

KW11-L

LINE FREQUENCY CLOCK TEST
CZKWAGO

AH-8874G-MC

COPYRIGHT © 70-78

FICHE 1 OF 1

JUL 1978

digital

MADE IN USA

The microfiche card contains a grid of frames. The leftmost column of frames contains technical specifications and data tables. The subsequent columns contain waveforms, likely representing clock signals, with varying patterns of pulses and gaps. The frames are arranged in a regular grid, typical of a microfiche format.

IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-8872G-MC
PRODUCT NAME: CZKWAGO KW11-L LINE FREQUENCY CLOCK TEST
PRODUCT DATE: 1-MAY-78
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) 1970, 1978 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

1.0 GENERAL PROGRAM INFORMATION

1.1 ABSTRACT
THIS PROGRAM TESTS THE KW11L LINE FREQUENCY CLOCK. IT VALIDATES PROPER OPERATION UNDER BOTH INTERRUPT AND NON-INTERRUPT MODES.

1.2 SYSTEM REQUIREMENTS
THIS PROGRAM IS DESIGNED TO RUN ON ANY PDP-11 WITH 4K OF MEMORY AND A KW11 LINE FREQUENCY CLOCK.

1.3 DEVICE ADDRESSES

THE DIAGNOSTIC ASSUMES THE LINE CLOCK ADDRESS IS 177546. VARIOUS TESTS IN THE DIAGNOSTIC ENSURE THAT THE LINE CLOCK IS NOT AFFECTED WHEN THE PAPER TAPE PUNCH (ADDRESS 177556) AND TELETYPE (ADDRESS 177566) ARE ACCESSED. THEREFORE, AN OCCASIONAL "a" MAY BE OUTPUT AT THE TELETYPE AND A CHARACTER MAY BE PUNCHED AT THE PAPER TAPE PUNCH. THESE ARE NOT ERRORS.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING
PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED
A. ABSOLUTE LOADER PROGRAM MUST BE IN MEMORY
B. PLACE THE BINARY TAPE IN THE PAPER TAPE READER
C. LOAD ADDRESS 17500
D. DEPRESS START (TAPE SHOULD READ IN)

2.2 STARTING
PROGRAM STARTING ADDRESS IS 000200
A. LOAD ADDRESS 000200
B. SELECT SWITCH REGISTER OPTIONS (SEE SECTION 2.3)

NOTE: IF THE PROCESSOR DOES NOT HAVE A HARDWARE SWITCH REGISTER, LOCATION # 176 SHOULD BE LOADED WITH THE APPROPRIATE SWITCH SETTINGS AFTER THE DIAGNOSTIC HAS BEEN LOADED INTO MEMORY. THE PROGRAM IS THEN STARTED AT ADDRESS 200.

C. DEPRESS START (PROGRAM SHOULD START RUNNING)

2.3 SWITCH REGISTER OPTIONS
HERE IS A LIST OF CONSOLE SWITCHES AND THEIR EFFECT ON THE PROGRAM...

SWITCH	ACTION IF SET
15	HALT ON ERROR
14	LOOP ON CURRENTLY EXECUTING TEST
13	INHIBIT ERROR PRINTOUTS
12	(UNUSED)
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST SPECIFIED IN SWR<7:0>
7-0	# OF TEST TO LOOP ON (ONLY WHEN SWR8 = 1)

2.4 EXECUTION TIMES

D 1

EXECUTION TIME FOR THIS PROGRAM IS DEPENDENT ON THE
MODEL OF PDP-11 IT IS BEING RUN ON. FOR A PDP-11/40
ABOUT 5 SECONDS IS NECESSARY TO DO 1 PASS OF THE PROGRAM
WITHOUT ITERATIONS.

SEQ 0003

3.0 ERROR INFORMATION

3.1 STANDARD ERROR REPORTING PROCEDURES
ERROR PRINTOUTS CONSIST OF FROM 4 TO 8 COLUMNS OF DATA, A
DATA HEADER, AND POSSIBLY A SHORT ERROR MESSAGE DESCRIBING
THE ERROR. FOR EXAMPLE...

```
CLOCK FAILED TO INTERRUPT
PC      PS      SP      TEST#  LKS
002262 000344 000764 000007 000300
```

THE FIRST 4 COLUMNS OF OF THE ERROR MESSAGE ALWAYS SHOW THE
CONTENTS OF THE PC, PS, SP, AND THE TEST NUMBER. MORE COLUMNS
OF DATA ARE ADDED WHERE THEY MIGHT BE RELEVANT TO A PARTICULAR
ERROR.

3.2 UNEXPECTED TRAP ERROR REPORTING
AN UNEXPECTED TRAP TO ADDRESS 4 CAUSES THE FOLLOWING
MESSAGE TO BE PRINTED OUT...

```
TRAPPED TO LOC 4 FROM LOCATION "XXXXXX"
RESTARTING PROGRAM
```

IN THE ACTUAL MESSAGE THE "XXXXXX" IS REPLACED BY THE PC
ADDRESS PUSHED ONTO THE STACK WHEN THE UNEXPECTED TRAP
OCCURS. THE PROGRAM THEN TRYS TO RESTART ITSELF
DESPITE SWITCH REGISTER SETTINGS.

3.3 POWER FAIL
IF A POWER FAIL CONDITION IS DETECTED THE FOLLOWING MESSAGE
IS PRINTED...

POWER

AFTER PRINTING OUT THE MESSAGE THE PROGRAM TRYS TO
RESTART ITSELF.

5.0 DEVICE INFORMATION

5.1 GENERAL INFORMATION
THE LINE CLOCK INTERRUPT VECTOR ADDRESS IS 100
THE LINE CLOCK PRIORITY LEVEL IS BR6

5.2 REGISTERS

```
LINE CLOCK STATUS REGISTER (LKS)                                777546
-----
! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
! ! ! ! ! ! ! ! 7 6 ! ! ! ! ! !
-----
```

BIT6 IF SET MONITOR=1 CAUSES AN INTERRUPT

BIT7 MONITOR BIT. SET BY CLOCK, CLEARED BY USER

E 1

SEQ 0004

7.0 LISTING

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

167400
000000

001100

177776

177774

177772

177570

177570

000000

000001

000002

000003

000004

000005

000006

000007

000006

000007

```
.ABS
.ENABL AMA
.LIST ME
.NLIST MC, MD, CND
$SWR=167400
$TN=0

.ENABL ABS
.TITLE CZKWA-G LINE FREQUENCY CLOCK PROGRAM
;*COPYRIGHT (C) 1970,1977
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY J. COMEAU
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-B1),AUG 29,1975.
;*

.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* -----
;* 15 HALT ON ERROR
;* 14 LOOP ON TEST
;* 13 INHIBIT ERROR TYPEOUTS
;* 11 INHIBIT ITERATIONS
;* 10 BELL ON ERROR
;* 9 LOOP ON ERROR
;* 8 LOOP ON TEST IN SWR<7:0>
;* 7-0 #OF TEST TO LOOP ON IF SWR<8> IS SET

.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***.
STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 ;;HARDWARE SWITCH REGISTER
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

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

```

57
58      000000
59      000040
60      000100
61      000140
62      000200
63      000240
64      000300
65      000340
66
67
68      100000
69      040000
70      020000
71      010000
72      004000
73      002000
74      001000
75      000400
76      000200
77      000100
78      000040
79      000020
80      000010
81      000004
82      000002
83      000001
84
85
86
87
88
89
90
91
92
93
94
95
96      100000
97      040000
98      020000
99      010000
100     004000
101     002000
102     001000
103     000400
104     000200
105     000100
106     000040
107     000020
108     000010
109     000004
110     000002
111     000001
112

```

```

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

```

```

;*"SWITCH REGISTER" SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
.EQUIV SW09,SW9
.EQUIV SW08,SW8
.EQUIV SW07,SW7
.EQUIV SW06,SW6
.EQUIV SW05,SW5
.EQUIV SW04,SW4
.EQUIV SW03,SW3
.EQUIV SW02,SW2
.EQUIV SW01,SW1
.EQUIV SW00,SW0

```

```

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

```

```
113 .EQUIV BIT03,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 TPVEC= 64 ;;TTY PRINTER VECTOR
135 000240 PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR
136 ;MISCELANHUS EQUATES
137 177546 LKS=177546
138 000240 NOP=240
139 000774 BUF2=774
140 000776 BUF1=776
141
142 .SBTTL TRAP CATCHER
143
144 000000 .=0
145 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
146 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
147 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
148 000174 .=174
149 000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
150 000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
151
152 .SBTTL STARTING ADDRESS(ES)
153 000200 000137 001400 JMP @#KSTART ;;JUMP TO STARTING ADDRESS OF PROGRAM
154 ;*****
155
156 .SBTTL ACT11 HOOKS
157 ;HOOKS REQUIRED BY ACT11
158 000204 $SVPC=. ;SAVE PC
159 000046 .=46
160 000046 006232 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
161 000052 .=52
162 000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO
163 000204 .=$SVPC ;; RESTORE PC
```



```
164 ;*****
165
166 .SBTTL COMMON TAGS
167
168 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
169 ;*USED IN THE PROGRAM.
170
171 001100 . =1100
172 001100 SCMTAG: .WORD 0 ;: START OF COMMON TAGS
173 001100 $PASS: .WORD 0 ;: CONTAINS PASS COUNT
174 001102 $STNM: .BYTE 0 ;: CONTAINS THE TEST NUMBER
175 001103 $ERFLG: .BYTE 0 ;: CONTAINS ERROR FLAG
176 001104 $ICNT: .WORD 0 ;: CONTAINS SUBTEST ITERATION COUNT
177 001106 $LPADR: .WORD 0 ;: CONTAINS SCOPE LOOP ADDRESS
178 001110 $LPERR: .WORD 0 ;: CONTAINS SCOPE RETURN FOR ERRORS
179 001112 $ERTTL: .WORD 0 ;: CONTAINS TOTAL ERRORS DETECTED
180 001114 $ITEMB: .BYTE 0 ;: CONTAINS ITEM CONTROL BYTE
181 001115 $ERMAX: .BYTE 1 ;: CONTAINS MAX. ERRORS PER TEST
182 001116 $ERRPC: .WORD 0 ;: CONTAINS PC OF LAST ERROR INSTRUCTION
183 001120 $GDADR: .WORD 0 ;: CONTAINS ADDRESS OF 'GOOD' DATA
184 001122 $BDADR: .WORD 0 ;: CONTAINS ADDRESS OF 'BAD' DATA
185 001124 $GDDAT: .WORD 0 ;: CONTAINS 'GOOD' DATA
186 001126 $BDDAT: .WORD 0 ;: CONTAINS 'BAD' DATA
187 001130 .WORD 0 ;: RESERVED--NOT TO BE USED
188 001132 .WORD 0
189 001134 .WORD 0
190 001136 177570 SWR: .WORD DSWR ;: ADDRESS OF SWITCH REGISTER
191 001140 177570 DISPLAY: .WORD DDISP ;: ADDRESS OF DISPLAY REGISTER
192 001142 177560 $TKS: 177560 ;: TTY KBD STATUS
193 001144 177562 $TKB: 177562 ;: TTY KBD BUFFER
194 001146 177564 $TPS: 177564 ;: TTY PRINTER STATUS REG. ADDRESS
195 001150 177566 $TPB: 177566 ;: TTY PRINTER BUFFER REG. ADDRESS
196 001152 000 $NULL: .BYTE 0 ;: CONTAINS NULL CHARACTER FOR FILLS
197 001153 002 $FILLS: .BYTE 2 ;: CONTAINS # OF FILLER CHARACTERS REQUIRED
198 001154 012 $FILLC: .BYTE 12 ;: INSERT FILL CHARS. AFTER A "LINE FEED"
199 001155 000 $TPFLG: .BYTE 0 ;: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
200 001156 000000 $REGAD: .WORD 0 ;: CONTAINS THE ADDRESS FROM
201 ;: WHICH ($REG0) WAS OBTAINED
202 001160 000000 $REG0: .WORD 0 ;: CONTAINS (($REGAD)+0)
203 001162 000000 $REG1: .WORD 0 ;: CONTAINS (($REGAD)+2)
204 001164 000000 $REG2: .WORD 0 ;: CONTAINS (($REGAD)+4)
205 001166 000000 $REG3: .WORD 0 ;: CONTAINS (($REGAD)+6)
206 001170 000000 $REG4: .WORD 0 ;: CONTAINS (($REGAD)+10)
207 001172 000000 $REG5: .WORD 0 ;: CONTAINS (($REGAD)+12)
208 001174 000000 $REG6: .WORD 0 ;: CONTAINS (($REGAD)+14)
209 001176 000000 $REG7: .WORD 0 ;: CONTAINS (($REGAD)+16)
210 001200 000000 $TMP0: .WORD 0 ;: USER DEFINED
211 001202 000000 $TMP1: .WORD 0 ;: USER DEFINED
212 001204 000000 $TMP2: .WORD 0 ;: USER DEFINED
213 001206 000000 $TMP3: .WORD 0 ;: USER DEFINED
214 001210 000000 $TMP4: .WORD 0 ;: USER DEFINED
215 001212 000000 $TMP5: .WORD 0 ;: USER DEFINED
216 001214 000000 $TMP6: .WORD 0 ;: USER DEFINED
217 001216 000000 $TMP7: .WORD 0 ;: USER DEFINED
218 001220 000000 $TIMES: 0 ;: MAX. NUMBER OF ITERATIONS
219 001222 000000 $ESCAPE: 0 ;: ESCAPE ON ERROR ADDRESS
```

CZKWA-G LINE FREQUENCY CLOCK PROGRAM
CZKWAG.P11 10-APR-78 11:05

J 1
MACY11 30A(1052) 19-APR-78 14:25 PAGE 6
COMMON TAGS

SEQ 0009

220 001224 177607 000377
221 001230 077
222 001231 015
223 001232 000012

\$BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
\$QUES: .ASCII /?/ ;;QUESTION MARK
\$CRLF: .ASCII <15> ;;CARRIAGE RETURN
\$LF: .ASCIZ <12> ;;LINE FEED

224
225 001234 000000
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241 001236
242 001236 010462
243 001240 010543
244 001242 010624
245 001244 000000
246
247 001246 010642
248 001250 010674
249 001252 010746
250 001254 000000
251
252 001256 010762
253 001260 011055
254 001262 011126
255 001264 000000
256
257 001266 011142
258 001270 011241
259 001272 011376
260 001274 000000
261
262 001276 011414
263 001300 011465
264 001302 011536
265 001304 000000
266
267 001306 011552
268 001310 011610
269 001312 011662
270 001314 000000
271
272 001316 011676
273 001320 011765
274 001322 012044
275 001324 000000
276
277 001326 012062
278 001330 012136
279 001332 012222

WORD: 000000

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

EM1 ;;
DH1 ;;PC PS SP TEST# LKS LKS ;;
DT1 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,LKS,\$GDDAT S/B ;;
0
EM2 ;;"CLOCK FAILED TO INTERRUPT"
DH2 ;;PC PS SP TEST# LKS ;;
DT2 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,LKS
0
EM3 ;;"CLOCK INTERRUPTED WHEN THE PROCESSOR PRIORITY WAS TOO HIGH"
DH3 ;;PC PS SP TEST# LKS ;;
DT3 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,LKS
0
EM4 ;;"CLOCK GIVES UNEQUAL # OF PULSES OVER TWO EQUAL PERIODS OF TIME"
DH4 ;;PC PS SP TEST# 1ST 2ND
DT4 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,\$REG1,\$REG0
0
EM5 ;;"LKS REGISTER RESPONDS TO ANOTHER ADDRESS"
DH5 ;;PC PS SP TEST# ADDRESS
DT5 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,\$GDADR
0
EM6 ;;"A NO SACK TIMEOUT HAS OCCURED"
DH6 ;;PC PS SP TEST# LKS ;;
DT6 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,LKS
0
EM7 ;;"WRONG CONDITION CODES WERE PUT ONTO STACK BY INTERRUPT"
DH7 ;;PC PS SP TEST# CC CC
DT7 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,BUF1,\$GDDAT
0
EM10 ;;"WRONG PC PUT ONTO THE STACK BY AN INTERRUPT"
DH10 ;;PC PS SP TEST# ;;
DT10 ;;\$ERRPC,\$REG7,\$REG6,\$REG5,BUF2,\$GDDAT

LINE	FREQUENCY	CLOCK	PROGRAM	MACY11 30A(1052)	19-APR-78	14:25	PAGE 8
280	001334	000000		0			
281							
282	001336	012240		EM11	;	"TRAPPED (RYING TO ACCESS LKS REGISTER"	
283	001340	012316		DH11	;	"(PC) (PS) (SP) TEST#"	
284	001342	012354		DT11	;	\$ERRPC,\$REG7,\$REG6,\$REG5	
285	001344	000000		0			
286							
287							
288							
289		001400					
290	001400	012706	001000	KSTART:	MOV	#1000,SP	; INITIALIZE THE STACK SO WE CAN CALL THE TYPEOUT
291	001404	005046			CLR	-(SP)	::
292	001406	013746	000034		MOV	34,-(SP)	::SAVE CURRENT TRAP VECTOR
293	001412	012737	001422 000034		MOV	#64\$,34	::SETUP NEW TRAP VECTOT
294	001420	104400			TRAP		::PUSH OLD PSW AN PCON STACK
295	001422	016666	000002 000006	64\$:	MOV	2(SP),6(SP)	::
296	001430	012716	001436		MOV	#65\$,(SP)	::REPLACE OLD PC WITH NEW
297	001434	000002			RTI		::RESTORE PSW
298	001436	012637	000034	65\$:	MOV	(SP)+,34	::RESTORE OLD TRAP VECTOR
299	001442	004737	006612		JSR	PC,\$TYPE	::PRINTOUT STARTUP MESSAGE
300	001446	010322			STMES		::ADDRESS OF MESSAGE 'MAINDEC-11-DZKWA-F'
301	001450			START:			
302	001450	012706	001100		MOV	#\$CMTAG,R6	::FIRST LOCATION TO BE CLEARED
303	001454	005026			CLR	(R6)+	::CLEAR MEMORY LOCATION
304	001456	022706	001126		CMP	#\$BDDAT,R6	::DONE?
305	001462	001374			BNE	.-6	::LOOP BACK IF NO
306	001464	012706	001000		MOV	#1000,SP	::SETUP THE STACK POINTER
307	001470	012737	006340 000020		MOV	#\$SCOPE,@#IOTVEC	::IOT VECTOR FOR SCOPE ROUTINE
308	001476	012737	000340 000022		MOV	#340,@#IOTVEC+2	::LEVEL 7
309	001504	012737	007634 000030		MOV	#\$ERROR,@#EMTVEC	::EMT VECTOR FOR ERROR ROUTINE
310	001512	012737	000340 000032		MOV	#340,@#EMTVEC+2	::LEVEL 7
311	001520	012737	010040 000034		MOV	#\$TRAP,@#TRAPVEC	::TRAP VECTOR FOR TRAP CALLS
312	001526	012737	000340 000036		MOV	#340,@#TRAPVEC+2	::LEVEL 7
313	001534	012737	010074 000024		MOV	#\$PWDRN,@#PWVEC	::POWER FAILURE VECTOR
314	001542	012737	000340 000026		MOV	#340,@#PWVEC+2	::LEVEL 7
315	001550	013737	006200 006172		MOV	\$ENDCT,\$EOPCT	::SETUP END-OF-PROGRAM COUNTER
316	001556	005037	001220		CLR	\$TIMES	::INITIALIZE NUMBER OF ITERATIONS
317	001562	005037	001222		CLR	\$ESCAPE	::CLEAR THE ESCAPE ON ERROR ADDRESS
318	001566	112737	000001 001115		MOVB	#1,\$ERMAX	::ALLOW ONE ERROR PER TEST
319	001574	012737	001574 001106		MOV	#\$SLPADR	::INITIALIZE THE LOOP ADDRESS FOR SCOPE
320	001602	012737	001602 001110		MOV	#\$SLPERR	::SETUP THE ERROR LOOP ADDRESS
321	001610	013746	000004		MOV	@#4,-(SP)	::SAVE ERROR VECTOR
322	001614	013746	000006		MOV	@#6,-(SP)	
323	001620	012737	001634 000004		MOV	#64\$,4	::SET UP TIME OUT VECTOR
324	001626	005777	177304		TST	@SWR	::TRY TO REFERENCE HARDWARE SWR
325	001632	000407			BR	65\$::BRANCH IF NO TIMEOUT TRAP OCCURS
326	001634	012737	000176 001136	64\$:	MOV	#\$SWREG,SWR	::POINT TO SOFTWARE SWR
327	001642	012737	000174 001140		MOV	#\$DISPREG,DISPLAY	::POINT TO SOFTWARE DISPLAY REG
328	001650	022626			CMP	(SP)+,(SP)+	::RESTORE STACK
329	001652	012637	000006	65\$:	MOV	(SP)+,@#6	::RESTORE ERROR VECTOR
330	001656	012637	000004		MOV	(SP)+,@#4	
331	001662	005737	000042		TST	42	::LOADED BY A MONITOR
332	001666	001401			BEQ	T0001	::BR IF NO
333	001670	000005			RESET		::YES--GENERATE AN INIT
334							
335							

