681	1686
	1688	1696	1700	1703	1706	1708	1715	1719	1723	1729	1732	1739	1741	1745	1751
	1757	1764	1769	1773	1779	1782	1785	1792	1797	1801	1808	1810	1813	1817	1825
	1829	1833	1837	1841	1845	1849	1853	1857	1861	1865	1869	1873	1877	1881	1885
	1889	1893	1896	1898	1902	1906	1910	1912	1916	1920	1924	1926	1930	1934	1938
	1940	1944	1948	1952	1954	1957	1961	1965	1967	1971	1975	1979	1981	1985	1989
	1993	1995	2001	2005	2007	2012	2016	2022	2026	2030	2037	2043	2047	2050	2052
	2055	2058	2063	2071	2075	2079	2082	2085	2088	2093	2097	2101	2105	2107	2111
	2114	2120	2124	2128	2133	2136	2140	2144	2148	2152	2155	2159	2163	2166	2171
	2173	2177	2181	2184	2189	2193	2197	2208	2219	2242	2246	2269	2285	2306	2309
	2312	2316	2404	2406	2431	2434	2438	2442	2444	2631	2731	2838	3249	3257	3258
	3259	3260	3261	3262	3263	3264	3265	3266	3267	3268	3269	3307	3361	3365	3369
	3377	3380	3383	3387											
.PAGE	234	349													
.PSECT	14		30												
.REM	1	14													
.REPT	194	273													
.SBTTL	63	176	188	197	201	212	224	281	349	550	637	644	686	710	723
	736	749	762	780	796	809	823	836	849	867	883	896	909	922	935
	948	961	974	986	998	1015	1058	1087	1127	1165	1201	1237	1253	1269	1285
	1305	1326	1347	1362	1376	1390	1413	1435	1447	1470	1484	1498	1512	1525	1538
	1554	1567	1580	1593	1606	1625	1638	1651	1664	1677	1696	1719	1741	1769	1797
	1825	1841	1857	1873	1889	1902	1916	1930	1944	1957	1971	1985	2001	2043	2071
	2097	2124	2140	2155	2173	2193	2242	2258	2323	2621	2686	2736	2875	2914	2962
	3013	3090	3169	3226	3249										
.TITLE	53														
.WORD	194	195	196	209	228	229	230	231	232	233	242	245	246	247	248
	251	252	253	254	255	256	257	260	261	262	271	273	274	286	287
	288	289	290	291	292	293	297	298	299	312	316	319	322	323	324
	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339
	340	341	342	343	344	464	467	469	471	473	475	477	479	481	482
	484	486	600	660	696	717	719	730	732	743	745	756	758	771	773
	789	792	803	805	816	818	830	832	843	845	858	860	875	878	890
	892	903	905	916	918	929	931	942	944	955	957	968	970	981	983
	993	995	1005	1010	1023	1025	1028	1032	1039	1047	1050	1052	1055	1065	1068
	1071	1074	1077	1079	1082	1084	1095	1097	1101	1105	1114	1117	1119	1122	1135
	1137	1140	1144	1153	1156	1158	1161	1173	1175	1179	1183	1186	1188	1191	1195
	1209	1211	1215	1219	1222	1224	1227	1231	1245	1249	1261	1265	1277	1281	1293
	1296	1300	1315	1321	1336	1342	1357	1372	1386	1400	1407	1423	1430	1443	1454

1457	1463	1466	1477	1480	1491	1494	1505	1508	1519	1522	1532	1535	1545	1548
1554	1561	1563	1574	1576	1587	1589	1600	1602	1615	1617	1632	1634	1645	1647
1658	1660	1671	1673	1686	1688	1703	1706	1708	1715	1729	1732	1739	1751	1757
1764	1779	1782	1785	1792	1808	1810	1813	1817	1833	1837	1849	1853	1865	1869
1881	1885	1896	1898	1910	1912	1924	1926	1938	1940	1952	1954	1965	1967	1979
1981	1993	1995	2007	2012	2016	2022	2026	2030	2037	2050	2052	2055	2058	2063
2079	2082	2085	2088	2093	2105	2107	2111	2114	2120	2133	2136	2148	2152	2163
2166	2171	2181	2184	2189	2208	2219	2274	2277	2292	2306	2309	2312	2316	2404
2406	2431	2434	2438	2442	2444	2912	2940	2945	2997	2999	3089	3119	3166	3201
3256	3313	3410	3430	3565	3631	3632	3639	3674	3684	3705	3706	3720	3730	3731
3761	3765	3795	3796	3801	3812	3813	3816	3817	3847	3848	3854	3855	3861	

000000

ERRORS DETECTED: 0

*DRLPC,DRLPC/SOL/CRF=DRLPA.MAC,DRLPC
RUN-TIME: 35 23 3 SECONDS
CORE USED: 40K