

LSI-11

SYSTEM TEST
CWQAAB0

AH-8227B-MC

JUL 1978

COPYRIGHT © 76-78

digital

FICHE 1 OF 1

MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-8226B-MC
PRODUCT NAME: CWQAABO 11W03 SYSTEM TEST
PRODUCT DATE: 29-MARCH-1978
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1976, 1978 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

PROGRAM HISTORY

DWQAA-A	26-APR-76	JOHN EGOLF
CWQAABO	29-MAR-78	BARRY SUSSMAN

1. ABSTRACT

CWQAA IS A DIAGNOSTIC WHICH WAS WRITTEN TO TEST THE M5911 MODULE. THIS PROGRAM VERIFIES THE DATA WITHIN THE ROM, VERIFIES FUNCTIONS OF THE 600HZ CLOCK, AND CHECKS THE BUS BUFFER BY MEANS OF THE TEST BOX PROVIDED WITH THE 11W03-AA SYSTEM. THIS PROGRAM WAS WRITTEN EXPRESSLY FOR THE 11W03-AA SYSTEM AND TAKES ADVANTAGE OF ALL UNITS WITHIN THE SYSTEM. THIS PROGRAM MUST BE RUN AFTER ALL CPU TESTS, MEMORY TESTS, AND DLV11 TESTS.

2. REQUIREMENTS

2.1 EQUIPMENT

11W03-AA SYSTEM, WHICH CONSISTS OF:

- A. KD11-F LSI-11 PROCESSOR (WITH 4K OF MEMORY)
- B. 12K MOS RAM MEMORY
- C. 2 DLV11 SERIAL LINE UNITS
- D. SPECIAL MODULE
 - M5911 - BUS BUFFER
 - BOOTSTRAP ROM
 - 600HZ CLOCK
- E. H780 POWER SUPPLY
- F. ALUMINUM CASE 17 X 21 X 7.5"
- G. SPECIAL OPERATORS PANEL
- H. POWER CORD
- I. CABLE, EXTERNAL BUS -6FT.
- J. H9270 BACKPLANE ASSEMBLY
- K. TEST BOX

2.2 STORAGE

PROGRAM WILL LOAD IN 4K BUT RESERVES THE RIGHT TO USE 16K (ADDRESS 000000 - 077777).

3. LOADING PROCEDURE

THIS PROGRAM MAY BE LOADED LIKE ANY OTHER PROGRAM IS LOADED. THERE ARE NO SPECIAL LOADING PROCEDURES.

4. STARTING PROCEDURE

IF DESIRED, AFTER PROGRAM IS LOADED, SOFTWARE SWITCH REGISTER MAY BE SET AS PER SECTION 4.1 (USING LSI-ODT) BEFORE START OF EXECUTION OF PROGRAM. AS DISCUSSED IN SECTION 4.2, THERE ARE TWO STARTING ADDRESSES:

SA 200 NORMAL START

SA 210 NORMAL START BUT THE OPPORTUNITY TO "GOTO"
A SELECTED TEST IS GIVEN TO THE USER.

4.1 CONTROL SWITCH SETTINGS

THE SOFTWARE SWITCH REGISTER MAY BE ALTERED BY USING LSI-ODT AND MODIFYING ADDRESS 000176. AN EASIER WAY TO CHANGE THE SWITCH REGISTER IS THAT AFTER THE DIAGNOSTIC IS STARTED AND WHILE IT IS RUNNING THE USER MAY TYPE "CONTROL G" < G> AND IF THE CPU PRIORITY ALLOWS (AND A RESET INSTRUCTION ISN'T BEING EXECUTED) THE PROGRAM WILL PRINT OUT:

(SWR)=/000000/=/

THE USER MAY THEN HIT <CR> CARRIAGE RETURN OR TYPE (IN OCTAL) NEW SETTINGS. SWITCHES ARE:

SW15	SET: HALT ON ERROR
SW14	SET: LOOP ON TEST
SW13	SET: INHIBIT ERROR PRINT-OUT.
SW12	SET: ESCAPE TO NEXT TEST ON ERROR.
SW11	SET: INHIBIT ITERATIONS
SW10	SET: BELL ON ERROR
SW09	SET: LOOP ON ERROR
SW08	SET: CONFIDENCE; -PRINT TEST # AT 1ST END OF EACH TEST.
SW07	RESERVED
SW06	RESERVED
SW05	RESERVED
SW04	RESERVED
SW03	RESERVED
SW02	RESERVED
SW01	RESERVED
SW00	RESERVED

4.1.2 SWITCH REGISTER RESTRICTIONS

NONE. FOR THE EXCEPTION OF SW12 ALL SWITCH REGISTER OPTIONS ARE "SYSMAC.SML" COMPATABLE.

4.2 STARTING ADDRESS

AS MENTIONED BEFORE THERE ARE TWO STARTING ADDRESSES:

SA 200 NORMAL OPERATION

SA 210 NORMAL OPERATION WITH OPTION TO SPECIFY
STARTING TEST NUMBER.

5. OPERATING PROCEDURE

PROGRAM SHOULD BE STARTED AS PER SECTION 4.2. ON INITIAL START PROGRAM WILL PRINT OUT:

CWQAAB0 11W03 SYSTEM TEST

AND BEGIN EXECUTION (OR ASK FOR TEST NO. IF SA=210).

5.2 PROGRAM AND/OR OPERATOR ACTION

SWR SHOULD BE SET TO '100000' HALT ON ERROR. AFTER ERROR MODIFY SWR TO '060000' LOOP ON TEST AND INHIBIT PRINTOUT AND START AT ADDRESS 210 SPECIFYING FAILING TEST NUMBER.

6. ERRORS

ALL CONTROLABLE ERRORS WILL PRINT:

- | | |
|------------|--------------------------|
| 1) ERRPC | ERROR PC (IN OCTAL) |
| 2) \$TSTNM | TEST NUMBER (IN OCTAL) |
| 3) \$PASS | PASSES MADE (IN DECIMAL) |
| 4) \$ERTLL | ERRORS MADE (IN DECIMAL) |

ALL PROCESSOR REGISTERS ARE SAVED IN MEMORY LOCATIONS ON ERRORS.

SAVR0	LOC: 1422
SAVR1	LOC: 1424
SAVR2	LOC: 1426
SAVR3	LOC: 1430
SAVR4	LOC: 1432
SAVR5	LOC: 1434

ADDITIONAL INFORMATION WILL BE PRINTED AS NEEDED.

7. RESTRICTIONS

7.1 HARDWARE RESTRICTIONS

SYSTEM MUST BE CONFIGURED *EXACTLY* AS OUTLINED FOR
11W03-AA SYSTEM.

8. MISCELLANEOUS

8.1 CONTROL "G" < G>

THIS WILL ENABLE USER TO CHANGE SWITCH REGISTER (PROVIDED
PROGRAM IS RUNNING).

8.2 CONTROL "T" < T>

THIS WILL ENABLE USER TO "GOTO" SELECTED TEST. (CPU MUST
NOT BE DOING A RESET.)

8.3 PASS COMPLETE

THIS IS A "SYSMAC.SML" COMPATIBLE END PASS ROUTINE. AT
PASS COMPLETE THE FOLLOWING WILL BE PRINTED:

END PASS # 1
END PASS # 2

PASS COUNT IS IN DECIMAL.

8.4 EXECUTION TIME

WITH NO ERRORS AND SW11=0 PASS TIME IS <8 MINS.

WITH NO ERRORS AND SW11=1 PASS TIME IS <2 MINS.

```

1
2      .ENABLE ABS,AMA
3      .TITLE  CWQAABO 11W03 SYSTEM TEST
4      ;*COPYRIGHT (C) 1978
5      ;*DIGITAL EQUIPMENT CORP.
6      ;*MAYNARD, MASS. 01754
7      ;*
8      ;*PROGRAM BY JOHN C. EGOLF
9      ;*
10     ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
11     ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
12     ;*
13
14     .SBTTL  OPERATIONAL SWITCH SETTINGS
15     ;*
16     ;*      SWITCH                      USE
17     ;*      -----
18     ;*      15          HALT ON ERROR
19     ;*      14          LOOP ON TEST
20     ;*      13          INHIBIT ERROR TYPEOUTS
21     ;*      12          SWITCH 12: ESCAPE TO NEXT TEST ON ERROR.
22     ;*      11          INHIBIT ITERATIONS
23     ;*      10          BELL ON ERROR
24     ;*      9           LOOP ON ERROR
25     ;*      8           CONFIDENCE: -PRINT TEST # AT 1ST END OF EACH TEST
26     .SBTTL  BASIC DEFINITIONS
27
28     ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
29     001100  STACK= 1100
30     .EQUIV  EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
31     .EQUIV  IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
32
33     ;*MISCELLANEOUS DEFINITIONS
34     000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
35     000012  LF= 12          ;;CODE FOR LINE FEED
36     000015  CR= 15          ;;CODE FOR CARRIAGE RETURN
37     000200  CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
38     177776  PS= 177776     ;;PROCESSOR STATUS WORD
39     .EQUIV  PS,PSW
40     177774  STKLMT= 177774  ;;STACK LIMIT REGISTER
41     177772  PIRQ= 177772    ;;PROGRAM INTERRUPT REQUEST REGISTER
42     177570  DSWR= 177570    ;;HARDWARE SWITCH REGISTER
43     177570  DDISP= 177570   ;;HARDWARE DISPLAY REGISTER
44
45     ;*GENERAL PURPOSE REGISTER DEFINITIONS
46     000000  R0= X0          ;;GENERAL REGISTER
47     000001  R1= X1          ;;GENERAL REGISTER
48     000002  R2= X2          ;;GENERAL REGISTER
49     000003  R3= X3          ;;GENERAL REGISTER
50     000004  R4= X4          ;;GENERAL REGISTER
51     000005  R5= X5          ;;GENERAL REGISTER
52     000006  R6= X6          ;;GENERAL REGISTER
53     000007  R7= X7          ;;GENERAL REGISTER
54     000006  SP= X6          ;;STACK POINTER
55     000007  PC= X7          ;;PROGRAM COUNTER
56

```

```
57      ;*PRIORITY LEVEL DEFINITIONS
58      000000 PR0= 0 ;;PRIORITY LEVEL 0
59      000040 PR1= 40 ;;PRIORITY LEVEL 1
60      000100 PR2= 100 ;;PRIORITY LEVEL 2
61      000140 PR3= 140 ;;PRIORITY LEVEL 3
62      000200 PR4= 200 ;;PRIORITY LEVEL 4
63      000240 PR5= 240 ;;PRIORITY LEVEL 5
64      000300 PR6= 300 ;;PRIORITY LEVEL 6
65      000340 PR7= 340 ;;PRIORITY LEVEL 7
66
67      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
68      100000 SW15= 100000
69      040000 SW14= 40000
70      020000 SW13= 20000
71      010000 SW12= 10000
72      004000 SW11= 4000
73      002000 SW10= 2000
74      001000 SW09= 1000
75      000400 SW08= 400
76      000200 SW07= 200
77      000100 SW06= 100
78      000040 SW05= 40
79      000020 SW04= 20
80      000010 SW03= 10
81      000004 SW02= 4
82      000002 SW01= 2
83      000001 SW00= 1
84      .EQUIV SW09,SW9
85      .EQUIV SW08,SW8
86      .EQUIV SW07,SW7
87      .EQUIV SW06,SW6
88      .EQUIV SW05,SW5
89      .EQUIV SW04,SW4
90      .EQUIV SW03,SW3
91      .EQUIV SW02,SW2
92      .EQUIV SW01,SW1
93      .EQUIV SW00,SW0
94
95      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
96      100000 BIT15= 100000
97      040000 BIT14= 40000
98      020000 BIT13= 20000
99      010000 BIT12= 10000
100     004000 BIT11= 4000
101     002000 BIT10= 2000
102     001000 BIT09= 1000
103     000400 BIT08= 400
104     000200 BIT07= 200
105     000100 BIT06= 100
106     000040 BIT05= 40
107     000020 BIT04= 20
108     000010 BIT03= 10
109     000004 BIT02= 4
110     000002 BIT01= 2
111     000001 BIT00= 1
112     .EQUIV BIT09,BIT9
```

```

113      .EQUIV BIT08,BIT8
114      .EQUIV BIT07,BIT7
115      .EQUIV BIT06,BIT6
116      .EQUIV BIT05,BIT5
117      .EQUIV BIT04,BIT4
118      .EQUIV BIT03,BIT3
119      .EQUIV BIT02,BIT2
120      .EQUIV BIT01,BIT1
121      .EQUIV BIT00,BIT0
122
123      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
124      000004  ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
125      000010  RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
126      000014  TBITVEC=14        ;; "T" BIT
127      000014  TRTVEC= 14        ;;TRACE TRAP
128      000014  BPTVEC= 14        ;;BREAKPOINT TRAP (BPT)
129      000020  IOTVEC= 20        ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
130      000024  PWRVEC= 24        ;;POWER FAIL
131      000030  EMTVEC= 30        ;;EMULATOR TRAP (EMT) **ERROR**
132      000034  TRAPVEC=34        ;; "TRAP" TRAP
133      000060  TKVEC= 60         ;;TTY KEYBOARD VECTOR
134      000064  TFVEC= 64         ;;TTY PRINTER VECTOR
135      000240  PIRQVEC=240       ;;PROGRAM INTERRUPT REQUEST VECTOR
136      .SBTTL TRAP CATCHER
137
138      000000      .=0
139      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
140      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
141      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
142      000174      .=174
143      000174 000000  DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
144      000176 000000  SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
145      .SBTTL STARTING ADDRESS(ES)
146      000200 000137 003734  JMP      @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
147      000210 000210
148      000210 000137 003730  JMP      SETTST      ;ASK FOR TEST NUMBER.

```

```

149      .SBTTL COMMON TAGS
150
151      ;*****
152      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
153      ;*USED IN THE PROGRAM.
154
155      001100      .S1100
156      001100      SCMTAG:      ;; START OF COMMON TAGS
157      001100      SPASS: .WORD 0      ;; CONTAINS PASS COUNT
158      001102      STSTNM: .BYTE 0      ;; CONTAINS THE TEST NUMBER
159      001103      SERFLG: .BYTE 0      ;; CONTAINS ERROR FLAG
160      001104      SICNT: .WORD 0      ;; CONTAINS SUBTEST ITERATION COUNT
161      001106      SLPADR: .WORD 0      ;; CONTAINS SCOPE LOOP ADDRESS
162      001110      SLPERR: .WORD 0      ;; CONTAINS SCOPE RETURN FOR ERRORS
163      001112      SERTTL: .WORD 0      ;; CONTAINS TOTAL ERRORS DETECTED
164      001114      SITEMB: .BYTE 0      ;; CONTAINS ITEM CONTROL BYTE
165      001115      SERMAX: .BYTE 1      ;; CONTAINS MAX. ERRORS PER TEST
166      001116      SERRPC: .WORD 0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
167      001120      SGDADR: .WORD 0      ;; CONTAINS ADDRESS OF 'GOOD' DATA
168      001122      SBDADR: .WORD 0      ;; CONTAINS ADDRESS OF 'BAD' DATA
169      001124      SGDDAT: .WORD 0      ;; CONTAINS 'GOOD' DATA
170      001126      SBDDAT: .WORD 0      ;; CONTAINS 'BAD' DATA
171      001130      .WORD 0      ;; RESERVED--NOT TO BE USED
172      001132      .WORD 0
173      001134      $AUTOB: .BYTE 0      ;; AUTOMATIC MODE INDICATOR
174      001135      $INTAG: .BYTE 0      ;; INTERRUPT MODE INDICATOR
175      001136      .WORD 0
176      001140      SWR: .WORD DSWR      ;; ADDRESS OF SWITCH REGISTER
177      001142      DISPLAY: .WORD DDISP      ;; ADDRESS OF DISPLAY REGISTER
178      001144      $TKS: 177560      ;; TTY KBD STATUS
179      001146      $TKB: 177562      ;; TTY KBD BUFFER
180      001150      $TPS: 177564      ;; TTY PRINTER STATUS REG. ADDRESS
181      001152      $TPB: 177566      ;; TTY PRINTER BUFFER REG. ADDRESS
182      001154      $NULL: .BYTE 0      ;; CONTAINS NULL CHARACTER FOR FILLS
183      001155      $FILLS: .BYTE 2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
184      001156      $FILLC: .BYTE 12      ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
185      001157      $TPFLG: .BYTE 0      ;; 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
186      001160      $TMP0: .WORD 0      ;; USER DEFINED
187      001162      $TIMES: 0      ;; MAX. NUMBER OF ITERATIONS
188      001164      $ESCAPE: 0      ;; ESCAPE ON ERROR ADDRESS
189      001166      $BELL: .ASCIZ <207><377><377>      ;; CODE FOR BELL
190      001172      $QUES: .ASCII /?/      ;; QUESTION MARK
191      001173      $CRLF: .ASCII <15>      ;; CARRIAGE RETURN
192      001174      $LF: .ASCIZ <12>      ;; LINE FEED
193      ;*****

```

000377

```

194      .SBTTL  ERROR POINTER TABLE
195
196      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
197      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
198      ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
199      ;*NOTE1:      IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
200      ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
201
202      ;*      EM      ;;POINTS TO THE ERROR MESSAGE
203      ;*      DH      ;;POINTS TO THE DATA HEADER
204      ;*      DT      ;;POINTS TO THE DATA
205      ;*      DF      ;;POINTS TO THE DATA FORMAT
206
207
208      001176      $ERRTB:
209      ;ERROR 1
210      001176      001444      EM1      ;NO 'BREPLY' FROM ROM ADDRESS.
211      001200      002522      DH1      ;ERRPC  TSTNO  PASCNT  ERRCNT  ROM ADDRESS
212      001202      003652      DT1      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO
213      001204      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL
214
215      ;ERROR 2
216      001206      001502      EM2      ;ROM DATA COMPARISON ERROR.
217      001210      002576      DH2      ;ERRPC  TSTNO  PASCNT  ERRCNT  ROMADD  WANTED  FOUND
218      001212      003666      DT2      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO  SAVR5  SAVR4
219      001214      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL  OCTAL  OCTAL
220
221      ;ERROR 3
222      001216      001535      EM3      ;NO 'BREPLY' FROM 600 HZ CLOCK.
223      001220      002664      DH3      ;ERRPC  TSTNO  PASCNT  ERRCNT  CLKCSR
224      001222      003652      DT1      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO
225      001224      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL
226
227      ;ERROR 4
228      001226      001574      EM4      ;CLOCK R/W BIT6 FAILED.
229      001230      002733      DH4      ;ERRPC  TSTNO  PASCNT  ERRCNT  CLKCSR  WANTED  FOUND
230      001232      003666      DT2      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO  SAVR5  SAVR4
231      001234      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL  OCTAL  OCTAL
232
233      ;ERROR 5
234      001236      001626      EM5      ;UNEXPECTED INTERRUPT.
235      001240      003021      DH5      ;ERRPC  TSTNO  PASCNT  ERRCNT  (SP)-TRAP PC
236      001242      003652      DT1      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO
237      001244      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL
238
239      ;ERROR 6
240      001246      001653      EM6      ;NO INTERRUPT FROM CLOCK.
241      001250      002664      DH3      ;ERRPC  TSTNO  PASCNT  ERRCNT  CLKCSR
242      001252      003652      DT1      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO
243      001254      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL
244
245      ;ERPOR 7
246      001256      001703      EM7      ;CLOCK INTERRUPTS NOT CONSISTANT.
247      001260      002664      DH3      ;ERRPC  TSTNO  PASCNT  ERRCNT  CLKCSR
248      001262      003652      DT1      ;$ERRPC MSKTST $PASS  $ERTTL  SAVRO
249      001264      003720      DF1      ;OCTAL  OCTAL  DEC    DEC    OCTAL

```

[illegible]

