

LPA/AR11

DIAGNOSTIC TEST 2
MD-11-DRLPD-A

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

MAR 1978
digital
MADE IN USA

The image displays a grid of approximately 150 small, illegible data tables or charts arranged in 10 columns and 15 rows. Each cell in the grid contains a small table with multiple columns and rows of text, which appears to be diagnostic test data. The text is too small and faded to be read, but the layout suggests a comprehensive set of test results for various aircraft components.

801

EOF1DRDP08501

00010000

780223

IDENP0P10A710N

HDR1DRLPDASEQ

00010000

780223
SEQ 0001

PRODUCT CODE: MAINDEC-11-DRLPD-A-D
PRODUCT NAME: LPA/AR11 DIAGNOSTIC TEST II
DATE: JAN. 1978
MAINTAINER: DIAGNOSTIC GROUP

COPYRIGHT (C) 1978
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE
COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT
BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR
USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS.
TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

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

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

1. ABSTRACT

THIS DIAGNOSTIC EXERCISES THE "AR11" ANALOG CIRCUITRY. THE PROGRAM WHEN STARTED WILL TYPE OUT THE PROGRAM TITLE. A MESSAGE IS THEN PRINTED GIVING THE LETTER DESIGNATORS TO BE TYPED TO RUN ANY ONE OF THE FIVE (5) SEPERATE TESTS OF WHICH THIS PROGRAM IS COMPRISED. THE PROGRAM THEN TYPES A 'CR .' AND THEN WAITS IN A KEYBOARD MONITOR MODE FOR A LETTER TO BE TYPED. ALTHOUGH THESE TESTS MAY BE RUN IN ANY ORDER IT IS IMPERATIVE THAT TEST I IS RUN FIRST AND PROVED FULLY OPERATIONAL.

THE PROGRAM IS SET UP TO GIVE THE OPERATOR AS MUCH CONTROL OVER THE PROGRAM AS POSSIBLE VIA THE TELETYPE. TYPING A 'IC' (OBTAINED VIA TYPING THE 'CNTR' AND 'C' KEYS SIMULTANEOUSLY) WHILE RUNNING ANY TEST WILL ENABLE THE PROGRAM TO RETURN TO THE KEYBOARD MONITOR AND AWAIT A NEW LETTER DESIGNATOR TO BE TYPED. TYPING A 'IA' WHILE IN MONITOR MODE WILL ENABLE THE LETTER DESIGNATORS TO BE RETYPED.

UNLIKE AR11 TEST I, THIS PROGRAM DOES NOT DETERMINE IF ADDITIONAL AR11'S ARE CONNECTED. TO RUN ANOTHER AR11, THE OPERATOR MUST SUPPLY THE BUS ADDRESS INDIVIDUALLY.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZARB-B". IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE AR11 OPTION WHEN IT IS ON THE LPA11-KX I/O BUS. NO RECABLING IS NEEDED. SOME TEST DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TEST, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN "MD-11-DZARB-B". YOU SHOULD RUN "MD-11+DRLPA" BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 15.

2. REQUIREMENTS (EQUIPMENT)

- A. PDP-11 COMPUTER WITH 8K OF MEMORY
- B. TELETYPE
- C. AR11 HEX OPTION MODULE INSTALLED
- D. VOLTAGE STANDARD (E.D.C.)
- E. VR14 OR STORAGE SCOPE
- F. LPA11-KX SYSTEM

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING BINARY TAPES.

4. STARTING PROCEDURE

THE PROGRAM STARTING ADDRESS IS '200'.
THE RESTART ADDRESS IS '204'.

5. CONSOLE SWITCH SETTINGS

- THIS PROGRAM HAS BEEN MODIFIED TO RUN WITH OR WITHOUT A HARDWARE SWITCH REGISTER.
- A. ALL SWITCHES SHOULD BE DOWN (0) WHEN THE PROGRAM IS STARTED.
 - B. REFER TO THE INDIVIDUAL TEST DESCRIPTIONS FOR APPLICABLE CONSOLE SWITCH SETTINGS

WHILE IN KEYBOARD MONITOR MODE, TYPING 'S WILL ENABLE THE SOFTWARE SWITCH REGISTER TO BE LOADED FROM THE TELETYPE WITHOUT HALTING THE PROCESSOR.
* TYPE 'CARRIAGE RETURN' (CR) TO TERMINATE ALL INPUT DATA.

6. QUICK STATIC REGISTER TEST

A. THIS TEST IS DESIGNED TO PROVIDE A REGISTER VERIFICATION TEST IN THIS PROGRAM.
IF THIS SUB-TEST FAILS, AR11 TEST I SHOULD BE LOADED.

B. STARTING SEQUENCE

1. TYPE 'A' TO RUN THE QUICK REGISTER TEST.
2. THE PROGRAM WILL THEN EXECUTE THE QUICK REGISTER TEST.

C. CONTROL SWITCHES

1. TYPING ↑C AT ANY TIME WILL ENABLE THE PROGRAM TO EXIT AND RETURN TO THE MONITOR.

<u>CONSOLE SWITCHES</u>	<u>FUNCTION</u>
CONSOLE SW15=1	HALT ON ERROR
CONSOLE SW13=1	INHIBIT ERROR TYPEOUTS

D. ERRORS

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

E. RESTRICTIONS

NONE

7. POINT PLOT VISUAL DISPLAY TEST

A. THIS TEST IS DESIGNED TO AID IN THE ADJUSTING AND ALIGNMENT OF THE VR14 OR STORAGE SCOPE SCOPE ON THE AR11 DISPLAY CONTROL.

B. STARTING SEQUENCE

1. TYPE 'B' TO RUN THE VISUAL DISPLAY TEST.
2. THE PROGRAM WILL THEN EXECUTE THE VISUAL DISPLAY TEST.

C. CONTROL SWITCHES

1. TYPING 'C' AT ANY TIME WILL ENABLE THE PROGRAM TO EXIT AND RETURN TO THE MONITOR.

<u>CONSOLE SWITCHES</u>	<u>FUNCTION</u>
CONSOLE SW08=0	LOOP THRU DISPLAY TEST
CONSOLE SW08=1	SELECT TEST IN SW 00-02
CONSOLE SW00-02=0	DISPLAY A HORIZONTAL LINE
CONSOLE SW00-02=1	DISPLAY A VERTICAL LINE
CONSOLE SW00-02=2	DISPLAY A SQUARE
CONSOLE SW00-02=3	DISPLAY AN "X"

D. ERRORS

NO PROVISIONS ARE MADE FOR LOGIC ERRORS. THE ONLY ERRORS IN THIS TEST ARE CHECKED VISUALLY.

E. RESTRICTIONS

IF VR14 CHANNEL SWITCH MUST BE SET TO "1 & 2" POSITION.
IF STORAGE SCOPE, POWER MUST BE APPLIED.

F. EXECUTION TIME

IT TAKES APPROXIMATELY 60 SECONDS TO THIS TEST.

8. VISUAL DISPLAY TEST DESCRIPTIONS

DISPLAY HORIZONTAL LINE

A HORIZONTAL LINE IS DISPLAYED ON THE SCOPE BY INITIALLY SETTING THE X AND Y DAC'S TO ZERO AND THEN INCREMENTING THE X VALUE WHILE HOLDING THE Y VALUE AT 1000. THE POINTS ARE DISPLAYED USING THE DISPLAY INTERRUPT ENABLED.

DISPLAY VERTICAL LINE

A VERTICAL LINE IS DISPLAYED ON THE SCOPE IN THE SAME MANNER AS FOR A HORIZONTAL LINE EXCEPT NOW THE Y VALUE IS INCREMENTED WHILE HOLDING THE X VALUE AT 1000.

DISPLAY SQUARE

A SQUARE IS DISPLAYED BY INITIALLY SETTING THE X AND Y VALUES TO NEGATIVE FULL SCALE, THEN X IS INCREMENTED TO POSITIVE FULL SCALE (BOTTOM LINE) THEN Y IS INCREMENTED TO POSITIVE FULL SCALE (RIGHT LINE) THEN X IS DECREMENTED TO NEGATIVE FULL SCALE (TOP LINE) AND FINALLY Y IS DECREMENTED TO NEGATIVE FULL SCALE (LEFT LINE). MODE 01 (INTENSIFY ON LOADING X) AND MODE 10 (INTENSIFY ON LOADING Y) ARE USED.

DISPLAY X

AN X IS DISPLAYED BY INITIALLY SETTING THE X AND Y VALUES TO NEGATIVE FULL SCALE AND THEN INCREMENTING BOTH TO POSITIVE FULL SCALE (LOWER LEFT TO UPPER RIGHT DIAGONAL) THEN X IS RESET TO NEGATIVE FULL SCALE, Y REMAINS AT POSITIVE FULL SCALE AND THEN X IS INCREMENTED WHILE Y IS DECREMENTED UNTIL BOTH REACH FULL SCALE AGAIN (UPPER LEFT TO LOWER RIGHT DIAGONAL). MODE 01 (INTENSIFY ON LOADING X) IS USED.

9. A TO D CALIBRATION TEST

SEQ 0007

A. THE 'A/D CALIBRATION' TEST IS DESIGNED TO ACCEPT AN INPUT FROM THE TELETYPE TO INDICATE THE TYPE OF SYNC (EXTERNAL, INTERNAL OR AR11 CLOCK TO BE USED AND THEN TAKES CONTINUOUS CONVERSIONS USING THE 'CH.' SELECTED VIA THE CONSOLE SWITCHES. THESE SETTINGS MAY BE CHANGED AT ANY TIME. THIS CAN ALSO BE USED FOR FINDING 50-50 POINT AND FOR FINDING MIDDLE OF A STATE FOR SUBSEQUENT REPEATABILITY TEST.

B. STARTING SEQUENCE

1. TYPE 'C' TO RUN THE A/D CALIBRATION TEST.
2. THE TEST HEADER PLUS A REQUEST FOR A SYNC TYPE WILL THEN BE TYPED.
3. TYPE IN THE DESIRED SYNC, 'I' FOR INTERNAL, 'E' FOR EXTERNAL 'C' FOR AR11 CLOCK FOLLOWED BY 'CR'.
4. THE TEST WILL START.

C. CONTROL SWITCHES1. ↑A (CONTROL A)

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

2. ↑C (CONTROL C)

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

3. CONSOLE SWITCH FUNCTION

CONSOLE SW '0-3'	CHANNEL SELECT
CONSOLE SW 05 = 0	SELECT BIPOLAR CHANNEL
CONSOLE SW 05 = 1	SELECT UNIPOLAR CHANNEL
CONSOLE SW 06 = 0	CONTINUOUS SAMPLES
CONSOLE SW 06 = 1	FREEZE ON CURRENT DATA
CONSOLE SW 10 = 1	PRINT CONVERSION VALUE

D. CALIBRATION ERRORS

NONE

10. A TO D REPEATABILITY TEST

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

B. STARTING SEQUENCE

1. TYPE 'D' TO RUN THE 'REPEATABILITY' TEST.
2. A REQUEST IS THEN MADE FOR CH.(S) TO BE TESTED AND COUNT SPREAD (RANGE IN WHICH ALL 512 COUNTS MUST FALL FOR THE CH. TO BE CONSIDERED ACCEPTABLE).
3. IF THE CHANNEL IS FOUND TO BE WITHIN THE SELECTED COUNT SPREAD, THE PROGRAM WILL EITHER CONTINUE TO THE NEXT CHANNEL IF SELECTED OR RETEST THE CURRENT CHANNEL.

C. CONTROL SWITCHES

1. ↑A (CONTROL A)

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

2. ↑C (CONTROL)

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

3. CONSOLE SWITCHES

FUNCTION

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

D. REPEATABILITY ERRORS

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

1. ERROR FORMAT

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

WHERE:

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

E. RESTRICTIONS

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

11. A TO D RECOVERY TEST

SEQ 0010

A. THE "RECOVERY TEST" IS DESIGNED TO DETERMINE THE INTER-CHANNEL SETTling CAPABILITY OF THE 'ARI1 A TO D'. THE TEST REQUESTS FOR TWO (2) CH. INPUTS TO BE TYPED IN. THE TEST THEN TAKES A SERIES OF SIXTEEN (16) CONVERSIONS (8 ON EACH CH.) AND THEN TYPES OUT THE 'B' AVERAGE VALUES IN THE ORDER THEY WERE TAKEN ON THE SECOND CH.

B. STARTING SEQUENCE

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

EXAMPLE:

CH. A XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

WHERE:

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

C. CONTROL SWITCHES

1. ↑A (CONTROL A)

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

2. ↑C (CONTROL C)

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

3. CONSOLE SWITCHES

FUNCTION

CONSOLE SW 10 = 0
CONSOLE SW 10 = 1

PRINT RECOVERY REPORT
INHIBIT PRINTING OF RECOVERY REPORT

D. RESTRICTIONS

NONE

12. MISC. INFORMATION

12.1 ACT-11 OR XXDP

IF THE PROGRAM WAS LOADED FROM ACT-11 OR CHAINED FROM XYDP, THE VISUAL DISPLAY TEST WILL BE RUN.

12.2 APT

THE APT HOOKS HAVE BEEN INSTALLED BUT NOT TESTED.

12.3 USER LINK TO I/O DEVICE

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

PROCEDURE:

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

```

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

```

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

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

```

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

```

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

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

13. PROGRAM VARIABLE LOCATIONS

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

NOTE: IF LOCATIONS \$BASE OR \$VECT1 ARE CHANGED, THE TEST MUST BE RE-INITIALIZED AT 200.

14. TABLE OF CONTENTS

ATTACHED

15. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

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

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

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

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

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

THE LPA11-KX DIAGNOSTIC KIT WILL INCLUDE:

SEQ 0013

<u>OPTION</u>	<u>GROUP</u>	<u>DIAG. #</u>	<u>DIAG. TITLE</u>
LPA11-KX	LEVEL 2	MD-11-DRLPA	LPA11-K SYSTEM DIAG.
M8254	"B"	MD-11-DRLPN	M8254 (IPBM) DIAG.
AA11-K	A	MD-11-DRLPB	AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11	A	MD-11-DRLPC	LPA/AR11 DIAG. #1
	A	MD-11-DRLPD	LPA/AR11 DIAG. #2
	A	MD-11-DRLPE	LPA/AR11 DIAG. #3
	B	MD-11-DZARA	AR11 DIAG. #1
	B	MD-11-DZARB	AR11 DIAG. #2
	B	MD-11-DZARC	AR11 DIAG. #3
DR11-K	A	MD-11-DRLPF	LPA/DR11-K DIAG.
	B	MD-11-DZDRG	DR11-K DIAG.
KW11-K	A	MD-11-DRLPG	LPA/KW11-K DIAG.
	B	MD-11-DZKWK	KW11-K DIAG.
LPS11	A	MD-11-DRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-DRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-DRLPJ	LPA/LPS11 DIAG. #3
	B	MD-11-DZLPC	LPS11 DIAG. #1
	B	MD-11-DZLPD	LPS11 DIAG. #2
	B	MD-11-DZLPI	LPS11 DIAG. #3
AD11-K	A	MD-11-DRLPK	LPA/AD11-K DIAG.
	B	MD-11-DZADL	AD11-K DIAG.
M8200-YC	B	MD-11-DZLPL	LPA/M8200-YC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/M8200-YC JMP+ROM READ TESTS

41	BASIC DEFINITIONS
156	OPERATIONAL SWITCH SETTINGS
168	TRAP CATCHER
177	STARTING ADDRESS(ES)
181	ACT11 HOOKS
192	APT PARAMETER BLOCK
214	COMMON TAGS
262	APT MAILBOX-ETABLE
330	ERROR POINTER TABLE
426	INITIALIZE THE COMMON TAGS
584	T1 DISPLAY HORIZONTAL LINE
598	T2 DISPLAY A VERTICAL LINE
645	T3 PINCUSHION TEST (DISPLAY SQUARE)
741	T4 PLOT AN X
804	END OF PASS ROUTINE
964	T5 CALIBRATION ROUTINE
1151	T6 REPEATABILITY TEST
1274	T7 RECOVERY TEST
1358	T10 LOAD DIFFERENT NUMBERS INTO DIFFERENT REG.
1458	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2011	SCOPE HANDLER ROUTINE
2077	ERROR HANDLER ROUTINE
2127	ERROR MESSAGE TIMEOUT ROUTINE
2174	POWER DOWN AND UP ROUTINES
2225	BINARY TO OCTAL (ASCII) AND TYPE
2302	TYPE ROUTINE
2381	TTY INPUT ROUTINE
2520	READ AN OCTAL NUMBER FROM THE TTY
2558	APT COMMUNICATIONS ROUTINE
2615	TRAP DECODER
2638	TRAP TABLE

.REM [

LPA.MAC

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

GOOD LUCK !

[
.GLOBL DRLPX2

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

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

001100

000011
000012
000015
000200
177776

177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

000000
000040
000100
000140
000200
000240
000300
000340

100000
040000

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

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

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

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

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

.*"SWITCH REGISTER" SWITCH DEFINITIONS
SW15= 100000
SW14= 40000

84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137

020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

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

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

; *BASIC "CPU" TRAP VECTOR ADDRESSES

138 000004
139 000010
140 000014
141 000014
142 000014
143 000020
144 000024
145 000030
146 000034
147 000060
148 000064
149 000240

ERRVEC= 4
RESVEC= 10
TBITVEC=14
TRTVEC= 14
BPTVEC= 14
IOTVEC= 20
PWRVEC= 24
EMTVEC= 30
TRAPVEC=34
TKVEC= 60
TPVEC= 64
PIRQVEC=240

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

150
151 170400
152 000340
153 000200

ABASE=170400
AVECT1=340
APRIOR=200

.SBTTL OPERATIONAL SWITCH SETTINGS

154
155
156
157
158
159
160
161
162
163
164
165
166
167

;;
;; SWITCH USE

;; 15 HALT ON ERROR
;; 14 LOOP ON TEST
;; 13 INHIBIT ERROR TYPEOUTS
;; 12 STORAGE SCOPE CONNECTED
;; 11 INHIBIT ITERATIONS
;; 10 BELL ON ERROR
;; 9 LOOP ON ERROR
;; 8 LOOP ON TEST IN SWR<7:0>

.SBTTL TRAP CATCHER

168
169 000000

.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

170
171
172
173 000174
174 000174 000000
175 000176 000000
176
177 000200 000137 001454
178 000204 000137 001524
179

.=174
DISPREG: .WORD 0 ;; SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#BEGIN ;; JUMP TO STARTING ADDRESS OF PROGRAM
JMP BEGIN1 ;; JUMP TO RESTART ADDRESS

.SBTTL ACT11 HOOKS

180
181
182
183
184 000210
185 000046
186 000046 004110
187 000052
188 000052 000000
189 000210
190 001000
191

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

```

192
193
194
195
196           001000
197           000024
198 000024    000200
199           000044
200 000044    001000
201           00100C
202
203
204
205
206 001000
207 001000    000000
208 001002    001202
209 001004    000030
210 001006    000060
211 001010    000120
212 001012    000052

```

```

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

$APTHD:
$HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL     ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 30        ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 60        ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 120       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
SETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

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

.SBTTL COMMON TAGS

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

```

SCMTAG:      .=1100      ;; START OF COMMON TAGS
              .WORD      0
STSNM:       .BYTE      000      ;; CONTAINS THE TEST NUMBER
SERFLG:      .BYTE      000      ;; CONTAINS ERROR FLAG
SICNT:       .WORD      000      ;; CONTAINS SUBTEST ITERATION COUNT
SLPADR:      .WORD      000      ;; CONTAINS SCOPE LOOP ADDRESS
SLPERR:      .WORD      000      ;; CONTAINS SCOPE RETURN FOR ERRORS
SERTTL:      .WORD      000      ;; CONTAINS TOTAL ERRORS DETECTED
SITEMB:      .BYTE      000      ;; CONTAINS ITEM CONTROL BYTE
SERMAX:      .BYTE      1       ;; CONTAINS MAX. ERRORS PER TEST
SERRPC:      .WORD      000      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
SGDADR:      .WORD      000      ;; CONTAINS ADDRESS OF 'GOOD' DATA
SBDADR:      .WORD      000      ;; CONTAINS ADDRESS OF 'BAD' DATA
SGDDAT:      .WORD      000      ;; CONTAINS 'GOOD' DATA
SBDDAT:      .WORD      000      ;; CONTAINS 'BAD' DATA
              .WORD      000      ;; RESERVED--NOT TO BE USED
SAUTOB:      .BYTE      000      ;; AUTOMATIC MODE INDICATOR
SINTAG:      .BYTE      000      ;; INTERRUPT MODE INDICATOR
              .WORD      0
SWR:         .WORD      DSWR      ;; ADDRESS OF SWITCH REGISTER
DISPLAY:     .WORD      DDI:P     ;; ADDRESS OF DISPLAY REGISTER
STKS:        177560           ;; TTY KBD STATUS
STKB:        177562           ;; TTY KBD BUFFER
STPS:        177564           ;; TTY PRINTER STATUS REG. ADDRESS
STPB:        177566           ;; TTY PRINTER BUFFER REG. ADDRESS
SNUL:        .BYTE      0       ;; CONTAINS NULL CHARACTER FOR FILLS
SFILLS:      .BYTE      2       ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
SFILLC:      .BYTE      12      ;; INSERT FILL CHARS. AFTER A "LINE FEED"
STPFLG:      .BYTE      0       ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
SREGAD:      .WORD      0       ;; CONTAINS THE ADDRESS FROM WHICH ($REGO) WAS OBTAINED
SREGO:       .WORD      0       ;; CONTAINS (($REGAD)+0)
SREG1:       .WORD      0       ;; CONTAINS (($REGAD)+2)
STIMES:      0               ;; MAX. NUMBER OF ITERATIONS
SESCAPE:     0               ;; ESCAPE ON ERROR ADDRESS
SBELL:       .ASCIZ <207><377><377> ;; CODE FOR BELL
SQUES:       .ASCII /?/      ;; QUESTION MARK
SCRLF:       .ASCII <15>     ;; CARRIAGE RETURN
SLF:         .ASCIZ <12>     ;; LINE FEED

```

.SBTTL APT MAILBOX-ETABLE

.EVEN
SMAIL: ;; APT MAILBOX
MSGTY: .WORD MSGTY ;; MESSAGE TYPE CODE

267	001204	000000	\$FATAL: .WORD	AFATAL	:: FATAL ERROR NUMBER
268	001206	000000	\$TESTN: .WORD	ATESTN	:: TEST NUMBER
269	001210	000000	\$PASS: .WORD	APASS	:: PASS COUNT
270	001212	000000	\$DEVCT: .WORD	ADEVCT	:: DEVICE COUNT
271	001214	000000	\$UNIT: .WORD	AUNIT	:: I/O UNIT NUMBER
272	001216	000000	\$MSGAD: .WORD	AMSGAD	:: MESSAGE ADDRESS
273	001220	000000	\$MSGLG: .WORD	AMSGLG	:: MESSAGE LENGTH
274	001222		\$ETABLE:		:: APT ENVIRONMENT TABLE
275	001222	000	\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
276	001223	000	\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
277	001224	000000	\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
278	001226	000000	\$USWR: .WORD	AUSWR	:: USER SWITCHES
279	001230	000000	\$CPUOP: .WORD	ACPUOP	:: CPU TYPE, OPTIONS
280			::*		BITS 15-11=CPU TYPE
281			::*		11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
282			::*		11/70=06, PDQ=07, Q=10
283			::*		BIT 10=REAL TIME CLOCK
284			::*		BIT 9=FLOATING POINT PROCESSOR
285			::*		BIT 8=MEMORY MANAGEMENT
286	001232	000	\$MAMS1: .BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
287	001233	000	\$MTYP1: .BYTE	AMTYP1	:: MEM. TYPE, BLK#1
288			::*		MEM. TYPE BYTE -- (HIGH BYTE)
289			::*		900 NSEC CORE=001
290			::*		300 NSEC BIPOLAR=002
291			::*		500 NSEC MOS=003
292	001234	000000	\$MADR1: .WORD	AMADR1	:: HIGH ADDRESS, BLK#1
293			::*		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
294	001236	000	\$MAMS2: .BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
295	001237	000	\$MTYP2: .BYTE	AMTYP2	:: MEM. TYPE, BLK#2
296	001240	000000	\$MADR2: .WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
297	001242	000	\$MAMS3: .BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
298	001243	000	\$MTYP3: .BYTE	AMTYP3	:: MEM. TYPE, BLK#3
299	001244	000000	\$MADR3: .WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
300	001246	000	\$MAMS4: .BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
301	001247	000	\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
302	001250	000000	\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
303	001252	000340	\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
304	001254	000000	\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
305	001256	170400	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
306	001260	000000	\$DEVm: .WORD	ADEVm	:: DEVICE MAP
307	001262	000000	\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
308	001264	000000	\$CDW2: .WORD	ACDW2	:: CONTROLLER DESCRIPTION WORD#2
309	001266	000000	\$DDW0: .WORD	ADDW0	:: DEVICE DESCRIPTOR WORD#0
310	001270	000000	\$DDW1: .WORD	ADDW1	:: DEVICE DESCRIPTOR WORD#1
311	001272	000000	\$DDW2: .WORD	ADDW2	:: DEVICE DESCRIPTOR WORD#2
312	001274	000000	\$DDW3: .WORD	ADDW3	:: DEVICE DESCRIPTOR WORD#3
313	001276	000000	\$DDW4: .WORD	ADDW4	:: DEVICE DESCRIPTOR WORD#4
314	001300	000000	\$DDW5: .WORD	ADDW5	:: DEVICE DESCRIPTOR WORD#5
315	001302	000000	\$DDW6: .WORD	ADDW6	:: DEVICE DESCRIPTOR WORD#6
316	001304	000000	\$DDW7: .WORD	ADDW7	:: DEVICE DESCRIPTOR WORD#7
317	001306	000000	\$DDW8: .WORD	ADDW8	:: DEVICE DESCRIPTOR WORD#8
318	001310	000000	\$DDW9: .WORD	ADDW9	:: DEVICE DESCRIPTOR WORD#9
319	001312	000000	\$DDW10: .WORD	ADDW10	:: DEVICE DESCRIPTOR WORD#10
320	001314	000000	\$DDW11: .WORD	ADDW11	:: DEVICE DESCRIPTOR WORD#11

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

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

SETEND:

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

.SBTTL ERROR POINTER TABLE

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

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

001326

\$ERRTB:

;ITEM 1

001326 012734
001330 012777
001332 013036
001334 000000

EM1
DH1
DT1
0

;DUAL REGISTER ADDRESSING DETECTED
;ERRPC BUFADR EXPECT READ
;\$ERRPC BUFADR \$GDDAT \$BDDAT

001336 000000

NBEXT: 0

001340 170400
001342 170402

ADCS: 170400
ADDBR: 170402

;A TO D STATUS/CONTROL REGISTER
;A TO D CONVERTED VALUE <READ>

001344 170404
001346 170406

CSR: 170404
CSB: 170406

;CLOCK STATUS REGISTER
;CLOCK PRESET BUFFER

001350 170410
001352 170412
001354 170414

VCSTAT: 170410
VCXREG: 170412
VCYREG: 170414

001356 170416
001360 000000
001362 000003

CSC: 170416
EAFK: 0
\$VERSN: .WORD 3

;;ADDRESS OF KMC-11 OF LPA-11

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

001364 170460

LPCI:
KMADO: .WORD 170460

;BASE KMC ADDR. MAY BE PATCHED BY USER.

001366 170461
001370

LPMR:
KMAD1: .WORD 170460+1
LPCO:

; >DO NOT <;KMC-CSR ADDR