```

336
337
338      .SBTTL TEST THAT THE LKS CAN BE REFERENCED WITHOUT A BUS ERROR
339      ;LKS ACCESS TEST
339      001672 000004
340      001674 012737 001730 000004
341      001702 012737 000340 000006
342      001710 012737 001716 001106
343      001716 012706 001000
344      001722 005037 177546
345      001726 000401
346      001730 104011
347
348
349
350      .SBTTL TEST THAT START CLEARS LINE CLOCK INTERRUPT ENABLE BIT
351      ;TEST THAT START CLEARS LINE CLOCK INTERRUPT ENABLE BIT
352      001732 000004
353      001734 012737 006266 000004
354      001742 012737 000340 000006
355      001750 012737 001756 001106
356      001756 000005
357      001760 012737 000200 001124
358      001766 032737 000100 177546
359      001774 001401
360      001776 104001
361
362
363
364      .SBTTL TEST THAT START SETS CLOCK FLAG
365      ;TEST THAT START SETS CLOCK FLAG
366      002000 000004
367      002002 012737 006266 000004
368      002010 012737 000340 000006
369      002016 012737 000200 001124
370      002024 012737 002032 001106
371      002032 000005
372      002034 105737 177546
373      002040 100401
374      002042 104001
375
376
377
378      .SBTTL TEST THAT CLOCK FLAG WILL SET AFTER SUFFICIENT PERIOD OF TIME (20 MS MIN)
379      ;TEST THAT CLOCK FLAG WILL SET AFTER SUFFICIENT PERIOD OF TIME (20 MS MIN)
380      002044 000004
381      002046 012737 006266 000004
382      002054 012737 000340 000006
383      002062 012737 002070 001106
384      002070 012737 000200 001124
385      002076 005037 177546
386      002102 005000
387      002104 105737 177546
388      002110 100403
389      002112 005200
390      002114 001373
391      002116 104001

```

T0001: SCOPE
 MOV #E0001,@#4 ;PREPARE FOR ADDRESSING THE LKS REGISTER. BAD HARDWARE
 MOV #340,@#6 ;COULD CAUSE A TRAP TO 4
 MOV #R0001,\$LPADR ;TIGHTEN UP THE SCOPE LOOP A BIT IN CASE OF AN ERROR
 R0001: MOV #1000,SP ;SETUP THE STACK POINTER IN CASE OF AN ERROR
 I0001: CLR @#LKS ;JUST REFERENCE LKS. DONT WORRY IF IT DIDNT CLEAR YET
 BR T0002 ;WE DIDNT TRAP IF WE REACH HERE. GO ON TO NEXT TEST
 E0001: ERROR 1 ;ERROR:::TRAPED TRYING TO ACCESS THE LKS REGISTER

T0002: SCOPE
 MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
 MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
 MOV #R0002,\$LPADR ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
 R0002: RESET
 MOV #200,\$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
 BIT #100,LKS ;TEST THE INTERRUPT ENABLE BIT
 BEQ T0003
 E0002: ERROR 1 ;ERROR, CLOCK INTERRUPT ENABLE NOT CLEARED BY INIT

T0003: SCOPE
 MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
 MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
 MOV #200,\$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
 MOV #R0003,\$LPADR ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
 R0003: RESET ;SHOULD SET THE CLOCK FLAG
 TSTB LKS ;FIND OUT IF IT DID
 BMI T0004 ;GO ON TO THE NEXT TEST IF IT SET THE CLOCK FLAG
 E0003: ERROR 1 ;ERROR, CLOCK FLAG NOT SET BY INIT

T0004: SCOPE
 MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
 MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
 MOV #R0004,\$LPADR ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
 R0004: MOV #200,\$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
 CLR LKS ;CLEAR THE CLOCK FLAG
 CLR RO ;AND A TIMER LOCATION
 A0004: TSTB LKS ;IS CLOCK FLAG SET
 BMI T0005
 INC RO ;NO, INCREMENT COUNT 003 WAIT FOR SOMEMORE
 BNE A0004 ;WAIT SUFFICIENT AMOUNT OF TIME FOR CLOCK
 E0004: ERROR 1 ;ERROR, CLOCK FLAG FAILED TO SET

```

392
393
394
395
396
397 002120 000004
398 002122 012737 006266 000004
399 002130 012737 000340 000006
400 002136 012737 000100 001124
401 002144 012737 002164 001106
402 002152 013746 000340
403 002156 012746 002164
404 002162 000002
405 002164
406 002164 005037 177546
407 002170 005003
408 002172 105737 177546
409 002176 100404
410 002200 005203
411 002202 001373
412 002204 104001
413 002206 000410
414 002210
415 002210 012737 000100 177546
416 002216 032737 000100 177546
417 002224 001001
418 002226 104001
419
420
421
422
423
424 002230 000004
425 002232 012737 006266 000004
426 002240 012737 000340 000006
427 002246 012737 000000 001124
428 002254 012737 002274 001106
429 002262 013746 000340
430 002266 012746 002274
431 002272 000002
432 002274
433 002274 005037 177546
434 002300 005003
435 002302 105737 177546
436 002306 100404
437 002310 005203
438 002312 001373
439 002314 104001
440 002316 000412
441 002320
442 002320 012737 000100 177546
443 002326 005037 177546
444 002332 032737 000100 177546
445 002340 001401
446 002342 104001
447

.SBTTL TEST THAT INTERRUPT ENABLE BIT MAY BE SET
:TEST THAT INTERRUPT ENABLE BIT MAY BE SET
T0005: SCOPE
      MOV      #TRAP0,@#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
      MOV      #340,@#6      ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
      MOV      #100,$GDDAT    ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
      MOV      #R0005,$LPADR  ;SETUP LOOP BACK ADDRESS IN CASE OF ERROR
      MOV      PR7,-(SP)      ;;PUT NEW PS ON STACK
      MOV      #64$,-(SP)    ;;PUT NEW PC ON STACK
      RTI                    ;; POP NEW PC AND PS

64$:
R0005: CLR      LKS
      CLR      R3            ;INITIALIZE A COUNTER LOCATION
A0005: TSTB    LKS          ;IS THE CLOCK FLAG SET?
      BMI     B0005         ;IF SO, CONTINUE ON WITH THE TEST
      INC     R3            ;IF NOT INCREMENT THE COUNTER LOCATION
      BNE     A0005         ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
E0005: ERROR 1            ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
      BR      T0006

B0005: MOV      #100,LKS     ;CLEAR CLOCK FLAG AND SET INTERRUPT ENABLE
      BIT     #100,LKS     ;IS INTERRUPT ENABLE SET?
      BNE     T0006
E1005: ERROR 1            ;ERROR INTERRUPT ENABLE NOT SET

.SRTTL TEST THAT INTERRUPT ENABLE BIT MAY BE CLEARED
:TEST THAT INTERRUPT ENABLE BIT MAY BE CLEARED
T0006: SCOPE
      MOV      #TRAP0,@#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
      MOV      #340,@#6      ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
      MOV      #0,$GDDAT     ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
      MOV      #R0006,$LPADR  ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
      MOV      PR7,-(SP)      ;;PUT NEW PS ON STACK
      MOV      #64$,-(SP)    ;;PUT NEW PC ON STACK
      RTI                    ;; POP NEW PC AND PS

64$:
R0006: CLR      LKS
      CLR      R3            ;INITIALIZE A COUNTER LOCATION
A0006: TSTB    LKS          ;IS THE CLOCK FLAG SET?
      BMI     B0006         ;IF SO, CONTINUE ON WITH THE TEST
      INC     R3            ;IF NOT INCREMENT THE COUNTER LOCATION
      BNE     A0006         ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
E0006: ERROR 1            ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
      BR      T0007

B0006: MOV      #100,LKS     ;CLEAR CLOCK FLAG AND SET INTERRUPT ENABLE
      CLR     LKS          ;CLEAR INTERRUPT ENABLE
      BIT     #100,LKS     ;TEST THE INTERRUPT ENABLE BIT
      BEQ     T0007        ;IS INTERRUPT ENABLE CLEARED
E10006: ERROR 1          ;ERROR, ERROR INTERRUPT BIT CAN NOT BE CLEARED

```

```

448
449
450
451
452 002344 000004
453 002346 012737 006266 000004
454 002354 012737 000340 000006
455 002362 012737 000300 001124
456 002370 012737 002412 001106
457 002376 012737 002474 000100
458 002404 012737 000340 000102
459 002412 012706 001000
460 002416 005046
461 002420 012746 002426
462 002424 000002
463 002426 005037 177546
464 002432 005003
465 002434 105737 177546
466 002440 100404
467 002442 005203
468 002444 001373
469 002446 104001
470 002450 000415
471 002452
472 002452 012737 000100 177546
473 002460 005000
474 002462 005200
475 002464 000240
476 002466 001375
477 002470 104002
478 002472 000404
479 002474 105737 177546
480 002500 100401
481 002502 104001
482
483
484
485
486
487
488 002504 000004
489 002506 012737 006266 000004
490 002514 012737 000340 000006
491 002522 012737 002544 001106
492 002530 012737 002654 000100
493 002536 012737 000340 000102
494 002544 005037 177546
495 002550 013746 000004
496 002554 012737 002572 000004
497 002562 012737 000240 177776
498 002570 000404
499 002572 022626
500 002574 012637 000004
501 002600 000431
502 002602 012637 000004
503 002606 012706 001000