304	001376	100000			MEMSIZ:	100000		: 16K	100000	
305								:	12K	060000
306								:	8K	040000
307								:	4K	020000
308	001400	177546			CLKCSR:	177546		;CLOCK CONTROL REGISTER		
309	001402	000100			CLKVEC:	100				
310	001404	000102			CLKPTY:	102				
311	001406	172524			DEV.A:	172524				
312	001410	165250			DEV.B:	165250				
313	001412	000340			B.AVEC:	340				
314	001414	000342			B.APTY:	342				
315	001416	000344			B.BVEC:	344				
316	001420	000346			B.BPTY:	346				
317	001422	000000			SAVR0:	0				
318	001424	000000			SAVR1:	0				
319	001426	000000			SAVR2:	0				
320	001430	000000			SAVR3:	0				
321	001432	000000			SAVR4:	0				
322	001434	000000			SAVR5:	0				
323	001436	000000			MSKTST:	0				
324	001440	000000			TSTFLG:	0				
325	001442	000000			XXSCOP:	0				
326										
327	001444	047516	021040	051102	EM1:	.ASCIZ	/NO "BREPLY" FROM ROM ADDRESS./			
	001502	047522	020115	040504	EM2:	.ASCIZ	/ROM DATA COMPARISON ERROR./			
	001535	116	020117	041042	EM3:	.ASCIZ	/NO "BREPLY" FROM 600 HZ CLOCK./			
	001574	046103	041517	020113	EM4:	.ASCIZ	'CLOCK R/W OF BIT6 FAILED.'			
	001626	047125	054105	042520	EM5:	.ASCIZ	/UNEXPECTED INTERRUPT./			
	001653	116	020117	047111	EM6:	.ASCIZ	/NO INTERRUPT FROM CLOCK./			
	001703	103	047514	045503	EM7:	.ASCIZ	/CLOCK INTERRUPTS NOT CONSISTANT./			
	001743	104	053105	041511	EM10:	.ASCIZ	'DEVICE "A" R/W FAILURE.'			
	001773	116	020117	041042	EM11:	.ASCIZ	/NO "BREPLY" FROM DEVICE 'A'./			
	002030	047516	021040	051102	EM12:	.ASCIZ	/NO "BREPLY" FROM DEVICE 'B'./			
	002065	104	053105	041511	EM13:	.ASCIZ	'DEVICE "B" R/W FAILURE.'			
	002115	125	042516	050130	EM14:	.ASCIZ	/UNEXPECTED "BREPLY" FROM UNKNOWN DEVICE./			
	002166	047516	044440	052116	EM15:	.ASCIZ	/NO INTERRUPT FROM DEVICE 'B'./			
	002222	042507	042516	040522	EM16:	.ASCIZ	/GENERAL ERROR!/'			
	002241	104	053114	030461	EM17:	.ASCIZ	/DLV11 DATA ERROR./			
	002263	115	046505	051117	EM20:	.ASCIZ	/MEMORY ERROR./			
	002301	015	044412	041516	EM21:	.ASCII	<15><12>/INCORRECT MEMORY SIZE FOUND./			
	002337	015	041412	040510		.ASCII	<15><12>/CHANGE LOCATION "MEMSIZ" TO SPECIFIC/			
	002405	015	051412	055111		.ASCII	<15><12>/SIZE IN YOUR CONFIGURATION./			
	002442	005015	046442	046505		.ASCII	<15><12>/"MEMSIZ" SHOULD BE 100000 FOR A 16K SYSTEM./<15><12><0>			
	002522	051105	050122	020103	DH1:	.ASCIZ	/ERRPC TSTNO PASCNT ERRCNT ROM ADDRESS/			
	002576	051105	050122	020103	DH2:	.ASCIZ	/ERRPC TSTNO PASCNT ERRCNT ROMADD WANTED FOUND/			
	002664	051105	050122	020103	DH3:	.ASCIZ	/ERRPC TSTNO PASCNT ERRCNT CLKCSR/			
	002733	105	051122	041520	DH4:	.ASCIZ	/ERRPC TSTNO PASCNT ERPCNT CLKCSR WANTED FOUND/			
	003021	105	051122	041520	DH5:	.ASCIZ	/ERRPC TSTNO PASCNT ERRCNT (SP)-TRAP PC/			
	003076	051105	050122	020103						


```

328 003730 005237 001440      SETTST: INC      TSTFLG      ;
329 003734                      START:
330                      .SBTTL  INITIALIZE THE COMMON TAGS
                      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
331 003734 012706 001100      MOV      #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
332 003740 005026              CLR      (R6)+          ;;CLEAR MEMORY LOCATION
333 003742 022706 001140      CMP      #SWR,R6      ;;DONE?
334 003746 001374              BNE      .-6            ;;LOOP BACK IF NO
335 003750 012706 001100      MOV      #STACK,SP      ;;SETUP THE STACK POINTER
336                      ;;INITIALIZE A FEW VECTORS
337 003754 012737 011232 000020  MOV      #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
338 003762 012737 000340 000022  MOV      #340,@IOTVEC+2 ;;LEVEL 7
339 003770 012737 011544 000030  MOV      #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
340 003776 012737 000340 000032  MOV      #340,@EMTVEC+2 ;;LEVEL 7
341 004004 012737 013542 000034  MOV      #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
342 004012 012737 000340 000036  MOV      #340,@TRAPVEC+2;LEVEL 7
343 004020 013737 011144 011136  MOV      $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
344 004026 005037 001162              CLR      $TIMES      ;;INITIALIZE NUMBER OF ITERATIONS
345 004032 005037 001164              CLR      $ESCAPE      ;;CLEAR THE ESCAPE ON ERROR ADDRESS
346 004036 112737 000001 001115  MOVB     #1,$ERMAX      ;;ALLOW ONE ERROR PER TEST
347 004044 012737 004044 001106  MOV      #.,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
348 004052 012737 004052 001110  MOV      #.,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
349                      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
350                      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
351 004060 013746 000004              MOV      @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
352 004064 012737 004120 000004  MOV      #64,$@ERRVEC      ;;SET UP ERROR VECTOR
353 004072 012737 177570 001140  MOV      #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
354 004100 012737 177570 001142  MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
355 004106 022777 177777 175024  CMP      #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
356 004114 001012              BNE      66$            ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
357                      ;;AND THE HARDWARE SWR IS NOT = -1
358 004116 000403              BR       65$            ;;BRANCH IF NO TIMEOUT
359 004120 012716 004126      64$:  MOV      #65$,(SP)      ;;SET UP FOR TRAP RETURN
360 004124 000002              RTI
361 004126 012737 000176 001140  65$:  MOV      #SWREG,SWR      ;;POINT TO SOFTWARE SWR
362 004134 012737 000174 001142  MOV      #DISPREG,DISPLAY
363 004142 012637 000004      66$:  MOV      (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
364
365                      .SBTTL  TYPE PROGRAM NAME
366                      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
367 004146 005227 177777      INC      #-1            ;;FIRST TIME?
368 004152 001027              BNE      67$            ;;BRANCH IF NO
369 004154 022737 011176 000042  CMP      #$ENDAD,@#42      ;;ACT-11?
370 004162 001423              BEQ      67$            ;;BRANCH IF YES
371 004164 104401 004172      TYPE     ,68$            ;;TYPE ASCIZ STRING
372 004170 000420              BR       67$            ;;GET OVER THE ASCIZ
373                      ;;68$: .ASCIZ <CRLF><15><12>/CWQAABO 11W03 SYSTEM TEST/<15><12><CRLF>
374 004232                      67$:
375 004232 012737 000006 000004  MOV      #6,4            ;
376 004240 005037 000006              CLR      6            ;
377 004244 013777 001404 175130  MOV      CLKPTY,@CLKVEC      ;
378 004252 012777 000002 175124  MOV      #2,@CLKPTY      ;
379 004260 013777 001414 175124  MOV      B.APTY,@B.AVEC      ;
380 004266 005077 175122              CLR      @B.APTY      ;
381 004272 013777 001420 175116  MOV      B.BPTY,@B.BVEC      ;
382 004300 005077 175114              CLR      @B.BPTY      ;

```

```

383 004304 005037 000022      CLR      @#IOTVEC+2      ;ZERO SCOPE PRIO.
384 004310 005037 001442      CLR      XXSCOP      ;
385 004314 012737 014620 000060  MOV      #KBISR,@#60      ;SET VECTOR
386 004322 012737 000200 000062  MOV      #200,@#62      ;SET PRIO
387 004330 012777 000100 174606  MOV      #100,@$TKS      ;SET INTR ENABLE
388 004336 106427 000000      MTPS     #0      ;SET PSW TO 0
389 004342 005737 001440      TST      TSTFLG      ;FLAG SET?
390 004346 001431      BEQ      TST1      ;
391 004350 005037 001440      CLR      TSTFLG      ;ZERO FLAG
392 004354 104401 014733      X1$:    TYPE      ,XTSTN      ;TYPE 'TEST NO: '
393 004360 104410      RDOCT      ;GET NUMBER
394 004362 012600      MOV      (SP)+,R0      ;
395 004364 001773      BEQ      X1$      ;BR IF NO NUMBER INPUT
396 004366 020037 015242      CMP      R0,LASTN      ;VALID TEST?
397 004372 101370      BHI      X1$      ;BR IF NO
398 004374 110037 001102      MOVB     R0,$TSTNM      ;LOAD INITAL TEST NO.
399 004400 000241      CLC      ;
400 004402 006100      ROL      R0      ;MAKE POWER OF 2.
401 004404 016037 015114 001106  MOV      TEST.TABLE(R0),$LPADR
402      ;GET TEST PC.
403 004412 062737 000002 001106  ADD      #2,$LPADR      ;GET PAST SCOPE
404 004420 013737 001106 001110  MOV      $LPADR,$LPERR      ;
405 004426 000177 174454      JMP      @SLPADR      ;GOTO TEST.
406
407      ;*****
408      ;*TEST 1      ROM 'BREPLY' TEST
409      ;*TEST TO REFERENCE ALL 256. ADDRESS
410      ;*OF THE ROM ONLY MAKING SURE THAT
411      ;*THERE IS A 'BREPLY' RESPONSE - NO DATA IS
412      ;*CHECKED IN THIS TEST.
413      ;*****
414 004432 000004      TST1:    SCOPE
415 004434 012700 173000      MOV      #173000,R0      ;SET INITAL START ADDRESS OF ROM.
416 004440 012701 000400      MOV      #256.,R1      ;SET NUMBER OF WORDS TO BE TESTED.
417 004444 013746 000004      MOV      4,-(SP)      ;SAVE LOC 4.
418 004450 013746 000006      MOV      6,-(SP)      ;SAVE LOC 6.
419 004454 012737 004506 000004  MOV      #3$,4      ;SET TIME-OUT TRAP VECTOR.
420 004462 012737 000200 000006  MOV      #200,6      ;LOAD PRIORITY = 4
421 004470 005710      1$:      TST      (R0)      ;REFERENCE ROM ADDRESS.
422 004472 005710      TST      (R0)      ;DO IT AGAIN.
423 004474 062700 000002      2$:      ADD      #2,R0      ;UPDATE ADDRESS.
424 004500 005301      DEC      R1      ;ALL DONE?
425 004502 001372      BNE      1$      ;BR IF NO
426 004504 000404      BR      4$      ;CONTINUE TEST
427 004506 104001      3$:      ERROR     1      ;ROM DID NOT ISSUE 'BREPLY'.
428 004510 012716 004474      MOV      #2$,(SP)      ;SET RETURN ADDRESS
429 004514 000002      RTI      ;RETURN
430 004516 012637 000006      4$:      MOV      (SP)+,6      ;RESTORE ADD 6.
431 004522 012637 000004      MOV      (SP)+,4      ;RESTORE ADD 4.
432
433
434
435      ;*****
436      ;*TEST 2      ROM WRITE TEST
437      ;*TEST THAT WRITING THE ROM
438      ;*PRODUCES A TIME-OUT TRAP.

```

```

439
440
441 004526 000004
442 004530 012700 173000
443 004534 012701 000400
444 004540 013746 000004
445 004544 013746 000006
446 004550 012737 004574 000004
447 004556 012737 000200 000006
448 004564 005020
449 004566 000240
450 004570 104016
451 004572 024646
452 004574 105301
453 004576 001403
454 004600 012716 004564
455 004604 000002
456 004606 106427 000000
457 004612 022626
458 004614 012637 000006
459 004620 012637 000004
460
461
462
463
464
465
466
467
468 004624 000004
469 004626 012700 173000
470 004632 012701 000400
471 004636 012702 013620
472 004642 011004
473 004644 011205
474 004646 020504
475 004650 001401
476 004652 104002
477 004654 022022
478 004656 005301
479 004660 001370
480
481
482
483
484
485
486
487
488
489 004662 000004
490 004664 013746 000004
491 004670 013746 000006
492 004674 012701 000017
493 004700 013700 001400
494 004704 012737 004730 000004

; *
; *****
TST2:  SCOPE
      MOV    #173000,R0      ;SET ROM ADDRESS
      MOV    #256.,R1       ;DO ALL ADDRESS
      MOV    4,-(SP)        ;STORE 4
      MOV    6,-(SP)        ;STORE 6
      MOV    #2$,4          ;SET TRAP VECTOR
      MOV    #200,6         ;PRIO.
1$:    CLR    (R0)+          ;WRITE ROM WITH ZERO.
      NOP
      ERROR   16            ;WRITE ROM AND NO TRAP.
      CMP    -(SP),-(SP)    ;FAKE AN INTERRUPT.
2$:    DECB   R1            ;256 DONE?
      BEQ    3$            ;BR IF YES
      MOV    #1$,(SP)      ;SET RETURN
      RTI
3$:    MTPS   #0            ;SET PTY TO 0
      CMP    (SP)+,(SP)+    ;FAKE RTI
      MOV    (SP)+,6        ;RESTORE 6
      MOV    (SP)+,4        ;RESTORE 4

; *****
; *TEST 3      ROM DATA TEST
; *TEST TO READ AND COMPARE ALL
; *256 ROM ADDRESS TO THE ROM DATA
; *MAP IN MEMORY.  ALL ADDRESSES ARE
; *VERIFIED FOR GOOD DATA.
; *****
TST3:  SCOPE
      MOV    #173000,R0      ;SET START OF ROM ADDRESS.
      MOV    #256.,R1       ;SET NUMBER OF WORDS TO CHECK.
      MOV    #ROMMAP,R2     ;GET SOFTWARE ADDRESS
1$:    MOV    (R0),R4        ;READ ROM.
      MOV    (R2),R5        ;READ SOFTWARE IMAGE
      CMP    R5,R4          ;ARE THEY GOOD?
      BEQ    .+4            ;ROM DATA ERROR?
      ERROR   2            ;BAD DATA IN ROMS!!
      CMP    (R0)+,(R2)+    ;POP POINTERS
      DEC    R1            ;ALL DONE?
      BNE    1$            ;BR IF NOT DONE.

; *****
; *TEST 4      600 HZ ADDRESS TEST
; *TEST TO VERIFY A "BREPLY" RESPONSE
; *FROM THE 600 HZ CLOCK.
; *IT IS ASSUMED THAT THE CLOCK
; *IS AT ADDRESS "177546".
; *****
TST4:  SCOPE
      MOV    4,-(SP)        ;SAVE ADDRESS 4.
      MOV    6,-(SP)        ;SAVE ADDRESS 6.
      MOV    #15.,R1        ;SET INTERNAL ICOUNT TO 15.
      MOV    CLKCSR,R0      ;GET CLOCK CSR.
      MOV    #3$,4          ;SET TIME-OUT VECTOR.

```

```

495 004712 012737 000200 000006      MOV      #200,6      ;SET PRIO.
496 004720 005710      1$:      TST      (R0)      ;REFERENCE CLOCK CSR.
497 004722 005301      2$:      DEC      R1      ;ICOUNT = 0?
498 004724 001375      BNE      1$      ;BR IF NO.
499 004726 000404      BR      4$      ;CONT TEST
500 004730 104003      3$:      ERROR    3      ;TIME-OUT ERROR.
501 004732 012716 004722      MOV      #2$, (SP)      ;SET RETURN ADD.
502 004736 000002      RTI      ;RETURN.
503 004740 012637 000006      4$:      MOV      (SP)+,6      ;RESTORE 6
504 004744 012637 000004      MOV      (SP)+,4      ;RESTORE 4
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520

```

```

*****
;TEST 5      600 HZ 'EVENT ENABLE' R/W TEST
;TEST THAT BIT6 OF 600HZ
;CLOCK IS R/W.
;SET PSW TO '4' AND SET BIT6
;OF 600HZ CLOCK VERIFYING THAT NO
;INTERRUPT OCCURED AND THAT BIT6
;DID INDEED SET. THEN CLEAR BIT6
;AND VERIFY BIT6 IS CLEARED.
;VERIFY NO INTERRUPT OCCURED.
;THEN CLEAR PS VERIFYING THAT
;THE INTERRUPT PENDING OCCURES.
;
*****

```

```

521 004750 000004      TST5:      SCOPE
522 004752 012737 000005 001162      MOV      #5,$TIMES      ;;DO 5 ITERATIONS
523 004760 106427 000200      MTPS      #200      ;SET PRIO TO '4'
524 004764 013700 001400      MOV      CLKCSR,R0      ;SET CLOCK CSR
525 004770 017746 174406      MOV      @CLKVEC,-(SP)      ;SAVE CLKVEC
526 004774 017746 174404      MOV      @CLKPTY,-(SP)      ;SAVE CLKPTY
527 005000 012777 005074 174374      MOV      #3$,@CLKVEC      ;SET CLOCK VECTOR
528 005006 012777 000200 174370      MOV      #200,@CLKPTY      ;SET CLOCK PRIORITY
529 005014 012701 000017      MOV      #15.,R1      ;SET ICOUNT
530 005020 012705 000100      1$:      MOV      #BIT6,R5      ;SET EXPECTED
531 005024 010510      MOV      R5,(R0)      ;WRITE BIT6
532 005026 011004      MOV      (R0),R4      ;READ CSR
533 005030 020504      CMP      R5,R4      ;COMPARE
534 005032 001401      BEQ      .+4      ;DID BIT6 SET?
535 005034 104004      ERROR    4      ;BIT6 NOT SET!
536 005036 012727 007640      MOV      #4000.,(PC)+      ;DELAY TIME
537 005042 000000      65$:      0      ;STORAGE
538 005044 005337 005042      DEC      65$      ;DELAY
539 005050 001375      BNE      .-4      ;
540 005052 040510      BIC      R5,(R0)      ;CLEAR BIT6
541 005054 011004      MOV      (R0),R4      ;READ CSR
542 005056 005005      CLR      R5      ;CLEAR EXPECTED
543 005060 005704      TST      R4      ;IS R4=0?
544 005062 001401      BEQ      .+4      ;BR IF OK!
545 005064 104004      ERROR    4      ;REGISTER NOT=0!
546 005066 005301      2$:      DEC      R1      ;I COUNT=0?
547 005070 001353      BNE      1$      ;BR IF NO
548 005072 000410      BR      4$      ;CONT TEST.
549 005074 010037 001160      3$:      MOV      R0,$TMP0      ;SAVE R0
550 005100 104005      ERROR    5      ;UNEXPECTED INTERRUPT

```

551	005102	013700	001160		MOV	\$TMP0,R0	;RESTORE R0
552	005106	012716	005066		MOV	#2\$,(SP)	;SET RETURN ADDRESS
553	005112	000002			RTI		;EXIT
554	005114	005005		4\$:	CLR	R5	
555	005116	012777	005146	174256	MOV	#5\$,@CLKVEC	
556	005124	012777	000200	174252	MOV	#200,@CLKPTY	
557	005132	106427	000000		MTPS	#0	
558	005136	005205			INC	R5	
559	005140	001376			BNE	.-2	
560	005142	104006			ERROR	6	
561	005144	000401			BR	6\$	
562	005146	022626		5\$:	CMP	(SP)+,(SP)+	
563	005150	012677	174230	6\$:	MOV	(SP)+,@CLKPTY	;RESTORE CLKPTY
564	005154	012677	174222		MOV	(SP)+,@CLKVEC	;RESTORE CLKVEC
565	005160	106427	000000		MTPS	#0	;SET PRIO TO LVL 0