```

383 001370 170462 KMAD2: .WORD 170460+2 ;>PATCH <;
384 001372 LPS0:
385 001372 170463 KMAD3: .WORD 170460+3 ;>THIS AREA <
386 001374 LPADL:
387 001374 170464 KMAD4: .WORD 170460+4 ;
388 001376 LPADH:
389 001376 170465 KMAD5: .WORD 170460+5 ;>DO NOT <
390 001400 LPMS1:
391 001400 170466 KMAD6: .WORD 170460+6 ;>PATCH <
392 001402 LPMS2:
393 001402 170467 KMAD7: .WORD 170460+7 ;>THIS AREA <
394
395 001404 000340 VECTOR: .WORD AVECT1&777 ;BASE VECTOR OF KMC
396 001406 000344 VECTPS: .WORD 4+AVECT1&777 ;VECTOR ADDR.+2
397
398 001410 000004 VERSN: .WORD 4 ;CURRENT VERSION NUMBER OF MICROCODE.
399
400 001412 000000 .DVLS: .WORD 0 ;/DEVICE LIST OF I/O ADDR. DEFINED
401 001414 000020 .BLKW 16. ;/BY INIT.
402
403 001454 012706 001100 BEGIN: MOV #STACK,SP
404
405 ;; THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
406 ;;
407
408 001460 010046 MOV RO, -(SP)
409 001462 010146 MOV R1, -(SP)
410 001464 013700 001364 MOV KMAD0, RO ;GET KMC-11 ADDRESS.
411 001470 012701 001366 MOV #KMAD1, R1 ;GET ADDR. OF ADDR. LIST.
412
413 001474 005200 64$: INC RO ;UPDATE ADDR.
414 001476 010021 MOV RO, (1)+ ;WRITE ADDR.
415 001500 020127 001404 CMP R1, #KMAD7+2 ;DONE ALL ADDRESSES?
416 001504 001373 BNE 64$ ;NO - DO NEXT ADDR.
417 001506 005037 001412 CLR .DVLS ;CLR ADDR. LIST.
418 001512 012601 MOV (SP)+, R1
419 001514 012600 MOV (SP)+, RO
420 001516 005037 013130 CLR TEMP
421 001522 000405 BR BEG
422 001524 012737 000001 013130 BEGIN1: MOV #1, TEMP
423 001532 005037 001412 CLR .DVLS
424 001536 000240 BEG: NOP
425
426 .SBTTL INITIALIZE THE COMMON TAGS
427 ;; CLEAR THE COMMON TAGS ($CMTAG) AREA
428 001540 012706 001100 MOV #CMTAG, R6 ;; FIRST LOCATION TO BE CLEARED
429 001544 005026 CLR (R6)+ ;; CLEAR MEMORY LOCATION
430 001546 022706 001140 CMP #SWR, R6 ;; DONE?
431 001552 001374 BNE .-6 ;; LOOP BACK IF NO
432 001554 012706 001100 MOV #STACK, SP ;; SETUP THE STACK POINTER
433 ;; INITIALIZE A FEW VECTORS
434 001560 012737 013250 000020 MOV #SCOPE, @IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
435 001566 012737 000340 000022 MOV #340, @IOTVEC+2 ;; LEVEL 7
436 001574 012737 013532 000030 MOV #ERROR, @EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
437 001602 012737 000340 000032 MOV #340, @EMTVEC+2 ;; LEVEL 7

```



```

437 001610 012737 016130 000034      MOV      #STRAP,@TRAPVEC      ;; TRAP VECTOR FOR TRAP CALLS
438 001616 012737 000340 000036      MOV      #340,@TRAPVEC+2    ;; LEVEL 7
439 001624 012737 014054 000024      MOV      #SPWRDN,@PWRVEC    ;; POWER FAILURE VECTOR
440 001632 012737 000340 000026      MOV      #340,@PWRVEC+2    ;; LEVEL 7
441 001640 005037 001166          CLR      $TIMES              ;; INITIALIZE NUMBER OF ITERATIONS
442 001644 005037 001170          CLR      $ESCAPE            ;; CLEAR THE ESCAPE ON ERROR ADDRESS
443 001650 112737 000001 001115      MOV      #1,$SERMAX         ;; ALLOW ONE ERROR PER TEST
444 001656 012737 001656 001106      MOV      #.,$SLPADR         ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
445 001664 012737 001664 001110      MOV      #.,$LPERR         ;; SETUP THE ERROR LOOP ADDRESS
446                                     ;; SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
447                                     ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
448 001672 013746 000004          MOV      @ERRVEC,-(SP)      ;; SAVE ERROR VECTOR
449 001676 012737 001732 000004      MOV      #64$,@ERRVEC      ;; SET UP ERROR VECTOR
450 001704 012737 177570 001140      MOV      #DSWR,SWR          ;; SETUP FOR A HARDWARE SWICH REGISTER
451 001712 012737 177570 001142      MOV      #DDISP,DISPLAY     ;; AND A HARDWARE DISPLAY REGISTER
452 001720 022777 177777 177212      CMP      #-1,@SWR           ;; TRY TO REFERENCE HARDWARE SWR
453 001726 001012          BNE      66$                ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
454                                     ;; AND THE HARDWARE SWR IS NOT = -1
455 001730 000403          BR       65$                ;; BRANCH IF NO TIMEOUT
456 001732 012716 001740 64$:      MOV      #65$,(SP)         ;; SET UP FOR TRAP RETURN
457 001736 000002          RTI                          ;;
458 001740 012737 000176 001140 65$:      MOV      #SWREG,SWR         ;; POINT TO SOFTWARE SWR
459 001746 012737 000174 001142      MOV      #DISPREG,DISPLAY   ;;
460 001754 012637 000004 66$:      MOV      (SP)+,@ERRVEC     ;; RESTORE ERROR VECTOR
461
462 001760 005037 001210          CLR      $PASS              ;; CLEAR PASS COUNT
463 001764 132737 000200 001223      BIT     #APTSIZE,$ENVM      ;; TEST USER SIZE UNDER APT
464 001772 001403          BEQ     67$                ;; YES, USE NON-APT SWITCH
465 001774 012737 001224 001140      MOV      #$$SWREG,SWR      ;; NO, USE APT SWITCH REGISTER
466 002002          67$:
467 002002 104401 002010          TYPE    69$                ;; TYPE ASCIZ STRING
468 002006 000411          BR      68$                ;; GET OVER THE ASCIZ
469                                     ;; 69$: .ASCIZ <15><12>#MD-11-DRLPD-A#<15><12>
470 002032          68$:
471 002032 005037 177776          CLR      @#PS
472                                     ;; DETERMINE CPU TYPE
473                                     ;;
474                                     ;; RO=3 IF 11/45
475                                     ;; RO=2 IF 11/40
476                                     ;; RO=1 IF 11/20
477                                     ;; RO=0 IF 11/05
478 177772          PIRQ=177772
479 177776          PS=177776
480
481 002036 012737 000002 000006      MOV      #RTI,@#6
482 002044 012737 000006 000004      MOV      #6,@#4
483 002052 012700 000003          MOV      #3,RO              ;; LOAD RO
484 002056 000261          SEC                          ;; SET C BIT
485 002060 005737 177772      TST     @#PIRQ              ;; TEST PIRQ
486 002064 005600          SBC     RO
487 002066 000261          SEC
488 002070 105737 177777      TST     @#PS+1
489 002074 005600          SBC     RO
490 002076 005037 177700          CLR      @#177700

```

```

491 002102 005037 000006          CLR      @#6
492 002106 010037 010652          MOV      RO,CPTYPE
493 002112 005237 010652          INC      CPTYPE
494 002116 006300                    ASI      RO
495 002120 016037 010640 010650  MOV      CPDLAY(RO),CPTIME      ;GET CP DELAY TIME
496 002126 000137 002234          JMP      INIT1
497
498                                ;LOAD TRAP CATCHER
499
500 002132 012702 000242  LDTRAP: MOV      #242,R2          ;LOAD R2
501 002136 012701 000240          MOV      #240,R1          ;LOAD R1
502 002142 010221          SS:  MOV      R2,(R1)+      ;LOAD .+2
503 002144 005021          CLR      (R1)+          ;LOAD HALT
504 002146 010102          MOV      R1,R2          ;LOAD R2
505 002150 005722          TST      (R2)+          ;BUMP R2
506 002152 020227 001002          CMP      R2,#1002      ;TEST FOR LAST
507 002156 001371          BNE      SS            ;BR UNTIL DONE
508
509                                ;LOAD DEVICE ADDRESSES LOCATIONS
510
511 002160 012700 001340          MOV      #ADCS,RO      ;LOAD POINTER
512 002164 013720 001256 10$:  MOV      $BASE,(RO)+    ;LOAD BASE ADDRESS
513 002170 022700 001360          CMP      #EAFK,RO     ;TEST FOR DONE
514 002174 001373          BNE      10$          ;BR
515 002176 013737 001256 010654  MOV      $BASE,ADCS1   ;LOAD HIGH BYTE POINTER
516 002204 005237 010654          INC      ADCS1
517 002210 012700 001342          MOV      #ADDBR,RO    ;LOAD 2ND ADDRESS
518 002214 012701 000002          MOV      #2,R1        ;LOAD R1
519 002220 060120          12$:  ADD      R1,(RO)+     ;UPDATE REAL DEVICE WORD
520 002222 005721          TST      (R1)+        ;BUMP R1
521 002224 022701 000020          CMP      #20,R1      ;TEST FOR DONE
522 002230 001373          BNE      12$          ;BR
523 002232 000207          RTS      PC           ;EXIT
524
525 002234 004737 002132  INIT1: JSR      PC,LDTRAP
526 002240 005737 013130          TST      TEMP
527 002244 001011          BNE      INIT2
528 002246 005737 000042          TST      @#42
529 002252 001402          BEQ      1$
530 002254 000137 002540          JMP      VSUALO
531 002260 104401          1$:  TYPE
532 002262 011610          TITLE
533 002264 104401          INITA: TYPE
534 002266 011772          MES4
535 002270 012737 002264 013054  INIT2: MOV      #INITA,AVECTR ;PRINT THE TEST CALL LETTERS.
536 002276 004737 002132          JSR      PC,LDTRAP    ;SET UP 'IA' VECTOR ADDRESS.
537 002302 104401          TYPE                ;LOAD TRAP CATCHER AND ADDRESSES
538 002304 012245          CNTRLC
539 002306 012737 000000 013246  MOV      #0,$TMDAT    ;PRINT '.' TO INDICATE MONITOR READY
540
541                                ;*
542                                ;*
543                                ;*
544                                ;*

```

```

545 ;* MOV STMDAT,AVCSTAT ;/ PUT DATA FROM STMDAT TO DEVICE REG VCSTAT
546 ;* MOV STMDAT,ADDBR ;/ PUT DATA FROM STMDAT TO DEVICE REG ADDBR
547 ;* MOV STMDAT,ACSB ;/ PUT DATA FROM STMDAT TO DEVICE REG CSB
548 ;* MOV STMDAT,AVCXREG ;/ PUT DATA FROM STMDAT TO DEVICE REG VCXREG
549 ;* MOV STMDAT,AVCYREG ;/ PUT DATA FROM STMDAT TO DEVICE REG VCYREG
550 ;* MOV STMDAT,ACSC ;/ PUT DATA FROM STMDAT TO DEVICE REG CSC
551 JSR PC,XTTYIN ;WAIT FOR TTY ENTRY
552 CMPB #'A,INBUF ;TEST FOR 'A'
553 BNE 1$ ;NOT 'A'
554 JMP RESTST ;YES RUN QUICK REGISTER TEST
555 1$: CMPB #'B,INBUF ;TEST FOR 'B'
556 BNE 2$ ;NOT 'B'
557 JMP VISUAL ;YES RUN 'SCOPE ADJUSTMENT TEST'
558 2$: CMPB #'C,INBUF ;TEST FOR 'C'
559 BNE 3$ ;NOT 'C'
560 JMP CALBRT ;YES RUN 'A TO D CALIBRATION' TEST
561 3$: CMPB #'D,INBUF ;TEST FOR 'D'
562 BNE 4$ ;NOT 'D'
563 JMP REPTST ;YES RUN 'A TO D REPEATABILITY' TEST
564 4$: CMPB #'E,INBUF ;TEST FOR 'E'
565 BNE 5$ ;NOT 'E'
566 JMP RECVRY ;YES RUN 'A TO D RECOVERY TEST'
567 5$: TYPE ILLEGAL ENTRY
568 QMARK TYPE '?'
569 BR ;WAIT AGAIN
570 INIT2
571 VISUAL: MOV #VISAUI,AVECTR
572 TYPE ;HEADER ABOUT SCOPE ADJ.
573 MES6 ;TEXT ABOUT SWR
574 VISAUI: TYPE
575 MES15
576 VSUALD: NOP
577 *****
578 *TEST 1 DISPLAY HORIZONTAL LINE
579 *****
580 *ST1: SCOPE
581 MOV #1,STIMES ;DO 1 ITERATION
582 PICO: MOV VCXREG,R1Y
583 MOV VCYREG,ROX
584 MOV #4,TICKS ;LOAD TIMER FOR CP TYPE
585 JSR PC,CHTIME ;CHANGE TIMER FOR CP TYPE
586 1$: JSR PC,PBB
587 JSR PC,TIMER ;DONE ?
588 BR 1$
589 *****
590 *TEST 2 DISPLAY A VERTICAL LINE
591 *****
592 *****
593 *****
594 *****
595 *****
596 *****
597 *****
598 *****

```

```

599 002612 000004 TST2: SCOPE
600 002614 012737 000001 001166 MOV #1,STIMES ;;DO 1 ITERATION
601 002622 013737 001354 003044 PIC1: MOV VCYREG,R1Y
602 002630 013737 001352 003042 MOV VCXREG,ROX
603 002636 012737 000004 013112 MOV #4,TICKS ;LOAD TIME
604 002644 004737 004662 JSR PC,CHTIME ;CHANGE TIMER FOR CP TYPE
605 002650 004737 002664 1S: JSR PC,PBB
606 002654 004737 004550 JSR PC,TIMER ;DONE ?
607 002660 000773 BR 1S
608 002662 000475 BR PIC3
609
610 002664 012737 000000 013246 PBB: MOV #0,$TMDAT
611
612 ;* MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
613 002702 013704 001350 MOV VCSTAT,R4
614 002706 012703 001777 MOV #1777,R3 ;SET HIGH LIMIT
615 002712 012702 000001 MOV #1,R2 ;INITIALIZE INCREMENTS BETWEEN POINTS
616 002716 012737 001000 013246 MOV #1000,$TMDAT
617
618 ;* MOV $TMDAT,$R1Y ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1Y
619 002734 012737 000000 013246 MOV #0,$TMDAT
620
621 ;* MOV $TMDAT,$R1Y ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1Y
622 002752 1S:
623
624 ;* MOV $R1Y,$TMDAT ;/READ DEVICE REG R1Y,PUT DATA IN $TMDAT.
625 002762 060237 013246 ADD R2,$TMDAT
626
627 ;* MOV $TMDAT,$R1Y ;/ PUT DATA FROM $TMDAT TO DEVICE REG R1Y
628
629 ;* MOV $VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
630 003006 005237 001126 INC $BDDAT
631
632 ;* MOV $BDDAT,$VCSTAT ;/ PUT DATA FROM $BDDAT TO DEVICE REG VCSTAT
633
634 ;* MOV $R1Y,$TMDAT ;/READ DEVICE REG R1Y,PUT DATA IN $TMDAT.
635 003032 023703 013246 CMP $TMDAT,R3
636 003036 001345 1S BNE 1S ;NO
637 003040 000207 RTS PC

```

```

638
639 003042 000000          ROX:  .WORD  0
640 003044 000000          RIY:  .WORD  0
641
642      ;*****
643      ;*TEST 3          PINCUSHION TEST (DISPLAY SQUARE)
644      ;*****
645 003046 000004          †ST3:  SCOPE
646 003050 012737 000001 001166      MOV      #1,STIMES          ;;DO 1 ITERATION
647      ;PLOT A SQUARE FROM LOWER LEFT TO LOWER RIGHT TO
648      ;UPPER RIGHT TO UPPER LEFT TO LOWER LEFT.
649      ;NON STORE DISPLAY
650 003056 005037 013102      PIC3:  CLR      LOW
651
652      ;*      MOV      LOW,AVCSTAT          ;/ PUT DATA FROM LOW TO DEVICE REG VCSTAT
653 003072 012737 000005 013112      MOV      #5,TICKS
654 003100 004737 004662          JSR      PC,CHTIME
655 003104 013701 001352          MOV      VCXREG,R1
656 003110 013702 001354          MOV      VCYREG,R2
657 003114 013703 001350          MOV      VCSTAT,R3
658 003120 012704 000004          MOV      #4,R4
659 003124 013737 013102 013246      p3:    MOV      LOW,$TMDAT
660
661      ;*      MOV      $TMDAT,AVCXREG          ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
662
663      ;*      MOV      $TMDAT,AVCYREG          ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
664      ;DRAW BOTTOM LINE
665 003152 012700 000377          MOV      #377,R0
666 003156 012737 000004 013246      MOV      #4,$TMDAT          ;ENABLE INTENSIFY ON LOADNG X
667
668      ;*      MOV      $TMDAT,AVCSTAT          ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
669 003174          P3A:
670
671      ;*      MOV      AVCXREG,$TMDAT          ;/READ DEVICE REG VCXREG,PUT DATA IN $TMDAT.
672 003204 060437 013246          ADD      R4,$TMDAT
673
674      ;*      MOV      $TMDAT,AVCXREG          ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
675      ;WAIT FOR READY
676 003220 005300          DEC      R0
677 003222 001364          BNE     P3A          ;NO
678      ;DRAW RIGHT LINE
679 003224 012737 000010 013246      MOV      #10,$TMDAT          ;ENABLE INTENSIFY ON LOADING Y
680
681      ;*      MOV      $TMDAT,AVCSTAT          ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
682 003242 012700 000377          MOV      #377,R0
683 003246          P3B:
684
685      ;*      MOV      AVCYREG,$TMDAT          ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
686 003256 060437 013246          ADD      R4,$TMDAT
687
688      ;*      MOV      $TMDAT,AVCYREG          ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
689 003272          P3:
690
691      ;*      MOV      AVCSTAT,$BDDAT          ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.

```

```

692 003302 105737 001126      TSTB  $BDDAT
693 003306 100371              BPL   1$
694                                ;WAIT FOR READY
695 003310 005300              DEC   R0
696 003312 001355              BNE   P3B
697                                ;NO
698 003314 012737 000004 013246 ;DRAW TOP LINE
699                                MOV   #4,$TMDAT
700                                ;ENABLE INTENSIFY ON LOADING X
701 003332 012700 000377      ;*   MOV   $TMDAT,$VCSTAT
702 003336                                MOV   #377,R0
703                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
704                                ;*   MOV   $VCXREG,$TMDAT
705 003346 160437 013246      SUB   R4,$TMDAT
706                                ;/READ DEVICE REG VCXREG,PUT DATA IN $TMDAT.
707                                ;*   MOV   $TMDAT,$VCXREG
708 003362                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
709                                ;$:
710                                ;*   MOV   $VCSTAT,$BDDAT
711 003372 105737 001126      TSTB  $BDDAT
712 003376 100371              BPL   1$
713                                ;WAIT FOR READY
714 003400 005300              DEC   R0
715 003402 001355              BNE   P3C
716                                ;NO
717 003404 012737 000010 013246 ;DRAW LEFT LINE
718                                MOV   #10,$TMDAT
719                                ;ENABLE INTENSIFY LOADING Y
720                                ;*   MOV   $TMDAT,$VCSTAT
721 003422 012700 000377      MOV   #377,R0
722                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
723                                ;$:
724 003436 160437 013246      ;*   MOV   $VCYREG,$TMDAT
725                                SUB   R4,$TMDAT
726                                ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
727                                ;*   MOV   $TMDAT,$VCYREG
728 003452                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
729                                ;$:
730                                ;*   MOV   $VCSTAT,$BDDAT
731 003462 105737 001126      TSTB  $BDDAT
732 003466 100371              BPL   1$
733                                ;WAIT FOR READY
734                                ;*   DEC   R0
735 003470 005300              BNE   P3D
736 003474 004737 004550      JSR   PC,TIMER
737 003500 000611              BR    P3
738                                ;*****
739                                ;*TEST 4 PLOT AN X
740                                ;*****
741 003502 000004 000001 001166 ;TST4: SCOPE
742 003504 012737 000000 013102 MOV   #1,$TIMES
743 003512 012737 000000 013102 PIC4: MOV   #0,LOW
744 003520 012737 001774 013104 MOV   #1774,HIGH
745 003526 012737 000000 013246 MOV   #0,$TMDAT

```

E03

MAINDEC-11-DRLPD-A
DRLPD.P11 T4MACY11 27(654)
PLOT AN X

14-DEC-77 20:22 PAGE 17

SEQ 0031

```

746          ;*      MOV      $TMDAT, @VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
747 003544   012737 000005 013112 ;*      MOV      #5, TICKS
748 003552   004737 004662          ;*      JSR      PC, CHTIME ;CHANGE TIMER FOR CP TYPE
749 003556   013701 001352          PIC4B: ;*      MOV      VCXREG, R1
750 003562   013702 001354          ;*      MOV      VCYREG, R2
751 003566   013703 001350          ;*      MOV      VCSTAT, R3
752 003572   012704 000004          ;*      MOV      #4, R4
753 003576          ;*      P4:
754          ;*      MOV      LOW, @VCYREG ;/ PUT DATA FROM LOW TO DEVICE REG VCYREG
755          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
756          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
757          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
758          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
759          ;PLOT LINE BEGINNING IN LOWER LEFT CORNER
760 003616   012737 000004 013246 ;*      MOV      #4, $TMDAT ;ENABLE INTENSIFY ON LOADING X
761          ;*      MOV      $TMDAT, @VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
762 003634   012700 000377          ;*      MOV      $TMDAT, @VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
763 003640          ;*      MOV      $TMDAT, @VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
764          ;*      MOV      $TMDAT, @VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
765          ;*      MOV      $TMDAT, @VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
766 003650   060437 013246          ;*      MOV      @VCYREG, $TMDAT ;/READ DEVICE REG VCYREG, PUT DATA IN $TMDAT.
767          ;*      ADD      R4, $TMDAT
768          ;*      MOV      $TMDAT, @VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
769          ;*      MOV      $TMDAT, @VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
770          ;*      MOV      $TMDAT, @VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
771 003674   060437 013246          ;*      MOV      @VCXREG, $TMDAT ;/READ DEVICE REG VCXREG, PUT DATA IN $TMDAT.
772          ;*      ADD      R4, $TMDAT ;+4 TO X
773          ;*      MOV      $TMDAT, @VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
774 003710   005300          ;*      MOV      $TMDAT, @VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
775 003712   001352          ;*      DEC      R0
776          ;*      BNE     P4A ;NO
777          ;PLOT LINE BEGINNING IN UPPER LEFT CORNER
778          ;*      MOV      HIGH, @VCYREG ;/ PUT DATA FROM HIGH TO DEVICE REG VCYREG
779          ;*      MOV      HIGH, @VCYREG ;/ PUT DATA FROM HIGH TO DEVICE REG VCYREG
780          ;*      MOV      HIGH, @VCYREG ;/ PUT DATA FROM HIGH TO DEVICE REG VCYREG
781 003734   012700 000377          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
782 003740          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
783          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
784          ;*      MOV      LOW, @VCXREG ;/ PUT DATA FROM LOW TO DEVICE REG VCXREG
785 003750   160437 013246          ;*      MOV      @VCYREG, $TMDAT ;/READ DEVICE REG VCYREG, PUT DATA IN $TMDAT.
786          ;*      SUB      R4, $TMDAT
787          ;*      MOV      $TMDAT, @VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
788          ;*      MOV      $TMDAT, @VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
789          ;*      MOV      $TMDAT, @VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
790 003774   060437 013246          ;*      MOV      @VCXREG, $TMDAT ;/READ DEVICE REG VCXREG, PUT DATA IN $TMDAT.
791          ;*      ADD      R4, $TMDAT ;+4 TO X
792          ;*      MOV      $TMDAT, @VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
793 004010   005300          ;*      MOV      $TMDAT, @VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
794 004012   001352          ;*      DEC      R0
795 004014   004737 004550          ;*      BNE     P4B ;NO
796 004020   000666          ;*      JSR      PC, TIMER
797          ;*      BR      P4
798          ;*      BR      P4
799          .SBTTL END OF PASS ROUTINE

```

```

800
801
802
803
804
805
806
807
808 004022
809 004022 000004
810 004024 005037 001102
811 004030 005037 001166
812 004034 005237 001210
813 004040 042737 100000 001210
814 004046 005327
815 004050 000001
816 004052 003022
817 004054 012737
818 004056 000001
819 004060 004050
820 004062 104401 004127
821 004066 013746 001210
822 004072 104405
823 004074 104401 004124
824 004100 013700 000042
825 004104 001405
826 004106 000005
827 004110 004710
828 004112 000240
829 004114 000240
830 004116 000240
831 004120
832 004120 000137
833 004122 002540
834 004124 377 377 000
835 004127 015 042412 042116
836 004134 050040 051501 020123
837 004142 000043
838
839
840 004144
841 004144 012737 002000 013246
842
843
844
845
846 004172 052737 010000 001350
847
848
849 004210 000240
850 004212 012700 000020
851 004216 005001
852 004220
853

```

```

*****
; 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 VSUALO
SEOP:
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?
SEOPCT: .WORD 1
BGT $DOAGN ; YES
MOV (PC)+,$(PC)+ ; RESTORE COUNTER
SENDCT: .WORD 1
SEOPCT
TYPE $SENDMG ; TYPE "END PASS #"
MOV $PASS,-(SP) ; SAVE $PASS FOR TYPEOUT
TYPDS ; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $SENULL ; TYPE A NULL CHARACTER
SGET42: MOV #42,R0 ; GET MONITOR ADDRESS
BEQ $DOAGN ; BRANCH IF NO MONITOR
RESET ; CLEAR THE WORLD
SENDAD: JSR PC,(R0) ; GO TO MONITOR
NOP ; SAVE ROOM
NOP ; FOR
NOP ; ACT11
SDOAGN:
JMP $(PC)+ ; RETURN
SRTNAD: .WORD VSUALO
SENULL: .BYTE -1,-1,0 ; NULL CHARACTER STRING
SENDMG: .ASCIZ <15><12>/END PASS #/
CLRVC:
MOV #BIT10,$TMDAT ; ERASE THE SCREEN
;* MOV $TMDAT,$VCSTAT ; / PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
;* MOV $VCSTAT,$TMDAT ; / READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
BIS #BIT12,$VCSTAT
;* MOV $TMDAT,$VCSTAT ; / PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
NOP
MOV #20,R0 ; SET UP DELAY
CLR R1
CLRVCA:
; TEST FOR READY

```



```

854 ;* MOV @VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
855 004230 105737 013246 TSTB $TMDAT
856 004234 100416 BMI CLRVCB ;BRANCH IF SET
857 004236 005301 DEC R1 ;DELAY
858 004240 001367 BNE CLRVCA
859 004242 005300 DEC R0 ;DELAY
860 004244 001365 BNE CLRVCA
861 004246 037727 174666 010000 BIT @SWR,#SW12 ;TEST INHIBIT PRINTOUT
862 004254 001002 BNE IS
863 004256 104401 TYPE
864 004260 011725 MES3
865 004262 005777 174652 IS: TST @SWR ;TEST @SWR
866 004266 100001 BPL CLRVCB
867 004270 000000 HALT ;ERASE RETURN FAILED TO SET READY
868
869 004272 000207 CLRVCB: RTS PC
870
871 004274 LOADVC: ;CLEAR STATUS
872 004274 012737 000000 013246 MOV #0,$TMDAT
873
874 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
875 004312 012737 001777 013132 MOV #1777,TEMP1
876 004320 013700 001350 MOV VCSTAT,R0
877 004324 013701 001352 MOV VCXREG,R1
878 004330 013702 001354 MOV VCYREG,R2
879 004334 012737 002000 013246 MOV #BIT10,$TMDAT ;SET STORE MODE
880
881 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
882
883 ;* MOV TEMP1,@VCYREG ;/ PUT DATA FROM TEMP1 TO DEVICE REG VCYREG
884 004362 012737 001777 013246 LODVCA: MOV #1777,$TMDAT
885
886 ;* MOV $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
887 004400 000413 BR LODVCC
888 004402
889
890 ;* MOV @VCYREG,$TMDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
891 004412 162737 000004 013246 SUB #4,$TMDAT
892
893 ;* MOV $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
894 004430 LODVCC:
895
896 ;* MOV @VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
897 004440 005237 013246 INC $TMDAT
898
899 ;* MOV $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
900 004454 000240
901 004456 IS:
902
903 ;* MOV @VCSTAT,$BDDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
904 004466 105737 001126 TSTB $BDDAT
905 004472 100371 BPL IS
906
907 ;* MOV @VCYREG,$TMDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.

```

```

908 004504 022737 000003 013246      CMP      #3,$TMDAT
909 004512 001333                      BNE      LODVCB
910 004514 104407                      CKSWR           ;TEST FOR "CTRL G"
911
912                                     ;*      MOV      @VCYREG,$TMDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
913 004526 162737 000001 013246      SUB      #1,$TMDAT
914
915                                     ;*      MOV      $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
916 004544 001306                      BNE      LODVCA
917 004546 000207                      RTS         PC
918
919
920

```

```

921                                     ;TIMER ROUTINE
922                                     ; ENTER VIA JSR PC,TIMER
923
924 004550 104407      TIMER:  CKSWR
925 004552 017737 174362 013110      MOV      @SWR,TIMSV
926 004560 004737 011574                      JSR      PC,TSTFLG
927 004564 032737 000400 013110      TIMERA: BIT      #818,TIMSV
928 004572 001006                      BNE      TIMER2           ;BIT 8 SET ?
929 004574 005337 013112                      DEC      TICKS           ;NO, DECrEMENT TICKS
930 004600 001002                      BNE      TIMER1
931 004602 062716 000002                      ADD      #2,(6)           ;ADD 2 TO STACK POINTER
932 004606 000207      TIMER1: RTS         PC           ;RETURN
933

```

```

934                                     ; SWR 8=1 SELECT TEST TO LOCK ON
935                                     ; SWR 2-0= TEST NUMBER
936
937 004610 042737 177770 013110      TIMER2: BIC      #177770,TIMSV
938 004616 006337 013110                      ASL      TIMSV
939 004622 062737 004652 013110                      ADD      #ROUTPT,TIMSV
940 004630 017737 006254 013110                      MOV      @TIMSV,TIMSV
941 004636 022600                      CMP      (SP)+,R0
942 004640 000240                      NOP
943 004642 000240                      NOP
944 004644 000240                      NOP
945 004646 000177 006236      TIMER4: JMP      @TIMSV
946
947 004652 002552      ROUTPT: PICO           ;DISPLAY A HORIZONTAL LINE
948 004654 002622                      PIC1          ;DISPLAY A VERTICAL LINE
949 004656 003056                      PIC3          ;DISPLAY A SQUARE
950 004660 003512                      PIC4          ;DISPALY A "X"
951

```