.SBTTL TEST THAT CLOCK INTERRUPTS TO CORRECT VECTOR ADDRESS
:TEST THAT CLOCK INTERRUPTS TO CORRECT VECTOR ADDRESS
T0007: SCOPE
      MOV      #TRAP0,#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
      MOV      #340,#6       ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
      MOV      #300,$GDDAT    ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
      MOV      #R0007,$LPADR  ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
      MOV      #D0007,100     ;SET UP VECTOR RETURN POINTER
      MOV      #340,#102     ;1 INTERRUPT IS ENOUGH
R0007: MOV      #1000,SP      ;GET STACK READY FOR INTERRUPTS
      CLR      -(SP)         ;DROP CPU PRIORITY LEVEL
      MOV      #1$,-(SP)     ;SET RETURN ADDRESS ON STACK FROM 'RTI'
      RTI
1$:   CLR      LKS
      CLR      R3           ;INITIALIZE A COUNTER LOCATION
A0007: TSTB    LKS          ;IS THE CLOCK FLAG SET?
      BMI     B0007        ;IF SO, CONTINUE ON WITH THE TEST
      INC     R3           ;IF NOT INCREMENT THE COUNTER LOCATION
      BNE     A0007        ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
E0007: ERROR 1
      BR      T0010       ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
B0007: MOV      #100,LKS    ;ENABLE INTERRUPT
      CLR      RO
C0007: INC     RO
      NOP
      BNE     C0007        ;STALL FOR TIME
E10007: ERROR 2
      BR      T0010       ;WAIT FOR INTERRUPT
D0007: TSTB    LKS          ;ERROR, DIDNT GET INTERRUPT
      BMI     T0010
E20007: ERROR 1
      ;ENTER HERE IF INTERRUPTED
      ;ERROR, INTERRUPT NOT CAUSED BY CLOCK

.SBTTL TEST THAT CLOCK WILL INTERRUPT WITH PROCESSOR AT PRIORITY 5
:TEST THAT CLOCK WILL INTERRUPT WITH PROCESSOR AT PRIORITY 5
:NOTE: IF THERE IS NO HARDWARE 'PSW' THIS TEST WILL BE SKIPPED
T0010: SCOPE
      MOV      #TRAP0,#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
      MOV      #340,#6       ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
      MOV      #R0010,$LPADR  ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
      MOV      #D0010,100     ;SET UP VECTOR RETURN POINTER
      MOV      #340,#102     ;NO INTERRUPTS ALLOWED AFTER THE FIRST ONE
R0010: CLR      LKS
      MOV      4,-(SP)       ;SAVE BUS TIMEOUT VECTOR CONTENTS
      MOV      #1$,4        ;SET A SERVICE 'PC' IN CASE TIMEOUT OCCURS
      MOV      #PR5,PS      ;DO WE HAVE A HARDWARE 'PSW'?
      BR      2$           ;BRANCH IF YES
1$:   CMP      (SP)+,(SP)+  ;RESTORE STACK FROM TIMEOUT
      MOV      (SP)+,4      ;RESTORE BUS TIMEOUT VECTOR CONTENTS
      BR      T0011
2$:   MOV      (SP)+,4      ;RESTORE BUS TIMEOUT VECTOR CONTENTS
      MOV      #1000,SP    ;INITIALIZE THE STACK POINTER
  
```

504	002612	005003				CLR	R3		; INITIALIZE A COUNTER LOCATION
505	002614	105737	177546			A0010:	TSTB	LKS	; IS THE CLOCK FLAG SET?
506	002620	100404					BMI	B0010	; IF SO, CONTINUE ON WITH THE TEST
507	002622	005203					INC	R3	; IF NOT INCREMENT THE COUNTER LOCATION
508	002624	001373					BNE	A0010	; AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
509	002626	104001				E0010:	ERROR 1		; CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
510	002630	000415					BR	T0011	
511	002632					B0010:			
512	002632	012737	000100	177546			MOV	#100,LKS	; ENABLE INTERRUPT
513	002640	005000					CLR	R0	
514	002642	005200				C0010:	INC	R0	
515	002644	000240					NOP		; STALL FOR SOME TIME
516	002646	001375					BNE	C0010	; WAIT FOR INTERRUPT
517	002650	104002				E10010:	ERROR 2		; ERROR, INTERRUPT FAILED TO OCCUR
518	002652	000404					BR	T0011	
519	002654	105737	177546			D0010:	TSTB	LKS	; ENTER HERE IF INTERRUPTED
520	002660	100401					BMI	T0011	
521	002662	104001				E20010:	ERROR 1		; ERROR, INTERRUPT DID NOT CLEAR THE CLOCK FLAG
522									
523									
524									
525									
526									
527									
528	002664	000004							
529	002666	012737	006266	000004					
530	002674	012737	000340	000006					
531	002702	012737	002710	001106					
532	002710	005037	177546						
533	002714	005003				R0011:	CLR	LKS	
534	002716	105737	177546				CLR	R3	; INITIALIZE A COUNTER LOCATION
535	002722	100404				A0011:	TSTB	LKS	; IS THE CLOCK FLAG SET?
536	002724	005203					BMI	B0011	; IF SO, CONTINUE ON WITH THE TEST
537	002726	001373					INC	R3	; IF NOT INCREMENT THE COUNTER LOCATION
538	002730	104001					BNE	A0011	; AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
539	002732	000443				E0011:	ERROR 1		; CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
540	002734						BR	T0012	
541	002734	012706	001000			B0011:			
542	002740	013746	000004				MOV	#1000,SP	; INITIALIZE THE STACK POINTER
543	002744	012737	002762	000004			MOV	4,-(SP)	; SAVE BUS TIMEOUT VECTOR CONTENTS
544	002752	012737	000300	177776			MOV	#1\$,4	; SET SERVICE 'PC' IN CASE TIMEOUT OCCURS
545	002760	000404					MOV	#PR6,PS	; DO WE HAVE A HARDWARE 'PSW'?
546	002762	022626					BR	2\$; BRANCH IF YES
547	002764	012637	000004			1\$:	CMP	(SP)+,(SP)+	; RESTORE STACK FROM BUS TIMEOUT
548	002770	000424					MOV	(SP)+,4	; RESTORE BUS TIMEOUT VECTOR CONTENTS
549	002772	012637	000004				BR	T0012	
550	002776	012737	003026	000100		2\$:	MOV	(SP)+,4	; RESTORE BUS TIMEOUT VECTOR CONTENTS
551	003004	012737	000100	177546			MOV	#E1011,100	; SET UP VECTOR RETURN
552	003012	005003					MOV	#100,LKS	; ENABLE INTERRUPT
553	003014	105737	177546				CLR	R3	; INITIALIZE A COUNTER LOCATION
554	003020	100404				C0011:	TSTB	LKS	; IS THE CLOCK FLAG SET?
555	003022	005203					BMI	D0011	; IF SO, CONTINUE CN WITH THE TEST
556	003024	001373					INC	R3	; IF NOT INCREMENT THE COUNTER LOCATION
557	003026	104001					BNE	C0011	; AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
558	003030	000715				E1011:	ERROR 1		; CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
559	003032						BR	T0011	
						D0011:			

.SBTTL TEST THAT CLOCK WILL NOT INTERRUPT WITH PROCESSOR PRIORITY 6
 ;TEST THAT CLOCK WILL NOT INTERRUPT WITH PROCESSOR PRIORITY 6
 ;NOTE: IF THERE IS NO HARDWARE 'PSW' THIS TEST WILL BE SKIPPED

T0011: SCOPE
 MOV #TRAP0,@#4 ; SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
 MOV #340,@#6 ; NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
 MOV #R0011,\$LPADR ; SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR

R0011: CLR LKS
 CLR R3 ; INITIALIZE A COUNTER LOCATION
 A0011: TSTB LKS ; IS THE CLOCK FLAG SET?
 BMI B0011 ; IF SO, CONTINUE ON WITH THE TEST
 INC R3 ; IF NOT INCREMENT THE COUNTER LOCATION
 BNE A0011 ; AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
 E0011: ERROR 1 ; CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
 BR T0012

B0011: MOV #1000,SP ; INITIALIZE THE STACK POINTER
 MOV 4,-(SP) ; SAVE BUS TIMEOUT VECTOR CONTENTS
 MOV #1\$,4 ; SET SERVICE 'PC' IN CASE TIMEOUT OCCURS
 MOV #PR6,PS ; DO WE HAVE A HARDWARE 'PSW'?
 BR 2\$; BRANCH IF YES
 1\$: CMP (SP)+,(SP)+ ; RESTORE STACK FROM BUS TIMEOUT
 MOV (SP)+,4 ; RESTORE BUS TIMEOUT VECTOR CONTENTS
 BR T0012

2\$: MOV (SP)+,4 ; RESTORE BUS TIMEOUT VECTOR CONTENTS
 MOV #E1011,100 ; SET UP VECTOR RETURN
 MOV #100,LKS ; ENABLE INTERRUPT
 CLR R3 ; INITIALIZE A COUNTER LOCATION
 C0011: TSTB LKS ; IS THE CLOCK FLAG SET?
 BMI D0011 ; IF SO, CONTINUE CN WITH THE TEST
 INC R3 ; IF NOT INCREMENT THE COUNTER LOCATION
 BNE C0011 ; AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
 E1011: ERROR 1 ; CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
 BR T0011

D0011:


```

560 003032 000240      NOP
561 003034 000240      NOP
562 003036 000401      BR      T0012      ;GIVE CLOCK EXTRA TIME TO INTERRUPT
563 003040 104003      E20011: ERROR 3      ;ERROR, CLOCK INTERRUPTED WITHOUT HAVING PRIORITY
564
565
566
567
568      .SBTTL TEST THAT RESET SETS CLOCK FLAG
      ;TEST THAT RESET SETS CLOCK FLAG
569 003042 000004      T0012: SCOPE
570 003044 012737 000200 001124      MOV      #200,$GDDAT      ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
571 003052 012737 006266 000004      MOV      #TRAP0,a#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
572 003060 012737 000340 000006      MOV      #340,a#6      ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
573 003066 012737 003074 001106      MOV      #R0012,$LPADR    ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
574 003074 005037 177546      R0012: CLR      LKS      ;CLEAR CLOCK FLAG
575 003100 005003      CLR      R3      ;INITIALIZE A COUNTER LOCATION
576 003102 105737 177546      A0012: TSTB     LKS      ;IS THE CLOCK FLAG SET?
577 003106 100404      BMI      B0012      ;IF SO, CONTINUE ON WITH THE TEST
578 003110 005203      INC      R3      ;IF NOT INCREMENT THE COUNTER LOCATION
579 003112 001373      BNE      A0012      ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
580 003114 104001      E0012: ERROR 1      ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
581 003116 000407      BR      T0013
582 003120
583 003120 005037 177546      B0012: CLR      LKS
584 003124 000005      RESET
585 003126 105737 177546      TSTB     LKS      ;SHOULD SET CLOCK FLAG
586 003132 100401      BMI      T0013
587 003134 104001      E10012: ERROR 1      ;ERROR, RESET DIDN'T SET CLOCK FLAG
588
589
590
591      .SBTTL TEST LINE CLOCK REPEATABILITY
592      ;TEST LINE CLOCK REPEATABILITY
593      ;MAKE SURE THAT OVER TWO EQUAL PERIODS OF TIME
594      ;THE CLOCK PUTS OUT THE SAME NUMBER OF PULSES
595 003136 000004      T0013: SCOPE
596 003140 005000      R0013: CLR      R0      ;CLEAR 1ST TIME COUNT
597 003142 005001      CLR      R1      ;CLEAR 1ST CLOCK COUNT
598 003144 013746 000340      MOV      PR7,-(SP)      ;;PUT NEW PS ON STACK
599 003150 012746 003156      MOV      #64$,-(SP)     ;;PUT NEW PC ON STACK
600 003154 000002      RTI      ;; POP NEW PC AND PS
601 003156
602 003156 012737 003156 001106      64$: R1013: MOV     #R1013,$LPADR ;ERROR IN NEXT FEW INSTRUCTIONS CAUSES A SHORT SCOPE LO
603 003164 005037 177546      CLR      LKS
604
605 003170 005003      ;SYNC ON CLOCK FLAG A COUPLE OF TIMES
606 003172 105737 177546      A0013: CLR      R3      ;INITIALIZE A COUNTER LOCATION
607 003176 100404      TSTB     LKS      ;IS THE CLOCK FLAG SET?
608 003200 005203      BMI      B0013      ;IF SO, CONTINUE ON WITH THE TEST
609 003202 001373      INC      R3      ;IF NOT INCREMENT THE COUNTER LOCATION
610 003204 104001      BNE      A0013      ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
611 003206 000524      E0013: ERROR 1      ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
612 003210      BR      T0014
613 003210 012737 003210 001106      B0013: MOV     #R2013,$LPADR ;MAKE SCOPE LOOP SHORT IN CASE OF AN ERROR
614 003216 005037 177546      R2013: CLR      LKS
615 003222 005003      CLR      R3      ;INITIALIZE A COUNTER LOCATION

```

616	003224	105737	177546		C0013:	TSTB	LKS		: IS THE CLOCK FLAG SET?
617	003230	100404				BMI	D0013		: IF SO, CONTINUE ON WITH THE TEST
618	003232	005203				INC	R3		: IF NOT INCREMENT THE COUNTER LOCATION
619	003234	001373				BNE	C0013		: AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
620	003236	104001			E10013:	ERROR 1			: CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
621	003240	000507				BR	T0014		
622	003242				D0013:				
623	003242	005037	177546			CLR	LKS		
624	003246	032737	000200	177546	1\$:	BIT	#200,LKS		: IS THE CLOCK FLAG SET ?
625	003254	001374				BNE	1\$: NO, WAIT FOR IT
626	003256	005037	177546			CLR	LKS		
627	003262	105737	177546		F0013:	TSTB	LKS		: IS CLOCK FLAG SET
628	003266	100003				BPL	G0013		: NO
629	003270	005201				INC	R1		: +1 TO CLOCK COUNT
630	003272	005037	177546			CLR	LKS		: CLEAR CLOCK IF SET
631	003276	005200			G0013:	INC	R0		: +1 TO TIME COUNT
632	003300	001370				BNE	F0013		: REPEAT UNTIL R0=0
633	003302	005000				CLR	R0		: CLEAR 2ND TIME COUNT
634	003304	005002				CLR	R2		: CLEAR 2ND CLOCK COUNT
635	003306	012737	003306	001106	R3013:	MOV	#R3013,\$LPADR		: INSURE A SHORT SCOPE LOOP
636	003314	005037	177546			CLR	LKS		
637									: SYNC ON CLOCK FLAG TWICE
638	003320	005003				CLR	R3		: INITIALIZE A COUNTER LOCATION
639	003322	105737	177546		H0013:	TSTB	LKS		: IS THE CLOCK FLAG SET?
640	003326	100404				BMI	J0013		: IF SO, CONTINUE ON WITH THE TEST
641	003330	005203				INC	R3		: IF NOT INCREMENT THE COUNTER LOCATION
642	003332	001373				BNE	H0013		: AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
643	003334	104001			E20013:	ERROR 1			: CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
644	003336	000450				BR	T0014		
645	003340				J0013:				
646	003340	012737	003340	001106	R4013:	MOV	#R4013,\$LPADR		: INSURE A SHORT SCOPE LOOP
647	003346	005037	177546			CLR	LKS		
648	003352	005003				CLR	R3		: INITIALIZE A COUNTER LOCATION
649	003354	105737	177546		K0013:	TSTB	LKS		: IS THE CLOCK FLAG SET?
650	003360	100404				BMI	L0013		: IF SO, CONTINUE ON WITH THE TEST
651	003362	005203				INC	R3		: IF NOT INCREMENT THE COUNTER LOCATION
652	003364	001373				BNE	K0013		: AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
653	003366	104001			E30013:	ERROR 1			: CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
654	003370	000433				BR	T0014		
655	003372				L0013:				
656	003372	012737	003140	001106		MOV	#R0013,\$LPADR		: MUST LOOP BACK TO BEGINING OF TEST IF EEROR COMES NOW
657	003400	005037	177546			CLR	LKS		
658	003404	032737	000200	177546	1\$:	BIT	#200,LKS		: IS CLOCK FLAG SET ?
659	003412	001374				BNE	1\$: NO, WAIT
660	003414	005037	177546			CLR	LKS		
661	003420	105737	177546		M0013:	TSTB	LKS		: IS CLOCK FLAG SET
662	003424	100003				BPL	N0013		: NO
663	003426	005202				INC	R2		: +1 TO CLOCK COUNT
664	003430	005037	177546			CLR	LKS		: CLEAR CLOCK IF SET
665	003434	005200			N0013:	INC	R0		: +1 TO TIME COUNT
666	003436	001370				BNE	M0013		: REPEAT UNTIL R0=0
667	003440	020102				CMP	R1,R2		: IS 1ST CLOCK COUNT EQUAL TO 2ND CLOCK COUNT?
668	003442	001406				BEQ	T0014		: YES
669	003444	010137	001162		E40013:	MOV	R1,\$REG1		: GET R1 READY FOR PRINTOUT
670	003450	010237	001164			MOV	R2,\$REG2		: GET R2 READY FOR PRINTOUT
671	003454	104004				ERROR 4			: ERROR, CLOCK FLAG OCCURRED DIFFERENT