566
567
568
569
570
571
572
573
574
575

```

;*****
;*TEST 6      600HZ INTERRUPT TEST
;*TEST THAT THE 600HZ CLOCK
;*CAN INTERRUPT AND THAT THE
;*INTERRUPTS ARE WITHIN THE SAME
;*TIME SPAN. 200. INTERRUPTS WILL BE
;*EXECUTED
;*****

```

576	005164	000004		TST6:	SCOPE		
577	005166	017746	174210		MOV	@CLKVEC,-(SP)	;SAVE CLKVEC
578	005172	017746	174206		MOV	@CLKPTY,-(SP)	;SAVE CLKPTY
579	005176	012777	005252	174176	MOV	#2\$,@CLKVEC	;SET INTERRUPT VECTOR
580	005204	012777	000200	174172	MOV	#200,@CLKPTY	;SET PRIO.
581	005212	012704	000310		MOV	#200.,R4	;SET ICOUNT
582	005216	012705	000002		MOV	#2,R5	;SET COUNT POSITION
583	005222	005001			CLR	R1	;ZERO COUNT UP.
584	005224	106427	000000		MTPS	#0	;CLEAR PSW
585	005230	013700	001400		MOV	CLKCSR,R0	;SAVE FOR PRINTOUT
586	005234	052777	000100	174136	BIS	#BIT6,@CLKCSR	;SET EVENT ENABLE
587	005242	005201		1\$:	INC	R1	;COUNT TIME
588	005244	001376			BNE	1\$	;=0?
589	005246	104006			ERROR	6	;NO INTERRUPT
590	005250	000423			BR	5\$	;EXIT TEST
591	005252	005305		2\$:	DEC	R5	;HOW MANY INTERRUPTS?
592	005254	001414			BEQ	3\$	;BR IF 2ND!
593	005256	100014			BPL	4\$	;BR IF 1ST!
594	005260	010102			MOV	R1,R2	;MUST BE 3RD OR MORE!
595	005262	062702	000010		ADD	#10,R2	;GIVE TOLARENCE OF +10
596	005266	160302			SUB	R3,R2	;GET DIFFERENCE
597	005270	100001			BPL	+.4	;\$SKIP IF POSITIVE
598	005272	005402			NEG	R2	;MAKE POSITIVE
599	005274	020227	000270		CMP	R2,#270	;GIVE +270 TOLARENCE
600	005300	101401			BLOS	+.4	
601	005302	104007			ERROR	7	;INTERRUPTS OUT OF TOLERANCE
602	005304	000401			BR	4\$	;CONT TEST
603	005306	010103		3\$:	MOV	R1,R3	;SAVE COUNT
604	005310	005001		4\$:	CLR	R1	;ZERO COUNTER
605	005312	005304			DEC	R4	;ICOUNT =0?
606	005314	001401			BEQ	5\$	;BR IF YES

```

607 005316 000002      RTI      ;EXIT
608 005320 042777 000100 174052 5$: BIC      #BIT6,@CLKCSR ;CLEAR EVENT ENABLE
609 005326 022626      CMP      (SP)+,(SP)+ ;POP INTR OFF STACK
610 005330 012677 174050      MOV      (SP)+,@CLKPTY ;RESTORE CLKPTY
611 005334 012677 174042      MOV      (SP)+,@CLKVEC ;RESTORE CLKVEC.
612 005340 106427 000000      MTPS     #0      ;SET PRIO TO LEVEL 0
613
614
615      ;*****
616      ;*TEST 7      UNEXPECTED 'BREPLY' TEST.
617      ;*TEST TO SCAN THROUGH ALL ADDRESS
618      ;*VERIFYING THAT ONLY EXPECTED DEVICES
619      ;*RETURN 'BREPLY'. THIS TEST CHECKS
620      ;*ONLY THAT 'UNEXPECTED 'BREPLYS'' AREN'T RECEIVED
621      ;*NOT THAT ALL DEVICE ISSUE 'BREPLY'.
622      ;*
623      ;*****
624 005344 000004      TST7:  SCOPE
625 005346 012737 000003 001162      MOV      #3,$TIMES      ;;DO 3 ITERATIONS
626 005354 013746 000004      MOV      4,-(SP)      ;SAVE 4
627 005360 013746 000006      MOV      6,-(SP)      ;SAVE 6
628 005364 012737 005406 000004      MOV      #2$,4      ;SET TIME-OUT TRAP
629 005372 005037 000006      CLR      6
630 005376 005000      CLR      R0      ;SET FIRST ADDRESS TO 0
631 005400 005710      1$:  TST      (R0)      ;READ ADDRESS
632 005402 005720      TST      (R0)+      ;POP ADDRESS
633 005404 000775      BR      1$      ;CONTINUE TEST
634 005406 022626      2$:  CMP      (SP)+,(SP)+      ;FAKE AN RTI
635 005410 023700 001376      CMP      MEMSIZ,R0      ;16K ?
636 005414 001403      BEQ      .+10      ;BR IF MEMORY SIZE IS CORRECT
637 005416 104016      ERROR  16      ;INCORRECT MEMORY SIZE FOUND
638 005420 104401 002301      TYPE      ,EM21      ;REPORT CHANGE MSG.
639      ;LOC SAVRO HAS MEMORY FOUND.
640 005424 012737 005462 000004      MOV      #6$,4      ;RESET TIME-OUT TRAP
641 005432 005710      3$:  TST      (R0)
642 005434 012701 014752      MOV      #BREPLY.TABLE,R1 ;SET EXPECTED DEVICES
643 005440 022100      4$:  CMP      (R1)+,R0      ;EXPECTED DEVICE?
644 005442 001002      BNE      5$      ;BR IF NO
645 005444 062100      ADD      (R1)+,R0      ;POP MODULO OFFSET
646 005446 000771      BR      3$      ;CONT TEST
647 005450 005721      5$:  TST      (R1)+      ;POP PAST OFFSET
648 005452 005711      TST      (R1)      ;END OF TABLE?
649 005454 001371      BNE      4$      ;BR IF NO
650 005456 104014      ERROR  14      ;UNEXPECTED 'BREPLY' FROM DEVICE.
651 005460 000401      BR      7$      ;SKIP FAKE RTI
652 005462 022626      6$:  CMP      (SP)+,(SP)+      ;FAKE RTI
653 005464 062700 000002      7$:  ADD      #2,R0      ;POP TO NEXT ADDRESS
654 005470 022700 177700      CMP      #177700,R0      ;ALL DONE?
655 005474 001356      BNE      3$
656 005476 012637 000006      MOV      (SP)+,6      ;RESTOE 6
657 005502 012637 000004      MOV      (SP)+,4 ;RESTORE 4
658
659      ;*****
660      ;*TEST 10      'DEVICE 'A'' ADDRESS TEST
661      ;*TEST TO VERIFY A 'BREPLY' RESPONSE
662      ;*FROM 'DEVICE 'A''.

```

```

663      ;*IT IS ASSUMED THAT THE DEVICE
664      ;*IS AT ADDRESS '172524'.
665      ;*****
666      TST10:  SCOPE
667      MOV     4,-(SP)      ;SAVE ADDRESS 4.
668      MOV     6,-(SP)      ;SAVE ADDRESS 6.
669      MOV     #15.,R1      ;SET INTERNAL ICOUNT TO 15.
670      MOV     DEV.A,R0      ;GET DEVICE CSR.
671      MOV     #3$,4        ;SET TIME-OUT VECTOR.
672      MOV     #200,6        ;SET PRIO.
673      1$:     TST     (R0)    ;REFERENCE DEVICE CSR.
674      2$:     DEC     R1      ;ICOUNT = 0?
675      BNE     1$          ;BR IF NO.
676      BR      4$          ;CONT TEST
677      3$:     ERROR   11      ;TIME-OUT ERROR.
678      MOV     #2$, (SP)    ;SET RETURN ADD.
679      RTI                     ;RETURN.
680      4$:     MOV     (SP)+,6  ;RESTORE 6
681      MOV     (SP)+,4        ;RESTORE 4
682
683
684      ;*****
685      ;*TEST 11      R/W TEST OF BIT 0 IN DEVICE 'A'
686      ;*R/W TEST OF BIT 0 IN DEVICE 'A'.
687      ;*WRITE BIT 0 VERIFY ONLY BIT 0 IS SET.
688      ;*CLEAR BIT 0 VERIFY BIT 0 IS CLEARED.
689      ;*
690      ;*****
691      TST11:  SCOPE
692      MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
693      MOV     #BIT0,R5      ;SET EXPECTED.
694      MOV     R5,(R0)        ;WRITE BIT0
695      MOV     (R0),R4        ;READ CSR
696      CMP     R5,R4          ;BIT OK?
697      BEQ     .+4            ;BR IF OK
698      ERROR   10            ;REGISTER HAS WRONG DATA.
699      BIC     R5,(R0)        ;CLEAR BIT0
700      MOV     (R0),R4        ;READ REGISTER.
701      BIC     #BIT0,R5      ;ZERO EXPECTED
702      TST     R4             ;REGISTER OK?
703      BEQ     .+4            ;BR IF YES
704      ERROR   10            ;REGISTER NOT =0.
705
706      ;*****
707      ;*TEST 12      R/W TEST OF BIT 1 IN DEVICE 'A'
708      ;*R/W TEST OF BIT 1 IN DEVICE 'A'.
709      ;*WRITE BIT 1 VERIFY ONLY BIT 1 IS SET.
710      ;*CLEAR BIT 1 VERIFY BIT 1 IS CLEARED.
711      ;*
712      ;*****
713      TST12:  SCOPE
714      MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
715      MOV     #BIT1,R5      ;SET EXPECTED.
716      MOV     R5,(R0)        ;WRITE BIT1
717      MOV     (R0),R4        ;READ CSR
718      CMP     R5,R4          ;BIT OK?

```

```

719 005656 001401      BEQ      .+4      ;BR IF OK
720 005660 104010      ERROR    10      ;REGISTER HAS WRONG DATA.
721 005662 040510      BIC      R5,(R0)  ;CLEAR BIT1
722 005664 011004      MOV      (R0),R4  ;READ REGISTER.
723 005666 042705 000002 BIC      #BIT1,R5  ;ZERO EXPECTED
724 005672 005704      TST      R4      ;REGISTER OK?
725 005674 001401      BEQ      .+4      ;BR IF YES
726 005676 104010      ERROR    10      ;REGISTER NOT =0.

```

```

*****
;TEST 13      R/W TEST OF BIT 2 IN DEVICE 'A'
;R/W TEST OF BIT 2 IN DEVICE 'A'.
;WRITE BIT 2 VERIFY ONLY BIT 2 IS SET.
;CLEAR BIT 2 VERIFY BIT 2 IS CLEARED.
;

```

```

734 *****
735 005700 000004      TST13:  SCOPE
736 005702 013700 001406      MOV      DEV.A,R0      ;LOAD DEVICE 'A' CSR.
737 005706 012705 000004      MOV      #BIT2,R5      ;SET EXPECTED.
738 005712 010510      MOV      R5,(R0)      ;WRITE BIT2
739 005714 011004      MOV      (R0),R4      ;READ CSR
740 005716 020504      CMP      R5,R4      ;BIT OK?
741 005720 001401      BEQ      .+4      ;BR IF OK
742 005722 104010      ERROR    10      ;REGISTER HAS WRONG DATA.
743 005724 040510      BIC      R5,(R0)      ;CLEAR BIT2
744 005726 011004      MOV      (R0),R4      ;READ REGISTER.
745 005730 042705 000004      BIC      #BIT2,R5  ;ZERO EXPECTED
746 005734 005704      TST      R4      ;REGISTER OK?
747 005736 001401      BEQ      .+4      ;BR IF YES
748 005740 104010      ERROR    10      ;REGISTER NOT =0.

```

```

*****
;TEST 14      R/W TEST OF BIT 3 IN DEVICE 'A'
;R/W TEST OF BIT 3 IN DEVICE 'A'.
;WRITE BIT 3 VERIFY ONLY BIT 3 IS SET.
;CLEAR BIT 3 VERIFY BIT 3 IS CLEARED.
;

```

```

756 *****
757 005742 000004      TST14:  SCOPE
758 005744 013700 001406      MOV      DEV.A,R0      ;LOAD DEVICE 'A' CSR.
759 005750 012705 000010      MOV      #BIT3,R5      ;SET EXPECTED.
760 005754 010510      MOV      R5,(R0)      ;WRITE BIT3
761 005756 011004      MOV      (R0),R4      ;READ CSR
762 005760 020504      CMP      R5,R4      ;BIT OK?
763 005762 001401      BEQ      .+4      ;BR IF OK
764 005764 104010      ERROR    10      ;REGISTER HAS WRONG DATA.
765 005766 040510      BIC      R5,(R0)      ;CLEAR BIT3
766 005770 011004      MOV      (R0),R4      ;READ REGISTER.
767 005772 042705 000010      BIC      #BIT3,R5  ;ZERO EXPECTED
768 005776 005704      TST      R4      ;REGISTER OK?
769 006000 001401      BEQ      .+4      ;BR IF YES
770 006002 104010      ERROR    10      ;REGISTER NOT =0.

```

```

*****
;TEST 15      R/W TEST OF BIT 4 IN DEVICE 'A'
;R/W TEST OF BIT 4 IN DEVICE 'A'.

```

```

775      ;*WRITE BIT 4 VERIFY ONLY BIT 4 IS SET.
776      ;*CLEAR BIT 4 VERIFY BIT 4 IS CLEARED.
777      ;*
778      ;*****
779      TST15:  SCOPE
780      MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
781      MOV     #BIT4,R5     ;SET EXPECTED.
782      MOV     R5,(R0)       ;WRITE BIT4
783      MOV     (R0),R4       ;READ CSR
784      CMP     R5,R4         ;BIT OK?
785      BEQ     .+4           ;BR IF OK
786      ERROR   10           ;REGISTER HAS WRONG DATA.
787      BIC     R5,(R0)       ;CLEAR BIT4
788      MOV     (R0),R4       ;READ REGISTER.
789      BIC     #BIT4,R5     ;ZERO EXPECTED
790      TST     R4            ;REGISTER OK?
791      BEQ     .+4           ;BR IF YES
792      ERROR   10           ;REGISTER NOT =0.
793
794      ;*****
795      ;*TEST 16      R/W TEST OF BIT 5 IN DEVICE 'A'
796      ;*R/W TEST OF BIT 5 IN DEVICE 'A'.
797      ;*WRITE BIT 5 VERIFY ONLY BIT 5 IS SET.
798      ;*CLEAR BIT 5 VERIFY BIT 5 IS CLEARED.
799      ;*
800      ;*****
801      TST16:  SCOPE
802      MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
803      MOV     #BIT5,R5     ;SET EXPECTED.
804      MOV     R5,(R0)       ;WRITE BIT5
805      MOV     (R0),R4       ;READ CSR
806      CMP     R5,R4         ;BIT OK?
807      BEQ     .+4           ;BR IF OK
808      ERROR   10           ;REGISTER HAS WRONG DATA.
809      BIC     R5,(R0)       ;CLEAR BIT5
810      MOV     (R0),R4       ;READ REGISTER.
811      BIC     #BIT5,R5     ;ZERO EXPECTED
812      TST     R4            ;REGISTER OK?
813      BEQ     .+4           ;BR IF YES
814      ERROR   10           ;REGISTER NOT =0.
815
816      ;*****
817      ;*TEST 17      R/W TEST OF BIT 6 IN DEVICE 'A'
818      ;*R/W TEST OF BIT 6 IN DEVICE 'A'.
819      ;*WRITE BIT 6 VERIFY ONLY BIT 6 IS SET.
820      ;*CLEAR BIT 6 VERIFY BIT 6 IS CLEARED.
821      ;*
822      ;*****
823      TST17:  SCOPE
824      MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
825      MOV     #BIT6,R5     ;SET EXPECTED.
826      MOV     R5,(R0)       ;WRITE BIT6
827      MOV     (R0),R4       ;READ CSR
828      CMP     R5,R4         ;BIT OK?
829      BEQ     .+4           ;BR IF OK
830      ERROR   10           ;REGISTER HAS WRONG DATA.

```

```
831 006134 040510      BIC    R5,(R0)      ;CLEAR BIT6
832 006136 011004      MOV     (R0),R4      ;READ REGISTER.
833 006140 042705 000100 BIC     #BIT6,R5      ;ZERO EXPECTED
834 006144 005704      TST     R4        ;REGISTER OK?
835 006146 001401      BEQ     .+4        ;BR IF YES
836 006150 104010      ERROR   10        ;REGISTER NOT =0.
```

```
837
838 ;*****
839 ;*TEST 20      R/W TEST OF BIT 7 IN DEVICE 'A'
840 ;*R/W TEST OF BIT 7 IN DEVICE 'A'.
841 ;*WRITE BIT 7 VERIFY ONLY BIT 7 IS SET.
842 ;*CLEAR BIT 7 VERIFY BIT 7 IS CLEARED.
843 ;*
```

```
844 ;*****
845 006152 000004      TST20:  SCOPE
846 006154 013700 001406 MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
847 006160 012705 000200 MOV     #BIT7,R5      ;SET EXPECTED.
848 006164 010510      MOV     R5,(R0)      ;WRITE BIT7
849 006166 011004      MOV     (R0),R4      ;READ CSR
850 006170 020504      CMP     R5,R4        ;BIT OK?
851 006172 001401      BEQ     .+4        ;BR IF OK
852 006174 104010      ERROR   10        ;REGISTER HAS WRONG DATA.
853 006176 040510      BIC     R5,(R0)      ;CLEAR BIT7
854 006200 011004      MOV     (R0),R4      ;READ REGISTER.
855 006202 042705 000200 BIC     #BIT7,R5      ;ZERO EXPECTED
856 006206 005704      TST     R4        ;REGISTER OK?
857 006210 001401      BEQ     .+4        ;BR IF YES
858 006212 104010      ERROR   10        ;REGISTER NOT =0.
```

```
859
860 ;*****
861 ;*TEST 21      R/W TEST OF BIT 8 IN DEVICE 'A'
862 ;*R/W TEST OF BIT 8 IN DEVICE 'A'.
863 ;*WRITE BIT 8 VERIFY ONLY BIT 8 IS SET.
864 ;*CLEAR BIT 8 VERIFY BIT 8 IS CLEARED.
865 ;*
```

```
866 ;*****
867 006214 000004      TST21:  SCOPE
868 006216 013700 001406 MOV     DEV.A,R0      ;LOAD DEVICE 'A' CSR.
869 006222 012705 000400 MOV     #BIT8,R5      ;SET EXPECTED.
870 006226 010510      MOV     R5,(R0)      ;WRITE BIT8
871 006230 011004      MOV     (R0),R4      ;READ CSR
872 006232 020504      CMP     R5,R4        ;BIT OK?
873 006234 001401      BEQ     .+4        ;BR IF OK
874 006236 104010      ERROR   10        ;REGISTER HAS WRONG DATA.
875 006240 040510      BIC     R5,(R0)      ;CLEAR BIT8
876 006242 011004      MOV     (R0),R4      ;READ REGISTER.
877 006244 042705 000400 BIC     #BIT8,R5      ;ZERO EXPECTED
878 006250 005704      TST     R4        ;REGISTER OK?
879 006252 001401      BEQ     .+4        ;BR IF YES
880 006254 104010      ERROR   10        ;REGISTER NOT =0.
```

```
881
882 ;*****
883 ;*TEST 22      R/W TEST OF BIT 9 IN DEVICE 'A'
884 ;*R/W TEST OF BIT 9 IN DEVICE 'A'.
885 ;*WRITE BIT 9 VERIFY ONLY BIT 9 IS SET.
886 ;*CLEAR BIT 9 VERIFY BIT 9 IS CLEARED.
```

```

887
888
889 006256 000004
890 006260 013700 001406
891 006264 012705 001000
892 006270 010510
893 006272 011004
894 006274 020504
895 006276 001401
896 006300 104010
897 006302 040510
898 006304 011004
899 006306 042705 001000
900 006312 005704
901 006314 001401
902 006316 104010
903
904
905
906
907
908
909
910
911 006320 000004
912 006322 013700 001406
913 006326 012705 002000
914 006332 010510
915 006334 011004
916 006336 020504
917 006340 001401
918 006342 104010
919 006344 040510
920 006346 011004
921 006350 042705 002000
922 006354 005704
923 006356 001401
924 006360 104010
925
926
927
928
929
930
931
932
933 006362 000004
934 006364 013700 001406
935 006370 012705 004000
936 006374 010510
937 006376 011004
938 006400 020504
939 006402 001401
940 006404 104010
941 006406 040510
942 006410 011004

;*****
;TST22: SCOPE
;MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
;MOV #BIT9,R5 ;SET EXPECTED.
;MOV R5,(R0) ;WRITE BIT9
;MOV (R0),R4 ;READ CSR
;CMP R5,R4 ;BIT OK?
;BEQ .+4 ;BR IF OK
;ERROR 10 ;REGISTER HAS WRONG DATA.
;BIC R5,(R0) ;CLEAR BIT9
;MOV (R0),R4 ;READ REGISTER.
;BIC #BIT9,R5 ;ZERO EXPECTED
;TST R4 ;REGISTER OK?
;BEQ .+4 ;BR IF YES
;ERROR 10 ;REGISTER NOT =0.

;*****
;*TEST 23 R/W TEST OF BIT 10 IN DEVICE 'A'
;*R/W TEST OF BIT 10 IN DEVICE 'A'.
;*WRITE BIT 10 VERIFY ONLY BIT 10 IS SET.
;*CLEAR BIT 10 VERIFY BIT 10 IS CLEARED.
;*
;*****
;TST23: SCOPE
;MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
;MOV #BIT10,R5 ;SET EXPECTED.
;MOV R5,(R0) ;WRITE BIT10
;MOV (R0),R4 ;READ CSR
;CMP R5,R4 ;BIT OK?
;BEQ .+4 ;BR IF OK
;ERROR 10 ;REGISTER HAS WRONG DATA.
;BIC R5,(R0) ;CLEAR BIT10
;MOV (R0),R4 ;READ REGISTER.
;BIC #BIT10,R5 ;ZERO EXPECTED
;TST R4 ;REGISTER OK?
;BEQ .+4 ;BR IF YES
;ERROR 10 ;REGISTER NOT =0.

;*****
;*TEST 24 R/W TEST OF BIT 11 IN DEVICE 'A'
;*R/W TEST OF BIT 11 IN DEVICE 'A'.
;*WRITE BIT 11 VERIFY ONLY BIT 11 IS SET.
;*CLEAR BIT 11 VERIFY BIT 11 IS CLEARED.
;*
;*****
;TST24: SCOPE
;MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
;MOV #BIT11,R5 ;SET EXPECTED.
;MOV R5,(R0) ;WRITE BIT11
;MOV (R0),R4 ;READ CSR
;CMP R5,R4 ;BIT OK?
;BEQ .+4 ;BR IF OK
;ERROR 10 ;REGISTER HAS WRONG DATA.
;BIC R5,(R0) ;CLEAR BIT11
;MOV (R0),R4 ;READ REGISTER.