```

952 004662 013737 010652 013140      CHTIME: MOV      CPTYPE,BRLEVI
953 004670 004737 011574                      JSR      PC,TSTFLG
954 004674 005337 013140      CHTMA:  DEC      BRLEVI
955 004700 001403                      BEQ      CHTMB
956 004702 006337 013112                      ASL      TICKS
957 004706 000772                      BR       CHTMA
958 004710 000207      CHTMB:  RTS         PC
959
960                                     ;*****
961                                     ;*TEST 5 CALIBRATION ROUTINE
962                                     ;*****

```

```

962 004712 000004 TST5: SCOPE
963 004714 012737 000001 001166 MOV #1,$TIMES ;;DO 1 ITERATION
964 ;*****
965 ;ROUTINE REQUESTS THE TYPE OF 'SYNC' TO BE USED ('I' INTERNAL OR 'E' EXTERNAL
966 ;OR 'C' AR CLOCK)
967 ;THE PROGRAM THEN TAKES CONTINUOUS CONVERSIONS USING DATA SW'S 3-0
968 ;TO SELECT THE CH., AND SWS TO SELECT UNIPOLAR/BIPOLAR
969 ;USE SW '10' TO PRINT THE CONVERSION VALUE.
970
971
972 004722 012737 004734 013054 CALBRT: MOV #CALBT1,AVECTR ;SET UP '1A' RESTART ADDRESS
973 004730 104401 TYPE
974 004732 012322 MES7 ;TYPE TEST HEADER
975 004734 104401 CALBT1: TYPE
976 004736 012415 MES10 ;TEXT 'SYNC I OR E OR C'
977 004740 004737 010020 JSR PC,XTTYIN ;WAIT FOR INPUT.
978 004744 013737 010250 013056 MOV INBUF,PROC ;SAVE IT IN TEMP STORAGE
979 004752 104401 TYPE
980 004754 012260 ACRLF
981 004756 005037 013124 CLR USECLK ;CLEAR CLOCK FLAG
982 004762 012737 000001 013062 MOV #1,COUNT ;SET UP FOR '1' CONVERSION
983 004770 117737 174144 013051 CALBT2: MOVB @SWR,ADWRD2+1 ;GET CH. FROM THE SW REG.
984 004776 042737 150377 013050 BIC #150377,ADWRD2 ;CLR UNWANTED BITS, A/D WORD COMPLETE
985 005004 005037 005754 CLR ADWRD1
986 005010 017737 174124 013066 MOV @SWR,KSTOR2 ;SAVE ORIGINAL SWITCH SETTING.
987 005016 022737 000005 013056 CMP #5,PROC ;TEST SYNC SELECT
988 005024 001004 BNE 1$ ;BRANCH IF NOT 'E'
989 005026 052737 000020 005754 BIS #20,ADWRD1 ;OTHERWISE ADD 'EXT' SYNC BIT TO A/D.
990 005034 000416 BR CALB2A
991 005036 122737 000003 013056 1$: CMPB #3,PROC
992 005044 001007 BNE 2$ ;BRANCH IF NOT 'C'
993 005046 052737 000040 005754 BIS #40,ADWRD1 ;OTHERWISE ADD 'CLOCK' SYNC TO A/D
994 005054 012737 177777 013124 MOV #-1,USECLK
995 005062 000403 BR CALB2A
996 005064 052737 000001 005754 2$: BIS #1,ADWRD1
997 005072 000240 CALB2A: NOP
998 005074 012705 020470 MOV #ADBUFF,R5 ;RESET POINTER
999 005100 000240 11$: NOP
1000 005102 000240 NOP
1001 005104 000240 NOP
1002 005106 013737 013050 013246 MOV ADWRD2,$TMDAT ;LOAD A/D STATUS
1003 ;*
1004 MOV $TMDAT,$ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1005 NOP ;CLEAR STAT
1006 ;*
1007 005126 012737 000000 013246 MOV #0,$TMDAT
1008 ;*
1009 MOV $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1010 ;*
1011 MOV $TMDAT,$VCXREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1012 ;*
1013 MOV $TMDAT,$VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1014 005164 013703 001354 2$: MOV VCYREG,R3 ;LOAD Y ADDRESS
1015 005170 013704 001352 MOV VCXREG,R4 ;LOAD X ADDRESS

```

```

1016                                     ;INTEN. ON Y LOAD
1017 005174 012737 000010 013246      MOV      #10,$TMDAT
1018
1019                                     ;/*
1019                                     3$:      MOV      $TMDAT,$VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1020 005212                                     ;SAVE STATUS
1021
1021                                     ;/*
1021                                     ;/READ DEVICE REG VCSTAT,PUT DATA IN AXIS1.
1022                                     MOV      $VCSTAT,AXIS1
1023 005222 010337 005760                                     ;SAVE R3
1023                                     MOV      R3,AXIS2
1024 005226 010437 005762                                     ;SAVE R4
1024                                     MOV      R4,AXIS3
1025 005232 012737 000004 005472      MOV      #4,101$
1025                                     ;LOAD COUNT
1026 005240 012700 000160      10$:      MOV      #160,R0
1026                                     ;LOAD A CONVERSION COUNT
1027 005244 012701 000000      MOV      #0,R1
1027                                     ;CLEAR OFFSET
1028 005250 005037 005752      CLR      MARKER
1028                                     ;NOT A MARKER SAMPLE
1029 005254 004737 005512      JSR      PC,CNVTSV
1029                                     ;CONVERT AND SAVE
1030
1031 005260 012700 000020      MOV      #20,R0
1031                                     ;LOAD MARKER COUNT
1032 005264 012701 000011      MOV      #11,R1
1032                                     ;LOAD OFFSET
1033 005270 005237 005752      INC      MARKER
1033                                     ;SET MARKER SAMPLE
1034 005274 004737 005512      JSR      PC,CNVTSV
1034                                     ;CONVERT AND SAVE
1035
1036 005300 005337 005472      DEC      101$
1036                                     ;DONE ?
1037 005304 001355      BNE      10$
1037                                     ;BR IF NOT
1038
1039                                     ;CONVERT INTO DIGITS
1040 005306 012737 000077 013132      15$:      MOV      #77,TEMP1
1040                                     ;LOAD CONVERSION COUNT
1041 005314 012737 000006 011232      MOV      #6,CMPCNT
1041                                     ;LOAD SHIFT COUNT
1042 005322 004737 011070      JSR      PC,CMPTEB
1042                                     ;AVERAGE
1043 005326 013700 013170      MOV      AVRAVE,R0
1043                                     ;GET A VALUE
1044 005332 006200      ASR      R0
1045 005334 006200      ASR      R0
1046 005336 005200      INC      R0
1047 005340 006200      ASR      R0
1048 005342 010037 005474      MOV      R0,102$
1048                                     ;SAVE FOR TYPEOUT
1049 005346 012737 000004 005470      MOV      #4,100$
1049                                     ;LOAD A COUNT
1050 005354 012702 005506      MOV      #DGT4,R2
1050                                     ;LOAD A POINTER
1051 005360 000403      BR      16$
1052 005362 006200      17$:      ASR      R0
1052                                     ;MOVE RIGHT
1053 005364 006200      ASR      R0
1054 005366 006200      ASR      R0
1055 005370 010001      16$:      MOV      R0,R1
1055                                     ;GET VALUE
1056 005372 042701 177770      BIC      #177770,R1
1056                                     ;MASK
1057 005376 006301      ASL      R1
1057                                     ;X2
1058 005400 016142 005510      MOV      AND(R1),-(R2)
1058                                     ;LOAD ADDRESS
1059 005404 005337 005470      DEC      100$
1059                                     ;FINISHED ?
1060 005410 001364      BNE      17$
1060                                     ;BR IF NOT
1061
1061                                     ;DISPLAY CONVERTED DIGITS
1062 005412 012737 000001 013136      22$:      MOV      #1,TEMP3
1062                                     ;SETUP TO PRINT '1' VALUE
1063 005420 032777 002000 173512      BIT      #SW10,$SWR
1064 005426 001406      BEQ      21$
1065 005430 013746 005474      MOV      102$,-(SP)
1066 005434 104403      TYPOS
1067 005436 004 001      .BYTE 4,1
1068 005440 104401      TYPE
1069 005442 012260      ACRLF

```

```

1070 005444 004737 011574 21$: JSR PC,TSTFLG ;TEST FOR KEYBOARD INTERRUPT
1071 005450 023777 013066 173462 CMP K$TOR2, $SWR ;TEST IF SWITCH REGISTER HAS CHANGED
1072 005456 001402 BEQ 23$ ;BRANCH AND TAKE NEXT BURST OF 512 CONVERSIONS
1073 005460 000137 004770 JMP CALBT2 ;YES, COMPUTE NEW INPUT
1074 005464 000137 005072 JMP CALB2A
1075 005470 000000 100$: 0
1076 005472 000000 101$: 0
1077 005474 000000 102$: 0
1078 005476 000000 DGT0: 0
1079 005500 000000 DGT1: 0
1080 005502 000000 DGT2: 0
1081 005504 000000 DGT3: 0
1082 005506 000000 DGT4: 0
1083 005510 005764 AND: NO
1084
1085 005512 005737 005752 CNVTSV: TST MARKER ;TEST IF MARKER SAMPLE
1086 005516 001401 BEQ 4$ ;BR IF NOT
1087 005520 005301 DEC R1 ;DEC OFFSET
1088 005522 005737 013124 4$: TST USECLK ;CLOCK ENABLE
1089 005526 001416 BEQ 5$ ;BR IF NOT
1090
1091 005530 012737 177750 013246 MOV #-30, $TMDAT ;LOAD PRESET
1092
1093 ;* MOV $TMDAT, $CSB ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1094 ;* MOV #3, $TMDAT ;LOAD RATE
1095 005546 012737 000003 013246 MOV $TMDAT, $CSR ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1096
1097 5$: ;* MOV $TMDAT, $ADCS ;/ READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
1098 005564 5$: ;* MOV $ADCS, $TMDAT ;ENABLE A/D
1099
1100 ;* MOV $ADCS, $TMDAT ;/ READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
1101 005574 153737 005754 013246 BISR $ADCS, $TMDAT
1102
1103 ;* MOV $TMDAT, $ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1104 005612 CNSTS1: ;* MOV $ADCS, $TMDAT ;/ READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
1105
1106 ;* MOV $ADCS, $TMDAT ;/ READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
1107 005622 105037 013246 CLRB $TMDAT
1108
1109 ;* MOV $TMDAT, $ADCS ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1110 ;* MOV $ADCS, $TMDAT ;LOAD CONVERTED VALUE
1111
1112 ;* MOV $ADDBR, $TMDAT ;/ READ DEVICE REG ADDBR, PUT DATA IN $TMDAT.
1113 005646 013702 013246 MOV $TMDAT, R2
1114 005652 060102 ADD R1, R2 ;ADD OFFSET IF ANY
1115 005654 006302 ASL R2 ;XB
1116 005656 006302 ASL R2
1117 005660 006302 ASL R2
1118 005662 010237 013246 MOV R2, $TMDAT ;LOAD AN AXIS
1119
1120 ;* MOV $TMDAT, $VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1121 005676 010225 ;* MOV R2, (R5)+ ;SAVE IN BUFFER
1122 005700 5$: ;WAIT FOR SCOPE
1123

```

```

1124 ;* MOV @VCSTAT,$TMDAT ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
1125 005710 105737 013246 TSTB $TMDAT
1126 005714 100371 BPL SS
1127
1128 ;* MOV @VCYREG,$TMDAT ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
1129 005726 062737 000002 013246 ADD #2,$TMDAT
1130
1131 ;* MOV $TMDAT,@VCYREG ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1132 005744 005300 DEC RO ;DONE ALL CONVERSIONS ?
1133 005746 001261 BNE CNVTSV ;BR IF NOT
1134 005750 000207 RTS PC ;EXIT
1135 005752 000000
1136 005754 000101
1137 005756 000000
1138 005760 000000
1139 005762 000000
1140 005764 076 121 111 NO: .BYTE 76,121,111,105,76
1141 005767 105 076
1142 005772 .EVEN
1143
1144
1145 ;*****
1146 ;*TEST 6 REPEATABILITY TEST
1147 ;*****
1148 005772 000004 000001 001166 TST6: SCOPE
1149 005774 012737 MOV #1,$TIMES ;;DO 1 ITERATION
1150
1151 ;THIS ROUTINE TO DESIGNED TO SHOW REPEATABILITY BY TAKING A SERIES OF
1152 ;'512' CONVERSIONS, AVERAGING THEM AND THEN CATEGORIZING
1153 ;THEM IN BINS FROM THE AVERAGE PLUS & MINUS 6 COUNTS. THE ROUTINE
1154 ;REQUESTS FOR A CHANNEL OR CHANNELS, AND A COUNT SPREAD TO BE TYPED
1155 ;IN VIA THE OPERATOR. A CONTINUOUS SERIES OF CONVERSIONS ARE THEN TAKEN
1156 ;AND COMPARED AGAINST THE INPUT COUNT SPREAD. IF ALL '512' CONVERSIONS
1157 ;ARE FOUND TO BE WITHIN THE SPREAD THE NEXT CH, IS EXERCISED OTHERWISE
1158 ;THE COUNTS ARE TYPED OUT. SETTING SWITCH '10' TO A '1' WILL FORCE A PRINTOUT
1159 ;OF THE CH (S).
1160
1161 006002 012737 006014 013054 REPTST: MOV #REPT1,AVECTR ;SET UP CNTR 'A' VECTOR ADDRESS
1162 006010 104401 TYPE
1163 006012 012463 MES13 ;TEXT 'REPEATABILITY TEST'
1164 006014 005037 013146 REPT1: CLR MESPRT
1165 006020 005037 013126 CLR OPS1
1166 006024 104401 TYPE
1167 006026 012521 MES14 ;REQUEST CHANNEL (S)
1168 006030 004737 010020 JSR PC,XTTYIN ;WAIT FOR INPUT
1169 006034 004737 010266 JSR PC,BCDBIN ;CONVERT TO OCTAL
1170 006040 013737 010422 013064 MOV BCDTAB,KSTOR1 ;SAVE AS INITIAL CH.
1171 006046 013737 013064 013066 MOV KSTOR1,KSTOR2 ;ALSO SAVE AS 2ND CH. ENTRY
1172 006054 005737 010424 TST BCDTAB+2 ;TEST FOR SECOND ENTRY
1173 006060 001407 BEQ REPT2 ;BRANCH IF NO SECOND ENTRY
1174 006062 023737 010424 013064 CMP BCDTAB+2,KSTOR1 ;COMPARE ENTRY 1 TO ENTRY 2
1175 006070 100751 BMI REPT1 ;BRANCH AND RESTART IF ILLEGAL
1176 006072 013737 010424 013066 MOV BCDTAB+2,KSTOR2 ;OTHERWISE SAVE AS SECOND CH.
1177 006100 104401 REPT2: TYPE

```

```

1178 006102 012603          MES16
1179 006104 004737 010020    JSR    PC,XTTYIN      ;TEXT 'COUNT SPREAD ?'
1180 006110 004737 010266    JSR    PC,BCDBIN     ;WAIT FOR ENTRY
1181 006114 013737 010422 013070  REPT2A: MOV  BCOTAB,KSTOR3 ;SAVE IT
1182 006122 013737 013064 013072  REPT3: MOV  KSTOR1,KSTOR4 ;SAVE STARTING CH.
1183 006130 004737 011574          JSR    PC,TSTFLG     ;TEST FOR KEYBOARD FLAG
1184 006134 012737 001000 013062  MOV    #1000,COUNT   ;SET FOR '512' CONVERSIONS
1185 006142 113737 013072 013051  MOVB   KSTOR4,ADWRD2+1 ;MOV SELECTED CH. TO HIGH BYTE OF ADWORD
1186 006150 042737 100377 013050  BIC    #100377,ADWRD2 ;MASK
1187 006156 052737 000001 013050  BIS    #1,ADWRD2
1188 006164 004737 010432          JSR    PC,ADCNT      ;TAKE THE CONVERSIONS
1189 006170 004737 011054          JSR    PC,CMPTE      ;AVERAGE & COMPUTE DISTRIBUTION
1190 006174 004737 011234          JSR    PC,CATORZ
1191 006200 032777 002000 172732  BIT    #SW10,JSWR    ;TEST DATA SW10
1192 006206 001047          BNE   REPT4
1193 006210 032777 020000 172722  TSTCT4: BIT  #SW13,JSWR ;IF SET, FORCE TYPE OUT
1194 006216 001142          BNE   REPT7          ;TEST FOR INHIBIT TYPEOUT
1195 006220 022737 000004 013070  CMP    #4,KSTOR3     ;BRANCH IF SW SET
1196 006226 001005          BNE   TSTCT3        ;WAS 4 TYPED
1197 006230 022737 001000 013244  CMP    #1000,XSPRD4 ;NO, TEST FOR '3'
1198 006236 001033          BNE   REPT4         ;TOTAL COUNTS WITHIN 4 COUNTS
1199 006240 000531          BR    REPT7         ;BRANCH IF NO.
1200 006242 022737 000003 013070  TSTCT3: CMP  #3,KSTOR3 ;YES, TEST NEXT CH.
1201 006250 001005          BNE   TSTCT2        ;COUNT = TO 3
1202 006252 022737 001000 013242  CMP    #1000,XSPRD3 ;NO TEST COUNT 2
1203 006260 001022          BNE   REPT4         ;BRANCH IF COUNT NOT WITHIN 3
1204 006262 000520          BR    REPT7         ;YES, TEST NEXT CH.
1205
1206 006264 022737 000002 013070  TSTCT2: CMP  #2,KSTOR3 ;COUNT =TO 2
1207 006272 001005          BNE   TSTCT1        ;NO, TEST COUNT 1
1208 006274 022737 001000 013240  CMP    #1000,XSPRD2 ;BRANCH IF NOT WITHIN 2
1209 006302 001011          BNE   REPT4         ;YES, TEST NEXT CH.
1210 006304 000507          BR    REPT7
1211 006306 022737 000001 013070  TSTCT1: CMP  #1,KSTOR3 ;COUNT = TO 1
1212 006314 001004          BNE   REPT4         ;NO, REPORT EVEN IF NOT '0'
1213 006316 022737 001000 013236  CMP    #1000,XSPRD1 ;BRANCH IF TOTAL WITHIN 1 COUNT
1214 006324 001477          BEQ   REPT7
1215 006326 104401          REPT4: TYPE
1216 006330 012260          ACRLF
1217 006332 005737 013146          TST   MESPRT        ;TEST IF HEADER HAS BEEN TYPED
1218 006336 001002          BNE   REPTS         ;BRANCH IF YES
1219 006340 104401          TYPE
1220 006342 012622          MES19
1221 006344 104401          REPTS: TYPE
1222 006346 012260          ACRLF
1223 006350 013737 013072 006556  MOV    KSTOR4,REPT8A ;TEXT 'CH. LOW AVG. HIGH'
1224 006356 042737 177700 006556  BIC    #177700,REPT8A ;CARRIAGE RETURN, LINE FEED
1225 006364 013746 006556          REPT8: MOV  REPT8A,-(SP) ;MOV. CH.
1226 006370 104403          TYPOS
1227 006372 002          .BYTE 2,1
1228 006374 004737 007474          JSR   PC,XSPACE
1229 006400 013746 013154          MOV  ADLOW,-(SP)    ;SAVE LOW VALUE
1230 006404 104403          TYPOS
1231 006406 004          .BYTE 4,1

```

1232	006410	004737	007474			JSR	PC,XSPACE		
1233	006414	013746	013170			MOV	AVRAGE,-(SP)	;SAVE AVERAGE	
1234	006420	104403				TYPOS			
1235	006422	004	001			.BYTE	4,1		
1236	006424	004737	007474			JSR	PC,XSPACE		
1237	006430	013746	013152			MOV	ADHIGH,-(SP)		
1238	006434	104403				TYPOS			
1239	006436	004	001			.BYTE	4,1		
1240	006440	005737	013146			TST	MESPRT		
1241	006444	001002				BNE	REPT6		
1242	006446	104401				TYPE			
1243	006450	012645				MES20		;PRINT 'COUNT SPREAD' HEADER	
1244	006452	052737	000007	013146	REPT6:	BIS	#7,MESPRT	;INHIBIT OTHER HEADERS	
1245	006460	022737	001000	013220		CMP	#1000,AVGCNT	;TEST IF ALL COUNTS WERE AT AVG.	
1246	006466	000240				NOP		; <BEQ REPT7> BRANCH TO NEXT CH. IF YES.	
1247	006470	104401				TYPE			
1248	006472	012260				ACRLF			
1249	006474	012704	013204			MOV	#ORLOW,R4		
1250	006500	012402			REPT6A:	MOV	(R4)+,R2		
1251	006502	004737	010656			JSR	PC,DECPRT	;TYPE OUT COUNT SPREAD	
1252	006506	022704	013236			CMP	#XSPRD1,R4	;TEST FOR DONE	
1253	006512	001372				BNE	REPT6A	;BRANCH IF NO AND TYPE NEXT COUNT	
1254	006514	005777	172420			TST	JSWR		
1255	006520	100001				BPL	REPT7		
1256	006522	000000				HALT		;REPEATABILITY ERROR	
1257	006524	013704	013072		REPT7:	MOV	KSTOR4,R4		
1258	006530	042704	177700			BIC	#177700,R4		
1259	006534	023704	013066			CMP	KSTOR2,R4	;TESTED ALL CH.(S)?	


```

1260 006540 001404          BEQ      REPT7A
1261 006542 005237 013072    INC      KSTOR4          ;TEST NEXT CHANNEL
1262 006546 000137 006130    JMP      REPT3
1263 006552 000137 006122    REPT7A: JMP      REPT2A
1264
1265 006556 000000          REPT3A: 0
1266
1267
1268 ;:*****
1269 ;:TEST 7          RECOVERY TEST
1270 ;:*****
1271 ST7:  SCOPE
1272 ;THIS TEST IS DESIGNED TO TEST INTER-CHANNEL SETTLING OF THE A/D CONVERTER BY
1273 ;ACCEPTING TWO (2) CHANNELS FROM THE TELETYPE AND THEN TAKING A
1274 ;SERIES OF EIGHT CONVERSIONS ON EACH CHANNEL AND TYPING OUT THE CON-
1275 ;VERSION VALUES OF THE 2ND CHANNEL IN THE ORDER THEY WERE TAKEN.
1276 006562 012737 006574 013054 RECVRY: MOV      #RECVY1,AVECTR ;SET UP THE 'A' RETURN ADDRESS
1277 006570 104401          TYPE
1278 006572 012356          MES8
1279 006574 104401          RECVY1: TYPE      ;TEXT 'RECOVERY TEST'
1280 006576 012521          MES14          ;REQUEST CHANNELS
1281 006600 004737 010020          JSR      PC,XTTYIN          ;WAIT FOR INPUT
1282 006604 004737 010266          JSR      PC,BCDBIN          ;CONVERT TO OCTAL
1283 006610 013737 010422 013064    MOV      BCDTAB,KSTOR1      ;SAVE 1ST CH.
1284 006616 013737 010424 013066    MOV      BCDTAB+2,KSTOR2    ;SAVE 2ND CH.
1285 006624 004737 011574          RECVY2: JSR      PC,TSTFLG          ;CHECK FOR KEYBOARD FLAG
1286 006630 012737 000020 007144    MOV      #16.,10$          ;LOAD COUNT
1287 006636 005037 007146          CLR      11$
1288 006642 012737 000010 013136    MOV      #10,TEMP3          ;SET UP TO PRINT EIGHT
1289 006650 012737 000010 013062    MOV      #10,COUNT          ;SETUP TO TAKE '8' CONVERSIONS
1290 006656 113737 013064 013051    1$:  MOVB   KSTOR1,ADWRD2+1    ;LOAD 1ST CH.
1291 006664 042737 140377 013050    BIC      #140377,ADWRD2
1292 006672 052737 000001 013050    BIS      #1,ADWRD2
1293 006700 005237 013120          INC      DELAY              ;NO DELAY BETWEEN MUX CHANGE AND CONVERT
1294 006704 004737 010432          JSR      PC,ADCNVT          ;TAKE THE CONVERSIONS
1295 006710 113737 013066 013051    MOVB   KSTOR2,ADWRD2+1      ;SET UP 2ND CH.
1296 006716 042737 140377 013050    BIC      #140377,ADWRD2
1297 006724 052737 000001 013050    BIS      #1,ADWRD2
1298 006732 005237 013120          INC      DELAY              ;NO DELAY BETWEEN MUX CHANGE AND CONVERT
1299 006736 004737 010432          JSR      PC,ADCNVT          ;TAKE 2ND SERIES OF CONVERSIONS
1300 006742 013701 007146          MOV      11$,R1            ;LOAD A POINTER
1301 006746 012700 020470          MOV      #ADBUFF,RO        ;LOAD POINTER
1302 006752 012061 020532          MOV      (RO)+,ADTB0(R1)
1303 006756 012061 020574          MOV      (RO)+,ADTB1(R1)
1304 006762 012061 020636          MOV      (RO)+,ADTB2(R1)
1305 006766 012061 020700          MOV      (RO)+,ADTB3(R1)
1306 006772 012061 020742          MOV      (RO)+,ADTB4(R1)
1307 006776 012061 021004          MOV      (RO)+,ADTB5(R1)
1308 007002 012061 021046          MOV      (RO)+,ADTB6(R1)
1309 007006 012061 021110          MOV      (RO)+,ADTB7(R1)
1310 007012 062737 000002 007146    ADD      #2,11$            ;UPDATE POINTER
1311 007020 005337 007144          DEC      10$
1312 007024 001314          BNE     1$
1313

```

```

1314 007026 012700 000010      MOV      #8, R0          ;LOAD COUNT
1315 007032 012702 020470      MOV      #ADBUFF,R2     ;LOAD POINTER
1316 007036 012701 000000      MOV      #0, R1
1317 007042 012737 000017 013132 2$:  MOV      #15, TEMP1     ;LOAD TEMP
1318 007050 012737 000004 011232  MOV      #4, CMPCNT     ;LOAD AVRG. COUNT
1319 007056 016104 007152      MOV      LADTB(R1),R4   ;GET POINTER
1320 007062 004737 011074      JSR      PC, CMPTA      ;AVERAGE
1321 007066 013722 013170      MOV      AVRA, (R2)+    ;SAVE AVERAGE
1322 007072 005721      TST      (R1)+         ;UPDATE POINTER
1323 007074 005300      DEC      R0
1324 007076 001361      BNE      2$            ;BR IF NOT DONE
1325
1326
1327 007100 032777 002000 172032      BIT      #SW10, @SWR    ;TEST BIT 10
1328 007106 001246      BNE      RECVY2
1329 007110 104401      TYPE
1330 007112 012407      MES9
1331 007114 013746 013066      MOV      KSTOR2, -(SP) ;TEXT 'CH.'
1332 007120 104403      TYPOS
1333 007122 002 001      .BYTE 2,1
1334 007124 012737 000002 007524      MOV      #2, SPACEX
1335 007132 004737 007474      JSR      PC, XSPACE
1336 007136 004737 007526      JSR      PC, XPRTAV
1337 007142 000630      BR RECVY2             ;PRINT VALUES OF 2ND CH.
1338 007144 000000      10$: 0                ;DO IT AGAIN
1339 007146 000000      11$: 0
1340 007150 000000      12$: 0
1341
1342 007152 020532      LADTB: ADTB0
1343 007154 020574      ADTB1
1344 007156 020636      ADTB2
1345 007160 020700      ADTB3
1346 007162 020742      ADTB4
1347 007164 021004      ADTB5
1348 007166 021046      ADTB6
1349 007170 021110      ADTB7
1350
1351 *****
1352 ;*TEST 10 LOAD DIFFERENT NUMBERS INTO DIFFERENT REG.
1353 *****
1353 007172 000004      TST10: SCOPE
1354 007174 012737 000040 001166      MOV      #40, $TIMES   ;;DO 40 ITERATIONS
1355 007202 012737 007210 013054      RESTST: MOV      #20$, AVECTR
1356 007210 104401      20$:  TYPE
1357 007212 011674      MES2
1358 007214 005037 007450      CLR      71$          ;TYPE HEADER ABOUT TEST
1359
1360 007220 012737 000100 007446      11$:  MOV      #100, 70$ ;RESET ;LOAD COUNTER
1361 007226 000240      10$:  NOP
1362 007230 013700 001356      2$:  MOV      CSC, R0     ;LOAD STARTING ADDRESS
1363 007234 062700 000002      ADD      #2, R0
1364 007240 012701 007472      MOV      #BUFNUM+20, R1 ;LOAD STARTING TABLE ADDRESS
1365 007244 012702 000010      MOV      #8, R2        ;LOAD COUNT
1366 007250 010037 013246      1$:  MOV      R0, $TMDAT
1367 007254 162737 000002 013246      SUB      #2, $TMDAT

```