```

672 003456 000240          NOP          ;NUMBER OF TIMES IN EQUAL PERIODS
673
674
675
676          .SBTTL LINE CLOCK REGISTER ADDRESSING TEST
677          ;LINE CLOCK REGISTER ADDRESSING TEST
678          ;TEST THAT THE "LKS" REGISTER CAN NOT BE ADDRESSED AS ANYTHING BUT 177546
679          ;SET A LOCATION THAT IS CLOSE(DIFFERS BY 1 BIT) TO THE LKS REGISTER
680          ;TO 100. IF THE LKS ALSO CHANGES, THEN SIGNAL AN ERROR
681 003460 000004          T0014: SCOPE
682 003462 005037 001124          CLR          $GDDAT
683 003466 012737 003542 001106          MOV          #R0014,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
684 003474 012737 157546 001120          MOV          #157546,$GDADR ;SAME AS "LKS" ADDRESS EXCEPT WITH BIT 13 CLEAR
685 003502 012737 003560 000004          MOV          #A0014,4 ;SETUP VECTOR IN CASE IT IS NONEXISTANT
686 003510 012737 000340 000006          MOV          #340,6
687 003516 005037 177546          CLR          LKS
688 003522 017737 175372 012364          MOV          @ $GDADR, SAVE ;SAVE CONTENTS
689 003530 013746 000340          MOV          PR7,-(SP) ;:PUT NEW PS ON STACK
690 003534 012746 003542          MOV          #64$,-(SP) ;:PUT NEW PC ON STACK
691 003540 000002          RTI ;: POP NEW PC AND PS
692 003542
693 003542 012706 001000          64$:
R0014: MOV          #1000,SP ;SETUP THE STACK
694 003546 005037 177546          CLR          LKS
695 003552 012777 000100 175340 I0014: MOV          #100,@ $GDADR ;SET THE "CLOSE" ADDRESS TO = 100
696 003560 032737 000100 177546 A0014: BIT          #100,LKS ;MAKE SURE THAT "LKS" WAS NOT AFFECTED
697 003566 001401          BEQ          B0014
698 003570 104005          E0014: ERROR 5 ;IT AFFECTED "LKS" -- ERROR
699 003572 005037 177546          B0014: CLR          LKS
700 003576 012737 003612 000004          MOV          #T0015,4
701 003604 013777 012364 175306          MOV          SAVE, @ $GDADR ;RESTORE CONTENTS
702
703
704
705          .SBTTL LINE CLOCK REGISTER ADDRESSING TEST
706          ;LINE CLOCK REGISTER ADDRESSING TEST
707 003612 000004          T0015: SCOPE
708 003614 012737 003700 000004          MOV          #A0015,4 ;SETUP VECTOR IN CASE IT IS NONEXISTANT
709 003622 012737 000340 000006          MOV          #340,6
710 003630 005037 001124          CLR          $GDDAT
711 003634 012737 003662 001106          MOV          #R0015,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
712 003642 012737 177146 001120          MOV          #177146,$GDADR ;SAME AS "LKS" ADDRESS EXCEPT WITH BIT 8 CLEAR
713 003650 013746 000340          MOV          PR7,-(SP) ;:PUT NEW PS ON STACK
714 003654 012746 003662          MOV          #64$,-(SP) ;:PUT NEW PC ON STACK
715 003660 000002          RTI ;: POP NEW PC AND PS
716 003662
717 003662 012706 001000          64$:
R0015: MOV          #1000,SP ;SETUP THE STACK
718 003666 005037 177546          CLR          LKS
719 003672 012777 000100 175220 I0015: MOV          #100,@ $GDADR ;SET THE "CLOSE" ADDRESS TO = 100
720 003700 032737 000100 177546 A0015: BIT          #100,LKS ;MAKE SURE THAT "LKS" WAS NOT AFFECTED
721 003706 001401          BEQ          B0015
722 003710 104005          E0015: ERROR 5 ;IT AFFECTED "LKS" -- ERROR
723 003712 005037 177546          B0015: CLR          LKS
724 003716 012737 003730 000004          MOV          #T0016,4
725 003724 005077 175170          CLR          @ $GDADR ;CLEAR OUT THE "CLOSE" ADDRESS
726
727

```

```

728
729
730
731 003730 000004
732 003732 012737 004016 000004
733 003740 012737 000340 000006
734 003746 005037 001124
735 003752 012737 004000 001106
736 003760 012737 177446 001120
737 003766 013746 000340
738 003772 012746 004000
739 003776 000002
740 004000
741 004000 012706 001000
742 004004 005037 177546
743 004010 012777 000100 175102
744 004016 032737 000100 177546
745 004024 001401
746 004026 104005
747 004030 005037 177546
748 004034 012737 004046 000004
749 004042 005077 175052
750
751
752
753
754
755 004046 000004
756 004050 012737 004134 000004
757 004056 012737 000340 000006
758 004064 005037 001124
759 004070 012737 004116 001106
760 004076 012737 177556 001120
761 004104 013746 000340
762 004110 012746 004116
763 004114 000002
764 004116
765 004116 012706 001000
766 004122 005037 177546
767 004126 012777 000100 174764
768 004134 032737 000100 177546
769 004142 001401
770 004144 104005
771 004146 005037 177546
772 004152 012737 004164 000004
773 004160 005077 174734
774
775
776
777
778
779 004164 000004
780 004166 012737 004252 000004
781 004174 012737 000340 000006
782 004202 005037 001124
783 004206 012737 004234 001106

```

```

.SBTTL LINE CLOCK REGISTER ADDRESSING TEST
;LINE CLOCK REGISTER ADDRESSING TEST
T0016: SCOPE
      MOV      #A0016,4          ;SETUP VECTOR IN CASE IT IS NONEXISTANT
      MOV      #340,6
      CLR      $GDDAT
      MOV      #R0016,$LPADR     ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
      MOV      #177446,$GDADR   ;SAME AS "LKS" ADDRESS EXCEPT WITH BIT 6 CLEAR
      MOV      PR7,-(SP)        ;;PUT NEW PS ON STACK
      MOV      #64$,-(SP)      ;;PUT NEW PC ON STACK
      RTI                          ;; POP NEW PC AND PS

64$:
R0016: MOV      #1000,SP        ;SETUP THE STACK
      CLR      LKS
I0016: MOV      #100,@$GDADR    ;SET THE "CLOSE" ADDRESS TO = 100
A0016: BIT      #100,LKS       ;MAKE SURE THAT "LKS" WAS NOT AFFECTED
      BEQ      B0016
E0016: ERROR 5                ;IT AFFECTED "LKS" -- ERROR
B0016: CLR      LKS
      MOV      #T0017 ,4
      CLR      @$GDADR         ;CLEAR OUT THE "CLOSE" ADDRESS

.SBTTL LINE CLOCK REGISTER ADDRESSING TEST
;LINE CLOCK REGISTER ADDRESSING TEST
T0017: SCOPE
      MOV      #A0017,4          ;SETUP VECTOR IN CASE IT IS NONEXISTANT
      MOV      #340,6
      CLR      $GDDAT
      MOV      #R0017,$LPADR     ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
      MOV      #177556,$GDADR   ;SAME AS "LKS" ADDRESS EXCEPT WITH BIT 3 SET
      MOV      PR7,-(SP)        ;;PUT NEW PS ON STACK
      MOV      #64$,-(SP)      ;;PUT NEW PC ON STACK
      RTI                          ;; POP NEW PC AND PS

64$:
R0017: MOV      #1000,SP        ;SETUP THE STACK
      CLR      LKS
I0017: MOV      #100,@$GDADR    ;SET THE "CLOSE" ADDRESS TO = 100
A0017: BIT      #100,LKS       ;MAKE SURE THAT "LKS" WAS NOT AFFECTED
      BEQ      B0017
E0017: ERROR 5                ;IT AFFECTED "LKS" -- ERROR
B0017: CLR      LKS
      MOV      #T0020 ,4
      CLR      @$GDADR         ;CLEAR OUT THE "CLOSE" ADDRESS

.SBTTL LINE CLOCK REGISTER ADDRESSING TEST
;LINE CLOCK REGISTER ADDRESSING TEST
T0020: SCOPE
      MOV      #A0020,4          ;SETUP VECTOR IN CASE IT IS NONEXISTANT
      MOV      #340,6
      CLR      $GDDAT
      MOV      #R0020,$LPADR     ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR

```

```

784 004214 012737 177566 001120      MOV      #177566,$GDADR  ;SAME AS "LKS" ADDRESS EXCEPT WITH BIT 4 SET
785 004222 013746 000340      MOV      PR7,-(SP)      ;;PUT NEW PS ON STACK
786 004226 012746 004234      MOV      #64$,-(SP)    ;;PUT NEW PC ON STACK
787 004232 000002      RTI                    ;; POP NEW PC AND PS
788 004234
789 004234 012706 001000      64$:
R0020: MOV      #1000,SP  ;SETUP THE STACK
790 004240 005037 177546      CLR      LKS
791 004244 012777 000100 174646  I0020: MOV      #100,@$GDADR ;SET THE "CLOSE" ADDRESS TO = 100
792 004252 032737 000100 177546  A0020: BIT      #100,LKS   ;MAKE SURE THAT "LKS" WAS NOT AFFECTED
793 004260 001401      BEQ      B0020
794 004262 104005      E0020: ERROR 5        ;IT AFFECTED "LKS" -- ERROR
795 004264 005037 177546      B0020: CLR      LKS
796 004270 012737 004302 000004      MOV      #T0021 ,4
797 004276 005077 174616      CLR      @$GDADR      ;CLEAR OUT THE "CLOSE" ADDRESS
798
799
800
801
802      .SBTTL  LINE CLOCK REGISTER ADDRESSING TEST
803      ;LINE CLOCK REGISTER ADDRESSING TEST
803 004302 000004      T0021: SCOPE
804 004304 012737 004370 000004      MOV      #A0021,4      ;SETUP VECTOR IN CASE IT IS NONEXISTANT
805 004312 012737 000340 000006      MOV      #340,6
806 004320 005037 001124      CLR      $GDDAT
807 004324 012737 004352 001106      MOV      #R0021,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
808 004332 012737 177746 001120      MOV      #177746,$GDADR ;SAME AS "LKS" ADDRESS EXCEPT WITH BIT 7 SET
809 004340 013746 000340      MOV      PR7,-(SP)    ;;PUT NEW PS ON STACK
810 004344 012746 004352      MOV      #64$,-(SP)  ;;PUT NEW PC ON STACK
811 004350 000002      RTI                    ;; POP NEW PC AND PS
812 004352
813 004352 012706 001000      64$:
R0021: MOV      #1000,SP  ;SETUP THE STACK
814 004356 005037 177546      CLR      LKS
815 004362 012777 000100 174530  I0021: MOV      #100,@$GDADR ;SET THE "CLOSE" ADDRESS TO = 100
816 004370 032737 000100 177546  A0021: BIT      #100,LKS   ;MAKE SURE THAT "LKS" WAS NOT AFFECTED
817 004376 001401      BEQ      B0021
818 004400 104005      E0021: ERROR 5        ;IT AFFECTED "LKS" -- ERROR
819 004402 005037 177546      B0021: CLR      LKS
820 004406 012737 004420 000004      MOV      #T0022 ,4
821 004414 005077 174500      CLR      @$GDADR      ;CLEAR OUT THE "CLOSE" ADDRESS
822
823
824
825      .SBTTL  LINE CLOCK REGISTER HIGH BYTE TEST
826      ;LINE CLOCK REGISTER HIGH BYTE TEST
827      ;MAKE SURE THE LKS REGISTER LOW BYTE RESPONDS TO THE HIGH BYTES ADDRESS
828 004420 000004      T0022: SCOPE
829 004422 000137 004534      JMP      T0023
830 004426 012737 006266 000004      MOV      #TRAP0,@#4    ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
831 004434 012737 000340 000006      MOV      #340,@#6     ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
832 004442 012737 000100 001124      MOV      #100,$GDDAT
833 004450 012737 004476 001106      MOV      #R0022,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
834 004456 012737 177547 001120      MOV      #177547,$GDADR ;HIGH BYTE OF THE LKS REGISTER
835 004464 013746 000340      MOV      PR7,-(SP)    ;;PUT NEW PS ON STACK
836 004470 012746 004476      MOV      #64$,-(SP)  ;;PUT NEW PC ON STACK
837 004474 000002      RTI                    ;; POP NEW PC AND PS
838 004476
839 004476 012706 001000      64$:
R0022: MOV      #1000,SP  ;SETUP THE STACK
    
```

```

840 004502 005037 177546
841 004506 112777 000100 174404 I0022: CLR LKS
842 004514 032737 000100 177546 I0022: MOV #100,@$GDADR ;SET THE HIGH BYTE ADDRESS TO = 100
843 004522 001001 I0022: BIT #100,LKS ;MAKE SURE THAT "LKS" LOW BYTE WAS AFFECTED
844 004524 104005 I0022: BNE A0022
845 004526 005037 177546 E0022: ERROR 5 ;SHOULD HAVE SET BIT 7. ERROR
846 004532 000005 A0022: CLR LKS
847
848
849
850
851 .SBTTL CLOCK FLAG BIT TEST
852 004534 000004 ;CLOCK FLAG BIT TEST
853 004536 012737 006266 000004 T0023: SCOPE
854 004544 012737 000340 000006 T0023: MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
855 004552 012737 000200 001124 T0023: MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
856 004560 012737 004566 001106 T0023: MOV #200,$GDDAT
857 004566 005037 177546 R0023: MOV #R0023,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
858 004572 005003 R0023: CLR LKS
859 004574 105737 177546 A0023: CLR R3 ;INITIALIZE A COUNTER LOCATION
860 004600 100404 A0023: TSTB LKS ;IS THE CLOCK FLAG SET?
861 004602 005203 A0023: BMI B0023 ;IF SO, CONTINUE ON WITH THE TEST
862 004604 001373 A0023: INC R3 ;IF NOT INCREMENT THE COUNTER LOCATION
863 004606 104001 E0023: BNE A0023 ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
864 004610 000410 E0023: ERROR 1 ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
865 004612 B0023: BR T0024
866 004612 012737 000200 177546 I0023: MOV #200,LKS ;MOVE A 1 INTO THE CLOCK FLAG BIT
867 004620 023737 177546 001124 I0023: CMP LKS,$GDDAT ;SHOULD NOT AFFECT THE FLAG BIT
868 004626 001401 E10023: BEQ T0024
869 004630 104001 E10023: ERROR 1 ;CLOCK FLAG DID NOT CLEAR
870
871
872
873 .SBTTL INTERRUPT TEST
874 004632 000004 T0024: SCOPE
875 004634 012737 006266 000004 T0024: MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
876 004642 012737 000340 000006 T0024: MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
877 004650 012737 000200 001124 T0024: MOV #200,$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
878 004656 012737 004676 001106 T0024: MOV #R0024,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
879 004664 012737 004736 000100 T0024: MOV #E0024,100
880 004672 005037 177546 R0024: CLR LKS ;ALLOW CLOCK INTERRUPTS
881 004676 R0024:
882 004676 013746 000340 R0024: MOV PR7,-(SP) ;;PUT NEW PS ON STACK
883 004702 012746 004710 R0024: MOV #64$,-(SP) ;;PUT NEW PC ON STACK
884 004706 000002 R0024: RTI ;;POP NEW PC AND PS
885 004710 64$:
886 004710 012737 000300 177546 A0024: MOV #300,LKS
887 004716 005227 000000 A0024: INC #0 ;WAIT FOR 20+ MS
888 004722 001375 A0024: BNE A0024 ;LOOP BACK IF NOT DONE WAITING
889 004724 000005 A0024: RESET ;RESET SHOULD CLEAR INTERRUPT ENABLE
890 004726 023737 001124 177546 A0024: CMP $GDDAT,LKS ;AND LEAVE THE CLOCK FLAG SET
891 004734 001401 A0024: BEQ T0025 ;GO ON TO THE NEXT TEST IF IT DID
892 004736 104001 E0024: ERROR 1 ;RESET SET INTERRUPT ENABLE OR CLEARED CLOCK FLAG
893
894
895

```