```

CWQAABO 11W03 SYSTEM TEST
CWQAAB.P11 22-MAR-78 14:51

MACY11 30A(1052) 23-MAR-78 07:51 PAGE 20
T24 R/W TEST OF BIT 11 IN DEVICE 'A'

SEQ 0026

943 006412 042705 004000
944 006416 005704
945 006420 001401
946 006422 104010

BIC #BIT11,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

947
948

949 ;*****
950 ;*TEST 25 R/W TEST OF BIT 12 IN DEVICE 'A'
951 ;*R/W TEST OF BIT 12 IN DEVICE 'A'.
952 ;*WRITE BIT 12 VERIFY ONLY BIT 12 IS SET.
953 ;*CLEAR BIT 12 VERIFY BIT 12 IS CLEARED.
954 ;*

955
956 006424 000004
957 006426 013700 001406
958 006432 012705 010000
959 006436 010510
960 006440 011004
961 006442 020504
962 006444 001401
963 006446 104010
964 006450 040510
965 006452 011004
966 006454 042705 010000
967 006460 005704
968 006462 001401
969 006464 104010

TST25: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT12,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT12
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT12
MOV (R0),R4 ;READ REGISTER.
BIC #BIT12,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

970
971

972 ;*****
973 ;*TEST 26 R/W TEST OF BIT 13 IN DEVICE 'A'
974 ;*R/W TEST OF BIT 13 IN DEVICE 'A'.
975 ;*WRITE BIT 13 VERIFY ONLY BIT 13 IS SET.
976 ;*CLEAR BIT 13 VERIFY BIT 13 IS CLEARED.
977 ;*

978
979 006466 000004
980 006470 013700 001406
981 006474 012705 020000
982 006500 010510
983 006502 011004
984 006504 020504
985 006506 001401
986 006510 104010
987 006512 040510
988 006514 011004
989 006516 042705 020000
990 006522 005704
991 006524 001401
992 006526 104010

TST26: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT13,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT13
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT13
MOV (R0),R4 ;READ REGISTER.
BIC #BIT13,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

993
994

995 ;*****
996 ;*TEST 27 R/W TEST OF BIT 14 IN DEVICE 'A'
997 ;*R/W TEST OF BIT 14 IN DEVICE 'A'.
998 ;*WRITE BIT 14 VERIFY ONLY BIT 14 IS SET.
;*CLEAR BIT 14 VERIFY BIT 14 IS CLEARED.
;*

999
1000

999 006530 000004
1000 006532 013700 001406
1001 006536 012705 040000
1002 006542 010510
1003 006544 011004
1004 006546 020504
1005 006550 001401
1006 006552 104010
1007 006554 040510
1008 006556 011004
1009 006560 042705 040000
1010 006564 005704
1011 006566 001401
1012 006570 104010
1013
1014
1015
1016
1017
1018
1019
1020

TST27: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT14,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT14
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT14
MOV (R0),R4 ;READ REGISTER.
BIC #BIT14,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

*TEST 30 R/W TEST OF BIT 15 IN DEVICE 'A'
*R/W TEST OF BIT 15 IN DEVICE 'A'.
*WRITE BIT 15 VERIFY ONLY BIT 15 IS SET.
*CLEAR BIT 15 VERIFY BIT 15 IS CLEARED.
*

1021 006572 000004
1022 006574 013700 001406
1023 006600 012705 100000
1024 006604 010510
1025 006606 011004
1026 006610 020504
1027 006612 001401
1028 006614 104010
1029 006616 040510
1030 006620 011004
1031 006622 042705 100000
1032 006626 005704
1033 006630 001401
1034 006632 104010
1035
1036
1037
1038
1039
1040
1041
1042

TST30: SCOPE
MOV DEV.A,R0 ;LOAD DEVICE 'A' CSR.
MOV #BIT15,R5 ;SET EXPECTED.
MOV R5,(R0) ;WRITE BIT15
MOV (R0),R4 ;READ CSR
CMP R5,R4 ;BIT OK?
BEQ .+4 ;BR IF OK
ERROR 10 ;REGISTER HAS WRONG DATA.
BIC R5,(R0) ;CLEAR BIT15
MOV (R0),R4 ;READ REGISTER.
BIC #BIT15,R5 ;ZERO EXPECTED
TST R4 ;REGISTER OK?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER NOT =0.

*TEST 31 BINARY COUNT TEST
*TEST TO WRITE A BINARY COUNT PATTERN
*(000000-177777) THRU DEVICE 'A' REGISTER
*

1043 006634 000004
1044 006636 012737 000003 001162
1045 006644 013700 001406
1046 006650 005005
1047 006652 010510
1048 006654 011004
1049 006656 020504
1050 006660 001401
1051 006662 104010
1052 006664 005205
1053 006666 001371
1054

TST31: SCOPE
MOV #3,\$TIMES ;DO 3 ITERATIONS
MOV DEV.A,R0 ;GET DEVICE 'A' CSR.
CLR R5 ;ZERO EXPECTED
1\$: MOV R5,(R0) ;LOAD DATA
MOV (R0),R4 ;READ DATA.
CMP R5,R4 ;WAS DATA GOOD?
BEQ .+4 ;BR IF YES
ERROR 10 ;REGISTER DATA ERROR.
INC R5 ;UPADE DATA
BNE 1\$;BR IF NOT DONE

```

1055
1056
1057
1058
1059
1060
1061
1062 006670 000004
1063 006672 012737 000012 001162
1064 006700 013700 001406
1065 006704 005003
1066 006706 012705 125252
1067 006712 010510
1068 006714 011004
1069 006716 020504
1070 006720 001401
1071 006722 104010
1072 006724 005105
1073 006726 105203
1074 006730 100370
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086 006732 000004
1087 006734 013700 001406
1088 006740 012710 001400
1089 006744 005002
1090 006746 012703 000003
1091 006752 010001
1092 006754 005201
1093 006756 105210
1094 006760 111004
1095 006762 105202
1096 006764 110205
1097 006766 020504
1098 006770 001401
1099 006772 104010
1100 006774 011004
1101 006776 110237 001160
1102 007002 110337 001161
1103 007006 013705 001160
1104 007012 020504
1105 007014 001401
1106 007016 104010
1107 007020 105705
1108 007022 001410
1109 007024 105211
1110 007026 111104

;*****
;*TEST 32      1'S AND 0'S TEST
;*TEST TO WRITE ALTERNATE 1'S AND 0'S
;*(125252 AND 052525) INTO DEVICE 'A'.
;*
;*****
TST32:  SCOPE
        MOV      #10, $TIMES      ;;DO 10. ITERATIONS
        MOV      DEV.A, R0        ;GET DEVICE 'A' CSR
        CLR      R3              ;SET TO DO 128. TIMES
        MOV      #125252, R5      ;LOAD DATA
1$:     MOV      R5, (R0)         ;WRITE DATA
        MOV      (R0), R4        ;READ REGISTER
        CMP      R5, R4         ;DATA OK?
        BEQ      .+4            ;BR IF OK?
        ERROR    10             ;REGISTER R/W ERROR
        COM      R5             ;CHANGE DATA
        INCB     R3             ;DONE 128. TIMES?
        BPL      1$            ;BR IF NO.

;*****
;*TEST 33      DEVICE 'A' BYTE OPERATIONS
;*TEST OF DEVICE 'A' BYTE OPERATIONS
;*SET LOW BYTE OF DEVICE 'A' TO 0 AND
;*SET HIGH BYTE OF DEVICE 'A' TO 3 THEN
;*DO A 'INCB' ON LOW BYTE CHECK RESULTS
;*DO A 'INCB' ON HIGH BYTE CHECK RESULTS
;*CONTINUE TILL LOW BYTE GOES TO ZERO
;*****
TST33:  SCOPE
        MOV      DEV.A, R0        ;LOAD DEVICE 'A' CSR.
        MOV      #3*400, (R0)    ;PLACE 3 IN HIGH BYTE
        CLR      R2              ;ZERO LOW BYTE IMAGE
        MOV      #3, R3          ;SET HIGH BYTE IMAGE TO 3.
        MOV      R0, R1          ;GET CSR
        INC      R1              ;MAKE EQUAL TO HIGH BYTE.
1$:     INCB     (R0)             ;ALTER LOW BYTE
        MOVB     (R0), R4        ;READ IT
        INCB     R2              ;ALTER LOW BYTE IMAGE
        MOVB     R2, R5          ;READ GOOD DATA
        CMP      R5, R4         ;IS DATA OK?
        BEQ      .+4            ;BR IF YES
        ERROR    10             ;REGISTER ERROR
        MOV      (R0), R4        ;READ WORD
        MOVB     R2, $TMP0       ;GET LOW BYTE
        MOVB     R3, $TMP0+1     ;GET HIGH BYTE
        MOV      $TMP0, R5       ;GOOD DATA
        CMP      R5, R4         ;STILL OK?
        BEQ      .+4            ;
        ERROR    10             ;COMPARISON ERROR
        TSTB     R5             ;IF LOW BYTE =0 THEN DONE
        BEQ      2$             ;BR IF END OF TEST
        INCB     (R1)           ;ALTER HIGH BYTE
        MOVB     (R1), R4        ;READ IT

```

```
1111 007030 105203      INCB      R3          ;ALTER HIGH BYTE IMAGE
1112 007032 110305      MOVB      R3,R5      ;LOAD GOOD DATA
1113 007034 020504      CMP        R5,R4      ;DATA OK?
1114 007036 001401      BEQ        .+4        ;BR IF YES.
1115 007040 104010      ERROR      10         ;REGISTER ERROR
1116 007042 000745      BR         1$         ;CONT TEST
1117 007044 000240      2$:      NOP
```

```
1118
1119
1120 ;*****
1121 ;*TEST 34      BUS INITIALIZE TEST FOR DEVICE 'A'
1122 ;*TEST THAT ALL BITS IN DEVICE 'A'
1123 ;*REGISTER CAN BE CLEARED BY "INIT" (RESET INSTR).
1124 ;*
1125 ;*****
```

```
1126 007046 000004      TST34:  SCOPE
1127 007050 012737 000005 001162      MOV      #5,$TIMES      ;;DO 5 ITERATIONS
1128 007056 013700 001406      MOV      DEV.A,R0      ;GET CSR
1129 007062 012710 177777      MOV      #-1,(R0)      ;SET ALL BITS
1130 007066 005005      CLR        R5          ;SET EXPECTED TO ALL ZEROS
1131 007070 000005      RESET      ;ISSUE INIT
1132 007072 052777 000100 172044      BIS      #100,@$TKS      ;SET INTR ENABLE
1133 007100 011004      MOV      (R0),R4      ;READ REGISTER
1134 007102 001401      BEQ        .+4        ;BR IF ALL ZEROS
1135 007104 104010      ERROR      10         ;REGISTER NOT ALL ZEROS
1136
```

```

1137
1138
1139      ;*****
1140      ;*TEST 35      'DEVICE 'B'' ADDRESS TEST (SEL 0)
1141      ;*TEST TO VERIFY A 'BREPLY' RESPONSE
1142      ;*FROM 'DEVICE 'B'' (SEL 0).
1143      ;*IT IS ASSUMED THAT THE DEVICE
1144      ;*IS AT ADDRESS '165250' (SEL 0).
1145      ;*****
1145      007106 000004      TST35:  SCOPE
1146      007110 013746 000004      MOV      4,-(SP)      ;SAVE ADDRESS 4.
1147      007114 013746 000006      MOV      6,-(SP)      ;SAVE ADDRESS 6.
1148      007120 012701 000017      MOV      #15.,R1      ;SET INTERNAL ICOUNT TO 15.
1149      007124 013700 001410      MOV      DEV.B,R0      ;GET DEVICE CSR.
1150      007130 012737 007154 000004      MOV      #3$,4      ;SET TIME-OUT VECTOR.
1151      007136 012737 000200 000006      MOV      #200,6      ;SET PRIO.
1152      007144 005710      1$:      TST      (R0)      ;REFERENCE DEVICE CSR.
1153      007146 005301      2$:      DEC      R1      ;ICOUNT = C?
1154      007150 001375      BNE      1$      ;BR IF NO.
1155      007152 000404      BR      4$      ;CONT TEST
1156      007154 104012      3$:      ERROR    12      ;TIME-OUT ERROR.
1157      007156 012716 007146      MOV      #2$, (SP)      ;SET RETURN ADD.
1158      007162 000002      RTI      ;RETURN.
1159      007164 012637 000006      4$:      MOV      (SP)+,6      ;RESTORE 6
1160      007170 012637 000004      MOV      (SP)+,4      ;RESTORE 4
1161
1162
1163      ;*****
1164      ;*TEST 36      R/W TEST OF BIT 6 IN DEVICE 'B' (SEL 0)
1165      ;*TEST THAT BIT 6 IN DEVICE 'B' (SEL 0)
1166      ;*IS R/W.
1167      ;*SET BIT 6 VERIFY IT IS SET.
1168      ;*CLEAR BIT 6 VERIFY IT IS CLEAR.
1169      ;*
1170      ;*****
1171      007174 000004      TST36:  SCOPE
1172      007176 106427 000200      MTPS     #200
1173      007202 013700 001410      MOV      DEV.B,R0      ;LOAD CSR INTO R0
1174      007206 012705 000100      MOV      #BIT6,R5      ;SET BIT 6 INTO R5
1175      007212 010510      MOV      R5,(R0)      ;WRITE BIT
1176      007214 011004      MOV      (R0),R4      ;READ REGISTER
1177      007216 020504      CMP      R5,R4      ;OK?
1178      007220 001401      BEQ      .+4      ;BR IF OK.
1179      007222 104013      ERROR    13      ;REGISTER R/W ERROR
1180      007224 040510      BIC      R5,(R0)      ;CLEAR BIT 6
1181      007226 011004      MOV      (R0),R4      ;READ DEVICE
1182      007230 042705 000100      BIC      #BIT6,R5      ;CLEAR EXPECTED
1183      007234 005704      TST      R4      ;REGISTER =0?
1184      007236 001401      BEQ      .+4      ;BR IF =0!
1185      007240 104013      ERROR    13      ;REGISTER ERROR
1186
1187      ;*****
1188      ;*TEST 37      R/W TEST OF BIT 7 IN DEVICE 'B' (SEL 0)
1189      ;*TEST THAT BIT 7 IN DEVICE 'B' (SEL 0)
1190      ;*IS R/W.
1191      ;*SET BIT 7 VERIFY IT IS SET.
1192      ;*CLEAR BIT 7 VERIFY IT IS CLEAR.