```

1368 007262 162700 000002      SUB      #2,R0
1369 007266 014137 001124      MOV      -(R1),SGDDAT
1370
1371          ;*      MOV      SGDDAT,@STMDAT ;/ PUT DATA FROM SGDDAT TO DEVICE REG STMDAT
1372 007302 005302          DEC      R2 ;DONE ALL ?
1373 007304 001361          BNE     1$ ;BR IF NOT
1374
1375 007306 013700 001340      MOV      ADCS,R0 ;LOAD STARTING POINTER
1376 007312 012701 007452      MOV      #BUFNUM,R1 ;LOAD STARTING POINTER <EXPECTED>
1377 007316 012702 000010      MOV      #B.,R2 ;LOAD # OF REG
1378
1379 007322 011137 001124      3$:     MOV      (R1),SGDDAT ;READ REG
1380 007326 010037 013246      MOV      R0,STMDAT
1381
1382          ;*      MOV      @STMDAT,$BDDAT ;/READ DEVICE REG STMDAT,PUT DATA IN $BDDAT.
1383 007342 023737 001124 001126      CMP      SGDDAT,$BDDAT ;COMPARE
1384 007350 001403          BEQ     4$ ;;BR IF EQUAL
1385 007352 010037 007472      MOV      R0,BUFADR ;SAVE BUS ADDRESS
1386 007356 104001          ERROR   1 ;INCORRECT DATA, REG. WAS CHANGED IN ERROR
1387 007360 005721          4$:     TST      (R1)+ ;UPDATE POINTERS
1388 007362 062700 000002      ADD      #2,R0
1389 007366 005302          DEC      R2 ;DONE ALL REG ?
1390 007370 001354          BNE     3$ ;BR IF NOT
1391 007372 012737 000000 013246      MOV      #0,STMDAT
1392
1393          ;*      MOV      $TMDAT,@VCSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1394 007410 004737 011574      JSR      PC,TSTFLG
1395 007414 005337 007446      DEC      70$ ;DONE
1396 007420 001302          BNE     10$ ;NO
1397 007422 104401          TYPE
1398 007424 004127          SENDMG
1399 007426 005237 007450      INC      71$
1400 007432 013746 007450      MOV      71$,-(SP)
1401 007436 104405          TYPDS
1402 007440 104401 004124          TYPE   $ENULL
1403 007444 000665          BR      11$ ;LOOP BACK
1404 007446 000000          70$:   0
1405 007450 000000          71$:   0
1406
1407          ;# TO BE LOADED INTO DIFFERENT REG
1408
1409 007452 027560      BUFNUM: 27560 ;A TO D STATUS
1410 007454 000000          0 ;A TO D BUFFER
1411 007456 140736          140736 ;CLOCK STATUS
1412 007460 000377          377 ;CLOCK PRESET
1413 007462 007214          7214 ;VC STATUS
1414 007464 001252          1252 ;VC X POS
1415 007466 000525          525 ;VC Y POS
1416 007470 000377          377 ;CLOCK COUNTER
1417
1418 007472 170400      BUFADR: 170400 ;BUS ADDRESS OF REG IN ERROR
1419
1420          ;*****
1421          ;SUBROUTINE TO ISSUE N SPACES

```

```

1422 ;N IS ONE PLUS VALUE CONTAINED IN SPACEX
1423 ;SPACEX IS CLEARED WITHIN THE SUBROUTINE, SO THAT A CALL ON
1424 ;SPACE WITHOUT LOADING SPACEX ISSUES ONLY ONE SPACE
1425
1426 007474 105777 171450 XSPACE: TSTB @STPS ;WAIT FOR TTY READY
1427 007500 100375 BPL -4
1428 007502 012777 000240 171442 MOV #240,@STPB ;OUTPUT A SPACE
1429 007510 005337 007524 DEC SPACEX ;DECREMENT COUNT
1430 007514 003367 BGT XSPACE ;LOOP IF NOT DONE
1431 007516 005037 007524 CLR SPACEX ;RESET COUNT TO ZERO
1432 007522 000207 RTS PC ;RETURN
1433 007524 000000 SPACEX: 0
1434
1435 ;SUBROUTINE TO TYPE OUT AVERAGES FOR THE RECOVERY TEST
1436
1437 007526 012737 020470 007536 XPRTAV: MOV #ADBUFF,AVGTAB
1438 007534 013746 XPTAI: MOV @PC+,-(SP)
1439 007536 020470 AVGTAB: ADBUFF
1440 007540 104403 TYPDS
1441 007542 004 001 .BYTE 4,1
1442 007544 062737 000002 007536 ADD #2,AVGTAB
1443 007552 012737 000002 007524 MOV #2,SPACEX
1444 007560 004737 007474 JSR PC,XSPACE
1445 007564 005337 013136 DEC TEMP3
1446 007570 001361 BNE XPTAI
1447 007572 000207 RTS PC
1448
1449 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1450
1451 ;*****
1452 ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1453 ;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1454 ;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1455 ;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1456 ;REPLACED WITH SPACES.
1457 ;CALL:
1458 ;* MOV NUM,-(SP) ;:PUT THE BINARY NUMBER ON THE STACK
1459 ;* TYPDS ;:GO TO THE ROUTINE
1460
1461 007574 STYPDS:
1462 007574 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
1463 007576 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
1464 007600 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
1465 007602 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
1466 007604 010546 MOV R5,-(SP) ;:PUSH R5 ON STACK
1467 007606 012746 020200 MOV #20200,-(SP) ;:SET BLANK SWITCH AND SIGN
1468 007612 016605 000020 MOV 20(SP),R5 ;:GET THE INPUT NUMBER
1469 007616 100004 BPL 1$ ;:BR IF INPUT IS POS.
1470 007620 005405 NEG R5 ;:MAKE THE BINARY NUMBER POS.
1471 007622 112766 000055 000001 MOVB #'-,1(SP) ;:MAKE THE ASCII NUMBER NEG.
1472 007630 005000 1$: CLR R0 ;:ZERO THE CONSTANTS INDEX
1473 007632 012703 010010 MOV #SDBLK,R3 ;:SETUP THE OUTPUT POINTER
1474 007636 112723 000040 MOVB #'',(R3)+ ;:SET THE FIRST CHARACTER TO A BLANK
1475 007642 005002 2$: CLR R2 ;:CLEAR THE BCD NUMBER

```

```

1476 007644 016001 010000      MOV      $DTBL(R0),R1      ;; GET THE CONSTANT
1477 007650 160105      3$: SUB      R1,R5          ;; FORM THIS BCD DIGIT
1478 007652 002402      BLT      4$              ;; BR IF DONE
1479 007654 005202      INC      R2              ;; INCREASE THE BCD DIGIT BY 1
1480 007656 000774      BR       3$
1481 007660 060105      4$: ADD      R1,R5          ;; ADD BACK THE CONSTANT
1482 007662 005702      TST      R2              ;; CHECK IF BCD DIGIT=0
1483 007664 001002      BNE      5$              ;; FALL THROUGH IF 0
1484 007666 105716      TSTB     (SP)            ;; STILL DOING LEADING 0'S?
1485 007670 100407      BMI     7$              ;; BR IF YES
1486 007672 106316      5$: ASLB     (SP)            ;; MSD?
1487 007674 103003      BCC      6$              ;; BR IF NO
1488 007676 116663 000001 177777  MOVB     1(SP),-1(R3)     ;; YES--SET THE SIGN
1489 007704 052702 000060      6$: BIS      #'0,R2       ;; MAKE THE BCD DIGIT ASCII
1490 007710 052702 000040      7$: BIS      #' ,R2       ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
1491 007714 110223      MOVB     R2,(R3)+       ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
1492 007716 005720      TST      (R0)+          ;; JUST INCREMENTING
1493 007720 020027 000010      CMP      R0,#10         ;; CHECK THE TABLE INDEX
1494 007724 002746      BLT      8$              ;; GO DO THE NEXT DIGIT
1495 007726 003002      BGT      8$              ;; GO TO EXIT
1496 007730 010502      MOV      R5,R2          ;; GET THE LSD
1497 007732 000764      BR       6$              ;; GO CHANGE TO ASCII
1498 007734 105726      8$: TSTB     (SP)+          ;; WAS THE LSD THE FIRST NON-ZERO?
1499 007736 100003      BPL      9$              ;; BR IF NO
1500 007740 116663 177777 177776  MOVB     -1(SP),-2(R3)   ;; YES--SET THE SIGN FOR TYPING
1501 007746 105013      9$: CLRB     (R3)         ;; SET THE TERMINATOR
1502 007750 012605      MOV      (SP)+,R5       ;; POP STACK INTO R5
1503 007752 012603      MOV      (SP)+,R3       ;; POP STACK INTO R3
1504 007754 012602      MOV      (SP)+,R2       ;; POP STACK INTO R2
1505 007756 012601      MOV      (SP)+,R1       ;; POP STACK INTO R1
1506 007760 012600      MOV      (SP)+,R0       ;; POP STACK INTO R0
1507 007762 104401 010010      TYPE     $DBLK          ;; NOW TYPE THE NUMBER
1508 007766 016666 000002 000004  MOV      2(SP),4(SP)    ;; ADJUST THE STACK
1509 007774 012616      MOV      (SP)+,(SP)
1510 007776 000002      RTI
1511 010000 023420      $DTBL: 1000.           ;; RETURN TO USER
1512 010002 001750      1000.
1513 010004 000144      100.
1514 010006 000012      10.
1515 010010 000004      $DBLK: .BLKW 4
1516      ;KEYBOARD SERVICE ROUTINE
1517
1518 010020 012704 010250      XTTYIN: MOV      #INBUF,R4      ;; SETUP CHARACTER BUFFER
1519 010024 042777 000100 171112  BIC      #BIT6,$STKS
1520 010032 005037 013060      CLR      CHRCNT         ;; CLEAR CHARACTER COUNTER
1521 010036 005037 010250      CLR      INBUF
1522 010042 005037 010252      CLR      INBUF+2
1523 010046 105777 171072      11$: TSTB     $STKS         ;; CHARACTER READY?
1524 010052 100375      BPL      11$           ;; NO, WAIT IT OUT
1525 010054 017701 171066  MOV      $STKB,R1       ;; SAVE CHARACTER
1526 010060 042701 177600      BIC      #177600,R1    ;; STRIP PARITY BIT
1527 010064 120127 000060      CMPB     R1,#60        ;; IS IT A SPECIAL CHARACTER ?
1528 010070 100420      BMI     14$           ;; YES, TEST IT
1529 010072 122701 000137      CMPB     #137,R1

```

1530	010076	100415			BMI	14\$		
1531	010100	010124			12\$: MOV	R1,(R4)+		;SAVE CHARACTER
1532	010102	005237	013060		INC	CHRCNT		;INCREMENT THE CHARACTER COUNT.
1533	010106	022737	000006	013060	CMP	#6,CHRCNT		
1534	010114	100451			BMI	4\$		
1535	010116	105777	171026		13\$: TSTB	2\$TPS		;ECHO CHARACTER
1536	010122	100375			BPL	13\$		
1537	010124	110177	171022		MOV	R1,2\$TPB		
1538	010130	000746			BR	11\$;WAIT FOR NEXT CHARACTER
1539					;SUBROUTINE TO TEST FOR SPECIAL CHARACTERS: '↑A', '↑C', '↑G', 'CR', ',', ' OR 'RUBOUT'			
1540								
1541	010132	122701	000001		14\$: CMPB	#1,R1		;CHAR.= '↑A' ?
1542	010136	001005			BNE	1\$;NO, NOT '↑A'
1543	010140	104401			TYPE			;ECHO '↑A'
1544	010142	012253			CNTRLA			
1545	010144	005726			TST	(SP)+		;RESTORE SP
1546	010146	000177	002702		JMP	2\$VECTR		;YES, EXIT VIA '↑A' VECTOR ADDRESS.
1547	010152	122701	000003		1\$: CMPB	#3,R1		;CHAR.= '↑C' ?
1548	010156	001003			BNE	2\$;NO, NOT '↑C'
1549	010160	005726			TST	(SP)+		
1550	010162	000137	002270		JMP	INIT2		;YES, EXIT TO MONITOR
1551	010166	122701	000137		2\$: CMPB	#137,R1		;CHAR.= 'RUBOUT' ?
1552	010172	001011			BNE	3\$;IGNORE CHAR. & EXIT
1553	010174	005737	013060		TST	CHRCNT		;IS RUBOUT LEGAL?
1554	010200	001722			BEQ	11\$;NO, IGNORE IT
1555	010202	005337	013060		DEC	CHRCNT		
1556	010206	012701	000134		MOV	#134,R1		;TYPE '\ ' TO INDICATE RUBOUT
1557	010212	005744			TST	-(R4)		;POP OFF LAST CHARACTER
1558	010214	000740			BR	13\$;WAIT FOR NEXT CHARACTER
1559	010216	122701	000054		3\$: CMPB	#54,R1		;TEST FOR ' '
1560	010222	001726			BEQ	12\$;LEGAL CHAR. SAVE IT
1561	010224	122701	000015		CMPB	#15,R1		;=TO 'CARRIAGE RETURN' TO TERMINATE?
1562	010230	001003			BNE	4\$;NO, CONTINUE
1563	010232	104401			TYPE			;YES, TYPE 'CR-LF'
1564	010234	012260			ACRLF			
1565	010236	000207			RTS	PC		;EXIT
1566	010240	104407			4\$: CKSWR			;TEST FOR CTRL G
1567	010242	104401			TYPE			;OTHERWISE TYPE '??'
1568	010244	012267			QMARK			
1569	010246	000664			BR	XTTYIN		;WAIT FOR NEW ENTRY
1570	010250	000000			INBUF:	0		;CHARACTER STORAGE BUFFER
1571								
1572								
1573		010266						
1574					;SUBROUTINE WILL CONVERT 'N' WORDS (SEPARATED VIA COMMA'S)			
1575					;WHICH WERE STORED IN A TABLE VIA 'TTYIN' TO OCTAL AND STORE THEM.			
1576								
1577	010266	012704	010250		BCDBIN:	MOV #INBUF,R4		;SETUP ASCII STORAGE TABLE
1578	010272	012703	010422		MOV	#BCDTAB,R3		;TABLE FOR STORAGE OF CONVERTED WORDS
1579	010276	005037	010424		CLR	BCDTAB+2		
1580	010302	005001			BCDBN1:	CLR R1		;REG. TO STORE RUNNING TOTAL
1581	010304	005002			CLR	R2		;TEMP. STORAGE FOR 'R1'
1582	010306	005737	013060		BCDBN2:	TST CHRCNT		;END OF DATA?
1583	010312	003424			BLE	BCDEND		;YES, EXIT

```

1584 010314 005337 013060      DEC      CHRCNT      ; DECREMENT CHARACTER COUNTER
1585 010320 122714 000054      CMPB     #54,(R4)   ; IS CHARACTER = TO ' '?
1586 010324 001417      BEQ      BCDEND     ; YES, DECODE NEW WORD
1587 010326 121427 000060      CMPB     (R4),#60
1588 010332 002425      BLT      BCDERR     ; TEST FOR LEGAL NO.
1589 010334 021427 000067      CMP      (R4),#67
1590 010340 003022      BGT      BCDERR
1591 010342 042714 177770      BIC      #177770,(R4) ; STRIP NO.
1592 010346 012400      MOV      (R4)+,R0   ; SAVE NO. IN R0.
1593 010350 010102      MOV      R1,R2     ; SAVE CURRENT TOTAL
1594 010352 006301      ASL      R1        ; NX2
1595 010354 006301      ASL      R1        ; NX4
1596 010356 006301      ASL      R1        ; NX8
1597 010360 060001      ADD      R0,R1     ; N+NEW NO.
1598 010362 000751      BR       BCDEND
1599 010364 020127 000077      BCDEND: CMP      R1,#77
1600 010370 003006      BGT      BCDERR
1601 010372 005724      TST      (R4)+     ; UPDATE BUFFER
1602 010374 010123      MOV      R1,(R3)+  ; SAVE CONVERTED VALUE & SETUP TO SAVE NEXT
1603 010376 005737 013060      TST      CHRCNT    ; FINISHED?
1604 010402 001337      BNE      BCDEN1    ; NO, CONVERT NEXT WORD
1605 010404 000207      RTS      PC        ; YES, EXIT
1606 010406 104401      BCDERR: TYPE      QMARK ; TYPE '?'
1607 010410 012267      JSR      PC,XITYIN ; TO BE TYPED ON QUESTIONABLE ENTRIES.
1608 010412 004737 010020      JMP      BCDTAB    ; OCTAL STORAGE TABLE
1609 010416 000137 010266      BCDTAB: 0
1610 010422 000000      0
1611 010424 000000      0
1612 010426 000000      0
1613 010430 000000      0
1614
1615      ; SUBROUTINE TO TAKE 'N' CONVERSIONS AND STORE THEM IN AN A/D BUFFER. ROUTINE
1616      ; IS ENTERED WITH 'N' IN COUNT AND THE CH TO BE CONVERTED IN 'ADWORD'.
1617      ; ENTERING WITH DELAY = 0 CAUSES DELAY BETWEEN MUX LOAD AND START OF
1618      ; FIRST CONVERSION
1619 010432 005077 167340      ADCNVT: CLR      @PSW
1620 010436 013737 013062 013132      MOV      COUNT,TEMP1 ; SET UP NO. OF CONVERSIONS TO BE TAKEN
1621 010444 012704 020470      MOV      @ADBUFF,R4  ; SET UP BUFFER ADDRESS.
1622 010450 005737 013120      TST      DELAY       ; CHECK IF DELAY SET
1623 010454 001037      BNE      3$         ; BR IF INHIBIT DELAY
1624
1625      ;*      MOV      @ADCS,$TMDAT ; / READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
1626 010466 113737 013051 013247      MOV      ADWORD2+1,$TMDAT+1
1627
1628      ;*      MOV      $TMDAT,@ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1629 010504 013737 010650 013122      MOV      CPTIME,DELAY1 ; SET UP TIMER
1630 010512 005337 013122      4$:      DEC      DELAY1    ; LOOP
1631 010516 001375      BNE      4$
1632
1633      ;*      MOV      @ADCS,$TMDAT ; / READ DEVICE REG ADCS, PUT DATA IN $TMDAT.
1634      ; LOAD LOW BYTE
1635 010530 113737 013050 013246      MOV      ADWORD2,$TMDAT
1636
1637      ;*      MOV      $TMDAT,@ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS

```

```

1638 010546 000411          BR      1$
1639 010550 000240          2$:   NOP
1640 010552 000240          NOP
1641 010554 013737 013050 013246 3$:   MOV      ADWRD2,$TMDAT ;LOAD CONTROL
1642                                     ;*   MOV      $TMDAT,$ADCS ; / PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1643                                     ;LOAD CH. & START CONVERT
1644                                     ;WAIT FOR DONE
1645 010572          1$:
1646                                     ;*   MOV      $ADCS,$TMDAT ; /READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
1647                                     TSTB   $TMDAT
1648 010602 105737 013246          BPL      1$ ;DELAY
1649 010606 100371          ;SAVE DATA
1650
1651                                     ;*   MOV      $ADDBR,$TMDAT ; /READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
1652                                     MOV     $TMDAT,(R4)+
1653 010620 013724 013246          DEC     TEMP1 ;DECREMENT COUNTER
1654 010624 005337 013132          BGT     2$ ;IF NOT '0' TAKE NEXT CONVERSION
1655 010630 003347
1656 010632 005037 013120          CLR     DELAY
1657 010636 000207          RTS     PC
1658
1659 010640 002000          CPDLAY: 2000 ;PDP-11/05
1660 010642 002400          ;PDP-11/20
1661 010644 003500          ;PDP-11/40
1662 010646 006000          ;PDP-11/45
1663
1664 010650 002000          CPTIME: 2000
1665 010652 000001          CPTYPE: 1
1666 010654 170401          ADCS1: 170401
1667
1668                                     ;PRINT DECIMAL VALUE IN R2
1669
1670
1671 010656 005077 167114          DECPRT: CLR     $PSW
1672 010662 012737 177774 011036          MOV     #-4,DIGCNT
1673 010670 012737 011044 011042          MOV     $DECPNT+2,DECPNT
1674 010676 012737 000240 011040          MOV     #240,ZERO
1675 010704 012737 177777 011034          TYPT1: MOV     #-1,DIGIT
1676 010712 005237 011034          TYPT2: INC     DIGIT
1677 010716 167702 000120          SUB     $DECPNT,%2
1678 010722 100373          BPL     TYPT2
1679 010724 067702 000112          ADD     $DECPNT,%2
1680 010730 004737 010754          JSR     PC,DECOU
1681 010734 005237 011036          INC     DIGCNT
1682 010740 001001          BNE     TYPT3
1683 010742 000207          RTS     PC
1684 010744 062737 000002 011042          TYPT3: ADD     #2,DECPNT
1685 010752 000754          BR      TYPT1
1686 010754 005737 011034          DECOU: TST     DIGIT
1687 010760 001010          BNE     DEC1
1688 010762 022737 177777 011036          CMP     #-1,DIGCNT
1689 010770 001404          BEQ     DEC1
1690 010772 013737 011040 011034          MOV     ZERO,DIGIT
1691 011000 000406          BR      DEC2

```


1692 011002 012737 000260 011040
 1693 011010 052737 000260 011034
 1694 011016 105777 170126
 1695 011022 100375
 1696 011024 013777 011034 170120
 1697 011032 000207
 1698 011034 000000
 1699 011036 000000
 1700 011040 000240
 1701 011042 011044
 1702 011044 001750
 1703 011046 000144
 1704 011050 000012
 1705 011052 000001

DEC1: MOV #260,ZERO
 BIS #260,DIGIT
 DEC2: TSTB @STPB
 BPL -4
 MOV DIGIT,@STPB
 RTS 7
 DIGIT: 0
 DIGCNT: 0
 ZERO: 240
 DECPNT: .+2
 1000.
 100.
 10.
 1.

; COMPUTE THE RESULTS OF 512 CONVERSIONS AS HIGH, LOW AND AVERAGE

1706
 1707
 1708
 1709 011054 012737 000777 013132
 1710 011062 012737 000011 011232
 1711 011070 012704 020470
 1712 011074 005037 013150
 1713 011100 012437 013170
 1714 011104 013737 013170 013152
 1715 011112 013737 013170 013154
 1716 011120 012437 013134
 1717 011124 023737 013134 013152
 1718 011132 003403
 1719 011134 013737 013134 013152
 1720 011142 023737 013134 013154
 1721 011150 003003
 1722 011152 013737 013134 013154
 1723 011160 063737 013134 013170
 1724 011166 005537 013150
 1725 011172 005337 013132
 1726 011176 001350
 1727 011200 013737 011232 013132
 1728 011206 006237 013150
 1729 011212 006037 013170
 1730 011216 005337 013132
 1731 011222 001371
 1732 011224 005537 013170
 1733 011230 000207

CMPTE: MOV #777,TEMP1 ; SET UP TO COMPARE '511' NUMBERS
 MOV #11,CMPCNT ; LOAD COUNTER
 CMPTEB: MOV #ADBUFF,R4 ; LOAD STARTING ADDRESS
 CMPTEA: CLR HIORDV ; CLR HI ORDER DIVIDEND
 MOV (R4)+,AVRAGE ; STORE 1ST VALUE AS AVERAGE
 MOV AVRAGE,ADHIGH ; HIGH
 MOV AVRAGE,ADLOW ; & LOW
 GETDAT: MOV (R4)+,TEMP2
 CMP TEMP2,ADHIGH ; IS NEW NO. GREATER THAN OLD NO. ?
 BLE TSLO ; BRANCH IF NOT GREATER
 MOV TEMP2,ADHIGH ; OTHERWISE SAVE AS NEW HIGH
 TSLO: CMP TEMP2,ADLOW
 BGT TAGA
 MOV TEMP2,ADLOW ; OTHERWISE SAVE AS NEW LOW
 TAGA: ADD TEMP2,AVRAGE ; ADD LOW ORDER
 ADC HIORDV ; ADD CARRY TO HI ORDER
 DEC TEMP1
 BNE GETDAT ; 512 ADDITIONS?
 MOV CMPCNT,TEMP1 ; YES, DIVIDE/512
 AVGDAT: ASR HIORDV ; SHIFT CARRY BIT INTO LO ORDER
 ROR AVRAGE
 DEC TEMP1
 BNE AVGDAT ; DONE?
 ADC AVRAGE ; YES, ADD REMAINDER TO LO ORDER
 RTS PC

CMPCNT: 11
; SUBROUTINE TO CALCULATE THE PLUS & MINUS 5 COUNT LIMITS FROM AN AVERAGE

1734
 1735 011232 000011
 1736
 1737
 1738 011234 012737 000005 013132
 1739 011242 013737 013170 013134
 1740 011250 012703 013172
 1741 011254 005237 013134
 1742 011260 013723 013134
 1743 011264 005337 013132
 1744 011270 001371
 1745

CATORZ: MOV #5,TEMP1 ; MOV AVER. TO WORK AREA
 MOV AVRAGE,TEMP2 ; SETUP DISTRIBUTION TABLE (POS.)
 #AVERP1,R3
 FILE1: INC TEMP2 ; A=A+1
 MOV TEMP2,(R3)+ ; SAVE A+1
 DEC TEMP1 ; SAVED '5' COUNTS?
 BNE FILE1 ; BRANCH IF NO
 ; SET UP TABLE OF AVG. -1 TO -5

```

1746 011272 012737 000005 013132      MOV      #5,TEMP1
1747 011300 013737 013170 013134      MOV      AVRAVE,TEMP2      ;MOV AVG. TO WORK AREA.
1748 011306 012703 013170      MOV      #AVRAGE,R3      ;SET UP DISTRIBUTION TABLE NEG.
1749 011312 005337 013134      FILE2:  DEC      TEMP2      ;A=1-1
1750 011316 013743 013134      MOV      TEMP2,-(R3)      ;SAVE 'A-1'
1751 011322 005337 013132      DEC      TEMP1      ;SAVED '5' COUNTS?
1752 011326 001371      BNE      FILE2      ;BRANCH IF NO
1753
1754      ;CATEGORIZE THE COUNT SPREAD AS '+6 & -6' COUNTS FROM THE AVERAGE
1755
1756 011330 012703 013204      MOV      #ORLOW,R3      ;CLEAR COUNTS
1757 011334 005023      CATR1:  CLR      (R3)+
1758 011336 022703 013236      CMP      #ORHIGH+2,R3      ;FINISHED?
1759 011342 001374      BNE      CATR1      ;NO, CLEAR NEXT COUNTER
1760 011344 012737 001001 013132      MOV      #1001,TEMP1      ;COMPARE '512' COUNTS
1761 011352 012700 020470      MOV      #ADBUFF,R0      ;SET UP A/D BUFFER
1762 011356 005337 013132      CATR2:  DEC      TEMP1
1763 011362 001437      BEQ      CATR5      ;EXIT IF '0'
1764 011364 012037 013134      MOV      (R0)+,TEMP2
1765 011370 023737 013202 013134      CMP      AVERP5,TEMP2
1766 011376 100423      BMI      OVRHI
1767 011400 023737 013134 013156      CMP      TEMP2,AVERM5
1768 011406 100422      BMI      OVRLO
1769 011410 005001      CLR      R1
1770 011412 012702 013156      MOV      #AVERM5,R2
1771 011416 022237 013134      CATR3:  CMP      (R2)+,TEMP2
1772 011422 001405      BEQ      CATR4
1773 011424 005201      INC      R1
1774 011426 022701 000013      CMP      #13,R1
1775 011432 001371      BNE      CATR3
1776 011434 000000      HALT
1777 011436 006301      CATR4:  ASL      R1      ;FATAL ERROR
1778 011440 005261 013206      INC      MINUS5(R1)      ;MULTIPLY 'OFFSET' X2
1779 011444 000744      BR      CATR2
1780 011446 005237 013234      OVRHI:  INC      ORHIGH
1781 011452 000741      BR      CATR2
1782 011454 005237 013204      OVRLO:  INC      ORLOW
1783 011460 000736      BR      CATR2
1784
1785      ;ADD THE COUNTS AND SAVE TOTAL IN SPREADS OF '1-4'
1786
1787 011462 013737 013220 013236      CATR5:  MOV      AVGCNT,XSPRD1
1788 011470 063737 013222 013236      ADD      PLUS1,XSPRD1
1789 011476 063737 013216 013236      ADD      MINUS1,XSPRD1      ;=TO NO. COUNTS AT SPREAD OF '1'
1790 011504 013737 013236 013240      MOV      XSPRD1,XSPRD2
1791 011512 063737 013224 013240      ADD      PLUS2,XSPRD2
1792 011520 063737 013214 013240      ADD      MINUS2,XSPRD2      ;=TO NO. COUNTS AT SPREAD OF '2'
1793 011526 013737 013240 013242      MOV      XSPRD2,XSPRD3
1794 011534 063737 013226 013242      ADD      PLUS3,XSPRD3
1795 011542 063737 013212 013242      ADD      MINUS3,XSPRD3      ;=TO NO. COUNTS AT SPREAD OF '3'
1796 011550 013737 013242 013244      MOV      XSPRD3,XSPRD4
1797 011556 063737 013230 013244      ADD      PLUS4,XSPRD4
1798 011564 063737 013210 013244      ADD      MINUS4,XSPRD4      ;=TO NO. COUNTS AT SPREAD OF '4'
1799 011572 000207      RTS      PC      ;EXIT

```