```

896      .SBTTL NO SACK TIMEOUT TEST
897      ;NO SACK TIMEOUT TEST
898 004740 000004 T0025: SCOPE
899 004742 012737 006266 000004 MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
900 004750 012737 000340 000006 MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
901 004756 012737 000300 001124 MOV #300,$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
902 004764 012737 005006 001106 MOV #R0025,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
903 004772 012737 000340 000102 MOV #340,102 ;NO INTERRUPTS AFTER THE FIRST ONE
904 005000 012737 005070 000100 MOV #C0025,100
905 005006 005037 177546 R0025: CLR LKS
906 005012 013746 000340 MOV PR7,-(SP) ;;PUT NEW PS ON STACK
907 005016 012746 005024 MOV #64$,-(SP) ;;PUT NEW PC ON STACK
908 005022 000002 RTI ;;POP NEW PC AND PS
909 005024
910 005024 005003 64$: CLR R3 ;INITIALIZE A COUNTER LOCATION
911 005026 105737 177546 A0025: TSTB LKS ;IS THE CLOCK FLAG SET?
912 005032 100404 BMI B0025 ;IF SO, CONTINUE ON WITH THE TEST
913 005034 005203 INC R3 ;IF NOT INCREMENT THE COUNTER LOCATION
914 005036 001373 BNE A0025 ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
915 005040 104001 E0025: ERROR 1 ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
916 005042 000417 BR T0026
917 005044
918 005044 005046 B0025: CLR -(SP) ;DROP CPU PRIORITY LEVEL
919 005046 012746 005054 MOV #1$,-(SP) ;SET RETURN 'PC' FROM THE RTI
920 005052 000002 RTI ;RETURN TO NEXT INSTRUCTION
921 005054 012737 000100 1$: MOV #100,LKS ;ENABLE CLOCK INTERRUPTS
922 005062 000001 WAIT ;THE ONLY WAY TO LEAVE HERE WITHOUT ERROR IS TO INTERRUPT
923 005064 104006 E10025: ERROR 6 ;IF IT ERROR 1S HERE ITS BECAUSE OF A "NO-SACK" TIMEOUT
924 005066 000405 BR T0026
925 005070 022737 000300 177546 C0025: CMP #300,LKS
926 005076 001401 BEQ T0026 ;FIND OUT WHAT THE INTERRUPT DID TO THE CLOCK STATUS REG
927 005100 104001 E20025: ERROR 1 ;IT CLEARED THE INTERRUPT ENABLE OR THE FLAG BIT
928
929
930
931      .SBTTL RESET TEST
932      ;RESET TEST
933 005102 000004 T0026: SCOPE
934 005104 012737 006266 000004 MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
935 005112 012737 000340 000006 MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
936 005120 012737 000200 001124 MOV #200,$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
937 005126 012737 005162 001106 MOV #R0026,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
938 005134 013746 000340 MOV PR7,-(SP) ;;PUT NEW PS ON STACK
939 005140 012746 005146 MOV #64$,-(SP) ;;PUT NEW PC ON STACK
940 005144 000002 RTI ;;POP NEW PC AND PS
941 005146
942 005146 012737 005220 000100 64$: MOV #E0026,100
943 005154 012737 000140 000102 MOV #140,102 ;SETUP STATUS FOR AFTER THE INTERRUPT
944 005162 005037 177546 R0026: CLR LKS
945 005166 012706 001000 MOV #1000,SP ;SETUP THE STACK
946 005172 012737 000100 177546 MOV #100,LKS ;SET INTERRUPT ENABLE BIT NOW
947 005200 005227 000000 A0026: INC #0 ;WAIT FOR CLOCK FLAG TO SET
948 005204 001375 BNE A0026
949 005206 000005 I0026: RESET ;RESET SHOULD NOT CLEAR THE FLAG
950 005210 023737 177546 001124 CMP LKS,$GDDAT ;FIND OUT IF ID DIT DID OR NOT
951 005216 001401 BEQ T0027

```

```

952 005220 104001 E0026: ERROR 1 ;RESET DID NOT INITIALIZE THE LKS WORD CORRECTLY
953
954
955
956
957
958
959 005222 000004
960 005224 012737 006266 000004
961 005232 012737 000340 000006
962 005240 012737 000200 001124
963 005246 012737 005266 001106
964 005254 005037 000102
965 005260 012737 005350 000100
966 005266 005037 177546
967 005272 013746 000340
968 005276 012746 005304
969 005302 000002
970 005304
971 005304 012706 001000
972 005310 005037 001234
973 005314 005237 001234
974 005320 001375
975 005322 012737 000300 177546
976 005330 012737 000200 177546
977 005336 023737 177546 001124
978 005344 001401
979 005346 104001
980
981
982
983
984
985 005350 000004
986 005352 012737 006266 000004
987 005360 012737 000340 000006
988 005366 005037 177546
989 005372 012737 000100 001124
990 005400 012737 005406 001106
991 005406 012737 005454 000100
992 005414 005037 000102
993 005420 005046
994 005422 012746 005430
995 005426 000002
996 005430 012706 001000
997 005434 005037 177546
998 005440 105037 001234
999 005444 012737 000100 177546
1000 005452 000001
1001 005454 012737 005506 000100
1002 005462 005046
1003 005464 012746 005472
1004 005470 000002
1005 005472 105237 001234
1006 005476 001375
1007 005500 005037 177546

.SBTTL CLOCK FLAG BIT TEST
;CLOCK FLAG BIT TEST
;MAKE SURE IT DOESNT CLEAR WHEN YOU TRY TO SET IT VIA A 'MOV' INSTRUCTION
T0027: SCOPE
MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
MOV #200,$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
MOV #R0027,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
CLR 102
MOV #T0030 ,100
R0027: CLR LKS
MOV PR7,-(SP) ;;PUT NEW PS ON STACK
MOV #64$,-(SP) ;;PUT NEW PC ON STACK
RTI ;; POP NEW PC AND PS

64$: MOV #1000,SP ;SETUP THE STACK
CLR WORD ;SETUP A COUNTER LOCATION TO = 0
A0027: INC WORD
BNE A0027 ;WASTE TIME LOOPING UNTIL THE COUNTER REACHES 0
MOV #300,LKS ;SET INTERRUPT ENABLE AND TRY TO SET THE CLOCK FLAG
MOV #200,LKS ;TRY TO SET IT AGAIN
CMP LKS,$GDDAT ;DID THE CLOCK FLAG STAY SET?
BEQ T0030 ;IF NOT GO ON TO THE NEXT TEST
E0027: ERROR 1 ;ERROR - MOVED A '1' INTO THE CLOCK FLAG BIT AND IT STAY

.SBTTL CLOCK FLAG AFTER INTERRUPT TEST
;SEE IF AN INTERRUPT CLEARS THE CLOCK FLAG
T0030: SCOPE
MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
CLR LKS ;NO CLOCK INTERRUPTS BEFORE WE ARE READY
MOV #100,$GDDAT ;HAVE GOOD DATA INFO READY FOR TYPEOUT IN CASE OF AN ERR
MOV #R0030,$LPADR ;INITIALIZE THE LOOPBACK ADDRESS IN CASE OF AN ERROR
R0030: MOV #A0030,100 ;SETUP CLOCK INTERRUPT VECTOR
CLR 102 ;PRIORITY LEVEL WILL ALLOW FURTHER INTERRUPTS
CLR -(SP) ;DROP CPU PRIORITY LEVEL
MOV #1$,-(SP) ;SET RETURN 'PC' FROM THE RTI
RTI ;RETURN TO NEXT INSTRUCTION
1$: MOV #1000,SP ;SETUP THE STACK
CLR LKS
CLRB WORD ;CLEAR OUT A COUNTER LOCATION
MOV #100,LKS ;ENABLE CLOCK INTERRUPTS NOW
WAIT ;WAIT FOR AN INTERRUPT
A0030: MOV #E0030,100 ;ERROR IF WE INTERRUPT AGAIN
CLR -(SP) ;DROP CPU PRIORITY LEVEL
MOV #B0030,-(SP) ;SET RETURN 'PC' FROM THE RTI
RTI ;RETURN TO NEXT INSTRUCTION
B0030: INCB WORD ;DO A NOTHING LOOP FOR A VERY SHORT PERIOD OF TIME
BNE B0030 ;WE SHOULD INCREMENT TO 0 LONG BEFORE AN INTERRUPT COMES
CLR LKS

```



```

1008 005504 000401
1009 005506 104001
1010
1011
1012
1013
1014
1015 005510 000004
1016 005512 012737 006266 000004
1017 005520 012737 000340 000006
1018 005526 012737 005534 001106
1019 005534 005037 177546
1020 005540 005003
1021 005542 105737 177546
1022 005546 100404
1023 005550 005203
1024 005552 001373
1025 005554 104001
1026 005556 000431
1027 005560
1028 005560 012706 001000
1029 005564 013746 000340
1030 005570 012746 005576
1031 005574 000002
1032 005576
1033 005576 012737 005554 000100
1034 005604 012737 000100 177546
1035 005612 005003
1036 005614 105737 177546
1037 005620 100404
1038 005622 005203
1039 005624 001373
1040 005626 104001
1041 005630 000404
1042 005632
1043 005632 000240
1044 005634 000240
1045 005636 000401
1046 005640 104003
1047
1048
1049
1050
1051
1052 005642 000004
1053 005644 012737 006266 000004
1054 005652 012737 000340 000006
1055 005660 012737 005666 001106
1056 005666 005037 177546
1057 005672 005003
1058 005674 105737 177546
1059 005700 100404
1060 005702 005203
1061 005704 001373
1062 005706 104001
1063 005710 000433

      BR      T0031
E0030:  ERROR 1      ; INTERRUPT DID NOT CLEAR THE CLOCK FLAG

      .SBTTL  NO INTERRUT AT PRIORITY 7 TEST
      ;TEST THAT CLOCK WILL NOT INTERRUPT WITH PROCESSR AT PRIORITY 7
T0031:  SCOPE
      MOV     #TRAP0,@#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
      MOV     #340,@#6        ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
      MOV     #R0031,$LPADR   ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
R0031:  CLR     LKS
      CLR     R3              ;INITIALIZE A COUNTER LOCATION
A0031:  TSTB   LKS            ;IS THE CLOCK FLAG SET?
      BMI    B0031           ;IF SO, CONTINUE ON WITH THE TEST
      INC    R3              ;IF NOT INCREMENT THE COUNTER LOCATION
      BNE    A0031           ;AND GO TEST THE CLOCK FLAG AGAIN, UNLESS...
E0031:  ERROR 1      ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
      BR      T0032
B0031:  MOV     #1000,SP      ;INITIALIZE THE STACK POINTER
      MOV     PR7,-(SP)      ;;PUT NEW PS ON STACK
      MOV     #64$,-(SP)    ;;PUT NEW PC ON STACK
      RTI                    ;; POP NEW PC AND PS
64$:   MOV     #E0031,100    ;SET UP VECTOR RETURN
      MOV     #100,LKS      ;ENABLE INTERRUPT
      CLR     R3            ;INITIALIZE A COUNTER LOCATION
C0031:  TSTB   LKS            ;IS THE CLOCK FLAG SET?
      BMI    D0031           ;IF SO, CONTINUE ON WITH THE TEST
      INC    R3              ;IF NOT INCREMENT THE COUNTER LOCATION
      BNE    C0031           ;AND GO TEST THE CLOCK FLAG AGAIN, UNLESS...
E10031: ERROR 1      ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
      BR      T0032
D0031:  NOP
      NOP                    ;GIVE CLOCK EXTRA TIME TO INTERRUPT
      BR      T0032
E20031: ERROR 3      ;ERROR, CLOCK INTERRUPTED WITHOUT HAVING PRIORITY

      .SBTTL  CC PUSH TEST FOR CLOCK INTERRUPTS
      ;TEST THAT CLOCK INTERRUPT PUSHES CONDITION CODES ONTO STACK
T0032:  SCOPE
      MOV     #TRAP0,@#4      ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
      MOV     #340,@#6        ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
      MOV     #R0032,$LPADR   ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
R0032:  CLR     LKS
      CLR     R3              ;INITIALIZE A COUNTER LOCATION
A0032:  TSTB   LKS            ;IS THE CLOCK FLAG SET?
      BMI    B0032           ;IF SO, CONTINUE ON WITH THE TEST
      INC    R3              ;IF NOT INCREMENT THE COUNTER LOCATION
      BNE    A0032           ;AND GO TEST THE CLOCK FLAG AGAIN, UNLESS...
E0032:  ERROR 1      ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
      BR      T0033
  
```

```

1064 005712
1065 005712 012706 001000
1066 005716 005037 000776
1067 005722 005037 000774
1068 005726 012737 005756 000100
1069 005734 012737 000100 177546
1070 005742 005046
1071 005744 012746 005752
1072 005750 000002
1073 005752 000277
1074 005754 000001
1075 005756 022737 000017 000776
1076 005764 001405
1077 005766 012737 000017 001124
1078 005774 104007
1079 005776 000721
1080
1081
1082
1083
1084
1085 006000 000004
1086 006002 012737 006266 000004
1087 006010 012737 000340 000006
1088 006016 012737 006024 001106
1089 006024 005037 177546
1090 006030 005003
1091 006032 105737 177546
1092 006036 100404
1093 006040 005203
1094 006042 001373
1095 006044 104001
1096 006046 000432
1097 006050
1098 006050 012706 001000
1099 006054 005037 000776
1100 006060 005037 000774
1101 006064 012737 006114 000100
1102 006072 012737 000100 177546
1103 006100 005046
1104 006102 012746 006110
1105 006106 000002
1106 006110 000277
1107 006112 000001
1108 006114 022737 006114 000774
1109 006122 001404
1110 006124 012737 006114 001124
1111 006132 104010
1112
1113
1114
1115
1116 006134 000004
1117 006136 005037 177546
1118 006142 000005
1119