```

```

1193
1194
1195 007242 000004
1196 007244 106427 000200
1197 007250 013700 001410
1198 007254 012705 000200
1199 007260 010510
1200 007262 011004
1201 007264 020504
1202 007266 001401
1203 007270 104013
1204 007272 040510
1205 007274 011004
1206 007276 042705 000200
1207 007302 005704
1208 007304 001401
1209 007306 104013
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221 007310 000004
1222 007312 106427 000000
1223 007316 017746 172070
1224 007322 017746 172066
1225 007326 012777 007442 172056
1226 007334 012777 000200 172052
1227 007342 005001
1228 007344 013700 001410
1229 007350 052710 000100
1230 007354 000240
1231 007356 042710 000100
1232 007362 052710 000200
1233 007366 000240
1234 007370 012777 007410 172014
1235 007376 052710 000100
1236 007402 000240
1237 007404 104015
1238 007406 000407
1239 007410 005010
1240 007412 105203
1241 007414 100403
1242 007416 012716 007350
1243 007422 000002
1244 007424 022626
1245 007426 005010
1246 007430 012677 171760
1247 007434 012677 171752
1248 007440 000404

;*****
;TST37: SCOPE
;MTPS #200
;MOV DEV.B,R0 ;LOAD CSR INTO R0
;MOV #BIT7,R5 ;SET BIT 7 INTO R5
;MOV R5,(R0) ;WRITE BIT
;MOV (R0),R4 ;READ REGISTER
;CMP R5,R4 ;OK?
;BEQ .+4 ;BR IF OK.
;ERROR 13 ;REGISTER R/W ERROR
;BIC R5,(R0) ;CLEAR BIT 7
;MOV (R0),R4 ;READ DEVICE
;BIC #BIT7,R5 ;CLEAR EXPECTED
;TST R4 ;REGISTER =0?
;BEQ .+4 ;BR IF =0!
;ERROR 13 ;REGISTER ERROR

;*****
;*TEST 40 DEVICE 'B' INTERRUPT TEST (SEL 0)
;*TEST OF DEVICE 'B' INTERRUPTS (SEL 0)
;*SET BIT6 VERIFY NO INTERRUPT
;*CLEAR BIT6; SET BIT7 -VERIFY NO INTERRUPT
;*NOW SET BIT6 AGAIN -EXPECT INTERRUPT AT
;*VECTOR '340'.
;*****
;TST40: SCOPE
;MTPS #0
;MOV @B.AVEC,-(SP) ;SAVE DVB VECTOR
;MOV @B.APTY,-(SP) ;SAVE DVB PRIO
;MOV #5$,@B.AVEC ;LOAD VECTOR
;MOV #200,@B.APTY ;LOAD PRIO
;CLR R1
;MOV DEV.B,R0
;1$: BIS #BIT6,(R0) ;SET EVENT ENABLE
;NOP ;WASTE TIME
;BIC #BIT6,(R0) ;CLEAR IT
;BIS #BIT7,(R0) ;SET OTHER INTR
;NOP ;WASTE TIME
;MOV #2$,@B.AVEC ;SET GOOD VECTOR
;BIS #BIT6,(R0) ;SET IE
;NOP ;WASTE TIME
;ERROR 15 ;NO INTERRUPT
;BR 4$ ;CONT TEST
;2$: CLR (R0) ;ZERO REGISTER
;INCB R3 ;UPDATE ICOUNT
;BMI 3$ ;DONE?
;MOV #1$,(SP) ;SET RETURN
;RTI ;EXIT
;3$: CMP (SP)+,(SP)+ ;POP PC+PSW
;4$: CLR (R0) ;DSABLE DEVICE
;MOV (SP)+,@B.APTY ;RESTORE PTY
;MOV (SP)+,@B.AVEC ;RESTORE VEC
;BR 6$ ;CONT TEST

```

1249 007442 010002
 1250 007444 011600
 1251 007446 104005
 1252 007450 010200
 1253 007452 106427 000000

5\$: MOV R0,R2 ;SAVE R0
 MOV (SP),R0 ;SAVE PC.
 ERROR 5 ;UNEXPECTED INTERRUPT
 MOV R2,R0 ;RESTORE R0
 6\$: MTPS #0 ;ZERO PTY

 ;*TEST 41 'DEVICE 'B'' ADDRESS TEST (SEL 2)
 ;*TEST TO VERIFY A 'BREPLY' RESPONSE
 ;*FROM 'DEVICE 'B'' (SEL 2).
 ;*IT IS ASSUMED THAT THE DEVICE
 ;*IS AT ADDRESS '165252' (SEL 2).
 ;*****

1263 007456 000004
 1264 007460 013746 000004
 1265 007464 013746 000006
 1266 007470 012701 000017
 1267 007474 013700 001410
 1268 007500 062700 000002
 1269 007504 012737 007530 000004
 1270 007512 012737 000200 000006
 1271 007520 005710
 1272 007522 005301
 1273 007524 001375
 1274 007526 000404
 1275 007530 104012
 1276 007532 012716 007522
 1277 007536 000002
 1278 007540 012637 000006
 1279 007544 012637 000004

TST41: SCOPE
 MOV 4,-(SP) ;SAVE ADDRESS 4.
 MOV 6,-(SP) ;SAVE ADDRESS 6.
 MOV #15,R1 ;SET INTERNAL ICOUNT TO 15.
 MOV DEV.B,R0 ;GET DEVICE CSR.
 ADD #2,R0 ;MAKE IT SEL 2
 MOV #3\$,4 ;SET TIME-OUT VECTOR.
 MOV #200,6 ;SET PRIO.
 1\$: TST (R0) ;REFERENCE DEVICE CSR.
 2\$: DEC R1 ;ICOUNT = 0?
 BNE 1\$;BR IF NO.
 BR 4\$;CONT TEST
 3\$: ERROR 12 ;TIME-OUT ERROR.
 MOV #2\$, (SP) ;SET RETURN ADD.
 RTI ;RETURN.
 4\$: MOV (SP)+,6 ;RESTORE 6
 MOV (SP)+,4 ;RESTORE 4

 ;*TEST 42 R/W TEST OF BIT 6 IN DEVICE 'B' (SEL 2)
 ;*TEST THAT BIT 6 IN DEVICE 'B' (SEL 2)
 ;*IS R/W.
 ;*SET BIT 6 VERIFY IT IS SET.
 ;*CLEAR BIT 6 VERIFY IT IS CLEAR.
 ;*

1289
 1290 007550 000004
 1291 007552 106427 000200
 1292 007556 013700 001410
 1293 007562 062700 000002
 1294 007566 012705 000100
 1295 007572 010510
 1296 007574 011004
 1297 007576 020504
 1298 007600 001401
 1299 007602 104013
 1300 007604 040510
 1301 007606 011004
 1302 007610 042705 000100
 1303 007614 005704
 1304 007616 001401

TST42: SCOPE
 MTPS #200
 MOV DEV.B,R0 ;LOAD CSR INTO R0
 ADD #2,R0 ;MAKE IT SEL 2
 MOV #BIT6,R5 ;SET BIT 6 INTO R5
 MOV R5,(R0) ;WRITE BIT
 MOV (R0),R4 ;READ REGISTER
 CMP R5,R4 ;OK?
 BEQ .+4 ;BR IF OK.
 ERROR 13 ;REGISTER R/W ERROR
 BIC R5,(R0) ;CLEAR BIT 6
 MOV (R0),R4 ;READ DEVICE
 BIC #BIT6,R5 ;CLEAR EXPECTED
 TST R4 ;REGISTER =0?
 BEQ .+4 ;BR IF =0!

```

CWQAAB0 11W03 SYSTEM TEST          MACY11 30A(1052) 23-MAR-78 07:51 PAGE 27
CWQAAB.P11 22-MAR-78 14:51          T42      R/W TEST OF BIT 6 IN DEVICE 'B' (SEL 2)
                                           SEQ 0033

1305 007620 104013          ERROR 13          ;REGISTER ERROR
1306 007622 106427 000000    MTPS  #0          ;SET PTY TO 0
1307
1308
1309 ;*****
1310 ;*TEST 43      R/W TEST OF BIT 7 IN DEVICE 'B' (SEL 2)
1311 ;*TEST THAT BIT 7 IN DEVICE 'B' (SEL 2)
1312 ;*IS R/W.
1313 ;*SET BIT 7 VERIFY IT IS SET.
1314 ;*CLEAR BIT 7 VERIFY IT IS CLEAR.
1315 ;*
1316 ;*****
1316 007626 000004    TST43: SCOPE
1317 007630 106427 000200    MTPS      #200
1318 007634 013700 001410    MOV      DEV.B,R0      ;LOAD CSR INTO R0
1319 007640 062700 000002    ADD      #2,R0      ;MAKE IT SEL 2
1320 007644 012705 000200    MOV      #BIT7,R5      ;SET BIT 7 INTO R5
1321 007650 010510          MOV      R5,(R0)      ;WRITE BIT
1322 007652 011004          MOV      (R0),R4      ;READ REGISTER
1323 007654 020504          CMP      R5,R4      ;OK?
1324 007656 001401          BEQ      .+4      ;BR IF OK.
1325 007660 104013          ERROR 13      ;REGISTER R/W ERROR
1326 007662 040510          BIC      R5,(R0)      ;CLEAR BIT 7
1327 007664 011004          MOV      (R0),R4      ;READ DEVICE
1328 007666 042705 000200    BIC      #BIT7,R5      ;CLEAR EXPECTED
1329 007672 005704          TST      R4      ;REGISTER =0?
1330 007674 001401          BEQ      .+4      ;BR IF =0!
1331 007676 104013          ERROR 13      ;REGISTER ERROR
1332 007700 106427 000000    MTPS      #0      ;SET PTY TO 0
1333
1334
1335 ;*****
1336 ;*TEST 44      DEVICE 'B' INTERRUPT TEST (SEL 2)
1337 ;*TEST OF DEVICE 'B' INTERRUPTS (SEL 2)
1338 ;*SET BIT6 VERIFY NO INTERRUPT
1339 ;*CLEAR BIT6; SET BIT7 -VERIFY NO INTERRUPT
1340 ;*NOW SET BIT6 AGAIN -EXPECT INTERUPT AT
1341 ;*VECTOR "344".
1342 ;*
1343 ;*****
1344 007704 000004    TST44: SCOPE
1345 007706 106427 000000    MTPS      #0
1346 007712 017746 171500    MOV      @B.BVEC,-(SP)      ;SAVE DVB VECTOR
1347 007716 017746 171476    MOV      @B.BPTY,-(SP)      ;SAVE DVB PRIO
1348 007722 012777 010042 171466    MOV      #5$,@B.BVEC      ;LOAD VECTOR
1349 007730 012777 000200 171462    MOV      #200,@B.BPTY      ;LOAD PRIO
1350 007736 005001          CLR      R1
1351 007740 013700 001410    MOV      DEV.B,R0      ;
1352 007744 062700 000002    ADD      #2,R0      ;MAKE IT SEL 2
1353 007750 052710 000100    1$: BIS      #BIT6,(R0)      ;SET EVENT ENABLE
1354 007754 000240          NOP          ;WASTE TIME
1355 007756 042710 000100    BIC      #BIT6,(R0)      ;CLEAR IT
1356 007762 052710 000200    BIS      #BIT7,(R0)      ;SET OTHER INTR
1357 007766 000240          NOP          ;WASTE TIME
1358 007770 012777 010010 171420    MOV      #2$,@B.BVEC      ;SET GOOD VECTOR
1359 007776 052710 000100    BIS      #BIT6,(R0)      ;SET IE
1360 010002 000240          NOP          ;WASTE TIME

```

1361	010004	104015		ERROR	15	;NO INTERRUPT
1362	010006	000407		BR	4\$	;CONT TEST
1363	010010	005010	2\$:	CLR	(R0)	;ZERO REGISTER
1364	010012	105203		INCB	R3	;UPDATE ICOUNT
1365	010014	100403		BMI	3\$	;DONE?
1366	010016	012716	007750	MOV	#1\$,(SP)	;SET RETURN
1367	010022	000002		RTI		;EXIT
1368	010024	022626	3\$:	CMP	(SP)+,(SP)+	;POP PC+PSW
1369	010026	005010	4\$:	CLR	(R0)	;DSABLE DEVICE
1370	010030	012677	171364	MOV	(SP)+,@B.BPTY	;RESTORE PTY
1371	010034	012677	171356	MOV	(SP)+,@B.BVEC	;RESTORE VEC
1372	010040	000404		BR	6\$	;CONT TEST
1373	010042	010002	5\$:	MOV	R0,R2	;SAVE R0
1374	010044	011600		MOV	(SP),R0	;SAVE PC.
1375	010046	104005		ERROR	5	;UNEXPECTED INTERUPT
1376	010050	010200		MOV	R2,R0	;RESTORE R0
1377	010052	106427	000000	6\$:	MTPS	#0
1378						;ZERO PTY

1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416

```

*****
; *TEST 45      TEST OF 'EVENT' AT DEVICE 'B'
; *TEST THAT SETTING BIT1 OF DEVICE 'B' (SEL 2)
; *CAUSES AN 'EVENT' VECTORING TO ADDRESS 100.
; *
; *
*****

```

```

TST45:  SCOPE
        MTPS      #0          ;ZERO PSW
        MOV       @CLKVEC,-(SP) ;SAVE VEC
        MOV       @CLKPTY,-(SP) ;SAVE PTY
        MOV       DEV.B,R0     ;GET CSR
        ADD       #2,R0        ;MAKE IT SEL 2
        CLR       R3          ;ICOUNTER=0
        MOV       #2$,@CLKVEC  ;SET VECTOR
        MOV       #200,@CLKPTY ;SET PRIO
        1$:      CLR       (R0) ;ZERO REGISTER
        BIS       #BIT1,(R0)   ;SET EVENT
        CLR       (PC)+        ;STALL FOR TIME.
        64$:     0
        ADD       #1,64$       ;INC TIMER
        BNE       ,-6          ;TIMER DONE?
        ERROR     15          ;NO INTERRUPT
        BR        4$          ;CONT
        2$:      INCB      R3   ;ICOUNT=ICOUT+1
        BMI       3$          ;BR IF DONE
        MOV       #1$,(SP)    ;SET RTN
        RTI
        3$:      CMP       (SP)+,(SP)+ ;POP PC+PSW
        4$:      CLR       (R0) ;DSABLE DEVICE
        MOV       (SP)+,@CLKPTY ;RESTORE PTY
        MOV       (SP)+,@CLKVEC ;RESTORE VEC
        MTPS      #0          ;PTY 0

```

```

*****
; *TEST 46      DLV11 CHARACTER TEST

```

```

1417 ;*TEST TO OUTPUT CHARACTERS ON THE DLV11
1418 ;*NOT BEING USED AS THE CONSOLE INTERFACE.
1419 ;*PROGRAM WILL DO 2 FULL BINARY COUNT PATTERNS
1420 ;*AND THEN A SLIDE ASCII PATTERN FOR 5 ROWS.
1421 ;*
1422 ;*****
1423 TST46: SCOPE
1424 010202 000004      MOV      #2,$TIMES      ;;DO 2 ITERATIONS
1425 010204 012737 000002 001162      MOV      #175610,R0      ;SET DEVICE ADDRESS
1426 010212 012700 175610      MOV      #2,R1      ;SET BINARY COUNTER
1427 010216 012701 000002      CLR      R3
1428 010222 005003      CLR      R5      ;CLEAR COUNTER
1429 010224 005005      TST      2(R0)      ;CLR RX DONE
1430 010226 005760 000002      TST      2(R0)
1431 010232 005760 000002      1$: TSTB     4(R0)      ;PRINTER READY?
1432 010236 105760 000004      BMI      64$      ;BR IF YES
1433 010242 100407      CMP      0,0      ;WASTE TIME
1434 010244 023737 000000 000000      ADD      #1,R3      ;DELAY COUNTER+1
1435 010252 062703 000001      BNE      1$      ;DELAY DONE? NO!
1436 010256 001367      ERROR     16      ;TPS BIT7 <>1 (NOT READY)
1437 010260 104016      64$: MOV      R5,6(R0)      ;LOAD DATA CHAR.
1438 010262 110560 000006      CLR      R3      ;CLEAR POINTER
1439 010266 005003      65$: TSTB     (R0)      ;RECEIVER READY(DONE)?
1440 010270 105710      BMI      66$      ;BR IF YES
1441 010272 100407      CMP      0,0      ;WASTE TIME.
1442 010274 023737 000000 000000      ADD      #1,R3
1443 010302 062703 000001      BNE      65$      ;DELAY DONE? NO!
1444 010306 001370      ERROR     16      ;RECEIVER NOT DONE.
1445 010310 104016      66$: MOV      2(R0),R4      ;READ DATA
1446 010312 016004 000002      CMP      R5,R4      ;DATA GOOD?
1447 010316 020504      BEQ      67$      ;BR IF YES
1448 010320 001401      ERROR     17      ;DATA COMPARE ERROR.
1449 010322 104017      67$: INCB     R5      ;NEXT CHAR
1450 010324 105205      BNE      1$
1451 010326 001343      DEC      R1
1452 010330 005301      BNE      1$
1453 010332 001341      TSTB     4(R0)
1454 010334 105760 000004      BPL      -4
1455 010340 100375      MOV      #15,6(R0)
1456 010342 112760 000015 000006      TSTB     4(R0)
1457 010344 100375      BPL      -4
1458 010346 112760 000012 000006      MOV      #12,6(R0)
1459 010348 012705 000005      MOV      #5.,R5
1460 010350 012703 000004      2$: MOV      #40,R3
1461 010352 005203      3$: INC      R3
1462 010354 022703 000176      CMP      #176,R3
1463 010402 001772      BEQ      2$
1464 010404 012701 000110      MOV      #72.,R1
1465 010406 010304      MOV      R3,R4
1466 010408 105760 000004      TSTB     4(R0)
1467 010410 100375      BPL      -4
1468 010412 112760 000015 000006      MOV      #15,6(R0)
1469 010414 105760 000004      TSTB     4(R0)
1470 010416 100375      BPL      -4
1471 010418 112760 000012 000006      MOV      #12,6(R0)
1472 010420 022704 000176      4$: CMP      #176,R4

```

1473	010446	001002		BNE	.+6	:
1474	010450	012704	000040	MOV	#40,R4	:
1475	010454	105760	000004	TSTB	4(R0)	:
1476	010460	100375		BPL	.-4	:
1477	010462	110460	000006	MOVB	R4,6(R0)	:
1478	010466	005204		INC	R4	:
1479	010470	005301		DEC	R1	:
1480	010472	001363		BNE	4\$	:
1481	010474	005305		DEC	R5	:
1482	010476	001336		BNE	3\$	:
1483	010500	105760	000004	TSTB	4(R0)	: TP READY?
1484	010504	100375		BPL	.-4	: BR IF NO
1485	010506	005060	000006	CLR	6(R0)	: XMIT A ZERO.
1486	010512	105760	000004	TSTB	4(R0)	: READY?
1487	010516	100375		BPL	.-4	: BR IF NO.
1488	010520	005060	000006	CLR	6(R0)	: LOAD A ZERO.
1489	010524	105760	000004	TSTB	4(R0)	: READY?
1490	010530	100375		BPL	.-4	: BR IF NO
1491	010532	005005		CLR	R5	:
1492	010534	005205		INC	R5	:
1493	010536	001376		BNE	.-2	:
1494	010540	005760	000002	TST	2(R0)	: CLEAR RX FLAG.