```

1800
1801 ;SUBROUTINE TO TEST FOR THE KEYBOARD FLAG BEING SET
1802
1803 011574 105777 167344 TSTFLG: TSTB 2STKS ;FLAG SET?
1804 011600 100002 BPL 1$ ;NO, EXIT
1805 011602 004737 010020 JSR PC,XTTYIN ;YES, INQUIRE
1806 011606 000207 RTS PC
1807
1808 011610 005015 040412 026522 TITLE: .ASCIZ <15><12><12>'AR-11 DIAGNOSTIC TEST II, (MAINDEC-11-DRLPD-A)'<<15><12>
1809 011616 030461 042040 040511
1810 011624 047107 051517 044524
1811 011632 020103 042524 052123
1812 011640 044440 026111 024040
1813 011646 040515 047111 042504
1814 011654 026503 030461 042055
1815 011662 046122 042120 040455
1816 011670 006451 000012
1817 011674 005015 052123 052101 MES2: .ASCIZ <15><12>"STATIC REGISTER TEST"<15><12>
1818 011702 041511 051040 043505
1819 011710 051511 042524 020122
1820 011716 042524 052123 005015
1821 011724 000
1822 011725 015 042412 040522 MES3: .ASCIZ <15><12>"ERASE RETURN FAILED TO SET READY"<15><12>
1823 011732 042523 051040 052105
1824 011740 051125 020116 040506
1825 011746 046111 042105 052040
1826 011754 020117 042523 020124
1827 011762 042522 042101 006531
1828 011770 000012
1829 011772 005015 054524 042520 MES4: .ASCII <15><12>"TYPE LETTER ' ' TO RUN DESIRED TEST:"<15><12>
1830 012000 046040 052105 042524
1831 012006 020122 020047 020047
1832 012014 047524 051040 047125
1833 012022 042040 051505 051111
1834 012030 042105 052040 051505
1835 012036 035124 005015
1836 012042 040447 036447 052123 .ASCII "'A'=STATIC REGISTER TEST"<15><12>
1837 012050 052101 041511 051040
1838 012056 043505 051511 042524
1839 012064 020122 042524 052123
1840 012072 005015
1841 012074 041047 036447 041523 .ASCII "'B'=SCOPE ADJUSTMENT TESTS"<15><12>
1842 012102 050117 020105 042101
1843 012110 052512 052123 042515
1844 012116 052116 052040 051505
1845 012124 051524 005015
1846 012130 041447 036447 020101 .ASCII "'C'=A TO D CALIBRATION TEST"<15><12>
1847 012136 047524 042040 041440
1848 012144 046101 041111 040522
1849 012152 044524 047117 052040
1850 012160 051505 006524 012
1851 012165 047 023504 040475 .ASCII "'D'=A TO D REPEATABILITY"<15><12>
1852 012172 052040 020117 020104
1853 012200 042522 042520 052101

```

1854	012206	041101	046111	052111	
1855	012214	006531	012		
1856	012217	047	023505	040475	.ASCIZ "'E'=A TO D RECOVERY"<15><12>
1857	012224	052040	020117	020104	
1858	012232	042522	047503	042526	
1859	012240	054522	005015	000	
1860	012245	136	006503	027012	CNTRLC: .ASCIZ '↑C'<15><12><56>
1861	012252	000			
1862	012253	136	006501	000012	CNTRLA: .ASCIZ '↑A'<15><12>
1863	012260	015	012	000	ACRLF: .BYTE 15,12,0
1864	012263	015	012	056	DOT: .BYTE 15,12,56,0
1865	012266	000			
1866	012267	077	000040		QMARK: .ASCIZ '? '
1867	012272	005015	041523	050117	MES6: .ASCIZ <15><12>'SCOPE ADJUSTMENT TEST'
1868	012300	020105	042101	052512	
1869	012306	052123	042515	052116	
1870	012314	052040	051505	000124	
1871	012322	005015	020101	047524	MES7: .ASCIZ <15><12>'A TO D CALIBRATION TEST'<15><12>
1872	012330	042040	041440	046101	
1873	012336	041111	040522	044524	
1874	012344	047117	052040	051505	
1875	012352	006524	000012		
1876	012356	005015	020101	047524	MES8: .ASCIZ <15><12>'A TO D RECOVERY TEST'<15><12>
1877	012364	042040	051040	041505	
1878	012372	053117	051105	020131	
1879	012400	042524	052123	005015	
1880	012406	000			
1881	012407	015	041412	020110	MES9: .ASCIZ <15><12>'CH '
1882	012414	000			
1883	012415	015	023412	023505	MES10: .ASCIZ <15><12>'E'XT. OR 'I'NT. OR 'C'LOCK. SYNC? "
1884	012422	052130	020056	051117	
1885	012430	023440	023511	052116	
1886	012436	020056	051117	023440	
1887	012444	023503	047514	045503	
1888	012452	020056	054523	041515	
1889	012460	020077	000		
1890	012463	015	040412	052040	MES13: .ASCIZ <15><12>'A TO D REPEATABILITY TEST'<15><12>
1891	012470	020117	020104	042522	
1892	012476	042520	052101	041101	
1893	012504	046111	052111	020131	
1894	012512	042524	052123	005015	
1895	012520	000			
1896	012521	015	005012	044103	MES14: .ASCIZ <15><12><12>'CH.(S)? '
1897	012526	024056	024523	020077	
1898	012534	000			
1899	012535	015	051412	051127	MES15: .ASCIZ <15><12>'SWRB AND 0 THRU 2 CONTROL PATTERN'<15><12>
1900	012542	020070	047101	020104	
1901	012550	020060	044124	052522	
1902	012556	031040	041440	047117	
1903	012564	051124	046117	050040	
1904	012572	052101	042524	047122	
1905	012600	005015	000		
1906	012603	103	052517	052116	MES16: .ASCIZ 'COUNT SPREAD? '
1907	012610	051440	051120	040505	

1908	012616	037504	000040		
1909	012622	005015	044103	020056	MES19: .ASCIZ <15><12>'CH. LO AV HI'
1910	012630	047514	020040	040440	
1911	012636	020126	020040	044510	
1912	012644	000			
1913	012645	015	020012	046040	MES20: .ASCIZ <15><12>' LO -5 -4 -3 -2 -1 AV +1 +2 +3 +4 +5 HI'
1914	012652	020117	026440	020065	
1915	012660	026440	020064	026440	
1916	012666	020063	026440	020062	
1917	012674	026440	020061	040440	
1918	012702	020126	025440	020061	
1919	012710	025440	020062	025440	
1920	012716	020063	025440	020064	
1921	012724	025440	020065	044040	
1922	012732	000111			
1923	012734	042522	044507	052123	EM1: .ASCIZ /REGISTER CONTENTS CHANGED IN ERROR/
1924	012742	051105	041440	047117	
1925	012750	042524	052116	020123	
1926	012756	044103	047101	042507	
1927	012764	020104	047111	042440	
1928	012772	051122	051117	000	
1929	012777	105	051122	041520	DH1: .ASCIZ /ERRPC BUFADR EXPECT BAD/
1930	013004	020040	041040	043125	
1931	013012	042101	020122	020040	
1932	013020	054105	042520	052103	
1933	013026	020040	020040	040502	
1934	013034	000104			
1935					
1936	013036	001116	007472	001124	DT1: .EVEN \$ERRPC, BUFADR, \$GDDAT, \$BDDAT, 0
1937	013044	001126	000000		
1938	013050	000000			ADWRD2: 0 ;LOW BYTE OF 'ADWORD'
1939	013052	000000			PRINT1: 0
1940	013054	002264			AVECTR: INITA ;'A' VECTOR ADDRESS
1941	013056	000000			PROC: 0 ;TEMP STORAGE FOR 'PSW'
1942	013060	000000			CHRCNT: 0 ;TEMP STORAGE
1943	013062	000000			COUNT: 0 ;TEMP STORAGE
1944	013064	000000			KSTOR1: 0 ;PERMANENT STORAGE
1945	013066	000000			KSTOR2: 0 ;PERMANENT STORAGE
1946	013070	000000			KSTOR3: 0 ;PERMANENT STORAGE
1947	013072	000000			KSTOR4: 0 ;PERMANENT STORAGE
1948	013074	000000			KSTOR5: 0 ;PERMANENT STORAGE
1949	013076	000000			KSTR11: 0
1950	013100	000000			KSTR12: 0
1951	013102	000000			LOW: 0
1952	013104	000000			HIGH: 0
1953	013106	000010			INCR: 10
1954	013110	000000			TIMSV: 0
1955	013112	000000			TICKS: 0
1956	013114	000000			STCHAN: 0
1957	013116	000000			FIRST: 0
1958	013120	000000			DELAY: 0
1959	013122	000000			DELAY1: 0
1960	013124	000000			USECLK: 0
1961	013126	000000			OPSI: 0

1962 013130 000000
 1963 013132 000000
 1964 013134 000000
 1965 013136 000000
 1966 013140 000000
 1967 013142 000000
 1968 013144 000000
 1969 013146 000000
 1970 013150 000000
 1971 013152 000000
 1972 013154 000000
 1973 013156 000000
 1974 013160 000000
 1975 013162 000000
 1976 013164 000000
 1977 013166 000000
 1978 013170 000000
 1979 013172 000000
 1980 013174 000000
 1981 013176 000000
 1982 013200 000000
 1983 013202 000000
 1984 013204 000000
 1985 013206 000000
 1986 013210 000000
 1987 013212 000000
 1988 013214 000000
 1989 013216 000000
 1990 013220 000000
 1991 013222 000000
 1992 013224 000000
 1993 013226 000000
 1994 013230 000000
 1995 013232 000000
 1996 013234 000000
 1997 013236 000000
 1998 013240 000000
 1999 013242 000000
 2000 013244 000000
 2001 013246 000000
 2002
 2003
 2004
 2005
 2006
 2007
 2008
 2009
 2010
 2011
 2012
 2013
 2014
 2015

TEMP: 0
 TEMP1: 00 ; TEMPORARY STORAGE
 TEMP2: 00 ; TEMPORARY STORAGE
 TEMP3: 00 ; TEMPORARY STORAGE
 BRLEV1: 00
 BRLEV2: 00
 BRLEV3: 00
 MESPRT: 00
 HIORDV: 00
 ADHIGH: 00
 ADLOW: 00
 AVERM5: 00
 AVERM4: 00
 AVERM3: 00
 AVERM2: 00
 AVERM1: 00
 AVRAGE: 00
 AVERP1: 00
 AVERP2: 00
 AVERP3: 00
 AVERP4: 00
 AVERP5: 00
 ORLOW: 00
 MINUS5: 00
 MINUS4: 00
 MINUS3: 00
 MINUS2: 00
 MINUS1: 00
 AVGCNT: 00
 PLUS1: 00
 PLUS2: 00
 PLUS3: 00
 PLUS4: 00
 PLUS5: 00
 ORHIGH: 00
 XSPRD1: 00
 XSPRD2: 00
 XSPRD3: 00
 XSPRD4: 00
 \$TMDAT: 0
 .SBTTL SCOPE HANDLER ROUTINE

```

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

```

```

2016 013250          $SCOPE:
2017 013250 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
2018 013252 104407          CKSWR
2019 013254 032777 040000 165656 1$: BIT #BIT14, $SWR          ;; LOOP ON PRESENT TEST?
2020 013262 001114          BNE $OVER          ;; YES IF SW14=1
2021          :*****START OF CODE FOR THE XOR TESTER*****
2022 013264 000416 $XTSTR: BR 6$          ;; IF RUNNING ON THE "XOR" TESTER CHANGE
2023          THIS INSTRUCTION TO A "NOP" (NOP=240)
2024 013266 013746 000004          MOV @#ERRVEC, -(SP)          ;; SAVE THE CONTENTS OF THE ERROR VECTOR
2025 013272 012737 013312 000004          MOV #5$, @#ERRVEC          ;; SET FOR TIMEOUT
2026 013300 005737 177060          TST @#177060          ;; TIME OUT ON XOR?
2027 013304 012637 000004          MOV (SP)+, @#ERRVEC          ;; RESTORE THE ERROR VECTOR
2028 013310 000463          BR $SVLAD          ;; GO TO THE NEXT TEST
2029 013312 022626          5$: CMP (SP)+, (SP)+          ;; CLEAR THE STACK AFTER A TIME OUT
2030 013314 012637 000004          MOV (SP)+, @#ERRVEC          ;; RESTORE THE ERROR VECTOR
2031 013320 000423          BR 7$          ;; LOOP ON THE PRESENT TEST
2032 013322          6$; *****END OF CODE FOR THE XOR TESTER*****
2033 013322 032777 000400 165610          BIT #BIT08, $SWR          ;; LOOP ON SPEC. TEST?
2034 013330 001404          BEQ 2$          ;; BR IF NO
2035 013332 127737 165602 001102          CMPB @SWR, $STSTM          ;; ON THE RIGHT TEST? SWR<7:0>
2036 013340 001465          BEQ $OVER          ;; BR IF YES
2037 013342 105737 001103          2$: TSTB $ERFLG          ;; HAS AN ERROR OCCURRED?
2038 013346 001421          BEQ 3$          ;; BR IF NO
2039 013350 123737 001115 001103          CMPB $ERMAX, $ERFLG          ;; MAX. ERRORS FOR THIS TEST OCCURRED?
2040 013356 101015          BHI 3$          ;; BR IF NO
2041 013360 032777 001000 165552          BIT #BIT09, $SWR          ;; LOOP ON ERROR?
2042 013366 001404          BEQ 4$          ;; BR IF NO
2043 013370 013737 001110 001106          7$: MOV $LPERR, $LPADR          ;; SET LOOP ADDRESS TO LAST SCOPE
2044 013376 000446          BR $OVER
2045 013400 105037 001103          4$: CLRB $ERFLG          ;; ZERO THE ERROR FLAG
2046 013404 005037 001166          CLR $TIMES          ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
2047 013410 000415          BR 1$          ;; ESCAPE TO THE NEXT TEST
2048 013412 032777 004000 165520          3$: BIT #BIT11, $SWR          ;; INHIBIT ITERATIONS?
2049 013420 001011          BNE 1$          ;; BR IF YES
2050 013422 005737 001210          TST $PASS          ;; IF FIRST PASS OF PROGRAM
2051 013426 001406          BEQ 1$          ;; INHIBIT ITERATIONS
2052 013430 005237 001104          INC $ICNT          ;; INCREMENT ITERATION COUNT
2053 013434 023737 001166 001104          CMP $TIMES, $ICNT          ;; CHECK THE NUMBER OF ITERATIONS MADE
2054 013442 002024          BGE $OVER          ;; BR IF MORE ITERATION REQUIRED
2055 013444 012737 000001 001104          1$: MOV #1, $ICNT          ;; REINITIALIZE THE ITERATION COUNTER
2056 013452 013737 013530 001166          MOV $MXCNT, $TIMES          ;; SET NUMBER OF ITERATIONS TO DO
2057 013460 105237 001102          $SVLAD: INCB $STSTM          ;; COUNT TEST NUMBERS
2058 013464 113737 001102 001206          MOVB $STSTM, $TESTN          ;; SET TEST NUMBER IN APT MAILBOX
2059 013472 011637 001106          MOV (SP), $LPADR          ;; SAVE SCOPE LOOP ADDRESS
2060 013476 011637 001110          MOV (SP), $LPERR          ;; SAVE ERROR LOOP ADDRESS
2061 013502 005037 001170          CLR $ESCAPE          ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2062 013506 112737 000001 001115          MOVB #1, $ERMAX          ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2063 013514 013777 001102 165420          $OVER: MOV $STSTM, @DISPLAY          ;; DISPLAY TEST NUMBER
2064 013522 013716 001106          MOV $LPADR, (SP)          ;; FUDGE RETURN ADDRESS
2065 013526 000002          RTI          ;; FIXES PS
2066 013530 000012          $MXCNT: 10.          ;; MAX. NUMBER OF ITERATIONS
2067          .SBTTL ERROR HANDLER ROUTINE
2068
2069          ;;*****

```

2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123

013532
013532 104407
013534 105237 001103
013540 001775
013542 013777 001102 165372
013550 032777 002000 165362
013556 001402
013560 104401 001172
013564 005237 001112
013570 011637 001116
013574 162737 000002 001116
013602 117737 165310 001114
013610 032777 020000 165322
013616 001004
013620 004737 013720
013624 104401 001177
013630
013630 122737 000001 001222
013636 001007
013640 113737 001114 013652
013646 004737 015700
013652 000
013653 000
013654 000777
013656 005777 165256
013662 100002
013664 000000
013666 104407
013670 032777 001000 165242
013676 001402
013700 013716 001110
013704 005737 001170
013710 001402
013712 013716 001170
013716
013716 000002

```
;;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,  
;;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL  
;;*AND GO TO SERRTYP ON ERROR  
;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  
;;*SW15=1 HALT ON ERROR  
;;*SW13=1 INHIBIT ERROR TYPEOUTS  
;;*SW10=1 BELL ON ERROR  
;;*SW09=1 LOOP ON ERROR  
;;*CALL  
* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER  
  
SERROR:  
7$: CKSWR  
INCB SERFLG ;; TEST FOR CHANGE IN SOFT-SWR  
BEQ 7$ SET THE ERROR FLAG  
MOV $STNM, $DISPLAY ;; DON'T LET THE FLAG GO TO ZERO  
BIT $BIT10, $SWR ;; DISPLAY TEST NUMBER AND ERROR FLAG  
BEQ 1$ BELL ON ERROR?  
TYPE $BELL NO - SKIP  
INC $ERTTL RING BELL  
MOV (SP), $ERRPC ;; COUNT THE NUMBER OF ERRORS  
SUB #2, $ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION  
MOVB $SERRPC, $ITEMB STRIP AND SAVE THE ERROR ITEM CODE  
BIT $BIT13, $SWR SKIP TYPEOUT IF SET  
BNE 20$ SKIP TYPEOUTS  
JSR PC, $ERRTYP GO TO USER ERROR ROUTINE  
TYPE , $SCLF  
  
20$: CMPB $APTENV, $ENV ;; RUNNING IN APT MODE  
BNE 2$ NO SKIP APT ERROR REPORT  
MOVB $ITEMB, 21$ SET ITEM NUMBER AS ERROR NUMBER  
JSR PC, $SATY4 REPORT FATAL ERROR TO APT  
  
21$: .BYTE 0  
.BYTE 0  
  
22$: BR 22$ APT ERROR LOOP  
2$: TST $SWR HALT ON ERROR  
BPL 3$ SKIP IF CONTINUE  
HALT HALT ON ERROR!  
CKSWR TEST FOR CHANGE IN SOFT-SWR  
BIT $BIT09, $SWR LOOP ON ERROR SWITCH SET?  
BEQ 4$ BR IF NO  
MOV $LPERR, (SP) FUDGE RETURN FOR LOOPING  
TST $ESCAPE CHECK FOR AN ESCAPE ADDRESS  
BEQ 5$ BR IF NONE  
MOV $ESCAPE, (SP) FUDGE RETURN ADDRESS FOR ESCAPE  
  
5$: RTI RETURN  
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE  
  
*****  
;;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH  
;;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),  
;;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
```



```

2124 013720                                SERRTYP:
2125 013720 104401 001177                TYPE      $SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
2126 013724 010046                        MOV      RO,-(SP)        ;; SAVE RO
2127 013726 005000                        CLR      RO              ;; PICKUP THE ITEM INDEX
2128 013730 153700 001114                BISB     @#$ITEMB,RO
2129 013734 001004                        BNE     1$              ;; IF ITEM NUMBER IS ZERO, JUST
2130                                     ;; TYPE THE PC OF THE ERROR
2131 013736 013746 001116                MOV      $ERRPC,-(SP)   ;; SAVE $ERRPC FOR TYPEOUT
2132                                     ;; ERROR ADDRESS
2133 013742 104402                        TYPOC
2134 013744 000426                        BR      6$              ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2135 013746 005300                        1$:    DEC      RO              ;; GET OUT
2136 013750 006300                        ASL     RO              ;; ADJUST THE INDEX SO THAT IT WILL
2137 013752 006300                        ASL     RO              ;; WORK FOR THE ERROR TABLE
2138 013754 006300                        ASL     RO
2139 013756 062700 001326                ADD     @SERRTB,RO      ;; FORM TABLE POINTER
2140 013762 012037 013772                MOV     (RO)+,2$
2141 013766 001404                        BEQ     3$              ;; PICKUP "ERROR MESSAGE" POINTER
2142 013770 104401                        TYPE
2143 013772 000000                        2$:    .WORD 0           ;; SKIP TYPEOUT IF NO POINTER
2144 013774 104401 001177                TYPE      $SCRLF          ;; TYPE THE "ERROR MESSAGE"
2145 014000 012037 014010                MOV     (RO)+,4$        ;; "ERROR MESSAGE" POINTER GOES HERE
2146 014004 001404                        BEQ     5$              ;; "CARRIAGE RETURN" & "LINE FEED"
2147 014006 104401                        TYPE
2148 014010 000000                        4$:    .WORD 0           ;; PICKUP "DATA HEADER" POINTER
2149 014012 104401 001177                TYPE      $SCRLF          ;; TYPE THE "DATA HEADER"
2150 014016 011000                        MOV     (RO),RO         ;; "DATA HEADER" POINTER GOES HERE
2151 014020 001004                        BNE     7$              ;; "CARRIAGE RETURN" & "LINE FEED"
2152 014022 012600                        MOV     (SP)+,RO        ;; PICKUP "DATA TABLE" POINTER
2153 014024 104401 001177                TYPE      $SCRLF          ;; GO TYPE THE DATA
2154 014030 000207                        RTS      PC              ;; RESTORE RO
2155 014032                                7$:    MOV     @ (RO)+,-(SP) ;; "CARRIAGE RETURN" & "LINE FEED"
2156 014032 013046                        MOV     @ (RO)+,-(SP)   ;; RETURN
2157 014034 104402                        TYPOC                ;; SAVE @ (RO)+ FOR TYPEOUT
2158 014036 005710                        TST     (RO)            ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2159 014040 001770                        BEQ     6$              ;; IS THERE ANOTHER NUMBER?
2160 014042 104401 014050                TYPE      $S              ;; BR IF NO
2161 014046 000771                        BR      7$              ;; TYPE TWO(2) SPACES
2162 014050 020040 000                8$:    .ASCIZ  / /        ;; LOOP
2163 014054                                .EVEN                ;; TWO(2) SPACES
2164                                     .SBTTL  POWER DOWN AND UP ROUTINES
2165
2166                                     ;; *****
2167                                     ;; POWER DOWN ROUTINE
2168 014054 012737 014220 000024          $PWRDN: MOV     @SILLUP,@PWRVEC ;; SET FOR FAST UP
2169 014062 012737 000340 000026          MOV     #340,@PWRVEC+2 ;; PRIO:7
2170 014070 010046                        MOV     RO,-(SP)        ;; PUSH RO ON STACK
2171 014072 010146                        MOV     R1,-(SP)        ;; PUSH R1 ON STACK
2172 014074 010246                        MOV     R2,-(SP)        ;; PUSH R2 ON STACK
2173 014076 010346                        MOV     R3,-(SP)        ;; PUSH R3 ON STACK
2174 014100 010446                        MOV     R4,-(SP)        ;; PUSH R4 ON STACK
2175 014102 010546                        MOV     R5,-(SP)        ;; PUSH R5 ON STACK
2176 014104 017746 165030                MOV     @SWR,-(SP)     ;; PUSH @SWR ON STACK
2177 014110 010637 014224                MOV     SP,$$SAVR6     ;; SAVE SP

```

```

2178 014114 012737 014126 000024      MOV    #SPWRUP,@PWRVEC ;;SET UP VECTOR
2179 014122 000000                      HALT
2180 014124 000776                      BR     -2                ;;HANG UP
2181
2182                                     ;:*****
2183                                     ;:POWER UP ROUTINE
2184 014126 012737 014220 000024 $PWRUP: MOV    #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
2185 014134 013706 014224          MOV    $SAVR6,SP        ;;GET SP
2186 014140 005037 014224          CLR    $SAVR6          ;;WAIT LOOP FOR THE TTY
2187 014144 005237 014224          IS:   INC    $SAVR6     ;;WAIT FOR THE INC
2188 014150 001375                      BNE   IS               ;;OF WORD
2189 014152 012677 164762          MOV    (SP)+,@SWR      ;;POP STACK INTO @SWR
2190 014156 012605                      MOV    (SP)+,R5        ;;POP STACK INTO R5
2191 014160 012604                      MOV    (SP)+,R4        ;;POP STACK INTO R4
2192 014162 012603                      MOV    (SP)+,R3        ;;POP STACK INTO R3
2193 014164 012602                      MOV    (SP)+,R2        ;;POP STACK INTO R2
2194 014166 012601                      MOV    (SP)+,R1        ;;POP STACK INTO R1
2195 014170 012600                      MOV    (SP)+,R0        ;;POP STACK INTO R0
2196 014172 012737 014054 000024      MOV    #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
2197 014200 012737 000340 000026      MOV    #340,@PWRVEC+2 ;;PRIO:7
2198 014206 104401                      TYPE   PWRMSG          ;;REPORT THE POWER FAILURE
2199 014210 014226          SPWRMG: .WORD PWRMSG   ;;POWER FAIL MESSAGE POINTER
2200 014212 012716          MOV    (PC)+,(SP)     ;;RESTART AT BEGIN
2201 014214 001454          SPWRAD: .WORD BEGIN   ;;RESTART ADDRESS
2202 014216 000002          RTI
2203 014220 000000          $SILLUP: HALT        ;; THE POWER UP SEQUENCE WAS STARTED
2204 014222 000776          BR     -2            ;; BEFORE THE POWER DOWN WAS COMPLETE
2205 014224 000000          $SAVR6: 0            ;; PUT THE SP HERE
2206 014226 005015 042522 052123      PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>
2207 014234 051101 044524 043516
2208 014242 040440 052106 051105
2209 014250 040440 050040 053517
2210 014256 051105 043040 044501
2211 014264 052514 042522 005015
2212 014272 000012
2213
2214 .EVEN
2215 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2216
2217 ;:*****
2218 ;:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2219 ;:OCTAL (ASCII) NUMBER AND TYPE IT.
2220 ;:$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2221 ;:CALL:
2222 ;:   MOV    NUM,-(SP)        ;;NUMBER TO BE TYPED
2223 ;:   TYPOS          ;;CALL FOR TYPEOUT
2224 ;:   .BYTE  N            ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2225 ;:   .BYTE  M            ;;M=1 OR 0
2226 ;:                               ;;1=TYPE LEADING ZEROS
2227 ;:                               ;;0=SUPPRESS LEADING ZEROS
2228 ;:
2229 ;:$TYPON-----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2230 ;:$TYPOS OR $TYPOC
2231 ;:CALL:

```