B0032:
MOV #1000,SP ;INITIALIZE THE STACK POINTER
CLR BUF1
CLR BUF2
MOV #C0032,100 ;SET UP VECTOR RETURN
MOV #100,LKS ;ENABLE INTERRUPT
CLR -(SP) ;DROP CPU PRIORITY LEVEL
MOV #1$,-(SP) ;SET RETURN 'PC' FROM THE RTI
RTI ;RETURN TO NEXT INSTRUCTION
1$: +SEC!SEV!SEZ!SEN ;SET ALL CONDITION CODES
WAIT ;WAIT FOR INTERRUPT
C0032: CMP #17,BUF1
BEQ T0033
MOV #17,$GDDAT
ERROR 7 ;ERROR DID NOT PUSH CORRECT PS ONTO STACK
BR T0032

.SBTTL PC PUSH TEST FOR CLOCK INTERRUPTS
;TEST THAT CLOCK INTERRUPT PUSHES THE PROGRAM COUNTER ONTO STACK
T0033: SCOPE
MOV #TRAP0,@#4 ;SETUP VECOR IN CASE OF UNFORSEEN PROBLEMS
MOV #340,@#6 ;NO INTERRUPTS WHILE PRINTING FATAL MESSAGE
MOV #R0033,$LPADR ;SETUP LOOPBACK ADDRESS IN CASE OF AN ERROR
R0033: CLR LKS
CLR R3 ;INITIALIZE A COUNTER LOCATION
A0033: TSTB LKS ;IS THE CLOCK FLAG SET?
BMI B0033 ;IF SO, CONTINUE ON WITH THE TEST
INC R3 ;IF NOT INCREMENT THE COUNTER LOCATION
BNE A0033 ;AND GO TEST THE CLOCK FLAG AGAIN. UNLESS...
E0033: ERROR 1 ;CLOCK FLAG DID NOT SET AFTER A WAITING PERIOD > 20 MS
BR T0034
B0033: MOV #1000,SP ;INITIALIZE THE STACK POINTER
CLR BUF1
CLR BUF2
MOV #C0033,100 ;SET UP VECTOR RETURN
MOV #100,LKS ;ENABLE INTERRUPT
CLR -(SP) ;DROP CPU PRIORITY LEVEL
MOV #1$,-(SP) ;SET RETURN 'PC' FROM THE RTI
RTI ;RETURN TO NEXT INSTRUCTION
1$: +SEC!SEV!SEZ!SEN ;SET ALL CONDITION CODES
WAIT ;WAIT FOR INTERRUPT
C0033: CMP #C0033,BUF2
BEQ T0034
E10033: MOV #C0033,$GDDAT
ERROR 10 ;ERROR, DID NOT PUSH CORRECT PC ONTO STACK

.SBTTL END OF PASS INDICATING
T0034: SCOPE
CLR LKS ;TURN THE CLOCK OFF
RESET ;TURN EVERYTHING OFF

```

```

1120 ;*****
1121
1122 .SBTTL END OF PASS ROUTINE
1123
1124 ;*INCREMENT THE PASS NUMBER ($PASS)
1125 ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
1126 ;*IF THERES A MONITOR GO TO IT
1127 ;*IF THERE ISN'T JUMP TO T0001
1128
1129 006144 $EOP:
1130 006144 000004 SCOPE
1131 006146 005037 001102 CLR $TSTNM ;;ZERO THE TEST NUMBER
1132 006152 005037 001220 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
1133 006156 005237 001100 INC $PASS ;;INCREMENT THE PASS NUMBER
1134 006162 042737 100000 001100 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
1135 006170 005327 DEC (PC)+ ;;LOOP?
1136 006172 000001 $EOPCT: .WORD 1
1137 006174 003022 BGT $DOAGN ;;YES
1138 006176 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
1139 006200 000001 $SENDCT: .WORD 1
1140 006202 006172 $EOPCT
1141 006204 104400 006246 TYPE ,SENDMG ;;TYPE "END PASS #"
1142 006210 013746 001100 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
1143 006214 104404 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
1144 006216 104400 006265 TYPE ,SENULL ;;TYPE A NULL CHARACTER
1145 006222
1146
1147 006222 013700 000042 MOV @#42,R0 ;;GET MONITOR ADDRESS
1148 006226 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
1149 006230 000005 RESET ;;CLEAR THE WORLD
1150 006232 004710 $SENDAD: JSR PC,(R0) ;;GO TO MONITOR
1151 006234 000240 NOP ;;SAVE ROOM
1152 006236 000240 NOP ;;FOR
1153 006240 000240 NOP ;;ACT11
1154 006242
1155 006242 000137 001672 $DOAGN:
1156 006246 005015 047105 020104 JMP @#T0001 ;;RETURN
1157 006254 040520 051523 021440 $SENDMG: .ASCIZ <15><12>/END PASS #/
1158 006262 000
1159 006263 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
1160 006266 005046 TRAPO: CLR -(SP)
1161 006270 004737 006612 JSR PC,$TYPE ;PRINTOUT"TRAPPED TO 4 FROM "
1162 006274 010371 TRPMES ;ADDRESS OF THE MESSAGE
1163 006276 011646 MOV (SP),-(SP) ;GET THE ADDRESS WHERE THE TRAP OCCURED
1164 006300 162716 000002 SUB #2,(SP) ;MAKE IT RIGHT
1165 006304 104401 TYPOC ;TYPE OUT ADDRESS IN OCTAL
1166 006306 104400 001231 TYPE ,$CRLF ;PRINTOUT A CARRIAGE RETURN-LINE FEED
1167 006312 005046 CLR -(SP)
1168 006314 004737 006612 JSR PC,$TYPE ;PRINTOUT RESTARTING MESSAGE
1169 006320 010432 TRPM2S ;ADDRESS OF RESTART MESSAGE
1170 006322 000240 NOP
1171 006324 000240 NOP
1172 006326 000240 NOP
1173 006330 000240 NOP
1174 006332 000005 RESET
1175 006334 000137 001450 JMP START

```

```

1176 ;*****
1177
1178 .SBTTL SCOPE HANDLER ROUTINE
1179
1180 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1181 ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1182 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1183 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1184 ;*SW14=1 LOOP ON TEST
1185 ;*SW11=1 INHIBIT ITERATIONS
1186 ;*SW09=1 LOOP ON ERROR
1187 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
1188 ;*CALL
1189 ;* SCOPE ;;SCOPE=IOT
1190
1191 006340 $SCOPE:
1192 006340 000240 NOP
1193 006342 032777 040000 172566 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
1194 006350 001111 BNE $OVER ;;YES IF SW14=1
1195 ;#####START OF CODE FOR THE XOR TESTER#####
1196 006352 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1197 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1198 006354 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1199 006360 012737 006400 000004 MOV #5$,@ERRVEC ;;SET FOR TIMEOUT
1200 006366 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
1201 006372 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
1202 006376 000463 BR $SVLAD ;;GO TO THE NEXT TEST
1203 006400 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
1204 006402 012637 000004 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
1205 006406 000423 BR 7$ ;;LOOP ON THE PRESENT TEST
1206 006410 6$:;#####END OF CODE FOR THE XOR TESTER#####
1207 006410 032777 000400 172520 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
1208 006416 001404 BEQ 2$ ;;BR IF NO
1209 006420 127737 172512 001102 CMPB @SWR,$STNM ;;ON THE RIGHT TEST? SWR<7:0>
1210 006426 001462 BEQ $OVER ;;BR IF YES
1211 006430 105737 001103 2$: TSTB SERFLG ;;HAS AN ERROR OCCURRED?
1212 006434 001421 BEQ 3$ ;;BR IF NO
1213 006436 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1214 006444 101015 BHI 3$ ;;BR IF NO
1215 006446 032777 001000 172462 BIT #BIT09,@SWR ;;LOOP ON ERROR?
1216 006454 001404 BEQ 4$ ;;BR IF NO
1217 006456 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
1218 006464 000443 BR $OVER
1219 006466 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
1220 006472 005037 001220 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1221 006476 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
1222 006500 032777 004000 172430 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
1223 006506 001011 BNE 1$ ;;BR IF YES
1224 006510 005737 001100 TST $PASS ;;IF FIRST PASS OF PROGRAM
1225 006514 001406 BEQ 1$ ;; INHIBIT ITERATIONS
1226 006516 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
1227 006522 023737 001220 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
1228 006530 002021 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
1229 006532 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
1230 006540 013737 006610 001220 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
1231 006546 105237 001102 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS

```

```

1232 006552 011637 001106      MOV      (SP), $LPADR      ;;SAVE SCOPE LOOP ADDRESS
1233 006556 011637 001110      MOV      (SP), $LPERR     ;;SAVE ERROR LOOP ADDRESS
1234 006562 005037 001222      CLR      $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1235 006566 112737 000001 001115  MOVB     #1, $ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1236 006574 013777 001102 172336 $OVER:  MOV      $TSTNM, @DISPLAY ;;DISPLAY TEST NUMBER
1237 006602 013716 001106      MOV      $LPADR, (SP)    ;;FUDGE RETURN ADDRESS
1238 006606 000002      RTI                     ;;FIXES PS
1239 006610 000010      $MXCNT: 10              ;;MAX. NUMBER OF ITERATIONS
1240      ;*****
1241
1242      .SBTTL  TYPE ROUTINE
1243
1244      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1245      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1246      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1247      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1248      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1249      ;*
1250      ;*CALL:
1251      ;*1) USING A TRAP INSTRUCTION
1252      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1253      ;*OR
1254      ;*      TYPE
1255      ;*      MESADR
1256      ;*
1257
1258 006612 105737 001155      $TYPE:  TSTB     $TPFLG      ;;IS THERE A TERMINAL?
1259 006616 100002      BPL      1$              ;;BR IF YES
1260 006620 000000      HALT                    ;;HALT HERE IF NO TERMINAL
1261 006622 000407      BR      3$              ;;LEAVE
1262 006624 010046      1$:  MOV      RO, -(SP)    ;;SAVE RO
1263 006626 017600 000002      MOV      @2(SP), RO      ;;GET ADDRESS OF ASCIZ STRING
1264 006632 112046      2$:  MOVB     (RO)+, -(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1265 006634 001005      BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1266 006636 005726      TST     (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
1267 006640 012600      60$:  MOV      (SP)+, RO    ;;RESTORE RO
1268 006642 062716 000002      3$:  ADD      #2, (SP)     ;;ADJUST RETURN PC
1269 006646 000002      RTI                     ;;RETURN
1270 006650 122716 000011      4$:  CMPB     #HT, (SP)    ;;BRANCH IF <HT>
1271 006654 001426      BEQ     8$
1272 006656 122716 000200      CMPB     #TCRLF, (SP)    ;;BRANCH IF NOT <CRLF>
1273 006662 001004      BNE     5$
1274 006664 005726      TST     (SP)+          ;;POP <CR><LF> EQUIV
1275 006666 104400      TYPE                    ;;TYPE A CR AND LF
1276 006670 001231      $CRLF
1277 006672 000757      BR      2$              ;;GET NEXT CHARACTER
1278 006674 004737 006756      5$:  JSR      PC, $TYPEPC  ;;GO TYPE THIS CHARACTER
1279 006700 123726 001154      6$:  CMPB     $FILLC, (SP)+ ;;IS IT TIME FOR FILLER CHARS.?
1280 006704 001352      BNE     2$              ;;IF NO GO GET NEXT CHAR.
1281 006706 013746 001152      MOV      $NULL, -(SP)   ;;GET # OF FILLER CHARS. NEEDED
1282      ;*AND THE NULL CHAR.
1283 006712 105366 000001      7$:  DECB     1(SP)        ;;DOES A NULL NEED TO BE TYPED?
1284 006716 002770      BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1285 006720 004737 006756      JSR      PC, $TYPEPC  ;;GO TYPE A NULL
1286 006724 105337 007022      DECB     $CHARCNT      ;;DO NOT COUNT AS A COUNT
1287 006730 000770      BR      7$              ;;LOOP

```

```

1288
1289
1290
1291 006732 112716 000040
1292 006736 004737 006756
1293 006742 132737 000007 007022
1294 006750 001372
1295 006752 005726
1296 006754 000726
1297 006756 105777 172164
1298 006762 100375
1299 006764 116677 000002 172156
1300 006772 122766 000015 000002
1301 007000 001003
1302 007002 105037 007022
1303 007006 000406
1304 007010 122766 000012 000002
1305 007016 002002
1306 007020 105227
1307 007022 000000
1308 007024 000207
1309
1310 000011
1311 000200
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326 007026
1327 007026 010046
1328 007030 010146
1329 007032 010246
1330 007034 010346
1331 007036 010546
1332 007040 012746 020200
1333 007044 016605 000020
1334 007050 100004
1335 007052 005405
1336 007054 112766 000055 000001
1337 007062 005000
1338 007064 012703 007242
1339 007070 112723 000040
1340 007074 005002
1341 007076 016001 007232
1342 007102 160105
1343 007104 002402

;HORIZONTAL TAB PROCESSOR
8$:   MOVB   #40,(SP)      ;;REPLACE TAB WITH SPACE
9$:   JSR    PC,$TYPEC    ;;TYPE A SPACE
      BITB   #7,$CHARCNT  ;;BRANCH IF NOT AT
      BNE   9$           ;;TAB STOP
      TST   (SP)+        ;;POP SPACE OFF STACK
      BR    2$           ;;GET NEXT CHARACTER
$TYPEC: TSTB  @STPS       ;;WAIT UNTIL PRINTER IS READY
      BPL  $TYPEC
      MOVB  2(SP),@STPB   ;;LOAD CHAR TO BE TYPED INTO DATA REG.
      CMPB  #15,2(SP)    ;;BRANCH IF
      BNE  1$           ;;NOT <CR>
      CLRB  $CHARCNT     ;;
      BR    $TYPEX      ;;EXIT
1$:   CMPB  #12,2(SP)    ;;BRANCH IF
      BGE  $TYPEX      ;;<LF>
      INCB  (PC)+        ;;INC SPACE
      $CHARCNT: .WORD 0  ;;COUNT
$TYPEX: RTS    PC
      ;; EQUATES
THT=11
TCRLF=200

;*****
.SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
; *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
; *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
; *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
; *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
; *REPLACED WITH SPACES.
; *CALL:
; *   MOV    NUM,-(SP)    ;;PUT THE BINARY NUMBER ON THE STACK
; *   TYPDS  ;;GO TO THE ROUTINE
$TYPDS:
      MOV    R0,-(SP)    ;;PUSH R0 ON STACK
      MOV    R1,-(SP)    ;;PUSH R1 ON STACK
      MOV    R2,-(SP)    ;;PUSH R2 ON STACK
      MOV    R3,-(SP)    ;;PUSH R3 ON STACK
      MOV    R5,-(SP)    ;;PUSH R5 ON STACK
      MOV    #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
      MOV    20(SP),R5   ;;GET THE INPUT NUMBER
      BPL   1$          ;;BR IF INPUT IS POS.
      NEG   R5          ;;MAKE THE BINARY NUMBER POS.
      MOVB  #'-,1(SP)   ;;MAKE THE ASCII NUMBER NEG.
1$:   CLR    R0          ;;ZERO THE CONSTANTS INDEX
      MOV   #SDBLK,R3   ;;SETUP THE OUTPUT POINTER
      MOVB  #'',(R3)+  ;;SET THE FIRST CHARACTER TO A BLANK
2$:   CLR    R2          ;;CLEAR THE BCD NUMBER
      MOV   $DTBL(R0),R1 ;;GET THE CONSTANT
3$:   SUB   R1,R5       ;;FORM THIS BCD DIGIT
4$:   BLT   4$          ;;BR IF DONE

```