1495						
1496						
1497						
1498						
1499						
1500						
1501						
1502						
1503						
1504	010544	000004		TST47: SCOPE		
1505	010546	012737	000002	MOV	#2,\$TIMES	: DO 2 ITERATIONS
1506	010554	012700	175610	MOV	#175610,R0	: GET DLV11 CSR
1507	010560	013746	000300	MOV	300,-(SP)	: SAVE
1508	010564	013746	000302	MOV	302,-(SP)	: VECTORS
1509	010570	013746	000304	MOV	304,-(SP)	: ON THE
1510	010574	013746	000306	MOV	306,-(SP)	: STACK.
1511	010600	012737	010674	MOV	#4\$,304	: SET TX VECTOR.
1512	010606	012737	000200	MOV	#200,306	: SET PRIO.
1513	010614	012737	010722	MOV	#5\$,300	: SET RX VECTOR
1514	010622	012737	000200	MOV	#200,302	: SET PRIO.
1515	010630	005005		CLR	R5	: SET DATA POINTER TO ZERO
1516	010632	052760	000100	BIS	#100,4(R0)	: SET TX IE.
1517	010640	005003		1\$: CLR	R3	: SET TIMER TO ZERO.
1518	010642	062703	000001	2\$: ADD	#1,R3	: TIME WAITING FOR INTERRUPTS.
1519	010646	001375		BNE	2\$	: KEEP COUNTING.
1520	010650	104016		ERROR	16	: DLV11 INTERRUPT PROBLEM.
1521	010652	012637	000306	3\$: MOV	(SP)+,306	: RESTORE
1522	010656	012637	000304	MOV	(SP)+,304	: ALL
1523	010662	012637	000302	MOV	(SP)+,302	: DLV11
1524	010666	012637	000300	MOV	(SP)+,300	: VECTORS.
1525	010672	000444		BR	TST50	: EXIT TEST
1526						
1527	010674			4\$:		: TRANSMITTER INTERRUPTS TO HERE.
1528	010674	110560	000006	MOVB	R5,6(R0)	: LOAD DATA CHAR.

```

1529 010700 042760 000100 000004 BIC #100,4(R0) ;DSABLE TX INTERRUPTS.
1530 010706 052760 000100 000000 BIS #100,0(R0) ;ENABLE RX INTERRUPTS.
1531 010714 012716 010640 MOV #1$, (SP) ;SET RETURN
1532 010720 000002 RTI ;EXIT I'R.
1533
1534 010722 5$: ;RECEIVER INTERRUPTS TO HERE.
1535 010722 000240 NOP ;
1536 010724 116004 000002 MOVB 2(R0),R4 ;GET DATA REVEIVERD
1537 010730 020504 CMP R5,R4 ;DATA GOOD?
1538 010732 001401 BEQ 6$ ;BE IF GOOD DATA
1539 010734 104017 ERROR 17 ;DLV11 DATA ERROR (INTERUPTS)
1540 010736 005205 6$: INC R5 ;UPDATE DATA CHAR.
1541 010740 022705 000040 CMP #32.,R5 ;ALL CHARS DONE?
1542 010744 001006 BNE 7$ ;BR IF NO
1543 010746 042760 000100 000000 BIC #100,0(R0) ;DSABLE RX INTERRUPTS.
1544 010754 012716 010652 MOV #3$, (SP) ;SET RETURN
1545 010760 000002 RTI ;RETURN
1546 010762 042760 000100 000000 7$: BIC #100,0(R0) ;DSABLE RX INTERRUPTS
1547 010770 052760 000100 000004 BIS #100,4(R0) ;ENABLE TX INTERRUPTS
1548 010776 012716 010640 MOV #1$, (SP) ;SET RETURN
1549 011002 000002 RTI
1550
1551
1552 ;*****
1553 ;*TEST 50 MINI MEMORY TEST
1554 ;*MINI MEMORY TESTS
1555 ;*ALL UNSED MEMORY WILL BE FILLED WITH ALL 1'S
1556 ;*AND THEN EACH ADDRESS WILL BE TESTED
1557 ;*WITH A FLOATING ZERO.
1558 ;*
1559 ;*****
1560 011004 000004 TST50: SCOPE
1561 011006 012737 000002 001162 MOV #2,$TIMES ;DO 2 ITERATIONS
1562 011014 013701 001376 MOV MEMSIZ,R1 ;GET MAX MEMORY SIZE.
1563 011020 162701 000310 SUB #310,R1 ;PROTECT ABL
1564 011024 012700 015244 MOV #CORMAX,R0 ;GET LAST ADDRESS USED BY PROGRAM
1565 011030 012720 177777 1$: MOV #177777,(R0)+ ;
1566 011034 020100 CMP R1,R0 ;ALL MEMORY FILLED?
1567 011036 001374 BNE 1$ ;
1568 011040 012700 015244 MOV #CORMAX,R0 ;
1569 011044 042710 000001 2$: BIC #BIT0,(R0) ;
1570 011050 012705 177776 MOV #C<1>,R5 ;
1571 011054 011004 3$: MOV (R0),R4 ;
1572 011056 020504 CMP R5,R4 ;
1573 011060 001401 BEQ 4$ ;
1574 011062 104020 ERROR 20 ;
1575 011064 000261 4$: SEC ;
1576 011066 006105 ROL R5 ;
1577 011070 000261 SEC ;
1578 011072 006110 ROL (R0) ;
1579 011074 103767 BCS 3$ ;
1580 011076 005720 TST (R0)+ ;
1581 011100 100401 BMI .+4 ;
1582 011102 104016 ERROR 16 ;GENERAL ERROR. BIT15<>1!
1583 011104 020100 CMP R1,R0 ;ALL MEMORY DONE?
1584 011106 001356 BNE 2$ ;

```

CWQAABO 11W03 SYSTEM TEST MACY11 30A(1052) 23-MAR-78 07:51 M 3
CWQAAB.P11 22-MAR-78 14:51 T50 MINI MEMORY TEST PAGE 32

SEQ 0038

1585

```

1586 .SBTTL END OF PASS ROUTINE
1587
1588 ;*****
1589 ;*INCREMENT THE PASS NUMBER ($PASS)
1590 ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
1591 ;*IF THERES A MONITOR GO TO IT
1592 ;*IF THERE ISN'T JUMP TO TST1
1593
1594 $EOP:
1595 011110 000004          SCOPE
1596 011112 005037 001102 CLR $TSTNM          ;;ZERO THE TEST NUMBER
1597 011116 005037 001162 CLR $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
1598 011122 005237 001100 INC $PASS          ;;INCREMENT THE PASS NUMBER
1599 011126 042737 100000 001100 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
1600 011134 005327          DEC (PC)+          ;;LOOP?
1601 011136 000001 $EOPCT: .WORD 1
1602 011140 003022          BGT $DOAGN          ;;YES
1603 011142 012737          MOV (PC)+,a(PC)+    ;;RESTORE COUNTER
1604 011144 000001 $ENDCT: .WORD 1
1605 011146 011136          $EOPCT
1606 011150 104401 011215 TYPE ,SENDMG          ;;TYPE 'END PASS #'
1607 011154 013746 001100 MOV $PASS,-(SP)      ;;SAVE $PASS FOR TYPEOUT
1608 011160 104405          TYPDS              ;;GO TYPE--DECIMAL ASCII WITH SIGN
1609 011162 104401 011212 TYPE ,SENUL          ;;TYPE A NULL CHARACTER
1610 011166 013700 000042 $GET42: MOV a#42,R0      ;;GET MONITOR ADDRESS
1611 011172 001405          BEQ $DOAGN          ;;BRANCH IF NO MONITOR
1612 011174 000005          RESET              ;;CLEAR THE WORLD
1613 011176 004710 $ENDAD: JSR PC,(R0)          ;;GO TO MONITOR
1614 011200 000240          NOP                ;;SAVE ROOM
1615 011202 000240          NOP                ;;FOR
1616 011204 000240          NOP                ;;ACT11
1617 011206          $DOAGN:
1618 011206 000137          JMP a(PC)+          ;;RETURN
1619 011210 004432 $RTNAD: .WORD TST1
1620 011212 377 377 000 $ENULL: .BYTE -1,-1,0    ;;NULL CHARACTER STRING
1621 011215 015 042412 042116 $ENDMG: .ASCIIZ <15><12>/END PASS #/
1622 011222 050040 051501 020123
1623 011230 000043
1624 .SBTTL SCOPE HANDLER ROUTINE
1625
1626 ;*****
1627 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1628 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1629 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1630 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1631 ;*SW14=1 LOOP ON TEST
1632 ;*SW11=1 INHIBIT ITERATIONS
1633 ;*SW09=1 LOOP ON ERROR
1634 ;*CALL
1635 ;* SCOPE ;;SCOPE=IOT
1636
1637 $SCOPE:
1638 011232 032777 000400 167700 BIT #BIT8,aSWR      ;;CONFIDENCE?
1639 011240 001425          BEQ 99$              ;;BR IF NO
1640 011242 123737 001102 001442 CMPB $TSTNM,XXSCOP ;;WAS THIS TEST ALREADY TYPED OUT?
1641 011250 001421          BEQ 99$              ;;BR IF YES

```

```

1642 011252 123737 001102 015242      CMPB   $TSTNM, LASTN    ;LEGAL TEST #
1643 011260 003015                      BGT     99$              ;BR IF NO
1644 011262 105737 001102                      TSTB   $TSTNM          ;TEST 0?
1645 011266 001412                      BEQ     99$              ;BR IF YES (THERE IS NO TEST 0)
1646 011270 113737 001102 001442      MOVB   $TSTNM, XXSCOP   ;LOAD FOR NEXT TIME
1647 011276 104401 014746                      TYPE   ,XTST           ;TYPE 'T '
1648 011302 013746 001442                      MOV     XXSCOP, -(SP)   ;
1649 011306 104402                      TYPOC                      ;
1650 011310 104401 014712                      TYPE   ,XSPA          ;
1651 011314                      99$:
1652 011314 032777 040000 167016 1$:   BIT     #BIT14, $SWR    ;:LOOP ON PRESENT TEST?
1653 011322 001101                      BNE     $OVER          ;:YES IF SW14=1
1654                      ;#####START OF CODE FOR THE XOR TESTER#####
1655 011324 000416      $XTSTR: BR     6$              ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
1656                      ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
1657 011326 013746 000004                      MOV     @#ERRVEC, -(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
1658 011332 012737 011352 000004      MOV     #5$, @#ERRVEC ;:SET FOR TIMEOUT
1659 011340 005737 177060                      TST     @#177060       ;:TIME OUT ON XOR?
1660 011344 012637 000004                      MOV     (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
1661 011350 000453                      BR      $SVLAD         ;:GO TO THE NEXT TEST
1662 011352 022626                      5$:   CMP     (SP)+, (SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
1663 011354 012637 000004                      MOV     (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
1664 011360 000413                      BR      7$            ;:LOOP ON THE PRESENT TEST
1665 011362                      6$:;#####END OF CODE FOR THE XOR TESTER#####
1666 011362 105737 001103      2$:   TSTB   $ERFLG          ;:HAS AN ERROR OCCURRED?
1667 011366 001421                      BEQ     3$            ;:BR IF NO
1668 011370 123737 001115 001103      CMPB   $ERMAX, $ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
1669 011376 101015                      BHI     3$            ;:BR IF NO
1670 011400 032777 001000 167532      BIT     #BIT09, $SWR   ;:LOOP ON ERROR?
1671 011406 001404                      BEQ     4$            ;:BR IF NO
1672 011410 013737 001110 001106 7$:   MOV     $LPERR, $LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
1673 011416 000443                      BR      $OVER          ;
1674 011420 105037 001103      4$:   CLRB   $ERFLG          ;:ZERO THE ERROR FLAG
1675 011424 005037 001162                      CLR     $TIMES        ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
1676 011430 000415                      BR      1$            ;:ESCAPE TO THE NEXT TEST
1677 011432 032777 004000 167500 3$:   BIT     #BIT11, $SWR   ;:INHIBIT ITERATIONS?
1678 011440 001011                      BNE     1$            ;:BR IF YES
1679 011442 005737 001100                      TST     $PASS         ;:IF FIRST PASS OF PROGRAM
1680 011446 001406                      BEQ     1$            ;:INHIBIT ITERATIONS
1681 011450 005237 001104                      INC     $ICNT         ;:INCREMENT ITERATION COUNT
1682 011454 023737 001162 001104      CMP     $TIMES, $ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
1683 011462 002021                      BGE     $OVER          ;:BR IF MORE ITERATION REQUIRED
1684 011464 012737 000001 001104 1$:   MOV     #1, $ICNT     ;:REINITIALIZE THE ITERATION COUNTER
1685 011472 013737 011542 001162      MOV     $MXCNT, $TIMES ;:SET NUMBER OF ITERATIONS TO DO
1686 011500 105237 001102      $SVLAD: INCB   $TSTNM ;:COUNT TEST NUMBERS
1687 011504 011637 001106                      MOV     (SP), $LPADR   ;:SAVE SCOPE LOOP ADDRESS
1688 011510 011637 001110                      MOV     (SP), $LPERR   ;:SAVE ERROR LOOP ADDRESS
1689 011514 005037 001164                      CLR     $ESCAPE        ;:CLEAR THE ESCAPE FROM ERROR ADDRESS
1690 011520 112737 000001 001115      MOVB   #1, $ERMAX     ;:ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1691 011526 013777 001102 167406 $OVER: MOV     $TSTNM, @DISPLAY ;:DISPLAY TEST NUMBER
1692 011534 013716 001106                      MOV     $LPADR, (SP)   ;:FUDGE RETURN ADDRESS
1693 011540 000002                      RTI                    ;:FIXES PS
1694 011542 000100      $MXCNT: 100         ;:MAX. NUMBER OF ITERATIONS
1695                      .SBTTL  ERROR HANDLER ROUTINE
1696
1697                      ;:*****

```

```

1698      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1699      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1700      ;*AND GO TO XERR1 ON ERROR
1701      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1702      ;*SW15=1      HALT ON ERROR
1703      ;*SW13=1      INHIBIT ERROR TYPEOUTS
1704      ;*SW10=1      BELL ON ERROR
1705      ;*SW09=1      LOOP ON ERROR
1706      ;*CALL
1707      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1708
1709      $ERROR:
1710      011544      105237      001103      7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
1711      011550      001775      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
1712      011552      013777      001102      167362      MOV      $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
1713      011560      032777      002000      167352      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
1714      011566      001402      BEQ      1$      ;;NO - SKIP
1715      011570      104401      001166      TYPE      , $BELL      ;;RING BELL
1716      011574      005237      001112      1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
1717      011600      011637      001116      MOV      (SP), $ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
1718      011604      162737      000002      001116      SUB      #2, $ERRPC
1719      011612      117737      167300      001114      MOVB      @ $ERRPC, $ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
1720      011620      032777      020000      167312      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
1721      011626      001004      BNE      20$      ;;SKIP TYPEOUTS
1722      011630      004737      012042      JSR      PC, XERR1      ;;GO TO USER ERROR ROUTINE
1723      011634      104401      001173      TYPE      , $CRLF
1724      011640      20$:
1725      011640      005777      167274      2$:      TST      @SWR      ;;HALT ON ERROR
1726      011644      100001      BPL      3$      ;;SKIP IF CONTINUE
1727      011646      000000      HALT      ;;HALT ON ERROR!
1728      011650      032777      001000      167262      3$:      BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
1729      011656      001402      BEQ      4$      ;;BR IF NO
1730      011660      013716      001110      MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
1731      011664      005737      001164      4$:      TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
1732      011670      001402      BEQ      5$      ;;BR IF NONE
1733      011672      013716      001164      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
1734      011676      5$:
1735      011676      022737      011176      000042      CMP      # $ENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
1736      011704      001001      BNE      6$      ;;BRANCH IF NO
1737      011706      000000      HALT      ;;YES
1738      011710      6$:
1739      011710      010046      MOV      R0,-(SP)      ;SAVE R0
1740      011712      010146      MOV      R1,-(SP)      ;SAVE R1
1741      011714      032777      010000      167216      BIT      #BIT12,@SWR      ;ESCAPE TO NEXT TEST ON ERROR?
1742      011722      001443      BEQ      90$      ;BR IF NO
1743      011724      000005      RESET      ;ISSUE INIT
1744      011726      052777      000100      167210      BIS      #100,@ $TKS      ;SET INTR ENABLE
1745      011734      106427      000000      MTPS      #0      ;SET PTY TO 0
1746      011740      012700      015116      MOV      #TEST.TABLE+2,R0      ;GET FIRST TEST PC
1747      011744      013701      001106      MOV      $LPADR,R1      ;GET CURRENT TEST PC
1748      011750      162701      000002      SUB      #2,R1      ;MAKE IT EQUAL TO TEST TABLE ENTRY
1749      011754      005710      91$:      TST      (R0)      ;AT END GOTO 90$
1750      011756      001425      BEQ      90$      ;
1751      011760      020120      CMP      R1,(R0)+      ;LOOK FOR CURRENT TEST.
1752      011762      001374      BNE      91$      ;BR IF NOT FOUND
1753      011764      011037      012040      MOV      (R0),92$      ;SAVE TEST PC

```

```

1754 011770 062737 000002 012040      ADD      #2,92$      ;POP PAST SCOPE
1755 011776 012737 000001 001104      MOV      #1,$ICNT      ;
1756 012004 013737 011542 001162      MOV      $MXCNT,$TIMES      ;
1757 012012 012706 001100      MOV      #STACK,SP      ;SET SP
1758 012016 012746 000000      MOV      #0,-(SP)      ;SET PTYO
1759 012022 013746 012040      MOV      92$,-(SP)      ;
1760 012026 000137 011500      JMP      $SVLAD      ;GO INTO SCOPE ROUTINE
1761 012032 012601      90$: MOV      (SP)+,R1      ;RESTORE R1
1762 012034 012600      MOV      (SP)+,R0      ;RESTORE R0
1763 012036 000002      RTI      ;EXIT
1764 012040 000000      92$: 0      ;SAVE TEST PC
1765
1766 012042 010037 001422      XERR1: MOV      R0,SAVR0
1767 012046 010137 001424      MOV      R1,SAVR1
1768 012052 010237 001426      MOV      R2,SAVR2
1769 012056 010337 001430      MOV      R3,SAVR3
1770 012062 010437 001432      MOV      R4,SAVR4
1771 012066 010537 001434      MOV      R5,SAVR5
1772 012072 113737 001102 001436      MOVB     $TSTNM,MSKTST
1773      .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
1774
1775      ;*****
1776      ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
1777      ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
1778      ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
1779
1780      $ERRTYP:
1781 012100 104401 001173      TYPE      , $CRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
1782 012104 010046      MOV      R0,-(SP)      ;:SAVE R0
1783 012106 005000      CLR      R0      ;:PICKUP THE ITEM INDEX
1784 012110 153700 001114      BISB     @#$ITEMB,R0
1785 012114 001004      BNE      1$      ;:IF ITEM NUMBER IS ZERO, JUST
1786      ;:TYPE THE PC OF THE ERROR
1787 012116 013746 001116      MOV      $ERRPC,-(SP)      ;:SAVE $ERRPC FOR TYPEOUT
1788      ;:ERROR ADDRESS
1789      ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1790 012122 104402      TYP0C
1791 012124 000445      BR      10$      ;:GET OUT
1792 012126 005300      1$: DEC      R0      ;:ADJUST THE INDEX SO THAT IT WILL
1793 012130 006300      ASL      R0      ;:
1794 012132 006300      ASL      R0      ;:
1795 012134 006300      ASL      R0      ;:
1796 012136 062700 001176      ADD      #$ERRTB,R0      ;:FORM TABLE POINTER
1797 012142 012037 012152      MOV      (R0)+,2$      ;:PICKUP "ERROR MESSAGE" POINTER
1798 012146 001404      BEQ      3$      ;:SKIP TYPEOUT IF NO POINTER
1799 012150 104401      TYPE      ;:TYPE THE "ERROR MESSAGE"
1800 012152 000000      2$: .WORD    0      ;: "ERROR MESSAGE" POINTER GOES HERE
1801 012154 104401 001173      TYPE      , $CRLF      ;:;"CARRIAGE RETURN" & "LINE FEED"
1802 012160 012037 012170      3$: MOV      (R0)+,4$      ;:PICKUP "DATA HEADER" POINTER
1803 012164 001404      BEQ      5$      ;:SKIP TYPEOUT IF 0
1804 012166 104401      TYPE      ;:TYPE THE "DATA HEADER"
1805 012170 000000      4$: .WORD    0      ;: "DATA HEADER" POINTER GOES HERE
1806 012172 104401 001173      TYPE      , $CRLF      ;:;"CARRIAGE RETURN" & "LINE FEED"
1807 012176 010146      5$: MOV      R1,-(SP)      ;:SAVE R1
1808 012200 012001      MOV      (R0)+,R1      ;:PICKUP "DATA TABLE" POINTER
1809 012202 001415      BEQ      9$      ;:BR IF NO DATA TO BE TYPED
1809 012204 012000      MOV      (R0)+,R0      ;:PICKUP "DATA FORMAT" POINTER