```

2232          *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2233          *      TYPON      ;;CALL FOR TYPEOUT
2234          *
2235          *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2236          *CALL:
2237          *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2238          *      TYPON      ;;CALL FOR TYPEOUT
2239          *
2240 014274 017646 000000          $TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
2241 014300 116637 000001 014517  MOVB     1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
2242 014306 112637 014521          MOVB     (SP)+,SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
2243 014312 062716 000002          ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
2244 014316 000406          BR      $TYPON
2245 014320 112737 000001 014517  $TYPOC: MOVB     #1,SOFILL      ;;SET THE ZERO FILL SWITCH
2246 014326 112737 000006 014521  MOVB     #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
2247 014334 112737 000005 014516  $TYPON: MOVB     #5,SOCNT      ;;SET THE ITERATION COUNT
2248 014342 010346          MOV      R3,-(SP)          ;;SAVE R3
2249 014344 010446          MOV      R4,-(SP)          ;;SAVE R4
2250 014346 010546          MOV      R5,-(SP)          ;;SAVE R5
2251 014350 113704 014521          MOVB     $SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
2252 014354 005404          NEG      R4
2253 014356 062704 000006          ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
2254 014362 110437 014520          MOVB     R4,SOMODE      ;;SAVE IT FOR USE
2255 014366 113704 014517          MOVB     $SOFILL,R4      ;;GET THE ZERO FILL SWITCH
2256 014372 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
2257 014376 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
2258 014400 006105          1$:     ROL      R5          ;;ROTATE MSB INTO "C"
2259 014402 000404          BR      3$
2260 014404 006105          2$:     ROL      R5          ;;GO DO MSB
2261 014406 006105          ROL      R5          ;;FORM THIS DIGIT
2262 014410 006105          ROL      R5
2263 014412 010503          MOV      R5,R3
2264 014414 006103          3$:     ROL      R3          ;;GET LSB OF THIS DIGIT
2265 014416 105337 014520          DEC8     $SOMODE      ;;TYPE THIS DIGIT?
2266 014422 100016          BPL      7$          ;;BR IF NO
2267 014424 042703 177770          BIC      #177770,R3      ;;GET RID OF JUNK
2268 014430 001002          BNE      4$          ;;TEST FOR 0
2269 014432 005704          TST      R4          ;;SUPPRESS THIS 0?
2270 014434 001403          BEQ      5$          ;;BR IF YES
2271 014436 005204          4$:     INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
2272 014440 052703 000060          BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
2273 014444 052703 000040          5$:     BIS      #' ',R3      ;;MAKE ASCII IF NOT ALREADY
2274 014450 110337 014514          MOVB     R3,8$          ;;SAVE FOR TYPING
2275 014454 104401 014514          TYPE     8$          ;;GO TYPE THIS DIGIT
2276 014460 105337 014516          7$:     DEC8     $SOCNT      ;;COUNT BY 1
2277 014464 003347          BGT      2$          ;;BR IF MORE TO DO
2278 014466 002402          BLT      6$          ;;BR IF DONE
2279 014470 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
2280 014472 000744          BR      2$          ;;GO DO THE LAST DIGIT
2281 014474 012605          6$:     MOV      (SP)+,R5      ;;RESTORE R5
2282 014476 012604          MOV      (SP)+,R4      ;;RESTORE R4
2283 014500 012603          MOV      (SP)+,R3      ;;RESTORE R3
2284 014502 016666 000002 000004  MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
2285 014510 012616          MOV      (SP)+,(SP)

```

2286 014512 000002
 2287 014514 000
 2288 014515 000
 2289 014516 000
 2290 014517 000
 2291 014520 000000
 2292
 2293
 2294
 2295
 2296
 2297
 2298
 2299
 2300
 2301
 2302
 2303
 2304
 2305
 2306
 2307
 2308
 2309 014522 105737 001157
 2310 014526 100002
 2311 014530 000000
 2312 014532 000430
 2313 014534 010046
 2314 014536 017600 000002
 2315 014542 122737 000001 001222
 2316 014550 001011
 2317 014552 132737 000100 001223
 2318 014560 001405
 2319 014562 010037 014572
 2320 014566 004737 015670
 2321 014572 000000
 2322 014574 132737 000040 001223
 2323 014602 001003
 2324 014604 112046
 2325 014606 001005
 2326 014610 005726
 2327 014612 012600
 2328 014614 062716 000002
 2329 014620 000002
 2330 014622 122716 000011
 2331 014626 001430
 2332 014630 122716 000200
 2333 014634 001006
 2334 014636 005726
 2335 014640 104401
 2336 014642 001177
 2337 014644 105037 015000
 2338 014650 000755
 2339 014652 004737 014734

```

RTI                                     ;; RETURN
8$: .BYTE 0                             ;; STORAGE FOR ASCII DIGIT
                                       ;; TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE 0                          ;; OCTAL DIGIT COUNTER
$DFILL: .BYTE 0                          ;; ZERO FILL SWITCH
$OMODE: .WORD 0                           ;; NUMBER OF DIGITS TO TYPE
.SBTTL TYPE ROUTINE

*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
*   TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
*   TYPE
*   MESADR
*
$TYPE: TSTB $TFPLG ;; IS THERE A TERMINAL?
      BPL 1$ ;; BR IF YES
      HALT ;; HALT HERE IF NO TERMINAL
      BR 3$ ;; LEAVE
1$: MOV RO, -(SP) ;; SAVE RO
      MOV #2(SP), RO ;; GET ADDRESS OF ASCIZ STRING
      CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
      BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
      BITB #APTSPOOL, $ENVm ;; SPOOL MESSAGE TO APT
      BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
      MOV RO, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
      JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
61$: .WORD 0 ;; MESSAGE ADDRESS
62$: BITB #APTCSUP, $ENVm ;; APT CONSOLE SUPPRESSED
      BNE 60$ ;; YES, SKIP TYPE OUT
2$: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
      BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
      TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+, RO ;; RESTORE RO
3$: ADD #2, (SP) ;; ADJUST RETURN PC
      RTI ;; RETURN
4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
      BEQ 8$
      CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
      BNE 5$
      TST (SP)+ ;; POP <CR><LF> EQUIV
                                       ;; TYPE A CR AND LF
                                       ;; CLEAR CHARACTER COUNT
                                       ;; GET NEXT CHARACTER
5$: JSR PC, $TYPEC ;; GO TYPE THIS CHARACTER

```

```

2340 014656 123726 001156 6$: CMPB $FILLC,(SP)+ ;: IS IT TIME FOR FILLER CHARS.?
2341 014662 001350 BNE 2$ ;: IF NO GO GET NEXT CHAR.
2342 014664 013746 001154 MOV $NULL,-(SP) ;: GET # OF FILLER CHARS. NEEDED
2343 ;: AND THE NULL CHAR.
2344 014670 105366 000001 7$: DECb 1(SP) ;: DOES A NULL NEED TO BE TYPED?
2345 014674 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
2346 014676 004737 014734 JSR PC,$TYPEC ;: GO TYPE A NULL
2347 014702 105337 015000 DECb $CHARCNT ;: DO NOT COUNT AS A COUNT
2348 014706 000770 BR 7$ ;: LOOP
2349
2350 ;: HORIZONTAL TAB PROCESSOR
2351
2352 014710 112716 000040 8$: MOVb #' (SP) ;: REPLACE TAB WITH SPACE
2353 014714 004737 014734 9$: JSR PC,$TYPEC ;: TYPE A SPACE
2354 014720 132737 000007 015000 BITb #7,$CHARCNT ;: BRANCH IF NOT AT
2355 014726 001372 BNE 9$ ;: TAB STOP
2356 014730 005726 TST (SP)+ ;: POP SPACE OFF STACK
2357 014732 000724 BR 2$ ;: GET NEXT CHARACTER
2358 014734 105777 164210 $TYPEC: TSTB @STPS ;: WAIT UNTIL PRINTER IS READY
2359 014740 100375 BPL $TYPEC
2360 014742 116677 000002 164202 MOVb 2(SP),@STPB ;: LOAD CHAR TO BE TYPED INTO DATA REG.
2361 014750 122766 000015 000002 CMPb #CR,2(SP) ;: IS CHARACTER A CARRIAGE RETURN?
2362 014756 001003 BNE 1$ ;: BRANCH IF NO
2363 014760 105037 015000 CLRb $CHARCNT ;: YES--CLEAR CHARACTER COUNT
2364 014764 000406 BR $TYPEX ;: EXIT
2365 014766 122766 000012 000002 1$: CMPb #LF,2(SP) ;: IS CHARACTER A LINE FEED?
2366 014774 001402 BEQ $TYPEX ;: BRANCH IF YES
2367 014776 105227 INCB (PC)+ ;: COUNT THE CHARACTER
2368 015000 000000 $CHARCNT: WORD 0 ;: CHARACTER COUNT STORAGE
2369 015002 000207 $TYPEX: RTS PC
2370
2371 .SBTTL TTY INPUT ROUTINE
2372
2373 ;: *****
2374 .ENABL LSB
2375
2376 ;: *****
2377 ;: *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2378 ;: *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2379 ;: *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2380 ;: *WHEN OPERATING IN TTY FLAG MODE.
2381 015004 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;: IS THE SOFT-SWR SELECTED?
2382 015012 001074 BNE 15$ ;: BRANCH IF NO
2383 015014 105777 164124 TSTB @STKS ;: CHAR THERE?
2384 015020 100071 BPL 15$ ;: IF NO, DON'T WAIT AROUND
2385 015022 117746 164120 MOVb @STKB,-(SP) ;: SAVE THE CHAR
2386 015026 042716 177600 BIC #177,(SP) ;: STRIP-OFF THE ASCII
2387 015032 022726 000007 CMP #7,(SP)+ ;: IS IT A CONTROL G?
2388 015036 001062 BNE 15$ ;: NO, RETURN TO USER
2389 015040 123727 001134 000001 CMPb $AUTOB,#1 ;: ARE WE RUNNING IN AUTO-MODE?
2390 015046 001456 BEQ 15$ ;: BRANCH IF YES
2391
2392 015050 104401 015531 $GTSWR: TYPE , $CNTLG ;: ECHO THE CONTROL-G (↑G)
2393 015054 104401 015536 TYPE , $MSWR ;: TYPE CURRENT CONTENTS

```

```

2394 015060 013746 000176      MOV      SWREG,-(SP)      ;;SAVE SWREG FOR TYPEOUT
2395 015064 104402              TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2396 015066 104401 015547      TYPE      ,SMNEW        ;;PROMPT FOR NEW SWR
2397 015072 005046              CLR      -(SP)          ;;CLEAR COUNTER
2398 015074 005046              CLR      -(SP)          ;;THE NEW SWR
2399 015076 105777 164042      7$:      TSTB      @STKS   ;;CHAR THERE?
2400 015102 100375              BPL      7$             ;;IF NOT TRY AGAIN
2401
2402 015104 117746 164036      MOVB     @STKB,-(SP)    ;;PICK UP CHAR
2403 015110 042716 177600      BIC      #1C177,(SP)   ;;MAKE IT 7-BIT ASCII
2404
2405
2406
2407 015114 021627 000025      9$:      CMP      (SP),#25     ;;IS IT A CONTROL-U?
2408 015120 001005              BNE      10$           ;;BRANCH IF NOT
2409 015122 104401 015524      TYPE      $CNTLU        ;;YES, ECHO CONTROL-U (↑U)
2410 015126 062706 000006      20$:     ADD      #6,SP         ;;IGNORE PREVIOUS INPUT
2411 015132 000757              BR       19$           ;;LET'S TRY IT AGAIN
2412
2413
2414 015134 021627 000015      10$:     CMP      (SP),#15     ;;IS IT A <CR>?
2415 015140 001022              BNE      16$           ;;BRANCH IF NO
2416 015142 005766 000004      TST      4(SP)          ;;YES, IS IT THE FIRST CHAR?
2417 015146 001403              BEQ      11$           ;;BRANCH IF YES
2418 015150 016677 000002 163762      MOV      2(SP),@SWR     ;;SAVE NEW SWR
2419 015156 062706 000006 11$:      ADD      #6,SP         ;;CLEAR UP STACK
2420 015162 104401 001177 14$:     TYPE      $CRLF        ;;ECHO <CR> AND <LF>
2421 015166 123727 001135 000001      CMPB     $INTAG,#1     ;;RE-ENABLE TTY KBD INTERRUPTS?
2422 015174 001003              BNE      15$           ;;BRANCH IF NOT
2423 015176 012777 000100 163740      MOV      #100,@STKS    ;;RE-ENABLE TTY KBD INTERRUPTS
2424 015204 000002              RTI                      ;;RETURN
2425 015206 004737 014734 16$:      JSR      PC,$TYPEC     ;;ECHO CHAR
2426 015212 021627 000060      CMP      (SP),#60     ;;CHAR < 0?
2427 015216 002420              BLT      18$           ;;BRANCH IF YES
2428 015220 021627 000067      CMP      (SP),#67     ;;CHAR > 7?
2429 015224 003015              BGT      18$           ;;BRANCH IF YES
2430 015226 042726 000060      BIC      #60,(SP)+    ;;STRIP-OFF ASCII
2431 015232 005766 000002      TST      2(SP)         ;;IS THIS THE FIRST CHAR
2432 015236 001403              BEQ      17$           ;;BRANCH IF YES
2433 015240 006316              ASL      (SP)          ;;NO, SHIFT PRESENT
2434 015242 006316              ASL      (SP)          ;;CHAR OVER TO MAKE
2435 015244 006316              ASL      (SP)          ;;ROOM FOR NEW ONE.
2436 015246 005266 000002 17$:     INC      2(SP)         ;;KEEP COUNT OF CHAR
2437 015252 056616 177776      BIS      -2(SP),(SP)   ;;SET IN NEW CHAR
2438 015256 000707              BR       7$           ;;GET THE NEXT ONE
2439 015260 104401 001176 18$:     TYPE      $QUES        ;;TYPE ?<CR><LF>
2440 015264 000720              BR       20$           ;;SIMULATE CONTROL-U
2441
2442
2443
2444
2445
2446
2447

```

; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
; CALL:
; RDCHR ; INPUT A SINGLE CHARACTER FROM THE TTY

```

2448          ;*      RETURN HERE          ;: CHARACTER IS ON THE STACK
2449          ;*                               ;: WITH PARITY BIT STRIPPED OFF
2450          ;*
2451          ;*
2452          ;*
2453 015266 011646 000004 000002 $RDCHR: MOV (SP), -(SP) ;: PUSH DOWN THE PC
2454 015270 016666 163642 15: MOV 4(SP), 2(SP) ;: SAVE THE PS
2455 015276 105777 163642 15: TSTB @STKS ;: WAIT FOR
2456 015302 100375 163636 000004 BPL 15 ;: A CHARACTER
2457 015304 117766 177600 000004 MOVB @STKB, 4(SP) ;: READ THE TTY
2458 015312 042766 177600 000004 BIC #177, 4(SP) ;: GET RID OF JUNK IF ANY
2459 015320 026627 000004 000023 CMP 4(SP), #23 ;: IS IT A CONTROL-S?
2460 015326 001013 163610 25: BNE 35 ;: BRANCH IF NO
2461 015330 105777 163604 25: TSTB @STKS ;: WAIT FOR A CHARACTER
2462 015334 100375 177600 000004 BPL 25 ;: LOOP UNTIL ITS THERE
2463 015336 117746 177600 000004 MOVB @STKB, -(SP) ;: GET CHARACTER
2464 015342 042716 177600 000021 BIC #177, (SP) ;: MAKE IT 7-BIT ASCII
2465 015346 022627 000021 25: CMP (SP)+, #21 ;: IS IT A CONTROL-Q?
2466 015352 001366 000750 35: BNE 25 ;: IF NOT DISCARD IT
2467 015354 000750 000004 000140 BR 15 ;: YES, RESUME
2468 015356 026627 000004 000175 35: CMP 4(SP), #140 ;: IS IT UPPER CASE?
2469 015364 002407 000004 000175 BLT 45 ;: BRANCH IF YES
2470 015366 026627 000004 000175 45: CMP 4(SP), #175 ;: IS IT A SPECIAL CHAR?
2471 015374 003003 000040 000004 BGT 45 ;: BRANCH IF YES
2472 015376 042766 000040 000004 BIC #40, 4(SP) ;: MAKE IT UPPER CASE
2473 015404 000002 45: RTI ;: GO BACK TO USER
2474          ;: *****
2475          ;: THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2476          ;: *CALL:
2477          ;*      RDLIN          ;: INPUT A STRING FROM THE TTY
2478          ;*      RETURN HERE    ;: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2479          ;*                               ;: TERMINATOR WILL BE A BYTE OF ALL 0'S
2480          ;*
2481          ;*
2482          ;*
2483 015406 010346 015514 $RDLIN: MOV R3, -(SP) ;: SAVE R3
2484 015410 012703 015524 15: MOV #STTYIN, R3 ;: GET ADDRESS
2485 015414 022703 015524 25: CMP #STTYIN+8., R3 ;: BUFFER FULL?
2486 015420 101405 104410 BLOS 45 ;: BR IF YES
2487 015422 104410 112613 RDCHR ;: GO READ ONE CHARACTER FROM THE TTY
2488 015424 112613 122713 MOVB (SP)+, (R3) ;: GET CHARACTER
2489 015426 122713 000177 105: CMPB #177, (R3) ;: IS IT A RUBOUT
2490 015432 001003 104401 BNE 35 ;: SKIP IF NOT
2491 015434 104401 000763 45: TYPE $QUES ;: TYPE A '?'
2492 015440 000763 111337 35: BR 15 ;: CLEAR THE BUFFER AND LOOP
2493 015442 111337 015512 35: MOVB (R3), 95 ;: ECHO THE CHARACTER
2494 015446 104401 015512 TYPE 95 ;: CHECK FOR RETURN
2495 015452 122723 000015 CMPB #15, (R3)+ ;: LOOP IF NOT RETURN
2496 015456 001356 177777 BNE 25 ;: CLEAR RETURN (THE 15)
2497 015460 105063 001200 CLRB -1(R3) ;: TYPE A LINE FEED
2498 015464 104401 012603 TYPE $LF ;: RESTORE R3
2499 015470 012603 000004 000002 MOV (SP)+, R3 ;: ADJUST THE STACK AND PUT ADDRESS OF THE
2500 015472 011646 015514 000004 MOV (SP), -(SP) ;: FIRST ASCII CHARACTER ON IT
2501 015474 016666 012766 000004 MOV 4(SP), 2(SP) ;:
2502 015502 012766 000002 RTI ;: RETURN
2503 015510 000002 95: .BYTE 0 ;: STORAGE FOR ASCII CHAR. TO TYPE

```

```

2502 015513 000 .BYTE 0 ; TERMINATOR
2503 015514 000010 $TTYIN: .BLKB 8 ; RESERVE 8 BYTES FOR TTY INPUT
2504 015524 052536 005015 000 $CNTLU: .ASCIZ /↑U/<15><12> ; CONTROL "U"
2505 015531 136 006507 000012 $CNTLG: .ASCIZ /↑G/<15><12> ; CONTROL "G"
2506 015536 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
2507 015544 020075 000
2508 015547 040 047040 053505 $MNEW: .ASCIZ / NEW = /
2509 015554 036440 000040
2510 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2511
2512 ; *****
2513 ; THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2514 ; CHANGE IT TO BINARY.
2515 ; CALL:
2516 ; * RDOCT ; READ AN OCTAL NUMBER
2517 ; * RETURN HERE ; LOW ORDER BITS ARE ON TOP OF THE STACK
2518 ; * ; HIGH ORDER BITS ARE IN $HIOCT
2519
2520 015560 011646 000004 000002 $RDOCT: MOV (SP), -(SP) ; PROVIDE SPACE FOR THE
2521 015562 016666 000004 000002 MOV 4(SP), 2(SP) ; INPUT NUMBER
2522 015570 010046 MOV RO, -(SP) ; PUSH RO ON STACK
2523 015572 010146 MOV R1, -(SP) ; PUSH R1 ON STACK
2524 015574 010246 MOV R2, -(SP) ; PUSH R2 ON STACK
2525 015576 104411 1$: RDLIN ; READ AN ASCIZ LINE
2526 015600 012600 MOV (SP)+, RO ; GET ADDRESS OF 1ST CHARACTER
2527 015602 005001 CLR R1 ; CLEAR DATA WORD
2528 015604 005002 CLR R2
2529 015606 112046 2$: MOVB (RO)+, -(SP) ; PICKUP THIS CHARACTER
2530 015610 001412 BEQ 3$ ; IF ZERO GET OUT
2531 015612 006301 ASL R1 ; *2
2532 015614 006102 ROL R2
2533 015616 006301 ASL R1 ; *4
2534 015620 006102 ROL R2
2535 015622 006301 ASL R1 ; *8
2536 015624 006102 ROL R2
2537 015626 042716 177770 BIC #↑C7, (SP) ; STRIP THE ASCII JUNK
2538 015632 062601 ADD (SP)+, R1 ; ADD IN THIS DIGIT
2539 015634 000764 BR 2$ ; LOOP
2540 015636 005726 3$: TST (SP)+ ; CLEAN TERMINATOR FROM STACK
2541 015640 010166 000012 MOV R1, 12(SP) ; SAVE THE RESULT
2542 015644 010237 015660 MOV R2, $HIOCT
2543 015650 012602 MOV (SP)+, R2 ; POP STACK INTO R2
2544 015652 012601 MOV (SP)+, R1 ; POP STACK INTO R1
2545 015654 012600 MOV (SP)+, RO ; POP STACK INTO RO
2546 015656 000002 RTI ; RETURN
2547 015660 000000 $HIOCT: .WORD 0 ; HIGH ORDER BITS GO HERE
2548 .SBTTL APT COMMUNICATIONS ROUTINE
2549
2550 ; *****
2551 015662 112737 000001 016126 $ATY1: MOVB #1, $FFLG ; TO REPORT FATAL ERROR
2552 015670 112737 000001 016124 $ATY3: MOVB #1, $MFLG ; TO TYPE A MESSAGE
2553 015676 000403 BR $ATYC
2554 015700 112737 000001 016126 $ATY4: MOVB #1, $FFLG ; TO ONLY REPORT FATAL ERROR
2555 015706 $ATYC:

```



```

2556 015706 010046      MOV      RO,-(SP)      ;; PUSH RO ON STACK
2557 015710 010146      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
2558 015712 105737 016124  TSTB    $MFLG         ;; SHOULD TYPE A MESSAGE?
2559 015716 001450      BEQ     5$            ;; IF NOT: BR
2560 015720 122737 000001 001222  CMPB    #APTENV,$ENV   ;; OPERATING UNDER APT?
2561 015726 001031      BNE     3$            ;; IF NOT: BR
2562 015730 132737 000100 001223  BITB    #APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
2563 015736 001425      BEQ     3$            ;; IF NOT: BR
2564 015740 017600 000004      MOV     @4(SP),RO     ;; GET MESSAGE ADDR.
2565 015744 062766 000002 000004  ADD     #2,4(SP)      ;; BUMP RETURN ADDR.
2566 015752 005737 001202      TST     $MSGTYPE     ;; SEE IF DONE W/ LAST XMISSION?
2567 015756 001375      BNE     1$            ;; IF NOT: WAIT
2568 015760 010037 001216      MOV     RO,$MSGAD     ;; PUT ADDR IN MAILBOX
2569 015764 105720      TSTB   (RO)+         ;; FIND END OF MESSAGE
2570 015766 001376      BNE     2$
2571 015770 163700 001216      SUB     $MSGAD,RO     ;; SUB START OF MESSAGE
2572 015774 006200      ASR    RO            ;; GET MESSAGE LNTH IN WORDS
2573 015776 010037 001220      MOV     RO,$MSGGLT    ;; PUT LENGTH IN MAILBOX
2574 016002 012737 000004 001202  MOV     #4,$MSGTYPE   ;; TELL APT TO TAKE MSG.
2575 016010 000413      BR      5$
2576 016012 017637 000004 016036 3$:     MOV     @4(SP),4$     ;; PUT MSG ADDR IN JSR LINKAGE
2577 016020 062766 000002 000004  ADD     #2,4(SP)     ;; BUMP RETURN ADDRESS
2578 016026 013746 177776      MOV     177776,-(SP) ;; PUSH 177776 ON STACK
2579 016032 004737 014522      JSR    PC,$TYPE      ;; CALL TYPE MACRO
2580 016036 000000      4$:     .WORD 0
2581 016040      5$:
2582 016040 105737 016126      10$:   TSTB    $FFLG         ;; SHOULD REPORT FATAL ERROR?
2583 016044 001416      BEQ     12$          ;; IF NOT: BR
2584 016046 005737 001222      TST     $ENV         ;; RUNNING UNDER APT?
2585 016052 001413      BEQ     12$          ;; IF NOT: BR
2586 016054 005737 001202      11$:   TST     $MSGTYPE     ;; FINISHED LAST MESSAGE?
2587 016060 001375      BNE     11$          ;; IF NOT: WAIT
2588 016062 017637 000004 001204  MOV     @4(SP),$FATAL ;; GET ERROR #
2589 016070 062766 000002 000004  ADD     #2,4(SP)     ;; BUMP RETURN ADDR.
2590 016076 005237 001202      INC    $MSGTYPE     ;; TELL APT TO TAKE ERROR
2591 016102 105037 016126      12$:   CLRB   $FFLG         ;; CLEAR FATAL FLAG
2592 016106 105037 016125      CLRB   $LFLG        ;; CLEAR LOG FLAG
2593 016112 105037 016124      CLRB   $MFLG        ;; CLEAR MESSAGE FLAG
2594 016116 012601      MOV    (SP)+,R1     ;; POP STACK INTO R1
2595 016120 012600      MOV    (SP)+,RO     ;; POP STACK INTO RO
2596 016122 000207      RTS    PC           ;; RETURN
2597 016124 000      $MFLG: .BYTE 0     ;; MESSG. FLAG
2598 016125 000      $LFLG: .BYTE 0     ;; LOG FLAG
2599 016126 000      $FFLG: .BYTE 0     ;; FATAL FLAG
2600      .EVEN
2601 000200  APTSIZE=200
2602 000001  APTENV=001
2603 000100  APTPOOL=100
2604 000040  APTCSUP=040
2605      .SBTTL TRAP DECODER
2606
2607      ;; *****
2608      ;; *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2609      ;; *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS

```

```

2610 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2611 ;*GO TO THAT ROUTINE.
2612
2613 016130 010046 STRAP: MOV RO, -(SP) ;:SAVE RO
2614 016132 016600 000002 MOV 2(SP),RO ;:GET TRAP ADDRESS
2615 016136 005740 TST -(RO) ;:BACKUP BY 2
2616 016140 111000 MOVB (RO),RO ;:GET RIGHT BYTE OF TRAP
2617 016142 006300 ASL RO ;:POSITION FOR INDEXING
2618 016144 016000 016164 MOV $TRPAD(RO),RO ;:INDEX TO TABLE
2619 016150 000200 RTS RO ;:GO TO ROUTINE
2620
2621
2622 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2623
2624 016152 011646 STRAP2: MOV (SP),-(SP) ;:MOVE THE PC DOWN
2625 016154 016666 000004 000002 MOV 4(SP),2(SP) ;:MOVE THE PSW DOWN
2626 016162 000002 RTI ;:RESTORE THE PSW
2627
2628 .SBTTL TRAP TABLE
2629
2630 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2631 ;*BY THE "TRAP" INSTRUCTION.
2632
2633 : ROUTINE
2634 : -----
2635 016164 016152 $TRPAD: .WORD STRAP2
2636 016166 014522 $TYPE ;:CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2637 016170 014320 $TYPOC ;:CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2638 016172 014274 $TYPOS ;:CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2639 016174 014334 $TYPON ;:CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2640 016176 007574 $TYPDS ;:CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
2641
2642 016200 015054 $GTSWR ;:CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
2643
2644 016202 015004 $CKSWR ;:CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
2645 016204 015266 $RDCHR ;:CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
2646 016206 015406 $RDLIN ;:CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
2647 016210 015560 $RDOCT ;:CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2648
2649
2650 ;*
2651 ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
2652 ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
2653 ;*NEXT WE WILL INIT BOTH UPROCESSORS
2654 ;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
2655 ;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
2656
2657 ;*
2658 ;* CALL= JSR R5,$LPAI
2659 ;* .WORD 0 ;ADDR. OF DEVICE ADDRESS.
2660 ;* ROUTINES REQUIRED: .LOADLP
2661 ;* PROGRAMS REQUIRED: DRLPX2
2662
2663 ;*
2664 ;* RETURNS WITH $AERR=1 IF SLAVE
2665 ;* MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
    
```

```

2664
2665 016212          SLPAI:  ;*
2666 016212 013746 000004      MOV    4,-(SP)
2667
2668 016216 000413          BR     31$
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681 016220 012737 016244 000004      MOV    #30$,4
2682 016226 005237 170000          INC    170000
2683 016232 104401 016240          TYPE  65$
2684 016236 000401          BR     64$
2685
2686 016242          ;;65$: .ASCIZ <7>##
2687 016242 000401          64$: BR     31$
2688 016244 022626          30$: CMP    (SP)+,(SP)+
2689 016246 012637 000004          31$: MOV    (SP)+,4
2690 016252 005037 017070          CLR    $AERR
2691 016256 004537 017072          JSR    R5,$LOAD
2692 016262 000000G        .WORD  DRLPX2
2693
2694 016264 052777 040000 163072      BIS    #BIT14,$KMADO
2695
2696 016272          1$:
2697
2698 016272 010146          MOV    R1,-(SP)
2699 016274 005001          CLR    R1
2700 016276 005201          2$: INC    R1
2701 016300 001376          BNE    2$
2702 016302 012777 104000 163054      MOV    #BIT15!BIT11,$KMADO
2703 016310 105201          25$: INCB  R1
2704 016312 001376          BNE    25$
2705
2706 016314 032777 000040 163042      BIT    #BITS,$KMADO
2707 016322 001401          BEQ    3$
2708
2709 016324 104000          ERROR
2710
2711 016326 012777 000004 163034      3$: MOV    #4,$KMAD2
2712 016334          4$:
2713 016334 004537 020002          JSR    R5,$TOUT
2714
2715 016340 104000          ERROR
2716
2717

```