```

1344 007106 005202          INC      R2          ;;INCREASE THE BCD DIGIT BY 1
1345 007110 000774          BR       3$
1346 007112 060105          4$: ADD    R1,R5          ;;ADD BACK THE CONSTANT
1347 007114 005702          TST     R2          ;;CHECK IF BCD DIGIT=0
1348 007116 001002          BNE     5$          ;;FALL THROUGH IF 0
1349 007120 105716          TSTB   (SP)        ;;STILL DOING LEADING 0'S?
1350 007122 100407          BMI     7$          ;;BR IF YES
1351 007124 106316          5$: ASLB   (SP)        ;;MSD?
1352 007126 103003          BCC     6$          ;;BR IF NO
1353 007130 116663 000001 177777 MOVB    1(SP),-1(R3) ;;YES--SET THE SIGN
1354 007136 052702 000060 6$: BIS   #'0,R2      ;;MAKE THE BCD DIGIT ASCII
1355 007142 052702 000040 7$: BIS   #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1356 007146 110223          MOVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1357 007150 005720          TST    (R0)+      ;;JUST INCREMENTING
1358 007152 020027 000010  CMP    RO,#10     ;;CHECK THE TABLE INDEX
1359 007156 002746          BLT    2$          ;;GO DO THE NEXT DIGIT
1360 007160 003002          BGT    8$          ;;GO TO EXIT
1361 007162 010502          MOV    R5,R2      ;;GET THE LSD
1362 007164 000764          BR     6$          ;;GO CHANGE TO ASCII
1363 007166 105726          8$: TSTB  (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
1364 007170 100003          BPL    9$          ;;BR IF NO
1365 007172 116663 177777 177776 MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
1366 007200 105013          9$: CLRB  (R3)      ;;SET THE TERMINATOR
1367 007202 012605          MOV    (SP)+,R5   ;;POP STACK INTO R5
1368 007204 012603          MOV    (SP)+,R3   ;;POP STACK INTO R3
1369 007206 012602          MOV    (SP)+,R2   ;;POP STACK INTO R2
1370 007210 012601          MOV    (SP)+,R1   ;;POP STACK INTO R1
1371 007212 012600          MOV    (SP)+,R0   ;;POP STACK INTO R0
1372 007214 104400 007242  TYPE    ,SDBLK      ;;NOW TYPE THE NUMBER
1373 007220 016666 000002 000004 MOV    2(SP),4(SP) ;;ADJUST THE STACK
1374 007226 012616          MOV    (SP)+,(SP)
1375 007230 000002          RTI
1376 007232 023420          SDTBL: 10000.
1377 007234 001750          1000.
1378 007236 000144          100.
1379 007240 000012          10.
1380 007242 000004          SDBLK: .BLKW 4
1381
1382
1383
1384          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1385          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1386          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1387          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1388          ;*CALL:
1389          ;*      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED
1390          ;*      TYPOS          ;;CALL FOR TYPEOUT
1391          ;*      .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1392          ;*      .BYTE  M              ;;M=1 OR 0
1393          ;*
1394          ;*
1395          ;*
1396          ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1397          ;*$TYPOS OR $TYPOC
1398          ;*CALL:
1399          ;*      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED

```

```

1400          :*          TYPON          ;;CALL FOR TYPEOUT
1401          :*
1402          :*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1403          :*CALL:
1404          :*          MOV          NUM,-(SP)          ;;NUMBER TO BE TYPED
1405          :*          TYPOC          ;;CALL FOR TYPEOUT
1406
1407 007252 017646 000000          $TYPOS: MOV          @ (SP),-(SP)          ;;PICKUP THE MODE
1408 007256 116637 000001 007475          MOVB          1(SP), $OFILL          ;;LOAD ZERO FILL SWITCH
1409 007264 112637 007477          MOVB          (SP)+, $SOMODE+1          ;;NUMBER OF DIGITS TO TYPE
1410 007270 062716 000002          ADD          #2, (SP)          ;;ADJUST RETURN ADDRESS
1411 007274 000406          BR          $TYPON
1412 007276 112737 000001 007475          $TYPOC: MOVB          #1, $OFILL          ;;SET THE ZERO FILL SWITCH
1413 007304 112737 000006 007477          MOVB          #6, $SOMODE+1          ;;SET FOR SIX(6) DIGITS
1414 007312 112737 000005 007474          $TYPON: MOVB          #5, $SOCNT          ;;SET THE ITERATION COUNT
1415 007320 010346          MOV          R3, -(SP)          ;;SAVE R3
1416 007322 010446          MOV          R4, -(SP)          ;;SAVE R4
1417 007324 010546          MOV          R5, -(SP)          ;;SAVE R5
1418 007326 113704 007477          MOVB          $SOMODE+1, R4          ;;GET THE NUMBER OF DIGITS TO TYPE
1419 007332 005404          NEG          R4
1420 007334 062704 000006          ADD          #6, R4          ;;SUBTRACT IT FOR MAX. ALLOWED
1421 007340 110437 007476          MOVB          R4, $SOMODE          ;;SAVE IT FOR USE
1422 007344 113704 007475          MOVB          $OFILL, R4          ;;GET THE ZERO FILL SWITCH
1423 007350 016605 000012          MOV          12(SP), R5          ;;PICKUP THE INPUT NUMBER
1424 007354 005003          CLR          R3          ;;CLEAR THE OUTPUT WORD
1425 007356 006105          1$: ROL          R5          ;;ROTATE MSB INTO 'C'
1426 007360 000404          BR          3$          ;;GO DO MSB
1427 007362 006105          2$: ROL          R5          ;;FORM THIS DIGIT
1428 007364 006105          ROL          R5
1429 007366 006105          ROL          R5
1430 007370 010503          MOV          R5, R3
1431 007372 006103          3$: ROL          R3          ;;GET LSB OF THIS DIGIT
1432 007374 105337 007476          DECB          $SOMODE          ;;TYPE THIS DIGIT?
1433 007400 100016          BPL          7$          ;;BR IF NO
1434 007402 042703 177770          BIC          #177770, R3          ;;GET RID OF JUNK
1435 007406 001002          BNE          4$          ;;TEST FOR 0
1436 007410 005704          TST          R4          ;;SUPPRESS THIS 0?
1437 007412 001403          BEQ          5$          ;;BR IF YES
1438 007414 005204          4$: INC          R4          ;;DON'T SUPPRESS ANYMORE 0'S
1439 007416 052703 000060          BIS          #'0, R3          ;;MAKE THIS DIGIT ASCII
1440 007422 052703 000040          5$: BIS          #' , R3          ;;MAKE ASCII IF NOT ALREADY
1441 007426 110337 007472          MOVB          R3, 8$          ;;SAVE FOR TYPING
1442 007432 104400 007472          TYPE          , 8$          ;;GO TYPE THIS DIGIT
1443 007436 105337 007474          7$: DECB          $SOCNT          ;;COUNT BY 1
1444 007442 003347          BGT          2$          ;;BR IF MORE TO DO
1445 007444 002402          BLT          6$          ;;BR IF DONE
1446 007446 005204          INC          R4          ;;INSURE LAST DIGIT ISN'T A BLANK
1447 007450 000744          BR          2$          ;;GO DO THE LAST DIGIT
1448 007452 012605          6$: MOV          (SP)+, R5          ;;RESTORE R5
1449 007454 012604          MOV          (SP)+, R4          ;;RESTORE R4
1450 007456 012603          MOV          (SP)+, R3          ;;RESTORE R3
1451 007460 016666 000002 000004          MOV          2(SP), 4(SP)          ;;SET THE STACK FOR RETURNING
1452 007466 012616          MOV          (SP)+, (SP)
1453 007470 000002          RTI
1454 007472 000          8$: .BYTE          0          ;;RETURN
1455 007473 000          .BYTE          0          ;;STORAGE FOR ASCII DIGIT
          ;;TERMINATOR FOR TYPE ROUTINE

```



```

1456 007474 000
1457 007475 000
1458 007476 000000
1459
1460
1461
1462
1463
1464
1465
1466
1467 007500
1468 007500 104400 001231
1469 007504 010046
1470 007506 005000
1471 007510 153700 001114
1472 007514 001004
1473
1474 007516 013746 001116
1475
1476 007522 104401
1477 007524 000426
1478 007526 005300
1479 007530 006300
1480 007532 006300
1481 007534 006300
1482 007536 062700 001236
1483 007542 012037 007552
1484 007546 001404
1485 007550 104400
1486 007552 000000
1487 007554 104400 001231
1488 007560 012037 007570
1489 007564 001404
1490 007566 104400
1491 007570 000000
1492 007572 104400 001231
1493 007576 011000
1494 007600 001004
1495 007602 012600
1496 007604 104400 001231
1497 007610 000207
1498 007612
1499 007612 013046
1500 007614 104401
1501 007616 005710
1502 007620 001770
1503 007622 104400 007630
1504 007626 000771
1505 007630 020040 000
1506 007634
1507
1508
1509
1510
1511
  
```

```

$OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
$OFILL: .BYTE 0 ;;ZERO FILL SWITCH
$OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
;*****
  
```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

; *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
; *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
  
```

\$ERRTYP:

```

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

.SBTTL ERROR HANDLER ROUTINE

```

; *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
  
```

```
1512 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1513 ;*AND GO TO $ERRTYP ON ERROR
1514 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1515 ;*SW15=1 HALT ON ERROR
1516 ;*SW13=1 INHIBIT ERROR TYPEOUTS
1517 ;*SW10=1 BELL ON ERROR
1518 ;*SW09=1 LOOP ON ERROR
1519 ;*CALL
1520 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1521
1522 007634 $ERROR:
1523 007634 010637 001174 MOV SP,$REG6 ;GET THE CURRENT STACK POINTER VALUE
1524 007640 162737 000004 001174 SUB #4,$REG6 ;RESTORE IT TO ITS 'PRE ERROR TRAP' VALUE FOR PR
1525 007646 016637 000002 001176 MOV 2(SP),$REG7 ;GET THE PS OFF OF THE STACK
1526 007654 005037 001172 CLR $REG5 ;PREPARE '$REG5' TO HOLD THE TEST #
1527 007660 113737 001102 001172 MOV $TSTNM,$REG5 ;TEST # IS HELD IN THE LOW BYTE OF 'TSTNM'
1528 007666 010037 001160 MOV RO,$REG0 ;MOST OF THE TIME RO HAS GOOD STUFF IN IT ALSO
1529 007672 105237 001103 7$: INCB $ERFLG ;;SET THE ERROR FLAG
1530 007676 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
1531 007700 013777 001102 171232 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
1532 007706 032777 002000 171222 BIT #BIT10,@SWR ;;BELL ON ERROR?
1533 007714 001402 BEQ 1$ ;;NO - SKIP
1534 007716 104400 001224 TYPE ,SBELL ;;RING BELL
1535 007722 005237 001112 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
1536 007726 011637 001116 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
1537 007732 162737 000002 001116 SUB #2,$ERRPC
1538 007740 117737 171152 001114 MOV $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
1539 007746 032777 020000 171162 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
1540 007754 001004 BNE 20$ ;;SKIP TYPEOUTS
1541 007756 004737 007500 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
1542 007762 104400 001231 TYPE ,SCRLF
1543 007766 20$:
1544 007766 005777 171144 2$: TST @SWR ;;HALT ON ERROR
1545 007772 100001 BPL 31$ ;;SKIP IF CONTINUE
1546 007774 000000 HALT ;;HALT ON ERROR!
1547 007776 022737 006232 000042 31$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
1548 010004 001001 BNE 3$ ;;BRANCH IF NO
1549 010006 000000 HALT ;;YES
1550 010010 032777 001000 171120 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
1551 010016 001402 BEQ 4$ ;;BR IF NO
1552 010020 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
1553 010024 005737 001222 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
1554 010030 001402 BEQ 5$ ;;BR IF NONE
1555 010032 013716 001222 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
1556 010036 5$:
1557 010036 000002 RTI ;;RETURN
1558 ;*****
1559
1560 .SBTTL TRAP DECODER
1561
1562 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
1563 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1564 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1565 ;*GO TO THAT ROUTINE.
1566
1567 010040 010046 $TRAP: MOV RO,-(SP) ;;SAVE RO
```

1568 010042 016600 000002
 1569 010046 005740
 1570 010050 111000
 1571 010052 006300
 1572 010054 016000 010062
 1573 010060 000200
 1574
 1575
 1576
 1577
 1578
 1579
 1580
 1581
 1582
 1583 010062
 1584 010062 006612
 1585 010064 007276
 1586 010066 007252
 1587 010070 007312
 1588 010072 007026
 1589
 1590
 1591
 1592
 1593
 1594 010074 012737 010236 000024
 1595 010102 012737 000340 000026
 1596 010110 010046
 1597 010112 010146
 1598 010114 010246
 1599 010116 010346
 1600 010120 010446
 1601 010122 010546
 1602 010124 013746 010456
 1603 010130 013746 010460
 1604 010134 013746 010254
 1605 010140 013746 001672
 1606 010144 010637 010242
 1607 010150 012737 010162 000024
 1608 010156 000000
 1609 010160 000776
 1610
 1611
 1612 010162 013706 010242
 1613 010166 005037 010242
 1614 010172 005237 010242
 1615 010176 001375
 1616 010200 012605
 1617 010202 012604
 1618 010204 012603
 1619 010206 012602
 1620 010210 012601
 1621 010212 012600
 1622 010214 012737 010074 000024
 1623 010222 012737 000340 000026

```

MOV 2(SP),R0      ;;GET TRAP ADDRESS
TST -(R0)         ;;BACKUP BY 2
MOVB (R0),R0      ;;GET RIGHT BYTE OF TRAP
ASL R0            ;;POSITION FOR INDEXING
MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0            ;;GO TO ROUTINE
  
```

.SBTTL TRAP TABLE

;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 ;*BY THE "TRAP" INSTRUCTION.

: ROUTINE

```

:-----
: $TRPAD:
: $TYPE      ;;CALL=TYPE      TRAP+0(104400) TTY TYPEOUT ROUTINE
: $TYPOC     ;;CALL=TYPOC     TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
: $TYPOS     ;;CALL=TYPOS     TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)
: $TYPON     ;;CALL=TYPON     TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)
: $TYPDS     ;;CALL=TYPDS     TRAP+4(104404) TYPE DECIMAL NUMBER (WITH SIGN)
:*****
  
```

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

```

$PWRDN: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST UP
MOV #340,@#PWRVEC+2 ;;PRIO:7
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV POWPUS,-(SP) ;;PUSH POWPUS ON STACK
MOV POWPOP,-(SP) ;;PUSH POWPOP ON STACK
MOV POWMES,-(SP) ;;PUSH POWMES ON STACK
MOV T0001,-(SP) ;;PUSH T0001 ON STACK
MOV SP,$SAVR6 ;;SAVE SP
MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
HALT
BR -2 ;;HANG UP
  
```

:POWER UP ROUTINE

```

$PWRUP: MOV $SAVR6,SP ;;GET SP
CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;;WAIT FOR THE INC
BNE 1$ ;;OF <POWPUS>,<POWPOP>,<POWMES>,<T0001> WORD
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
MOV #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
MOV #340,@#PWRVEC+2 ;;PRIO:7
  
```