```

1810	012206	105720		6\$:	TSTB	(R0)+	::'OCTAL' OR 'DECIMAL'
1811	012210	001003			BNE	7\$	::BR IF DECIMAL
1812	012212	013146			MOV	@(R1)+,-(SP)	::SAVE @(R1)+ FOR TYPEOUT
1813	012214	104402			TPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1814	012216	000402			BR	8\$	
1815	012220			7\$:			
1816	012220	013146			MOV	@(R1)+,-(SP)	::SAVE @(R1)+ FOR TYPEOUT
1817	012222	104405			TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
1818	012224	005711		8\$:	TST	(R1)	::IS THERE ANOTHER NUMBER?
1819	012226	001403			BEQ	9\$	::BR IF NO
1820	012230	104401	012250		TYPE	,11\$	::TYPE TWO(2) SPACES
1821	012234	000764			BR	6\$	::LOOP
1822							
1823	012236	012601		9\$:	MOV	(SP)+,R1	::RESTORE R1
1824	012240	012600		10\$:	MOV	(SP)+,R0	::RESTORE R0
1825	012242	104401	001173		TYPE	,SCRLF	::'CARRIAGE RETURN' & 'LINE FEED'
1826	012246	000207			RTS	PC	::RETURN
1827	012250	020040	000	11\$:	.ASCIZ	/ /	::TWO(2) SPACES
1828		012254			.EVEN		
1829					.SBTTL	TYPE ROUTINE	
1830							
1831							
1832							
1833							
1834							
1835							
1836							
1837							
1838							
1839							
1840							
1841							
1842							
1843							
1844							
1845							
1846	012254	105737	001157	\$TYPE:	TSTB	\$TPFLG	::IS THERE A TERMINAL?
1847	012260	100002			BPL	1\$	::BR IF YES
1848	012262	000000			HALT		::HALT HERE IF NO TERMINAL
1849	012264	000407			BR	3\$	::LEAVE
1850	012266	010046		1\$:	MOV	R0,-(SP)	::SAVE R0
1851	012270	017600	000002		MOV	@2(SP),R0	::GET ADDRESS OF ASCIZ STRING
1852	012274	112046		2\$:	MOVB	(R0)+,-(SP)	::PUSH CHARACTER TO BE TYPED ONTO STACK
1853	012276	001005			BNE	4\$	::BR IF IT ISN'T THE TERMINATOR
1854	012300	005726			TST	(SP)+	::IF TERMINATOR POP IT OFF THE STACK
1855	012302	012600		60\$:	MOV	(SP)+,R0	::RESTORE R0
1856	012304	062716	000002	3\$:	ADD	#2,(SP)	::ADJUST RETURN PC
1857	012310	000002			RTI		::RETURN
1858	012312	122716	000011	4\$:	CMPB	#HT,(SP)	::BRANCH IF <HT>
1859	012316	001430			BEQ	8\$	
1860	012320	122716	000200		CMPB	#CRLF,(SP)	::BRANCH IF NOT <CRLF>
1861	012324	001006			BNE	5\$	
1862	012326	005726			TST	(SP)+	::POP <CR><LF> EQUIV
1863	012330	104401			TYPE		::TYPE A CR AND LF
1864	012332	001173			\$CRLF		
1865	012334	105037	012470		CLRB	\$CHARCNT	::CLEAR CHARACTER COUNT

```

*****
*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
*

```

```

1866 012340 000755          BR      2$          ;;GET NEXT CHARACTER
1867 012342 004737 012424    5$:    JSR      PC,$TYPEC      ;;GO TYPE THIS CHARACTER
1868 012346 123726 001156    6$:    CMPB     $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1869 012352 001350          BNE      2$          ;;IF NO GO GET NEXT CHAR.
1870 012354 013746 001154          MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
1871                                ;;AND THE NULL CHAR.
1872 012360 105366 000001    7$:    DECB     1(SP)          ;;DOES A NULL NEED TO BE TYPED?
1873 012364 002770          BLT      6$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
1874 012366 004737 012424          JSR      PC,$TYPEC      ;;GO TYPE A NULL
1875 012372 105337 012470          DECB     $CHARCNT        ;;DO NOT COUNT AS A COUNT
1876 012376 000770          BR      7$          ;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

1880 012400 112716 000040    8$:    MOVB     #' ,(SP)          ;;REPLACE TAB WITH SPACE
1881 012404 004737 012424    9$:    JSR      PC,$TYPEC      ;;TYPE A SPACE
1882 012410 132737 000007 012470    BITB     #7,$CHARCNT      ;;BRANCH IF NOT AT
1883 012416 001372          BNE      9$          ;;TAB STOP
1884 012420 005726          TST      (SP)+          ;;POP SPACE OFF STACK
1885 012422 000724          BR      2$          ;;GET NEXT CHARACTER
1886 012424 105777 166520    $TYPEC: TSTB     @STPS          ;;WAIT UNTIL PRINTER IS READY
1887 012430 100375          BPL      $TYPEC
1888 012432 116677 000002 166512    MOVB     2(SP),@STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1889 012440 122766 000015 000002    CMPB     #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
1890 012446 001003          BNE      1$          ;;BRANCH IF NO
1891 012450 105037 012470          CLRB     $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1892 012454 000406          BR      $TYPEX
1893 012456 122766 000012 000002    1$:    CMPB     #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
1894 012464 001402          BEQ      $TYPEX      ;;BRANCH IF YES
1895 012466 105227          INCB     (PC)+          ;;COUNT THE CHARACTER
1896 012470 000000    $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
1897 012472 000207    $TYPEX: RTS      PC

```

.SRTTL BINARY TO OCTAL (ASCII) AND TYPE

```

1898
1899
1900
1901    ;*****
1902    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1903    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1904    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1905    ;*CALL:
1906    ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
1907    ;*      TYPOS          ;;CALL FOR TYPEOUT
1908    ;*      .BYTE     N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1909    ;*      .BYTE     M          ;;M=1 OR 0
1910    ;*                                  ;;1=TYPE LEADING ZEROS
1911    ;*                                  ;;0=SUPPRESS LEADING ZEROS
1912    ;*
1913    ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1914    ;*$TYPOS OR $TYPOC
1915    ;*CALL:
1916    ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
1917    ;*      TYPON          ;;CALL FOR TYPEOUT
1918    ;*
1919    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1920    ;*CALL:
1921    ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED

```

```

1922      ;*      TYPOC      ;;CALL FOR TYPEOUT
1923
1924 012474 017646 000000      $TYPOS: MOV 2(SP),-(SP)      ;;PICKUP THE MODE
1925 012500 116637 000001 012717      MOV 1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
1926 012506 112637 012721      MOV (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
1927 012512 062716 000002      ADD #2,(SP)      ;;ADJUST RETURN ADDRESS
1928 012516 000406      BR $TYPON
1929 012520 112737 000001 012717 $TYPOC: MOV #1,$OFILL      ;;SET THE ZERO FILL SWITCH
1930 012526 112737 000006 012721      MOV #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
1931 012534 112737 000005 012716 $TYPON: MOV #5,$OCNT      ;;SET THE ITERATION COUNT
1932 012542 010346      MOV R3,-(SP)      ;;SAVE R3
1933 012544 010446      MOV R4,-(SP)      ;;SAVE R4
1934 012546 010546      MOV R5,-(SP)      ;;SAVE R5
1935 012550 113704 012721      MOV $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
1936 012554 005404      NEG R4
1937 012556 062704 000006      ADD #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
1938 012562 110437 012720      MOV R4,$OMODE      ;;SAVE IT FOR USE
1939 012566 113704 012717      MOV $OFILL,R4      ;;GET THE ZERO FILL SWITCH
1940 012572 016605 000012      MOV 12(SP),R5      ;;PICKUP THE INPUT NUMBER
1941 012576 005003      CLR R3      ;;CLEAR THE OUTPUT WORD
1942 012600 006105      1$: ROL R5      ;;ROTATE MSB INTO 'C'
1943 012602 000404      BR 3$      ;;GO DO MSB
1944 012604 006105      2$: ROL R5      ;;FORM THIS DIGIT
1945 012606 006105      ROL R5
1946 012610 006105      ROL R5
1947 012612 010503      MOV R5,R3
1948 012614 006103      3$: ROL R3      ;;GET LSB OF THIS DIGIT
1949 012616 105337 012720      DECB $OMODE      ;;TYPE THIS DIGIT?
1950 012622 100016      BPL 7$      ;;BR IF NO
1951 012624 042703 177770      BIC #177770,R3      ;;GET RID OF JUNK
1952 012630 001002      BNE 4$      ;;TEST FOR 0
1953 012632 005704      TST R4      ;;SUPPRESS THIS 0?
1954 012634 001403      BEQ 5$      ;;BR IF YES
1955 012636 005204      4$: INC R4      ;;DON'T SUPPRESS ANYMORE 0'S
1956 012640 052703 000060      BIS #'0,R3      ;;MAKE THIS DIGIT ASCII
1957 012644 052703 000040      5$: BIS #' ,R3      ;;MAKE ASCII IF NOT ALREADY
1958 012650 110337 012714      MOV R3,8$      ;;SAVE FOR TYPING
1959 012654 104401 012714      TYPE ,8$      ;;GO TYPE THIS DIGIT
1960 012660 105337 012716      7$: DECB $OCNT      ;;COUNT BY 1
1961 012664 003347      BGT 2$      ;;BR IF MORE TO DO
1962 012666 002402      BLT 6$      ;;BR IF DONE
1963 012670 005204      INC R4      ;;INSURE LAST D.GIT ISN'T A BLANK
1964 012672 000744      BR 2$      ;;GO DO THE LAST DIGIT
1965 012674 012605      6$: MOV (SP)+,R5      ;;RESTORE R5
1966 012676 012604      MOV (SP)+,R4      ;;RESTORE R4
1967 012700 012603      MOV (SP)+,R3      ;;RESTORE R3
1968 012702 016666 000002 000004      MOV 2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
1969 012710 012616      MOV (SP)+,(SP)
1970 012712 000002      RTI      ;;RETURN
1971 012714      8$: .BYTE 0      ;;STORAGE FOR ASCII DIGIT
1972 012715      .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
1973 012716      000      $OCNT: .BYTE 0      ;;OCTAL DIGIT COUNTER
1974 012717      000      $OFILL: .BYTE 0      ;;ZERO FILL SWITCH
1975 012720 000000      000      $OMODE: .WORD 0      ;;NUMBER OF DIGITS TO TYPE
1976      .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1977

```

```

1978      ;*****
1979      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1980      ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1981      ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1982      ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1983      ;*REPLACED WITH SPACES.
1984      ;*CALL:
1985      ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
1986      ;*      TYPDS      ;;GO TO THE ROUTINE
1987
1988      $TYPDS:
1989      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
1990      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
1991      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
1992      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
1993      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
1994      MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
1995      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
1996      BPL      1$           ;;BR IF INPUT IS POS.
1997      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
1998      MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1999      CLR      R0           ;;ZERO THE CONSTANTS INDEX
2000      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
2001      MOVB     #'',(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2002      CLR      R2           ;;CLEAR THE BCD NUMBER
2003      MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
2004      SUB      R1,R5         ;;FORM THIS BCD DIGIT
2005      BLT      4$           ;;BR IF DONE
2006      INC      R2           ;;INCREASE THE BCD DIGIT BY 1
2007      BR      3$
2008      ADD      R1,R5         ;;ADD BACK THE CONSTANT
2009      TST      R2           ;;CHECK IF BCD DIGIT=0
2010      BNE      5$           ;;FALL THROUGH IF 0
2011      TSTB     (SP)         ;;STILL DOING LEADING 0'S?
2012      BMI      7$           ;;BR IF YES
2013      ASLB     (SP)         ;;MSD?
2014      BCC      6$           ;;BR IF NO
2015      MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN
2016      BIS      #'0,R2        ;;MAKE THE BCD DIGIT ASCII
2017      BIS      #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2018      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2019      TST      (R0)+         ;;JUST INCREMENTING
2020      CMP      R0,#10        ;;CHECK THE TABLE INDEX
2021      BLT      2$           ;;GO DO THE NEXT DIGIT
2022      BGT      8$           ;;GO TO EXIT
2023      MOV      R5,R2         ;;GET THE LSD
2024      BR      6$           ;;GO CHANGE TO ASCII
2025      TSTB     (SP)+         ;;WAS THE LSD THE FIRST NON-ZERO?
2026      BPL      9$           ;;BR IF NO
2027      MOVB     -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
2028      CLRB     (R3)          ;;SET THE TERMINATOR
2029      MOV      (SP)+,R5       ;;POP STACK INTO R5
2030      MOV      (SP)+,R3       ;;POP STACK INTO R3
2031      MOV      (SP)+,R2       ;;POP STACK INTO R2
2032      MOV      (SP)+,R1       ;;POP STACK INTO R1
2033      MOV      (SP)+,R0       ;;POP STACK INTO R0

```

2034	013110	104401	013136			TYPE	\$DBLK		::NOW TYPE THE NUMBER
2035	013114	016666	000002	000004		MOV	2(SP),4(SP)		::ADJUST THE STACK
2036	013122	012616				MOV	(SP)+,(SP)		
2037	013124	000002				RTI			::RETURN TO USER
2038	013126	023420			\$DTBL:	10000.			
2039	013130	001750				1000.			
2040	013132	000144				100.			
2041	013134	000012				10.			
2042	013136	000004			\$DBLK:	.BLKW 4			
2043					.SBTTL	TTY INPUT ROUTINE			
2044									
2045					::*****				
2046					.ENABL	LSB			
2047									
2048					.DSABL	LSB			
2049									
2050					::*****				
2051					::*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY				
2052					::*CALL:				
2053					::*	RDCHR			::INPUT A SINGLE CHARACTER FROM THE TTY
2054					::*	RETURN HERE			::CHARACTER IS ON THE STACK
2055					::*				::WITH PARITY BIT STRIPPED OFF
2056					::*				
2057					::*				
2058									
2059	013146	011646			\$RDCHR:	MOV	(SP),-(SP)		::PUSH DOWN THE PC
2060	013150	016666	000004	000002		MOV	4(SP),2(SP)		::SAVE THE PS
2061	013156	105777	165762		1\$:	TSTB	@\$TKS		::WAIT FOR
2062	013162	100375				BPL	1\$		::A CHARACTER
2063	013164	117766	165756	000004		MOVB	@\$TKB,4(SP)		::READ THE TTY
2064	013172	042766	177600	000004		BIC	# C<177>,4(SP)		::GET RID OF JUNK IF ANY
2065	013200	026627	000004	000023		CMP	4(SP),#23		::IS IT A CONTROL-S?
2066	013206	001013				BNE	3\$		::BRANCH IF NO
2067	013210	105777	165730		2\$:	TSTB	@\$TKS		::WAIT FOR A CHARACTER
2068	013214	100375				BPL	2\$		::LOOP UNTIL ITS THERE
2069	013216	117746	165724			MOVB	@\$TKB,-(SP)		::GET CHARACTER
2070	013222	042716	177600			BIC	# C177,(SP)		::MAKE IT 7-BIT ASCII
2071	013226	022627	000021			CMP	(SP)+,#21		::IS IT A CONTROL-Q?
2072	013232	001366				BNE	2\$		::IF NOT DISCARD IT
2073	013234	000750				BR	1\$		::YES, RESUME
2074	013236	026627	000004	000140	3\$:	CMP	4(SP),#140		::IS IT UPPER CASE?
2075	013244	002407				BLT	4\$		::BRANCH IF YES
2076	013246	026627	000004	000175		CMP	4(SP),#175		::IS IT A SPECIAL CHAR?
2077	013254	003003				BGT	4\$		::BRANCH IF YES
2078	013256	042766	000040	000004		BIC	#40,4(SP)		::MAKE IT UPPER CASE
2079	013264	000002			4\$:	RTI			::GO BACK TO USER
2080					::*****				
2081					::*THIS ROUTINE WILL INPUT A STRING FROM THE TTY				
2082					::*CALL:				
2083					::*	RDLIN			::INPUT A STRING FROM THE TTY
2084					::*	RETURN HERE			::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2085					::*				::TERMINATOR WILL BE A BYTE OF ALL 0'S
2086									
2087	013266	010346			\$RDLIN:	MOV	R3,-(SP)		::SAVE R3
2088	013270	012703	013374		1\$:	MOV	#\$TTYIN,R3		::GET ADDRESS
2089	013274	022703	013404		2\$:	CMP	#\$TTYIN+8.,R3		::BUFFER FULL?

```

2090 013300 101405      BLOS      4$      ;;BR IF YES
2091 013302 104406      RDCHR      ;;GO READ ONE CHARACTER FROM THE TTY
2092 013304 112613      MOV      (SP)+,(R3)  ;;GET CHARACTER
2093 013306 122713 000177 10$: CMPB    #177,(R3)  ;;IS IT A RUBOUT
2094 013312 001003      BNE      3$      ;;SKIP IF NOT
2095 013314 104401 001172 4$:  TYPE    ,SQUES  ;;TYPE A '?'
2096 013320 000763      BR      1$      ;;CLEAR THE BUFFER AND LOOP
2097 013322 111337 013372 3$:  MOV      (R3),9$  ;;ECHO THE CHARACTER
2098 013326 104401 013372      TYPE    ,9$
2099 013332 122723 000015      CMPB    #15,(R3)+  ;;CHECK FOR RETURN
2100 013336 001356      BNE      2$      ;;LOOP IF NOT RETURN
2101 013340 105063 177777      CLRB    -1(R3)  ;;CLEAR RETURN (THE 15)
2102 013344 104401 001174      TYPE    ,SLF  ;;TYPE A LINE FEED
2103 013350 012603      MOV      (SP)+,R3  ;;RESTORE R3
2104 013352 011646      MOV      (SP),-(SP)  ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2105 013354 016666 000004 000002 MOV      4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
2106 013362 012766 013374 000004 MOV      #TTYIN,4(SP)
2107 013370 000002      RTI      ;;RETURN
2108 013372 000      9$:  .BYTE    0      ;;STORAGE FOR ASCII CHAR. TO TYPE
2109 013373 000      .BYTE    0      ;;TERMINATOR
2110 013374 000010      $TTYIN: .BLKB    8.  ;;RESERVE 8 BYTES FOR TTY INPUT
2111 013404 052536 005015 000      $CNTLU: .ASCIZ / U/<15><12>  ;;CONTROL 'U'
2112 013411 136 006507 000012      $CNTLG: .ASCIZ / G/<15><12>  ;;CONTROL 'G'
2113 013416 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
2114 013424 020075 000
2115 013427 040 047040 053505 $MNEW: .ASCIZ / NEW = /
2116 013434 036440 000040
2117      .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2118
2119      ;;*****
2120      ;;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2121      ;;*CHANGE IT TO BINARY.
2122      ;;*CALL:
2123      ;;*      RDOCT      ;;READ AN OCTAL NUMBER
2124      ;;*      RETURN HERE  ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2125      ;;*      ;;HIGH ORDER BITS ARE IN $HIOCT
2126
2127 013440 011646      $RDOCT: MOV      (SP),-(SP)  ;;PROVIDE SPACE FOR THE
2128 013442 016666 000004 000002 MOV      4(SP),2(SP)  ;;INPUT NUMBER
2129 013450 010046      MOV      R0,-(SP)  ;;PUSH R0 ON STACK
2130 013452 010146      MOV      R1,-(SP)  ;;PUSH R1 ON STACK
2131 013454 010246      MOV      R2,-(SP)  ;;PUSH R2 ON STACK
2132 013456 104407 1$:  RDLIN      ;;READ AN ASCII LINE
2133 013460 012600      MOV      (SP)+,R0  ;;GET ADDRESS OF 1ST CHARACTER
2134 013462 005001      CLR      R1      ;;CLEAR DATA WORD
2135 013464 005002      CLR      R2
2136 013466 112046 2$:  MOV      (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
2137 013470 001412      BEQ      3$      ;;IF ZERO GET OUT
2138 013472 006301      ASL      R1      ;;*2
2139 013474 006102      ROL      R2
2140 013476 006301      ASL      R1      ;;*4
2141 013500 006102      ROL      R2
2142 013502 006301      ASL      R1      ;;*8
2143 013504 006102      ROL      R2
2144 013506 042716 177770      BIC      #C7,(SP)  ;;STRIP THE ASCII JUNK
2145 013512 062601      ADD      (SP)+,R1  ;;ADD IN THIS DIGIT

```

CW
CW
CC
EN
ER

ES
GE
GE
MS

MU
NE

PC
PU
RE
SC

SE
SE
SE
SK
SL
SP
ST

SW
TR
TY
TY
TY
TY
TY
TY
XE
XS
SE
SR
SS

MACY11 30A(1052) 23-MAR-78 07:51 PAGE 43
READ AN OCTAL NUMBER FROM THE TTY

SEQ 0049

Address	Hex	ASCII	Label	Operation	Comment
2146	013514	000764		BR	2\$
2147	013516	005726	3\$:	TST	(SP)+
2148	013520	010166		MOV	R1,12(SP)
2149	013524	010237	000012	MOV	R2,\$HIOCT
2150	013530	012602	013540	MOV	(SP)+,R2
2151	013532	012601		MOV	(SP)+,R1
2152	013534	012600		MOV	(SP)+,R0
2153	013536	000002		RTI	
2154	013540	000000	\$HIOCT:	.WORD	0

```

;;LOOP
;;CLEAN TERMINATOR FROM STACK
;;SAVE THE RESULT

;;POP STACK INTO R2
;;POP STACK INTO R1
;;POP STACK INTO R0
;;RETURN
;;HIGH ORDER BITS GO HERE