```

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

```

```

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

```

```

;;TYPE ASCIZ STRING
;;GET OVER THE ASCIZ

```

```

;ALL THIS JUNK MUST BE REMOVED!!

```

```

;LOAD MICRO-CODE.
;FILE "DRLPX2.OBJ"

```

```

;ISSUE KMC+DMC INIT.

```

```

;"HANGS" HERE THEN KMC-11 ERROR.

```

```

;STALL FOR DMC-UP

```

```

;SET RUN, AND ENABLE ARBITRATION.

```

```

;SLAVE READY? (READING IPBM SR)

```

```

;FATAL LPA-11 ERROR SLAVE NOT READY.

```

```

;READ FAST PATH

```

```

;-TOUT-CHECK FOR TIMEOUT

```

```

;/TIME-OUT ERROR
;/WE FAILED TO COMPLETE
;/CURRENT OPERATION.

```

```

2718 ;/CONTINUES IN THIS LOOP
2719 ;/WOULD MAKE US "HANG" HERE
2720
2721 016342 000774 BR 4$
2722
2723 ;/RETURNS HERE-FROM-TIMED OUT.
2724 016344 122777 000377 163016 CMPB #377,AKMAD2 ;WAIT TILL KMC DONE COMMAND.
2725 016352 001370 BNE 4$
2726 016354 122777 000377 163012 CMPB #377,AKMAD4 ;IF FAST PATH=377 THEN ERROR.
2727 016362 001001 BNE 35$
2728 016364 104000 ERROR ;IPBM ERROR (SLAVE SIDE)
2729 ;YOU MUST RUN IPBM DIAGNOSTIC.
2730
2731 016366 122777 000004 163000 35$: CMPB #4,AKMAD4 ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
2732 016374 001543 BEQ 5$ ;YES-CONTINUE.
2733 016376 005227 177777 INC #-1
2734 016402 001140 BNE 5$
2735 016404 005227 177777 INC #-1
2736 016410 001135 BNE 5$
2737 016412 104401 016420 TYPE ,67$ ;:TYPE ASCIZ STRING
2738 016416 000440 BR 66$ ;:GET OVER THE ASCIZ
2739 ;:67$: .ASCIZ <200>"W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
2740 016520 66$: TYPE ,69$ ;:TYPE ASCIZ STRING
2741 016520 104401 016526 BR 68$ ;:GET OVER THE ASCIZ
2742 016524 000430 ;:69$: .ASCIZ <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
2743 68$:
2744 016606 TYPE ,71$ ;:TYPE ASCIZ STRING
2745 016606 104401 016614 BR 70$ ;:GET OVER THE ASCIZ
2746 016612 000434 ;:71$: .ASCIZ <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
2747 70$:
2748 016704
2749
2750 016704 112737 177777 017036 5$: MOVB #0-1,11$ ;:DAC CODE FOR SLAVE.
2751 016712 012501 MOV (5)+,R1 ;:GET NEXT DEVICE ADDR.
2752 016714 021127 000000 6$: CMP (R1),#0 ;:TERM REACHED?
2753 016720 001444 BEQ 10$
2754 016722 105237 017036 INCB 11$
2755 016726 113777 017036 162440 MOVB 11$,AKMAD4 ;:FIFO DATA
2756 016734 004737 017040 JSR PC,20$ ;:ISSUE SEND
2757 016740 112177 162430 MOVB (R1)+,AKMAD4 ;:SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
2758 016744 004737 017040 JSR PC,20$ ;:ISSUE SEND
2759 016750 112177 162420 MOVB (R1)+,AKMAD4 ;:SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
2760 016754 004737 017040 JSR PC,20$
2761
2762 016760 032777 000002 162376 7$: BIT #BIT1,AKMAD0 ;:WAIT FOR FIFO DATA
2763 016766 001374 BNE 7$ ;:=1 NO DATA. =0 DATA.
2764 016770 112777 000002 162372 MOVB #2,AKMAD2 ;:READ FIFO.
2765
2766 016776 8$:
2767 016776 004537 020002 JSR R5, $TOU1 ;:-TOUT-CHECK FOR TIMEOUT
2768
2769 017002 104000 ERROR ;:/TIME-OUT ERROR
2770 ;:/WE FAILED TO COMPLETE
2771 ;:/CURRENT OPERATION.

```

```

2772 ;/CONTINUES IN THIS LOOP
2773 ;/WOULD MAKE US "HANG" HERE
2774
2775 017004 000774 BR 8$
2776
2777 ;/RETURNS HERE-FROM-TIMED OUT.
2778 017006 122777 000377 162354 CMPB #377,@KMA2 ;WAIT FOR READ.
2779 017014 001370 BNE 8$
2780 017016 105777 162352 ISTB @KMA4 ;WAS A ZERO RETURNED?
2781 017022 001734 BEQ 6$ ;YES GET NEXT ADDR.
2782 ;SLAVE WILL RETURN CODE 0 IF
2783 017024 005237 017070 INC $AERR ;DEV PRESENT ELSE
2784 ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
2785 017030 005041 CLR -(1) ;GET RID OF REFERENCE TO BAD ADDR.
2786 017032 012601 10$: MOV (SP)+,R1
2787 017034 000205 RTS R5 ;RETURN ALL ADDR. CHECKED.
2788
2789 017036 000000 11$: .WORD 0 ;HOLDS DAC CODE PLUS OFFSET
2790 ;TO SLAVES ADDR. TABLE.
2791
2792 017040 112777 000003 162322 20$: MOVB #3,@KMA2 ;ISSUE FIFO WRITE
2793 017046 004537 020002 21$: JSR R5,$TOUT ;-TOUT-CHECK FOR TIMEOUT
2794
2795 017052 104000 ERROR ;/TIME-OUT ERROR
2796 ;/WE FAILED TO COMPLETE
2797 ;/CURRENT OPERATION.
2798 ;/CONTINUES IN THIS LOOP
2799 ;/WOULD MAKE US "HANG" HERE
2800
2801
2802 017054 000774 BR 21$
2803
2804 ;/RETURNS HERE-FROM-TIMED OUT.
2805 017056 122777 000377 162304 CMPB #377,@KMA2 ;KMC CODE WILL RETURN A "377"
2806 017064 001370 BNE 21$ ;WHEN DONE COMMAND.
2807 017066 000207 RTS PC
2808
2809 017070 000000 $AERR: .WORD 0 ;=0 IF ADDR. LIST OK,=1 IF BAD.
2810
2811 ;*
2812 ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
2813 ;* CALL = JSR R5,$LOAD
2814 ;* .WORD XX ;ADDR. OF MICRO CODE.
2815 ;* ;RETURNS HERE
2816 ;* NOTE: MICRO CODE FILE MUST END IN -1 DATA.
2817 ;*
2818
2819 $LOAD: MOV R4,-(SP) ;SAVE R4.
2820 017074 010046 MOV R0,-(SP) ;SAVE R0.
2821 017076 012500 1$: MOV (5)+,R0 ;GET PROG. ADDR.
2822 017100 005077 162260 CLR @KMA0 ;CLEAR CSR
2823 017104 005077 162264 CLR @KMA4 ;CLEAR CRAM ADDR.
2824 017110 052777 002000 162246 2$: BIS #2000,@KMA0 ;SELECT CRAM.
2825 017116 012077 162256 MOV (0)+,@KMA6 ;WRITE DATA.

```

2826	017122	052777	020000	162234		BIS	#20000, @KMADO	; SET CRAM WRITE
2827	017130	005077	162230			CLR	@KMADO	; DISABLE CRAM.
2828	017134	005277	162234			INC	@KMAD4	; UPDATE CRAM ADDR.
2829	017140	021027	177777			CMP	(0), #-1	; ALL DONE?
2830	017144	001361				BNE	2\$; NO LOOP.
2831	017146	005077	162222			CLR	@KMAD4	; CLEAR CRAM ADDR.
2832	017152	016500	177776			MOV	-2(5), R0	; GET MICRO CODE ADDR.
2833								
2834	017156	052777	002000	162200	3\$:	BIS	#2000, @KMADO	; SELECT CRAM
2835	017164	022077	162210			CMP	(R0)+, @KMAD6	; DATA OK?
2836	017170	001013				BNE	5\$; NO - REPORT AN ERROR.
2837	017172	021027	177777			CMP	(0), #-1	; ALL DONE?
2838	017176	001405				BEQ	4\$; YES - EXIT
2839	017200	005077	162160			CLR	@KMADO	; NO - DESELECT CRAM.
2840	017204	005277	162164			INC	@KMAD4	; UPDATE CRAM ADDR.
2841	017210	000762				BR	3\$	
2842								
2843	017212	012600			4\$:	MOV	(SP)+, R0	; RESTORE R0
2844	017214	012604				MOV	(SP)+, R4	; RESTORE R4
2845	017216	000205				RTS	R5	; EXIT
2846								
2847	017220				5\$:			; COME HERE ON LOAD ERROR
2848	017220	005745				TST	-(5)	
2849	017222	105204				INCB	R4	; UPDATE ERROR COUNTER.
2850	017224	100324				BPL	1\$; IF NOT TOO MANY, TRY AGAIN.
2851	017226	000000				HALT		; MICRO CODE LOAD ERROR.
2852								; KMC-11 FAULT. YOU COULD TRY
2853	017230	000722				BR	1\$; TO PRESS CONTINUE TO GIVE IT
2854								; ANOTHER CHANCE, BUT I DOUBT
2855								; THAT THAT WOULD WORK. SINCE I'VE
2856								; ALREADY GIVEN IT 177 (OCTAL) CHANCES.
2857								; TRY RUNNING THE KMC-11 DIAGNOSTIC.
2858								
2859								
2860								
2861								
2862								
2863								
2864								
2865								
2866								
2867	017232	010046			\$TLKW:	MOV	R0, -(SP)	; SAVE R0
2868	017234	012500				MOV	(5)+, R0	; GET DEVICE OFFSET
2869	017236	052700	000340			BIS	#340, R0	; ADD WRITE CODE.
2870	017242	004737	017514			JSR	PC, \$LPW	; WAIT FOR FAST PATH READY
2871	017246	010037	017340			MOV	R0, W1	
2872	017252	010077	162116			MOV	R0, @KMAD4	
2873	017256	112777	000005	162104		MOVB	#5, @KMAD2	; ISSUE FAST PATH WRITE
2874	017264	004737	017514			JSR	PC, \$LPW	; WAIT FOR RDY
2875	017270	011537	017342			MOV	(5) W2	
2876	017274	112577	162074			MOVB	(5)+, @KMAD4	; WRITE LOW BYTE DATA.
2877								
2878	017300	112777	000005	162062		MOVB	#5, @KMAD2	; FP WRITE
2879	017306	004737	017514			JSR	PC, \$LPW	

```

; *THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
; *
; *      CALL = JSR      R5, $TLKW
; *              .WORD   0
; *              .WORD   0
; *              ; OFFSET OF DEVICE ADDR.
; *              ; DATA TO BE WRITTEN
    
```

```

2880 017312 111537 017344          MOVB      (5),W3
2881 017316 112577 162052          MOVB      (5)+,AKMAD4      ;WRITE HIGH BYTE
2882 017322 112777 000005 162040  MOVB      #5,AKMAD2
2883 017330 004737 017514          JSR      PC,$LPW
2884 017334 012600          MOV      (SP)+,RO
2885 017336 000205          RTS      R5                ;EXIT DONE.
2886 017340 000000          W1:      0
2887 017342 000000          W2:      0
2888 017344 000000          W3:      0
2889
2890          ;*
2891          ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
2892          ;*
2893          ;*      CALL = JSR      R5,$TLKR
2894          ;*      .WORD      0                ;OFFSET OF DEVICE
2895          ;*      ;RETURNS HERE
2896          ;*DATA IN WORD $DATR
2897          ;*
2898
2899 017346 010046          $TLKR:  MOV      RO,-(SP)      ;SAVE RO
2900 017350 012500          MOV      (5)+,RO          ;GET OFFSET
2901 017352 052700 000300          BIS      #300,RO         ;ADD READ CODE
2902 017356 004737 017514          JSR      PC,$LPW         ;WAIT TILL READY
2903 017362 110077 162006          MOVB     RO,AKMAD4
2904 017366 112777 000005 161774  MOVB     #5,AKMAD2      ;ISSUE WRITE FP
2905 017374 004737 017514          JSR      PC,$LPW
2906 017400 010037 017510          MOV      RO,RO1
2907 017404          1$:
2908 017404 004537 020002          JSR      R5,$TOUT        ;-TOUT-CHECK FOR TIMEOUT
2909
2910 017410 104000          ERROR          ;/TIME-OUT ERROR
2911          ;/WE FAILED TO COMPLETE
2912          ;/CURRENT OPERATION.
2913          ;/CONTINUES IN THIS LOOP
2914          ;/WOULD MAKE US "HANG" HERE
2915
2916 017412 000774          BR              1$
2917
2918          ;/RETURNS HERE-FROM-TIMED OUT.
2919 017414 032777 000040 161742          BIT      #BITS,AKMADO    ;FAST PATH GOT DATA?
2920 017422 001370          BNE      1$
2921 017424 112777 000004 161736          MOVB     #4,AKMAD2      ;ISSUE FAST PATH READ
2922 017432 004737 017514          JSR      PC,$LPW
2923 017436 117737 161732 017512  MOVB     AKMAD4,$DATR    ;GET LOW BYTE
2924 017444          2$:
2925 017444 004537 020002          JSR      R5,$TOUT        ;-TOUT-CHECK FOR TIMEOUT
2926
2927 017450 104000          ERROR          ;/TIME-OUT ERROR
2928          ;/WE FAILED TO COMPLETE
2929          ;/CURRENT OPERATION.
2930          ;/CONTINUES IN THIS LOOP
2931          ;/WOULD MAKE US "HANG" HERE
2932
2933 017452 000774          BR              2$

```

```

2934
2935
2936 017454 032777 000040 161702 BIT #BITS,AKMAD0 ;/RETURNS HERE-FROM-TIMED OUT.
2937 017462 001370 BNE 2$ ;FAST PATH READY?
2938 017464 112777 000004 161676 MOVB #4,AKMAD2 ;ISSUE FAST PATH READ
2939 017472 004737 017514 JSR PC,SLPW
2940 017476 117737 161672 017513 MOVB AKMAD4,$DATR+1 ;SAVE HIGH BYTE
2941 017504 012600 MOV (SP)+,R0
2942 017506 000205 RTS R5
2943 017510 000000
2944 017512 000000 RDI: 0
SDATR: .WORD 0
;
; THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
; AS FAST PATH TO BE READ.
;
; CALL = JSR PC,SLPW
;
; IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
; THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
;
2955 017514 010146 SLPW: MOV R1,-(SP) ;SAVE R1
2956 017516 005001 CLR R1
2957 017520 122777 000377 161642 1$: CMPB #377,AKMAD2 ;FINISHED INSTRUCTION?
2958 017526 001403 BEQ 2$
2959 017530 005201 INC R1 ;TIME OUT?
2960 017532 001372 BNE 1$
2961 017534 000411 BR 10$
2963 017536 032777 000020 161620 2$: BIT #BIT4,AKMAD0 ;FAST PATH READ?
2964 017544 001403 BEQ 3$
2965 017546 005201 INC R1 ;NO - TIME OUT?
2966 017550 001372 BNE 2$
2967 017552 000402 BR 10$ ;YES - REPORT AN ERROR
2969 017554 012601 3$: MOV (SP)+,R1 ;RESTORE R1
2970 017556 000207 RTS PC ;EXIT
2972 017560 104401 017566 10$: TYPE 65$ ;;TYPE ASCIZ STRING
2973 017560 000407 BR 64$ ;;GET OVER THE ASCIZ
2974 017564 000407 ;;65$: .ASCIZ <200>#LPA-11 FAULT#
2975 017604 64$:
2978 017604 000000 11$: HALT ;LPA-11 FAULT RUN LPA-11
2979 017606 000776 BR 11$ ;DIAGNOSTICS.
;
; *
; * THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
; * A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
; *
; *
; * FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED

```



```

2988
2989
2990
2991
2992
2993
2994 017610 010046 $OUTLP: MOV R0,-(SP) ;SAVE R0
2995 017612 010146 MOV R1,-(SP) ;SAVE R1
2996
2997 017614 012700 001412 MOV #.DVLS,R0 ;PROGRAM DEFINED LIST.
2998 017620 005001 CLR R1
2999 017622 005710 1$: TST (0) ;TERMINATOR REACHED?
3000 017624 001421 BEQ 10$ ;YES NEXT STEP.
3001 017626 027520 000000 CMP @($),(0)+ ;MATCH WITH ADDR IN LIST?
3002 017632 001402 BEQ 2$
3003 017634 005201 INC R1
3004 017636 000771 BR 1$
3005
3006 017640 010137 017656 2$: MOV R1,3$ ;SAVE OFFSET, DEVICE KNOWN.
3007 017644 005725 TST ($)+
3008 017646 013537 017660 MOV @($)+,4$ ;GET DATA TO BE WRITTEN
3009 017652 004537 017232 JSR R5,$TLKw ;DO WRITE
3010 017656 000000 3$: .WORD 0 ;DEVICE OFFSET
3011 017660 000000 4$: .WORD 0 ;DATA TO BE WRITTEN.
3012 017662 012601 MOV (SP)+,R1
3013 017664 012600 MOV (SP)+,R0
3014 017666 000205 RTS R5
3015 017670 017520 000000 10$: MOV @($),(0)+ ;SAVE ADDR.
3016 017674 005010 CLR (0)
3017 017676 004537 016212 JSR R5,$LPAI
3018 017702 001412 .WORD .DVLS
3019 017704 000755 BR 2$
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036 017706 010046 $INLP: MOV R0,-(SP) ;SAVE R0
3037 017710 010146 MOV R1,-(SP) ;SAVE R1
3038
3039 017712 012700 001412 MOV #.DVLS,R0 ;PROG DEFINED ADDR. LIST.
3040 017716 005001 CLR R1
3041 017720 005710 1$: TST (0) ;EOL REACHED?

```

```

;* BEFORE, IF NOT WE HAVE TO INITIALIZE THE LPA WITH
;* THAT ADDRESS.
;* WHEN THE ADDP. IS KNOWN BY THE LPA, DO THE OUTPUT BY
;* $TLKw
;*

```

```

;*
;* THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
;* TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
;*
;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
;* USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
;* WITH THE NEW ADDR.
;* WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
;* $TLKR
;*
;* CALL THROUGH MOVEI DATA,ADDR.
;* WHICH EQUALS:
;* JSR R5,$INLP
;* .WORD XX ADDR OF DEVICE
;* .WORD YY ADDR TO STORE READ DATA.

```

```

3042 017722 001420          BEQ      10$          ;YES - DEFINE NEW ADDR.
3043
3044 017724 027520 000000    CMP      @ (5), (0)+      ;ADDR. MATCH?
3045 017730 001402          BEQ      2$
3046 017732 005201          INC      R1
3047 017734 000771          BR       1$
3048
3049 017736 010137 017750    2$:      MOV      R1, 3$          ;SAVE LIST OFFSET
3050 017742 005725          TST     (5)+
3051 017744 004537 017346    JSR     R5, $TLKR        ;GO READ DEVICE
3052 017750
3053 017750 000000    $OFS=. 3$:      .WORD   0          ;OFFSET OF DEVICE
3054
3055 017752 013735 017512    MOV     $DATR, @ (5)+    ;STORE DATA.
3056 017756 012601          MOV     (SP)+, R1        ;RESTORE R1
3057 017760 012600          MOV     (SP)+, R0        ;RESTORE R2
3058 017762 000205          RTS     R5              ;EXIT
3059
3060 017764 017520 000000    10$:     MOV     @ (5), (0)+
3061 017770 005010          CLR     (0)
3062 017772 004537 016212    JSR     R5, $LPAI
3063 017776 001412          .WORD  .DVL$
3064 020000 000756          BR       2$
3065
3066          ;*
3067          ;* $STOUT ROUTINE USED TO WATCH IF
3068          ;* WE'RE IN A LOOP TOO-LONG
3069          ;* CALL= JSR R5, $STOUT
3070          ;* ERROR X ;RETURNS HERE ON TIMEOUT
3071          ;* BR
3072          ;* ;RETURNS HERE NO ERROR
3073          ;*
3074 020002 020537 020036    $STOUT:  CMP     R5, $SAD        ;SAME ADDR?
3075 020006 001405          BEQ     1$
3076 020010 010537 020036    MOV     R5, $SAD        ;NO-SAVE THIS ADDR.
3077 020014 005037 020040    CLR     $CNT            ;CLR CNT AT ADDR.
3078 020020 000403          BR       2$
3079 020022 005237 020040    1$:      INC     $CNT            ;OVERFLOW?
3080 020026 100402          BMI     3$              ;YES-ERROR RETURN
3081 020030 062705 000004    2$:      ADD     #4, R5          ;NO-NON ERROR RETURN
3082 020034 000205          3$:      RTS     R5              ;RETURN.
3083
3084 020036 000000    $SAD:   .WORD   0          ;CONTAINS LOOP ADDR.
3085 020040 000000    $CNT:   .WORD   0          ;# OF TIMES AT ADDR.
3086
3087
3088          ;*
3089          ;* THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
3090          ;* USE FOR A RESET. FIRST WE DO A RESET INSTRUCTION.
3091          ;* THEN WE CLR ".DVLST" WHICH FORCES US TO RESET BOTH THE
3092          ;* KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.
3093          ;*
3094          ;* CALL=JSR PC, $RESET ;REPLACES "RESET INSTRUCTION
3095          ;* ;RETURNS HERE.
3096          ;*
    
```

```

3096 020042 000005 $RESET: RESET ;RESET THE WORLD.
3097
3098 ;* MOV 22$,1$ ;/READ DEVICE REG 2$,PUT DATA IN 1$.
3099 020054 005737 017070 ;* TST $AERR ;IF NO ERROR,LOOP
3100 020060 001004 ;* BNE 10$ ;THERE WAS AN ERROR.
3101 020062 062737 000002 020076 ;* ADD #2,2$ ;UPDATE DEVICE ADDR.
3102 ;* ;YOU SEE , WE HAVE TO PROTECT OUR SELF!
3103 ;* ;IF 2$ CONTAINED A VALID ADDR,WE
3104 ;* ;MUST KEEP TRYING UNTIL WE GENERATE
3105 ;* ;AN INVALID ADDR.
3106 020070 000764 BR $RESET
3107 020072 10$:
3108 020072 000207 RTS PC
3109 020074 000000 1$: .WORD 0 ;JUNK LOC.
3110 020076 160000 2$: .WORD 160000 ;DUMB ADDR. FORCES INIT OF DMC/KMC.
3111
3112
3113 ;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
3114 ; IS NOT TIME DEPENDENT CODE SENCE
3115 ; NOT USED TO GET SPECIFIC TIME BUT
3116 ; JUST A LITTLE DELAY.
3117
3118 ;
3119 ; THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
3120 ; THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
3121 ;
3122 ;
3123 ; CALL= JSR PC, SDELAY
3124 ;
3125 SDELAY:
3126 020100 TST RTCCSR ;CLOCK PRESENT?
3127 020100 005737 020162 BPL 10$
3128 020104 100016 MOV #2,TIME
3129 020106 012737 000002 020152 000040 BIS #15,RTCCSR ;START CLOCK
3130 020114 052777 000115 CLR PS
3131 020122 005037 177776 1$: TST TIME
3132 020126 005737 020152 BNE 1$
3133 020132 001375 CLR RTCCSR ;STOP CLOCK
3134 020134 005077 000022
3135 020140 000207 RTS PC
3136 020142 105237 020152 10$: INCB TIME
3137 020146 001375 BNE 10$
3138 020150 000207 RTS PC
3139
3140 020152 000000 TIME: .WORD 0
3141
3142 020154 005337 020152 CLKINT: DEC TIME
3143 020160 000002 RTI
3144 020162 000000 RTCCSR: .WORD 0 ;CLOCK CSR IF USED.
3145
3146 ;
3147 ;*
3148 ;*THIS MACRO ALLOWS THE OPERATOR TO TALK TO
3149 ;*ANY DEVICE ON THE I/O BUS
; *USER MUST START AT THIS ADDR.

```


3204					
3205					
3206					
3207					
3208					
3209					
3210					
3211					
3212					
3213					
3214					
3215					
3216					
3217					
3218					
3219					
3220					
3221					
3222					
3223					
3224	020360	012537	020370		
3225	020364	004537	017706		
3226	020370	000000			
3227	020372	020466			
3228	020374	113777	017750	160776	
3229	020402	113777	017750	160772	
3230	020410	013737	020370	020430	
3231	020416	062737	000002	020430	
3232	020424	004537	017706		
3233	020430	000000			
3234	020432	020466			
3235	020434	113777	017750	160730	
3236	020442	152777	000340	160730	
3237	020450	152777	000300	160724	
3238	020456	152777	000300	160706	
3239	020464	000205			
3240	020466	000000			
3241					
3242					
3243	020470	000000			
3244		020532			
3245	020532	000000			
3246		020574			
3247	020574	000000			
3248		020636			
3249	020636	000000			
3250		020700			
3251	020700	000000			
3252		020742			
3253	020742	000000			
3254		021004			
3255	021004	000000			
3256		021046			
3257	021046	000000			

```

THIS ROUTINE LOOKS THROUGH CURENT DVLS FOR A/D ADDR.
IF UNFOUND GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
TO SET UP THE USER PROGRAM TO LINK TO FILE "DRLPX2" FOR
SAMPLE TAKEING PURPOSES.
TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
A/D CSR IN BSEL 4, AND 5.
(2) HE MUST CALL THIS ROUTINE:
      JSR      R5,$PUTS      ;CALL SET UP ROUTINE.
      .WORD   ADCSR        ;ADDR. OF A/D CSR.
      ;RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
                          ;(UNTILL ONE DOES A RESET)

(3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
START CONVERSION CAUTION*DO WITH MOVVB INSTR.!
(4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
(5)READ KMC REG 4,5 FOR A/D RESULT.
(6) TO TAKE MORE SAMPLES, SIMPLY PUT A/D CSR INTO
BSEL 4,5 AND CODE 6 INTO BSEL 2.

```