```

CW
CW
\$ \$
\$ \$
\$ \$
\$ \$
\$ \$
\$ \$
. E
. H
. S
. S
. S
. \$
. \$
. \$
. \$
. \$
. \$
. \$
. \$
. \$
.
E
C
R
R
C

```

2155 .SBTTL TRAP DECODER
2156
2157 ;*****
2158 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2159 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2160 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2161 ;*GO TO THAT ROUTINE.
2162
2163 013542 010046 $TRAP: MOV R0,-(SP) ;;SAVE R0
2164 013544 016600 000002 MOV 2(SP),R0 ;;GET TRAP ADDRESS
2165 013550 005740 TST -(R0) ;;BACKUP BY 2
2166 013552 111000 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
2167 013554 006300 ASL R0 ;;POSITION FOR INDEXING
2168 013556 016000 013576 MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
2169 013562 000200 RTS R0 ;;GO TO ROUTINE
2170
2171
2172 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2173
2174 013564 011646 $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
2175 013566 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
2176 013574 000002 RTI ;;RESTORE THE PSW
2177
2178 .SBTTL TRAP TABLE
2179
2180 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2181 ;*BY THE "TRAP" INSTRUCTION.
2182
2183 ; ROUTINE
2184 ; -----
2185 013576 013564 $TRPAD: .WORD $TRAP2
2186 013600 012254 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2187 013602 012520 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2188 013604 012474 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2189 013606 012534 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2190 013610 012722 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
2191
2192
2193 013612 013146 $RDCHR ;;CALL=RDCHR TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
2194 013614 013266 $RDLIN ;;CALL=RDLIN TRAP+7(104407) TTY TYPEIN STRING ROUTINE
2195 013616 013440 $RDOCT ;;CALL=RDOCT TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY

```

2196	013620				ROMMAP:	
2197	013620	000400	000440	000167	.WORD	000400,000440,000167,000416,132710,100200,001775
	013636	102511	016005	000002	.WORD	102511,016005,000002,032705,177560,001406,005703
	013654	001102	120527	000101	.WORD	001102,120527,000101,001077,000207,042705,177760
	013672	126560	173064	000002	.WORD	126560,173064,000002,001070,000207
	013704	100400	001602	002604	.WORD	100400,001602,002604,103406
	013714	004610	105412	106414	.WORD	004610,105412,106414,007616
	013724	012700	175610	012706	.WORD	012700,175610,012706,000500,005004,005024,012737
	013742	173130	000004	000773	.WORD	173130,000004,000773,162704,000004,010406,005003
	013760	004767	000130	020527	.WORD	004767,000130,020527,040001,001372,004767,000116
	013776	005705	001366	004767	.WORD	005705,001366,004767,000154,010502,162702,000004
	014014	004767	000142	010501	.WORD	004767,000142,010501,005702,001424,060205,020605
	014032	101003	004767	000144	.WORD	101003,004767,000144,000731,004767,000046,005702
	014050	100402	110521	000772	.WORD	100402,110521,000772,105703,001403,004767,000124
	014066	000716	004767	000124	.WORD	000716,004767,000124,000730,004767,000012,105703
	014104	001366	004767	000106	.WORD	001366,004767,000106,010107,004767,177510,010546
	014122	111666	000001	042716	.WORD	111666,000001,042716,007760,004767,177472,006305
	014140	006305	006305	006305	.WORD	006305,006305,006305,042705,170017,052605,060503
	014156	005302	000207	004767	.WORD	005302,000207,004767,177726,010504,004767,177720
	014174	000305	105005	150405	.WORD	000305,105005,150405,000207,112705,000317,000405
	014212	112705	000102	000402	.WORD	112705,000102,000402,112705,000215,105760,000004
	014230	100375	110560	000006	.WORD	100375,110560,000006,000207,017640,002415,112024
	014246	005004	005005	012700	.WORD	005004,005005,012700
	014254	177500	000005	010410	.WORD	177500,000005,010410
	014262	012701	173420	012702	.WORD	012701,173420,012702,000375,112103,112110,100407
	014300	130310	001776	105202	.WORD	130310,001776,105202,100772,116012,000002,000771
	014316	005710	100403	005002	.WORD	005710,100403,005002,120312,001406,005205,020527
	014334	000010	002001	000746	.WORD	000010,002001,000746,000000,000112

2198	014620	017746	164322		KBISR:	MOV	@\$TKB,-(SP)	;SAVE CHAR
2199	014624	042716	000200			BIC	#BIT7,(SP)	;CLEAR BIT SEVEN
2200	014630	022716	000024			CMP	#24,(SP)	;WAS IT < T> (TEST NO.)
2201	014634	001004				BNE	3\$	;BR IF NO
2202	014636	012706	001100			MOV	#STACK,SP	;SET STACK
2203	014642	000137	004354			JMP	X1\$	;GO TO ROUTINE
2204	014646	022726	000007		3\$:	CMP	#7,(SP)+	;WAS IT < G>?
2205	014652	001016				BNE	1\$	;BR IF NO
2206	014654	104401	014716			TYPE	,XSWR	;SWR=
2207	014660	017746	164254			MOV	@SWR,-(SP)	
2208	014664	104402				TYPOC		
2209	014666	104401	014727			TYPE	,EQUALS	
2210	014672	104410				RDOCT		
2211	014674	105737	013374			TSTB	\$TTYIN	;WAS JUST <CR> HIT?
2212	014700	001402				BEQ	2\$	;BR IF YES
2213	014702	011677	164232			MOV	(SP),@SWR	;SET UP SWR
2214	014706	005726			2\$:	TST	(SP)+	;SET STACK
2215	014710	000002			1\$:	RTI		;EXIT
	014712	020040	000040		XSPA:	.ASCIZ	/ /	
	014716	024200	053523	024522	XSWR:	.ASCIZ	<200>'(SWR)='/'	
	014727	057	027475	000	EQUALS:	.ASCIZ	'/'='/'	
	014733	200	042524	052123	XTSTN:	.ASCIZ	<200>/TEST NO: /	
	014746	052200	000040		XTST:	.ASCIZ	<200>/T /	
						.EVEN		

2216	014752				BREPLY.TABLE:	
2217	014752	165250	000004			165250,004
2218	014756	172524	000004			172524,004
2219	014762	173000	001000			173000,512.
2220	014766	177546	000002			177546,002
2221	014772	177550	000010			177550,010
2222	014776	177560	000010			177560,010
2223	015002	175610	000010			175610,010
2224	015006	000000	000000			000000,000
2225	015012	000000	000000			000000,000
2226	015016	000000	000000			000000,000
2227	015022	000000	000000			000000,000
2228	015026	000000	000000			000000,000
2229	015032	000000	000000			000000,000

2230						
2231					;.PRINT	;DUBUG PC
2232	015036	011637	015110		DEBUG:	MOV (SP),1\$
2233	015042	162737	000002	015110		SUB #2,1\$
2234	015050	017737	000034	015112		MOV @1\$,2\$
2235	015056	022737	106400	015112		CMP #MTPS,2\$
2236	015064	001401				BEQ .+4
2237	015066	000000				HALT ;REAL TRAP TO ADDRESS '4'.
2238	015070	011637	015110			MOV (SP),1\$
2239	015074	062716	000002			ADD #2,(SP)
2240	015100	017766	000004	000002		MOV @1\$,2(SP)
2241	015106	000002				RTI
2242	015110	000000			1\$:	0
2243	015112	000000			2\$:	0
2244	015114				TEST.TABLE:	0
2245	015114	000000				TST1
2246	015116	004432				;PC FOR TEST NUMBER 1

2247	015120	004526	TST2	;PC FOR TEST NUMBER 2
2248	015122	004624	TST3	;PC FOR TEST NUMBER 3
2249	015124	004662	TST4	;PC FOR TEST NUMBER 4
2250	015126	004750	TST5	;PC FOR TEST NUMBER 5
2251	015130	005164	TST6	;PC FOR TEST NUMBER 6
2252	015132	005344	TST7	;PC FOR TEST NUMBER 7
2253	015134	005506	TST10	;PC FOR TEST NUMBER 10
2254	015136	005574	TST11	;PC FOR TEST NUMBER 11
2255	015140	005636	TST12	;PC FOR TEST NUMBER 12
2256	015142	005700	TST13	;PC FOR TEST NUMBER 13
2257	015144	005742	TST14	;PC FOR TEST NUMBER 14
2258	015146	006004	TST15	;PC FOR TEST NUMBER 15
2259	015150	006046	TST16	;PC FOR TEST NUMBER 16
2260	015152	006110	TST17	;PC FOR TEST NUMBER 17
2261	015154	006152	TST20	;PC FOR TEST NUMBER 20
2262	015156	006214	TST21	;PC FOR TEST NUMBER 21
2263	015160	006256	TST22	;PC FOR TEST NUMBER 22
2264	015162	006320	TST23	;PC FOR TEST NUMBER 23
2265	015164	006362	TST24	;PC FOR TEST NUMBER 24
2266	015166	006424	TST25	;PC FOR TEST NUMBER 25
2267	015170	006466	TST26	;PC FOR TEST NUMBER 26
2268	015172	006530	TST27	;PC FOR TEST NUMBER 27
2269	015174	006572	TST30	;PC FOR TEST NUMBER 30
2270	015176	006634	TST31	;PC FOR TEST NUMBER 31
2271	015200	006670	TST32	;PC FOR TEST NUMBER 32
2272	015202	006732	TST33	;PC FOR TEST NUMBER 33
2273	015204	007046	TST34	;PC FOR TEST NUMBER 34
2274	015206	007106	TST35	;PC FOR TEST NUMBER 35
2275	015210	007174	TST36	;PC FOR TEST NUMBER 36
2276	015212	007242	TST37	;PC FOR TEST NUMBER 37
2277	015214	007310	TST40	;PC FOR TEST NUMBER 40
2278	015216	007456	TST41	;PC FOR TEST NUMBER 41
2279	015220	007550	TST42	;PC FOR TEST NUMBER 42
2280	015222	007626	TST43	;PC FOR TEST NUMBER 43
2281	015224	007704	TST44	;PC FOR TEST NUMBER 44
2282	015226	010056	TST45	;PC FOR TEST NUMBER 45
2283	015230	010202	TST46	;PC FOR TEST NUMBER 46
2284	015232	010544	TST47	;PC FOR TEST NUMBER 47
2285	015234	011004	TST50	;PC FOR TEST NUMBER 50
2286	015236	011110	\$EOP	;END PASS PC
2287	015240	000000	0000	;TABLE TERMINATOR
2288	015242	000050	LASTN: 50	
2289	015244	000000	CORMAX: 0	
2290		000001	.END	

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

CWQAABO 11W03 SYSTEM TEST
CWQAAB.P11 22-MAR-78 14:51

MACY11 30A(1052) 23-MAR-78 07:51 PAGE 54
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0059

\$RDDEC= ***** U	2196												
\$RDLIN 013266	2087#	2194											
\$RDOCT 013440	2127#	2195											
\$RDSZ = 000010	2080#												
\$RTNAD 011210	1619#												
\$R2A = ***** U	2196												
\$SAVRE= ***** U	2196												
\$SCOPE 011232	337	1637#											
\$SETUP= 000027	327#	336	337	339	341	343	344	345	347	369	372	1596	1638
	1710	1728	1735	2048	2117								
\$STUP = 177777	327#												
\$SVLAD 011500	1661	1686#	1760										
\$SWR = 167000	2#	13	18	19	20	21	22	23	24	25	187	188	189
	344	345	347	348	415	442	469	490	522	577	625	667	692
	714	736	758	780	802	824	846	868	890	912	934	956	978
	1000	1022	1044	1063	1087	1127	1146	1172	1196	1222	1264	1291	1317
	1345	1388	1424	1505	1561	1591	1597	1612	1618	1620	1630	1631	1632
	1633	1634	1652	1664	1666	1667	1668	1675	1676	1677	1688	1691	1694
	1701	1702	1703	1704	1705	1713	1720	1725	1728	1766			
\$SWRMK= 000000	1634												
\$TIMES 001162	187#	344*	522*	625*	1044*	1063*	1127*	1424*	1505*	1561*	1597*	1675*	1682
	1685*	1694	1756*										
\$TKB 001146	179#	2046	2063	2069	2197								
\$TKS 001144	178#	387*	1132*	1744*	2046	2061	2067						
\$TMP0 001160	186#	549*	551	1101*	1102*	1103							
\$TN = 000051	2#	13	390	407	415#	435	442#	461	469#	482	490#	507	522#
	568	577#	615	625#	659	667#	684	692#	706	714#	728	736#	750
	758#	772	780#	794	802#	816	824#	838	846#	860	868#	882	890#
	904	912#	926	934#	948	956#	970	978#	992	1000#	1014	1022#	1037
	1044#	1056	1063#	1077	1087#	1120	1127#	1138	1146#	1163	1172#	1187	1196#
	1212	1222#	1256	1264#	1282	1291#	1308	1317#	1335	1345#	1380	1388#	1415
	1424#	1497	1505#	1525	1552	1561#	2246	2288					
\$TPB 001152	181#	1888*	1899										
\$TPFLG 001157	185#	1846	1899										
\$TPS 001150	180#	1886	1899										
\$TRAP 013542	341	2163#											
\$TRAP2 013564	2174#	2185											
\$TRP = 000011	2178#	2187#	2188#	2189#	2190#	2191#	2193	2194#	2195#	2196#			
\$TRPAD 013576	2168	2185#											
\$TSTNM 001102	158#	398*	1596*	1629	1640	1642	1644	1646	1686*	1691	1695	1712	1766
	1772												
\$TTYIN 013374	2088	2089	2106	2110#	2210								
\$TYPBN= ***** U	2191												
\$TYPDS 012722	1988#	2190											
\$TYPE 012254	1846#	2178	2186										
\$TYPEC 012424	1867	1874	1881	1886#	1887								
\$TYPEX 012472	1892	1894	1897#										
\$TYPOC 012520	1929#	2187											
\$TYPON 012534	1928	1931#	2189										
\$TYPOS 012474	1924#	2188											
\$XTSTR 011324	1655#												
\$SET4= 000000	1612#												
\$OFILL 012717	1925*	1929*	1939	1974#									
\$4OCAT= ***** U	1652	1722											
.	138#	142#	147#	155#	193	334	347	348	475	534	539	544	559
	597	600	636	697	703	719	725	741	747	763	769	785	791

807	813	829	835	851	857	873	879	895	901	917	923	939
945	961	967	983	989	1005	1011	1027	1033	1050	1070	1098	1105
1114	1134	1178	1184	1202	1208	1298	1304	1324	1330	1401	1454	1457
1467	1470	1473	1476	1484	1487	1490	1493	1581	1620	1624	1694	1695
1766	1828#	1899	2042#	2046	2110#	2111	2117	2197#	2236			

COMMENT ENDCOM ERROR	136#														
	136#														
	30#	427	450	476	500	535	545	550	560	589	601	637	650	677	698
	704	720	726	742	748	764	770	786	792	808	814	830	836	852	858
	874	880	896	902	918	924	940	946	962	968	984	990	1006	1012	1028
	1034	1051	1071	1099	1106	1115	1135	1156	1179	1185	1203	1209	1237	1251	1275
	1299	1305	1325	1331	1361	1375	1402	1436	1444	1448	1520	1539	1574	1582	
ESCAPE	136#														
GETPRI	136#														
GETSWR	136#	372#													
MSG	407#	409	434#	437	461#	463	481#	484	506#	509	567#	570	614#	617	658#
	661	683#	686	705#	708	727#	730	749#	752	771#	774	793#	796	815#	818
	837#	840	859#	862	881#	884	903#	906	925#	928	947#	950	969#	972	991#
	994	1013#	1016	1036#	1039	1055#	1058	1076#	1079	1119#	1122	1137#	1140	1162#	1165
	1186#	1189	1211#	1214	1255#	1258	1281#	1284	1307#	1310	1334#	1337	1379#	1382	1414#
	1417	1496#	1499	1551#	1554										
MULT	136#														
NEWST	136#	407	435	461	482	507	568	615	659	684	706	728	750	772	794
	816	838	860	882	904	926	948	970	992	1014	1037	1056	1077	1120	1138
	1163	1187	1212	1256	1282	1308	1335	1380	1415	1497	1552				
POP	136#	2029	2150												
PUSH	136#	1988	2129												
REPORT	136#														
SCOPE	31#	414	441	468	489	521	576	624	666	691	713	735	757	779	801
	823	845	867	889	911	933	955	977	999	1021	1043	1062	1086	1126	1145
	1171	1195	1221	1263	1290	1316	1344	1387	1423	1504	1560	1595			
SETPRI	136#														
SETTRA	2178#	2187	2188	2189	2190	2193	2194	2195							
SETUP	136#	328													
SKIP	136#	390	1525												
SLASH	136#														
SPACE	136#														
STARS	136#	151	193	407	413	435	440	461	467	482	488	507	520	568	575
	615	623	659	665	684	690	706	712	728	734	750	756	772	778	794
	800	816	822	838	844	860	866	882	888	904	910	926	932	948	954
	970	976	992	998	1014	1020	1037	1042	1056	1061	1077	1085	1120	1125	1138
	1144	1163	1170	1187	1194	1212	1220	1256	1262	1282	1289	1308	1315	1335	1343
	1380	1386	1415	1422	1497	1503	1552	1559	1588	1626	1697	1775	1831	1901	1978
	2045	2051	2080	2119	2157										
SWRSU	136#	349#													
TRMTRP	2178#														
TYPBIN	136#														
TYPDEC	136#	1607	1815												
TYPNAM	3#	136#	365												
TYPNUM	136#														
TYPOCS	136#														
TYPOCT	136#	1787	1812												
TYPTXT	136#														
XERREN	1586#	1739													
XSCOP	1586#	1638													
\$EGOLF	1586#	1766													
\$RESET	3#	1131	1743												
\$SKIP	3#	475	534	544	597	600	697	703	719	725	741	747	763	769	785
	791	807	813	829	835	851	857	873	879	895	901	917	923	939	945
	961	967	983	989	1005	1011	1027	1033	1050	1070	1098	1105	1114	1134	1178
	1184	1202	1208	1298	1304	1324	1330								

SSCMRE	149#														
SSCMTM	149#	186													
SSESCA	136#														
SSNEWT	136#	407	435	461	482	507	568	615	659	684	706	728	750	772	794
	816	838	860	882	904	926	948	970	992	1014	1037	1056	1077	1120	1138
	1163	1187	1212	1256	1282	1308	1335	1380	1415	1497	1552				
SSSET	2178#	2187	2188	2189	2190	2193	2194	2195							
SSSKIP	136#	390	1525												
.EQUAT	3#	26													
.HEADE	3#														
.SETUP	3#	327													
.SWRHI	3#	14													
.SWRLO	26#														
.SCATC	3#	136													
.SCMTA	3#	149													
.SEOP	3#	1586													
.SERRO	3#	1695													
.SERRT	3#	1773													
.SRDOC	3#	2117													
.SREAD	3#	2043													
.SSCOP	3#	1624													
.STRAP	3#	2155													
.STYPD	3#	1976													
.STYPE	3#	1829													
.STYPO	3#	1899													

. ABS. 015246 000

ERRORS DETECTED: 0

CWQAAB.BIN,CWQAAB.LST/CRF/SOL/NL:TOC=CWQAAB.P11
 RUN-TIME: 20 10 .9 SECONDS
 RUN-TIME RATIO: 92/32=2.8
 CORE USED: 23K (45 PAGES)