```

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

ADBUFF: 0
      .=. +40
ADTB0:  0
      .=. +40
ADTB1:  0
      .=. +40
ADTB2:  0
      .=. +40
ADTB3:  0
      .=. +40
ADTB4:  0
      .=. +40
ADTB5:  0
      .=. +40
ADTB6:  0

```

3258		021110
3259	021110	000000
3260		021152
3261		000001

ADTB7: 0 .=.+40
 . .=.+40
 .END

CPTYPE	010652	492*	493*	952	1665#																	
CR	= 000015	50#	2361	2371																		
CRLF	= 000200	51#	2332	2371																		
CSB	001346	359#	550	1094																		
CSC	001356	365#	556	1362																		
CSR	001344	358#	544	1098																		
DDISP	= 177570	57#	241	451																		
DECOUT	010754	1680	1686#																			
DECPNT	011042	1673#	1677	1679	1684*	1701#																
DECPRT	010656	1251	1671#																			
DEC1	011002	1687	1689	1692#																		
DEC2	011016	1691	1694#																			
DELAY	013120	1293*	1298*	1622	1656*	1958#																
DELAY1	013122	1629*	1630*	1959#																		
DGTO	005476	1078#																				
DGT1	005500	1079#																				
DGT2	005502	1080#																				
DGT3	005504	1081#																				
DGT4	005506	1050	1082#																			
DH1	012777	348	1929#																			
DIGCNT	011036	1672*	1681*	1688	1699#																	
DIGIT	011034	1675*	1676*	1686	1690*	1693*	1696	1698#														
DISPLA	001142	241#	451*	459*	2063*	2085*																
DISPRE	000174	174#	459																			
DOT	012263	1864#																				
DRLPX2	= *****	14#	2692																			
DSWR	= 177570	56#	240	450																		
DT1	013036	349	1936#																			
EFAK	001360	366#	513																			
EMTVEC	= 000030	145#	435*	436*																		
EM1	012734	347	1923#																			
ERRVEC	= 000004	138#	448	449*	460*	2024	2025*	2027*	2030*													
FILE1	011254	1741#	1744																			
FILE2	011312	1749#	1752																			
FIRST	013116	1957#																				
GETDAT	011120	1716#	1726																			
GNS	= *****	173	469	2636	2637	2638	2639	2640	2642	2644	2645	2646	2647	2685								
		2739	2743	2747	2975	3161	3185															
GTSWR	= 104406	2642#																				
HIGH	013104	743*	780	1952#																		
HIORDV	013150	1712*	1724*	1728*	1970#																	
HT	= 000011	48#	2330	2371																		
INBUF	010250	557	560	563	566	569	978	1518	1521*	1522*	1570#	1577										
INCR	013106	1953#																				
INITA	002264	533#	535	1940																		
INIT1	002234	496	525#																			
INIT2	002270	527	535#	574	1550																	
IOTVEC	= 000020	143#	433*	434*																		
KM300	001364	378#	410	2694*	2702*	2706	2762	2822*	2824*	2826*	2827*	2834*	2839*	2919								
		2936	2963																			
KMAD1	001366	381#	411																			
KMAD2	001370	383#	2711*	2724	2764*	2778	2792*	2805	2873*	2878*	2882*	2904*	2921*	2938*								
		2957																				
KMAD3	001372	385#	3235*	3238*																		

G

U

REPT1	006014	1161	1164*	1175															
REPT2	006100	1173	1177*																
REPT2A	006122	1182*	1263																
REPT3	006130	1183*	1262																
REPT4	006326	1192	1198	1203	1209	1212	1215*												
REPT5	006344	1218	1221*																
REPT6	006452	1241	1244*																
REPT6A	006500	1250*	1253																
REPT7	006524	1194	1199	1204	1210	1214	1255	1257*											
REPT7A	006552	1260	1263*																
REPT8	006364	1225*																	
REPT8A	006556	1223*	1224*	1225	1265*														
RESTST	007202	559	1355*																
RESVEC=	000010	139*																	
ROUTPT	004652	939	947*																
RTCCSR	020162	3126	3129*	3133*	3144*														
RO	=%000000	60*	408	410*	413*	414	419*	483*	486*	489*	492	494*	495	511*					
		512*	513	517*	519*	665*	676*	682*	695*	701*	714*	720*	733*	763*					
		775*	782*	794*	824*	827	850*	859*	876*	941	1026*	1031*	1043*	1044*					
		1045*	1046*	1047*	1048	1052*	1053*	1054*	1055	1132*	1301*	1302	1303	1304					
		1305	1306	1307	1308	1309	1314*	1323*	1362*	1363*	1366	1368*	1375*	1380					
		1385	1388*	1462	1472*	1476	1492	1493	1506*	1592*	1597	1761*	1764	2126					
		2127*	2128*	2135*	2136*	2137*	2138*	2139*	2140	2145	2150*	2152*	2156	2158					
		2170	2195*	2313	2314*	2319	2324	2327*	2522	2526*	2529	2545*	2556	2564*					
		2568	2569	2571*	2572*	2573	2595*	2613	2614*	2615	2616*	2617*	2618*	2619*					
		2820	2821*	2832*	2835	2843*	2867	2868*	2869*	2871	2872	2884*	2899	2900*					
		2901*	2903	2906	2941*	2994	2997*	3013*	3036	3039*	3057*								
ROX	003042	589*	602*	639*															
R1	=%000001	61*	409	411*	415	418*	501*	502*	503*	504	518*	519	520	521					
		655*	749*	851*	857*	877*	1027*	1032*	1055*	1056*	1057*	1058	1087*	1114					
		1300*	1302*	1303*	1304*	1305*	1306*	1307*	1308*	1309*	1316*	1319	1322	1364*					
		1369	1376*	1379	1387	1463	1476*	1477	1481	1505*	1525*	1526*	1527	1529					
		1531	1537	1541	1547	1551	1556*	1559	1561	1580*	1593	1594*	1595*	1596*					
		1597*	1599	1602	1769*	1774	1777*	1778*	1778*	2171	2194*	2523	2527*	2531*					
		2533*	2535*	2538*	2541	2544*	2557	2594*	2698	2699*	2700*	2703*	2751*	2752					
		2757	2759	2786*	2955	2956*	2959*	2965*	2969*	2995	2998*	3003*	3006	3012*					
		3037	3040*	3046*	3049	3056*													
R1Y	003044	588*	601*	619	622	625	628	635	640*										
R2	=%000002	62*	500	502	504*	505	506	615*	625	656*	750*	878*	1050*	1058*					
		1113*	1114*	1115*	1116*	1117*	1118	1121	1250*	1315*	1321*	1365*	1372*	1377*					
		1389*	1464	1475*	1479*	1482	1489*	1490*	1491	1496*	1504*	1581*	1593*	1770*					
		1771	2172	2193*	2524	2528*	2532*	2534*	2536*	2542	2543*								
R3	=%000003	63*	614*	635	657*	751*	1014*	1023	1465	1473*	1474*	1488*	1491*	1500*					
		1501*	1503*	1578*	1602*	1740*	1742*	1748*	1750*	1756*	1757*	1758	2173	2192*					
		2248	2257*	2263*	2264*	2267*	2272*	2273*	2274	2283*	2480	2481*	2482	2485*					
		2486	2490	2492	2494*	2496*													
R4	=%000004	64*	613*	658*	672	686	705	724	752*	767	772	786	791	1015*					
		1024	1249*	1250	1252	1257*	1258*	1259	1319*	1518*	1531*	1557	1577*	1585					
		1587	1589	1591*	1592	1601	1621*	1653*	1711*	1713	1716	2174	2191*	2249					
		2251*	2252*	2253*	2254	2255*	2269	2271*	2279*	2282*	2819	2844*	2849*						
R5	=%000005	65*	542*	544*	546*	548*	550*	552*	554*	556*	613*	619*	622*	625*					
		628*	630*	633*	635*	653*	662*	664*	669*	672*	675*	682*	686*	689*					
		692*	701*	705*	708*	711*	720*	724*	727*	730*	747*	756*	758*	763*					
		767*	770*	772*	775*	780*	782*	786*	789*	791*	794*	844*	846*	849*					

855*	875*	882*	884*	887*	891*	894*	897*	900*	904*	908*	913*	916*
998*	1005*	1010*	1012*	1014*	1020*	1023*	1094*	1098*	1101*	1104*	1107*	1110*
1113*	1121*	1125*	1129*	1132*	1372*	1383*	1394*	1466	1468*	1470*	1477*	1481*
1496	1502*	1626*	1629*	1634*	1638*	1644*	1648*	1653*	2175	2190*	2250	2256*
2258*	2260*	2261*	2262*	2263	2281*	2691*	2713*	2767*	2787*	2794*	2845*	2885*
2908*	2925*	2942*	3009*	3014*	3017*	3051*	3058*	3062*	3074	3076	3081*	3082*
3099*	3173*	3190*	3225*	3232*	3239*							
66#	427*	428*	429									
67#												
3125*												
68#	403*	408*	409*	418	419	431*	448*	456*	460	821*	941	1065*
1225*	1229*	1233*	1237*	1331*	1400*	1438*	1462*	1463*	1464*	1465*	1466*	1467*
1468	1471*	1484	1486*	1488	1498	1500	1502	1503	1504	1505	1506	1508*
1509*	1545	1549	2024*	2027	2029	2030	2059	2060	2064*	2090	2111*	2114*
2126*	2131*	2152	2156*	2170*	2171*	2172*	2173*	2174*	2175*	2176*	2177	2185*
2189	2190	2191	2192	2193	2194	2195	2200*	2240*	2241	2242	2243*	2248*
2249*	2250*	2256	2281	2282	2283	2284*	2285*	2313*	2314	2324*	2326	2327*
2328*	2330	2332	2334	2340	2342*	2344*	2352*	2356	2360	2361	2365	2385*
2386*	2387	2394*	2397*	2398*	2402*	2403*	2407	2410*	2414	2416	2418	2419*
2426	2428	2430*	2431	2433*	2434*	2435*	2436*	2437*	2452*	2453*	2456*	2457*
2458	2462*	2463*	2464	2467	2469	2471*	2480*	2485	2496	2497*	2498*	2499*
2520*	2521*	2522*	2523*	2524*	2526	2529*	2537*	2538	2540	2541*	2543	2544
2545	2556*	2557*	2564	2565*	2576	2577*	2578*	2588	2589*	2594	2595	2613*
2614	2624*	2625*	2666*	2688	2689	2698*	2786	2819*	2820*	2843	2844	2867*
2884	2899*	2941	2955*	2969	2994*	2995*	3012	3013	3036*	3037*	3056	3057
3169	3177*	3188										
1334*	1429*	1431*	1433*	1443*								
43#	403	431										
1956*												
54#												
240#	429	450*	452	458*	465*	861	865	925	983	986	1063	1071
1191	1193	1254	1327	2019	2033	2035	2041	2048	2086	2093	2105	2109
2176	2189*	2381	2418*									
175#	458	2381	2394									
107#												
97#	107											
96#	106											
95#	105											
94#	104											
93#	103											
92#	102											
91#	101											
90#	100											
89#	99											
88#	98											
106#												
87#	1063	1191	1327									
86#												
85#	861											
84#	1193											
83#												
82#												
105#												
104#												

R6 =%000006
R7 =%000007
SDELAY 020100
SP =%000006

SPACEX 007524
STACK = 001100
STCHAN 013114
STKMT= 177774
SWR 001140

SWREG 000176
SW0 = 000001
SW00 = 000001
SW01 = 000002
SW02 = 000004
SW03 = 000010
SW04 = 000020
SW05 = 000040
SW06 = 000100
SW07 = 000200
SW08 = 000400
SW09 = 001000
SW1 = 000002
SW10 = 002000
SW11 = 004000
SW12 = 010000
SW13 = 020000
SW14 = 040000
SW15 = 100000
SW2 = 000004
SW3 = 000010

.LPAIN	22#	2649	
.PUTCS	24#	3204	
.RESET	22#	3086	
.SETUP	30#	345	425
.SWRHI	30#	155	
.SWRLO	167#		
.UTK	29#	3146	
.SACT1	30#	180	
.SAPT8	30#	261#	
.SAPTH	30#	191	
.SAPTY	30#	2548	
.SCATC	30#	167	
.SCMTA	30#	213	
.SEO	30#	799	
.SERR0	30#	2067	
.SERRT	30#	2117	
.SINLP	28#	3021	
.SMMAC	18#		
.SOUTL	27#	2982	
.SPARM	30#		
.SPOWE	30#	2164	
.SRDOC	30#	2510	
.SREAD	30#	2371	
.SSAVE	30#		
.SSCOP	30#	2002	
.SSPAC	30#		
.SSWDO	30#		
.STLKW	26#	2860	
.STOUT	403#	3065	
.STRAP	30#	2605	
.STYPD	30#	1449	
.STYPE	30#	2292	
.STYPO	30#	2215	

ADC	1724	1732													
ADD	519	625	672	686	767	772	791	931	939	1114	1129	1310	1363	1388	1442
	1481	1597	1679	1684	1723	1788	1789	1791	1792	1794	1795	1797	1798	2139	2243
	2253	2328	2410	2419	2538	2565	2577	2589	3081	3101	3231				
ASL	494	938	956	1057	1115	1116	1117	1594	1595	1596	1777	2136	2137	2138	2433
	2434	2435	2531	2533	2535	2617									
ASLB	1486														
ASR	1044	1045	1047	1052	1053	1054	1728	2572							
BCC	1487														
BEQ	464	529	825	955	1064	1072	1086	1089	1173	1214	1260	1384	1554	1560	1586
	1689	1763	1772	2034	2036	2038	2042	2051	2084	2087	2110	2113	2141	2146	2159
	2270	2318	2331	2366	2390	2417	2432	2530	2559	2563	2583	2585	2707	2732	2753
	2781	2838	2958	2964	3000	3002	3042	3045	3075	3171					
BGE	2054														
BGT	816	1430	1495	1590	1600	1655	1721	2277	2429	2470					
BHI	2040														
BIC	813	937	984	1056	1186	1224	1258	1291	1296	1519	1526	1591	2267	2386	2403
	2430	2457	2463	2471	2537										
BICB	3167														
BIS	846	989	993	996	1187	1244	1292	1297	1489	1490	1693	2272	2273	2437	2694
	2824	2826	2834	2869	2901	3129									
BISB	1101	2128	3236	3237	3238										
BIT	861	927	1063	1191	1193	1327	2019	2033	2041	2048	2086	2093	2109	2706	2762
	2919	2936	2963												
BITB	463	2317	2322	2354	2562										
BLE	1583	1718													
BLOS	2483														
BLT	1478	1494	1588	2278	2345	2427	2468								
BMI	856	1175	1485	1528	1530	1534	1766	1768	3080						
BNE	416	430	453	507	514	522	527	558	561	564	567	570	636	677	696
	715	734	776	795	858	860	862	909	916	928	930	988	992	1037	1060
	1133	1192	1194	1196	1198	1201	1203	1207	1209	1212	1218	1241	1253	1312	1324
	1328	1373	1390	1396	1446	1483	1542	1548	1552	1562	1604	1623	1631	1682	1687
	1726	1731	1744	1752	1759	1775	2020	2049	2094	2099	2129	2151	2188	2268	2316
	2323	2325	2333	2341	2355	2362	2382	2388	2408	2415	2422	2459	2465	2487	2493
	2561	2567	2570	2587	2701	2704	2725	2727	2734	2736	2763	2779	2806	2830	2836
	2920	2937	2960	2966	3100	3132	3137								
BPL	693	712	731	866	905	1126	1255	1427	1469	1499	1524	1536	1649	1678	1695
	1804	2106	2266	2310	2359	2384	2400	2455	2461	2850	3127	3164			
BR	421	455	468	574	594	607	608	736	797	887	957	990	995	1051	1199
	1204	1210	1337	1403	1480	1497	1538	1558	1569	1598	1638	1685	1691	1779	1781
	1783	2022	2028	2031	2044	2047	2104	2134	2161	2180	2204	2244	2259	2280	2312
	2338	2348	2357	2364	2411	2438	2440	2466	2489	2539	2553	2575	2668	2684	2687
	2721	2738	2742	2746	2775	2802	2841	2853	2916	2933	2961	2967	2974	2979	3004
	3019	3047	3064	3078	3106	3160	3179	3184	3193						
CLR	417	420	423	428	441	442	462	471	490	491	503	650	810	811	851
	981	985	1028	1164	1165	1287	1358	1431	1472	1475	1520	1521	1522	1579	1590
	1581	1619	1656	1671	1712	1757	1769	2046	2061	2127	2186	2257	2397	2398	2527
	2528	2690	2699	2785	2822	2823	2827	2831	2839	2956	2998	3016	3040	3061	3077
	3130	3133	3157												
CLRB	1107	1501	2045	2337	2363	2494	2591	2592	2593						
CMP	415	429	452	506	513	521	635	908	941	987	1071	1174	1195	1197	1200
	1202	1206	1208	1211	1213	1245	1252	1259	1383	1493	1533	1589	1599	1688	1717
	1720	1758	1765	1767	1771	1774	2029	2053	2381	2387	2407	2414	2426	2428	2458

CMPB	2464	2467	2469	2482	2688	2752	2829	2835	2837	3001	3044	3074			
	557	560	563	566	569	991	1527	1529	1541	1547	1551	1559	1561	1585	1587
	2035	2039	2098	2315	2330	2332	2340	2361	2365	2389	2421	2486	2492	2560	2724
DEC	2726	2731	2778	2805	2957	3170									
	676	695	714	733	775	794	814	857	859	929	954	1036	1059	1087	1132
	1311	1323	1372	1389	1395	1429	1445	1555	1584	1630	1654	1725	1730	1743	1749
DECB	1751	1762	2135	3142											
EMT	2265	2276	2344	2347											
HALT	44														
INC	173	867	1256	1776	2107	2179	2203	2311	2851	2978					
	413	493	516	630	812	897	1033	1046	1261	1293	1298	1399	1479	1532	1676
	1681	1741	1773	1778	1780	1782	2052	2089	2187	2271	2279	2436	2590	2682	2700
INCB	2733	2735	2783	2828	2840	2959	2965	3003	3046	3079					
IOT	2057	2083	2367	2703	2754	2849	3136								
JMP	45														
	177	178	496	530	559	562	565	568	571	832	945	1073	1074	1262	1263
JSR	1546	1550	1609												
	525	536	542	544	546	548	550	552	554	556	591	592	593	604	605
	606	613	619	622	625	628	630	633	635	653	654	662	664	669	672
	675	682	686	689	692	701	705	708	711	720	724	727	730	735	747
	748	756	758	763	767	770	772	775	780	782	786	789	791	794	796
	827	844	846	849	855	875	882	884	887	891	894	897	900	904	908
	913	916	926	953	977	1005	1010	1012	1014	1020	1023	1029	1034	1042	1070
	1094	1098	1101	1104	1107	1110	1113	1121	1125	1129	1132	1168	1169	1179	1180
	1183	1188	1189	1190	1228	1232	1236	1251	1281	1282	1285	1294	1299	1320	1335
	1336	1372	1383	1394	1444	1608	1626	1629	1634	1638	1644	1648	1653	1680	1805
	2095	2101	2320	2339	2346	2353	2425	2579	2691	2713	2756	2758	2760	2767	2794
	2870	2874	2879	2883	2902	2905	2908	2922	2925	2939	3009	3017	3051	3062	3099
MOV	3173	3190	3225	3232											
	403	408	409	410	411	414	418	419	422	427	431	433	434	435	436
	437	438	439	440	444	445	448	449	450	451	456	458	459	460	465
	481	482	483	492	495	500	501	502	504	511	512	515	517	518	535
	539	576	587	588	589	590	600	601	602	603	610	613	614	615	616
	619	646	653	655	656	657	658	659	665	666	679	682	698	701	717
	720	741	742	743	744	747	749	750	751	752	760	763	782	817	821
	824	841	850	872	875	876	877	878	879	884	925	940	952	963	972
	978	982	986	994	998	1002	1007	1014	1015	1017	1023	1024	1025	1026	1027
	1031	1032	1040	1041	1043	1048	1049	1050	1055	1058	1062	1065	1091	1095	1113
	1118	1121	1149	1161	1170	1171	1176	1181	1182	1184	1223	1225	1229	1233	1237
	1249	1250	1257	1276	1283	1284	1286	1288	1289	1300	1301	1302	1303	1304	1305
	1306	1307	1308	1309	1314	1315	1316	1317	1318	1319	1321	1331	1334	1354	1355
	1360	1362	1364	1365	1366	1369	1375	1376	1377	1379	1380	1385	1391	1400	1428
	1437	1438	1443	1462	1463	1464	1465	1466	1467	1468	1473	1476	1496	1502	1503
	1504	1505	1506	1508	1509	1518	1525	1531	1556	1577	1578	1592	1593	1602	1620
	1621	1629	1641	1653	1672	1673	1674	1675	1690	1692	1696	1709	1710	1711	1713
	1714	1715	1716	1719	1722	1727	1738	1739	1740	1742	1746	1747	1748	1750	1756
	1760	1761	1764	1770	1787	1790	1793	1796	2024	2025	2027	2030	2043	2055	2056
	2059	2060	2063	2064	2085	2090	2111	2114	2126	2131	2140	2145	2150	2152	2156
	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2184	2185	2189	2190
	2191	2192	2193	2194	2195	2196	2197	2200	2240	2248	2249	2250	2255	2263	2281
	2282	2283	2284	2285	2313	2314	2319	2327	2342	2394	2418	2423	2452	2453	2480
	2481	2496	2497	2498	2499	2520	2521	2522	2523	2524	2526	2541	2542	2543	2544
	2545	2556	2557	2564	2568	2573	2574	2576	2578	2588	2594	2595	2613	2614	2618
	2624	2625	2666	2681	2689	2698	2702	2711	2751	2786	2819	2820	2821	2825	2832

MOV B	2843	2844	2867	2868	2871	2872	2875	2884	2899	2900	2906	2941	2955	2969	2994
	2995	2997	3006	3008	3012	3013	3015	3036	3037	3039	3049	3055	3056	3057	3060
	3076	3128	3169	3177	3188	3224	3230								
	443	983	1185	1290	1295	1471	1474	1488	1491	1500	1537	1626	1635	2058	2062
	2092	2100	2241	2242	2245	2246	2247	2251	2254	2255	2274	2324	2352	2360	2385
	2402	2456	2462	2485	2490	2529	2551	2552	2554	2616	2750	2755	2757	2759	2764
	2792	2873	2876	2878	2880	2881	2862	2903	2904	2921	2923	2938	2940	3165	3228
	3229	3235													
NEG	1470	2252													
NOP	424	581	828	829	830	849	900	942	943	944	997	999	1000	1001	1005
	1246	1361	1639	1640											
RESET	826	3096													
ROL	2258	2260	2261	2262	2264	2532	2534	2536							
ROR	1729														
RTI	457	481	1510	2065	2116	2202	2286	2329	2424	2472	2500	2546	2626	3143	
RTS	523	637	869	917	932	958	1134	1432	1447	1565	1605	1657	1683	1697	1733
	1799	1806	2154	2369	2596	2619	2787	2807	2845	2885	2942	2970	3014	3058	3082
	3108	3135	3138	3239											
SBC	486	489													
SEC	484	487													
SUB	705	724	786	891	913	1367	1368	1477	1677	2091	2571				
TRAP	2628	2637	2638	2639	2640	2642	2644	2645	2646	2647					
TST	485	505	520	526	528	865	1085	1088	1172	1217	1240	1254	1322	1387	1482
	1492	1545	1549	1553	1557	1582	1601	1603	1622	1686	2026	2050	2105	2112	2158
	2269	2326	2334	2356	2416	2431	2540	2566	2584	2586	2615	2848	2999	3007	3041
	3050	3099	3126	3131											
TSTB	488	692	711	730	855	904	1125	1426	1484	1498	1523	1535	1648	1694	1803
.ASCII	2037	2309	2358	2383	2399	2454	2460	2558	2569	2582	2780	3163			
.ASCIZ	257	258	1829	1836	1841	1846	1851	1856	1860	1862	1866	1867	1871	1876	1881
	256	259	470	835	1808	1817	1822	1856	1860	1862	1866	1867	1871	1876	1881
	1883	1890	1896	1899	1906	1909	1913	1923	1929	2162	2206	2504	2505	2506	2508
	2686	2740	2744	2748	2976	3162	3186	3197							
.ASECT	14														
.BLKB	2503														
.BLKW	401	1515													
.BYTE	222	223	228	229	237	238	246	247	248	249	275	276	286	287	294
	295	297	298	300	301	834	1067	1140	1227	1231	1235	1239	1333	1441	1863
	1864	2102	2103	2287	2288	2289	2290	2501	2502	2597	2598	2599			
.DSABL	2441														
.ENABL	30	2374													
.END	3261														
.ENDC	35	44	136	150	163	165	166	167	178	183	187	189	194	196	203
	216	220	222	250	254	255	256	257	261	264	286	294	297	300	303
	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318
	319	320	321	322	323	324	325	329	345	398	420	425	431	432	435
	437	439	441	442	444	446	467	470	584	585	586	587	588	597	598
	599	600	601	643	644	645	646	647	738	739	740	741	742	802	804
	805	807	810	816	819	820	824	826	832	834	835	838	960	961	962
	963	964	1146	1147	1148	1149	1150	1268	1269	1270	1271	1351	1352	1353	1354
	1355	1385	1452	2005	2008	2013	2019	2021	2032	2035	2036	2037	2039	2041	2048
	2052	2057	2059	2063	2066	2067	2070	2073	2083	2090	2095	2096	2097	2105	2116
	2117	2120	2135	2164	2167	2176	2177	2183	2189	2190	2200	2202	2206	2218	2295
	2324	2374	2375	2377	2405	2441	2445	2473	2474	2481	2483	2486	2488	2504	2510
	2513	2515	2548	2551	2552	2555	2582	2597	2608	2614	2617	2636	2637	2638	2639

	2640	2641	2642	2643	2644	2645	2646	2647	2648	2686	2716	2723	2740	2744	2748
	2640	2641	2642	2643	2644	2645	2646	2647	2648	2686	2716	2723	2740	2744	2748
.EQUIV	2770	2777	2797	2804	2911	2918	2928	2935	2976	3146	3162	3186			
	44	45	53	98	99	100	101	102	103	104	105	106	107	126	127
.EVEN	128	129	130	131	132	133	134	135							
.GLOBL	264	470	1142	1935	2163	2213	2600	2686	2740	2744	2748	2976	3162	3186	3200
.IF	14														
	31	42	108	136	162	164	165	166	167	176	182	185	187	193	195
	202	215	219	221	250	254	255	256	260	261	263	286	294	297	300
	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317
	318	319	320	321	322	323	324	325	329	345	395	420	425	426	431
	433	435	437	439	441	442	444	462	469	583	585	587	588	596	598
	600	601	642	644	646	647	737	739	741	742	801	802	803	804	805
	806	807	809	815	818	820	824	826	832	834	835	959	961	963	964
	1145	1147	1149	1150	1267	1269	1271	1352	1354	1355	1384	1384	1451	2004	2007
	2012	2018	2019	2031	2033	2034	2035	2037	2038	2039	2048	2050	2058	2060	2065
	2066	2067	2069	2072	2083	2086	2093	2095	2096	2098	2105	2109	2116	2117	2119
	2134	2150	2166	2176	2177	2182	2189	2190	2198	2200	2202	2206	2217	2294	2315
	2373	2375	2376	2377	2405	2444	2445	2473	2481	2482	2486	2487	2503	2504	2510
	2512	2515	2527	2550	2552	2555	2582	2597	2607	2613	2617	2628	2637	2638	2639
	2640	2641	2642	2644	2645	2646	2647	2648	2685	2715	2721	2739	2743	2747	2769
.IFF	2775	2796	2802	2910	2916	2927	2933	2975	3146	3161	3185				
	42	162	165	166	167	183	187	189	194	196	203	216	219	222	250
	261	264	431	584	585	586	587	588	597	598	599	600	601	643	644
	645	646	647	738	739	740	741	742	802	806	810	815	818	834	960
	961	962	963	964	1146	1147	1148	1149	1150	1268	1269	1270	1271	1351	1352
	1353	1354	1384	1452	2005	2032	2035	2036	2039	2066	2070	2072	2086	2116	2117
	2120	2135	2164	2167	2183	2198	2218	2295	2374	2377	2445	2447	2452	2473	2474
	2483	2487	2504	2513	2551	2608	2614	2715	2721	2769	2775	2796	2802	2910	2916
.IFT	2927	2933													
.IFTF	470	2047	2096	2447	2452	2531	2547	2548	2686	2740	2744	2748	2976	3162	3186
	470	2045	2095	2392	2445	2448	2527	2531	2547	2686	2740	2744	2748	2976	3162
.IIF	3186														
	30	35	40	159	160	161	163	166	167	173	260	264	432	435	441
	442	444	445	804	810	811	822	834	838	2008	2009	2010	2011	2012	2013
	2017	2046	2047	2063	2066	2067	2073	2074	2075	2076	2077	2082	2108	2116	2117
	2132	2157	2371	2374	2395	2496	2504	2510	2636	2637	2638	2639	2640	2642	2644
.IRP	2645	2646	2647	3178											
	345	425	583	596	642	737	959	1145	1267	1350	1462	1502	2018	2170	2176
.LIST	2189	2190	2522	2543	2556	2557	2578	2594	2595						
	14	30	150	166	173	250	252	253	254	261	264	345	425	446	470
	542	544	546	548	550	552	554	556	583	587	596	600	613	619	622
	625	628	630	633	635	642	646	653	662	664	669	672	675	682	686
	689	692	701	705	708	711	720	724	727	730	737	741	747	756	758
	763	767	770	772	775	780	782	786	789	791	794	810	826	844	846
	849	855	875	882	884	887	891	894	897	900	904	908	913	916	959
	963	1005	1010	1012	1014	1020	1023	1094	1098	1101	1104	1107	1110	1113	1121
	1125	1129	1132	1145	1149	1267	1271	1350	1354	1372	1383	1394	1626	1629	1634
	1638	1644	1648	1653	2012	2116	2473	2628	2636	2637	2638	2639	2640	2641	2642
.MACRO	2643	2644	2645	2646	2647	2648	2686	2740	2744	2748	2976	3099	3162	3186	403
	17	18	19	20	22	24	25	26	27	28	29	30	167	213	
	462	2628													
.MCALL	30	150	261	446											
.NLIST	14	30	150	166	173	250	252	253	254	261	264	345	425	446	470
	542	544	546	548	550	552	554	556	583	587	596	600	613	619	622

	625	628	630	633	635	642	646	653	662	664	669	672	675	682	686
	689	692	701	705	708	711	720	724	727	730	737	741	747	756	758
	763	767	770	772	775	780	782	786	789	791	794	810	826	844	846
	849	855	875	882	884	887	891	894	897	900	904	908	913	916	959
	963	1005	1010	1012	1014	1020	1023	1094	1098	1101	1104	1107	1110	1113	1121
	1125	1129	1132	1145	1149	1267	1271	1350	1354	1372	1383	1394	1626	1629	1634
	1638	1644	1648	1653	2012	2116	2473	2628	2636	2637	2638	2639	2640	2641	2642
	2643	2644	2645	2646	2647	2648	2686	2740	2744	2748	2976	3099	3162	3186	
.PAGE	213	329													
.PSECT	14														
.REM	1	14													
.REPT	173	252													
.SBTTL	40	155	167	176	180	191	213	261	329	425	583	596	642	737	799
	959	1145	1267	1350	1449	2002	2067	2117	2164	2215	2292	2371	2510	2548	2605
	2628														
.TITLE	30														
.WORD	173	174	175	188	207	208	209	210	211	212	221	224	225	226	227
	230	231	232	233	234	235	236	239	240	241	250	252	253	266	267
	268	269	270	271	272	273	277	278	279	292	296	299	302	303	304
	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319
	320	321	322	323	324	367	378	381	383	385	387	389	391	393	395
	396	398	400	542	544	546	548	550	552	554	556	613	619	622	625
	628	630	633	635	639	640	653	662	664	669	672	675	682	686	689
	692	701	705	708	711	720	724	727	730	747	756	758	763	767	770
	772	775	780	782	786	789	791	794	815	818	833	844	846	849	855
	875	882	884	887	891	894	897	900	904	908	913	916	1005	1010	1012
	1014	1020	1023	1094	1098	1101	1104	1107	1110	1113	1121	1125	1129	1132	1372
	1383	1394	1626	1629	1634	1638	1644	1648	1653	2143	2148	2199	2201	2291	2321
	2368	2547	2580	2635	2692	2789	2809	2944	3010	3011	3018	3053	3063	3084	3085
	3099	3109	3110	3140	3144	3174	3175	3180	3191	3192	3195	3196	3226	3227	3233
	3234	3240													

000000

ERRORS DETECTED: 0

*DRLPD,DRLPD/SOL/CRF=DRLPA.MAC,DRLPD
RUN-TIME: 25 13 2 SECONDS
CORE USED: 40K