```
1624 010230 104400
1625 010232 010244
1626 010234 000002
1627 010236 000000
1628 010240 000776
1629 010242 000000
1630 010244 005015 047520 042527
1631 010252 000122
1632
1633 010254 005015 042522 052123
1634 010262 051101 044524 043516
1635 010270 040440 052106 051105
1636 010276 040440 050040 053517
1637 010304 051105 043040 040511
1638 010312 052514 042522 005015
1639 010320 000012
1640 010322 005015 041412 045532
1641 010330 040527 043455 046040
1642 010336 047111 020105 051106
1643 010344 050505 042525 041516
1644 010352 020131 046103 041517
1645 010360 020113 042524 052123
1646 010366 005015 000
1647
1648 010371 124 040522 050120
1649 010376 042105 052040 020117
1650 010404 047514 020103 020064
1651 010412 051106 046517 046040
1652 010420 041517 052101 047511
1653 010426 020116 000040
1654 010432 042522 052123 051101
1655 010440 044524 043516 050040
1656 010446 047522 051107 046501
1657 010454 000
1658
1659 010456 001234
1660 010460 001234
1661 010462 020040 020040 020040
1662 010470 020040 020040 020040
1663 010476 020040 020040 020040
1664 010504 020040 020040 020040
1665 010512 020040 020040 020040
1666 010520 020040 045514 020123
1667 010526 020040 020040 045514
1668 010534 020123 020040 020040
1669 010542 000
1670 010543 050 041520 020051
1671 010550 020040 024040 051520
1672 010556 020051 020040 024040
1673 010564 050123 020051 020040
1674 010572 052040 051505 021524
1675 010600 020040 053440 051501
1676 010606 020040 020040 051440
1677 010614 041057 020040 020040
1678 010622 000040
1679
```

TYPE ;REPORT THE POWER FAILURE
SPWRMG: .WORD \$POWER ;:POWER FAIL MESSAGE POINTER
RTI
\$ILLUP: HALT ;:THE POWER UP SEQUENCE WAS STARTED
BR .-2 ;: BEFORE THE POWER DOWN WAS COMPLETE
\$SAVR6: 0 ;:PUT THE SP HERE
\$POWER: .ASCIZ <15><12>'POWER'
.EVEN
POWMES: .ASCIZ <15> <12> 'RESTARTING AFTER A POWER FIALURE' <15> <12> <12>
STMES: .ASCIZ <15><12><12>'CZKWA-G LINE FREQUENCY CLOCK TEST'<15><12>
TRPMES: .ASCIZ "TRAPPED TO LOC 4 FROM LOCATION "
TRPM2S: .ASCIZ "RESTARTING PROGRAM"
.EVEN
POWPUS: WORD
POWPOP: WORD
EM1: .ASCIZ " LKS LKS "
DH1: .ASCIZ "(PC) (PS) (SP) TEST# WAS S/B "

.EVEN

LINE	FREQUENCY	CLOCK	PROGRAM	DT1:	DT2:	DT3:	DT4:
1680	010624	001116	001176	001174	DT1:		
1681	010632	001172	177546	001124			
1682	010640	000000					
1683	010642	046103	041517	020113	EM2:		
1684	010650	040506	046111	042105			
1685	010656	052040	020117	047111			
1686	010664	042524	051122	050125			
1687	010672	000124					
1688	010674	050050	024503	020040	DH2:		
1689	010702	020040	050050	024523			
1690	010710	020040	020040	051450			
1691	010716	024520	020040	020040			
1692	010724	042524	052123	020043			
1693	010732	020040	046050	051513			
1694	010740	020051	020040	000			
1695		010746			.EVEN		
1696	010746	001116	001176	001174	DT2:		
1697	010754	001172	177546				
1698	010760	000000			0		
1699	010762	046103	041517	020113	EM3:		
1700	010770	047111	042524	051122			
1701	010776	050125	042524	020104			
1702	011004	044127	047105	052040			
1703	011012	042510	050040	047522			
1704	011020	042503	051523	051117			
1705	011026	050040	044522	051117			
1706	011034	052111	020131	040527			
1707	011042	020123	047524	020117			
1708	011050	044510	044107	000			
1709	011055	050	041520	020051	DH3:		
1710	011062	020040	024040	051520			
1711	011070	020051	020040	024040			
1712	011076	050123	020051	020040			
1713	011104	052040	051505	021524			
1714	011112	020040	024040	045514			
1715	011120	024523	020040	000040			
1716					.EVEN		
1717	011126	001116	001176	001174	DT3:		
1718	011134	001172	177546				
1719	011140	000000			0		
1720	011142	046103	041517	020113	EM4:		
1721	011150	044507	042526	020123			
1722	011156	047125	050505	040525			
1723	011164	020114	020043	043117			
1724	011172	050040	046125	042523			
1725	011200	020123	053117	051105			
1726	011206	052040	047527	042440			
1727	011214	052521	046101	050040			
1728	011222	051105	047511	051504			
1729	011230	047440	020106	044524			
1730	011236	042515	000				
1731	011241	050	041520	020051	DH4:		
1732	011246	020040	024040	051520			
1733	011254	020051	020040	024040			
1734	011262	050123	020051	020040			
1735	011270	052040	051505	021524			

1736	011276	020040	030440	052123		
1737	011304	020040	020040	031040		
1738	011312	042116	005015	020040		
1739	011320	020040	020040	020040		
1740	011326	020040	020040	020040		
1741	011334	020040	020040	020040		
1742	011342	020040	020040	020040		
1743	011350	020040	020040	020040		
1744	011356	042520	044522	042117		
1745	011364	020040	042520	044522		
1746	011372	042117	000			
1747		011376				
1748	011376	001116	001176	001174	.EVEN	
1749	011404	001172	001162	001164	DT4:	\$ERRPC,\$REG7,\$REG6,\$REG5,\$REG1,\$REG2
1750	011412	000000			0	
1751	011414	045514	020123	042522	EM5:	.ASCIZ 'LKS REGISTER RESPONDS TO ANOTHER ADDRESS''
1752	011422	044507	052123	051105		
1753	011430	051040	051505	047520		
1754	011436	042116	020123	047524		
1755	011444	040440	047516	044124		
1756	011452	051105	040440	042104		
1757	011460	042522	051523	000		
1758	011465	050	041520	020051	DH5:	.ASCIZ ''(PC) (PS) (SP) TEST# ADDRESS''
1759	011472	020040	024040	051520		
1760	011500	020051	020040	024040		
1761	011506	050123	020051	020040		
1762	011514	052040	051505	021524		
1763	011522	020040	040440	042104		
1764	011530	042522	051523	000		
1765		011536			.EVEN	
1766	011536	001116	001176	001174	DT5:	\$ERRPC,\$REG7,\$REG6,\$REG5,\$GDADR
1767	011544	001172	001120			
1768	011550	000000			0	
1769	011552	020101	047516	051440	EM6:	.ASCIZ ''A NO SACK TIMEOUT HAS OCCURED''
1770	011560	041501	020113	044524		
1771	011566	042515	052517	020124		
1772	011574	040510	020123	041517		
1773	011602	052503	042522	000104		
1774	011610	050050	024503	020040	DH6:	.ASCIZ ''(PC) (PS) (SP) TEST# (LKS) ''
1775	011616	020040	050050	024523		
1776	011624	020040	020040	051450		
1777	011632	024520	020040	020040		
1778	011640	042524	052123	020043		
1779	011646	020040	046050	051513		
1780	011654	020051	020040	000		
1781		011662			.EVEN	
1782	011662	001116	001176	001174	DT6:	\$ERRPC,\$REG7,\$REG6,\$REG5,LKS
1783	011670	001172	177546			
1784	011674	000000			0	
1785	011676	051127	047117	020107	EM7:	.ASCIZ ''WRONG CONDITION CODES WERE PUT ONTO STACK BY INTERRUPT''
1786	011704	047503	042116	052111		
1787	011712	047511	020116	047503		
1788	011720	042504	020123	042527		
1789	011726	042522	050040	052125		
1790	011734	047440	052116	020117		
1791	011742	052123	041501	020113		

1792	011750	054502	044440	052116	
1793	011756	051105	052522	052120	
1794	011764	000			
1795	011765	050	041520	020051	DH7: .ASCIZ "(PC) (PS) (SP) TEST# CC WAS CC S/B"
1796	011772	020040	024040	051520	
1797	012000	020051	020040	024040	
1798	012006	050123	020051	020040	
1799	012014	052040	051505	021524	
1800	012022	020040	041440	020103	
1801	012030	040527	020123	041440	
1802	012036	020103	027523	000102	
1803					.EVEN
1804	012044	001116	001176	001174	DT7: \$ERRPC,\$REG7,\$REG6,\$REG5,BUF1,\$GDDAT
1805	012052	001172	000776	001124	
1806	012060	000000			0
1807	012062	051127	047117	020107	EM10: .ASCIZ "WRONG PC PUT ONTO THE STACK BY AN INTERRUPT"
1808	012070	041520	050040	052125	
1809	012076	047440	052116	020117	
1810	012104	044124	020105	052123	
1811	012112	041501	020113	054502	
1812	012120	040440	020116	047111	
1813	012126	042524	051122	050125	
1814	012134	000124			
1815	012136	050050	024503	020040	DH10: .ASCIZ "(PC) (PS) (SP) TEST# @(SP)WAS @(SP)S/B "
1816	012144	020040	050050	024523	
1817	012152	020040	020040	051450	
1818	012160	024520	020040	020040	
1819	012166	042524	052123	020043	
1820	012174	040040	051450	024520	
1821	012202	040527	020123	024100	
1822	012210	050123	051451	041057	
1823	012216	020040	000		
1824		012222			.EVEN
1825	012222	001116	001176	001174	DT10: \$ERRPC,\$REG7,\$REG6,\$REG5,BUF2,\$GDDAT
1826	012230	001172	000774	001124	
1827	012236	000000			0
1828	012240	051124	042531	020104	EM11: .ASCIZ "TRIED TO ACCESS THE LKS REGISTER, AND TRAPPED"
1829	012246	047524	040440	041503	
1830	012254	051505	020123	044124	
1831	012262	020105	045514	020123	
1832	012270	042522	044507	052123	
1833	012276	051105	020054	047101	
1834	012304	020104	051124	050101	
1835	012312	042520	000104		
1836	012316	050050	024503	020040	DH11: .ASCIZ "(PC) (PS) (SP) TEST#"
1837	012324	020040	050050	024523	
1838	012332	020040	020040	051450	
1839	012340	024520	020040	020040	
1840	012346	042524	052123	000043	
1841					.EVEN
1842	012354	001116	001176	001174	DT11: \$ERRPC,\$REG7,\$REG6,\$REG5
1843	012362	001172			
1844	012364	000000			SAVE: 0 ;SAVE @#157546 IN TEST 14
1845	012366	000000			0
1846		000001			.END

A0004	002104	387#	390				
A0005	002172	408#	411				
A0006	002302	435#	438				
A0007	002434	465#	468				
A0010	002614	505#	508				
A0011	002716	534#	537				
A0012	003102	576#	579				
A0013	003172	606#	609				
A0014	003560	685	696#				
A0015	003700	708	720#				
A0016	004016	732	744#				
A0017	004134	756	768#				
A0020	004252	780	792#				
A0021	004370	804	816#				
A0022	004526	843	845#				
A0023	004574	859#	862				
A0024	004716	887#	888				
A0025	005026	911#	914				
A0026	005200	947#	948				
A0027	005314	973#	974				
A0030	005454	991	1001#				
A0031	005542	1021#	1024				
A0032	005674	1058#	1061				
A0033	006032	1091#	1094				
BIT0	= 000001	121#					
BIT00	= 000001	111#	121				
BIT01	= 000002	110#	120				
BIT02	= 000004	109#	119				
BIT03	= 000010	108#	118				
BIT04	= 000020	107#	117				
BIT05	= 000040	106#	116				
BIT06	= 000100	105#	115				
BIT07	= 000200	104#	114				
BIT08	= 000400	103#	113	1207			
BIT09	= 001000	102#	112	1215	1550		
BIT1	= 000002	120#					
BIT10	= 002000	101#	1532				
BIT11	= 004000	100#	1222				
BIT12	= 010000	99#					
BIT13	= 020000	98#	1539				
BIT14	= 040000	97#	1193				
BIT15	= 100000	96#					
BIT2	= 000004	119#					
BIT3	= 000010	118#					
BIT4	= 000020	117#					
BIT5	= 000040	116#					
BIT6	= 000100	115#					
BIT7	= 000200	114#					
BIT8	= 000400	113#					
BIT9	= 001000	112#					
BPTVEC	= 000014	128#					
BUF1	= 000776	140#	1066*	1075	1099*	1804	
BUF2	= 000774	139#	1067*	1100*	1108	1825	
B0005	002210	409	414#				
B0006	002320	436	441#				
B0007	002452	466	471#				

B0010	001632	506	511#		
B0011	002734	535	540#		
B0012	003120	577	582#		
B0013	003210	607	612#		
B0014	003572	697	699#		
B0015	003712	721	723#		
B0016	004030	745	747#		
B0017	004146	769	771#		
B0020	004264	793	795#		
B0021	004402	817	819#		
B0023	004612	860	865#		
B0025	005044	912	917#		
B0030	005472	1003	1005#	1006	
B0031	005560	1022	1027#		
B0032	005712	1059	1064#		
B0033	006050	1092	1097#		
C0007	002462	474#	476		
C0010	002642	514#	516		
C0011	003014	553#	556		
C0013	003224	616#	619		
C0025	005070	904	925#		
C0031	005614	1036#	1039		
C0032	005756	1068	1075#		
C0033	006114	1101	1108#	1110	
DDISP =	177570	43#	191		
DH1	010543	243	1670#		
DH10	012136	278	1815#		
DH11	012316	283	1836#		
DH2	010674	248	1688#		
DH3	011055	253	1709#		
DH4	011241	258	1731#		
DH5	011465	263	1758#		
DH6	011610	268	1774#		
DH7	011765	273	1795#		
DISPLA	001140	191#	327*	1236*	1531*
DISPRE	000174	149#	327		
DSWR =	177570	42#	190		
DT1	010624	244	1680#		
DT10	012222	279	1825#		
DT11	012354	284	1842#		
DT2	010746	249	1696#		
DT3	011126	254	1717#		
DT4	011376	259	1748#		
DT5	011536	264	1766#		
DT6	011662	269	1782#		
DT7	012044	274	1804#		
D0007	002474	457	479#		
D0010	002654	492	519#		
D0011	003032	554	559#		
D0013	003242	617	622#		
D0031	005632	1037	1042#		
EMTVEC=	000030	131#	309*	310*	
EM1	010462	242	1661#		
EM10	012062	277	1807#		
EM11	012240	282	1828#		
EM2	010642	247	1683#		

EM3	010762	252	1699#				
EM4	011142	257	1720#				
EM5	011414	262	1751#				
EM6	011552	267	1769#				
EM7	011676	272	1785#				
ERRVEC=	000004	124#	1198	1199*	1201*	1204*	
E0001	001730	340	346#				
E0002	001776	360#					
E0003	002042	374#					
E0004	002116	391#					
E0005	002204	412#					
E0006	002314	439#					
E0007	002446	469#					
E0010	002626	509#					
E0011	002730	538#					
E0012	003114	580#					
E0013	003204	610#					
E0014	003570	698#					
E0015	003710	722#					
E0016	004026	746#					
E0017	004144	770#					
E0020	004262	794#					
E0021	004400	818#					
E0022	004524	844#					
E0023	004606	863#					
E0024	004736	879	892#				
E0025	005040	915#					
E0026	005220	942	952#				
E0027	005346	979#					
E0030	005506	1001	1009#				
E0031	005554	1025#	1033				
E0032	005706	1062#					
E0033	006044	1095#					
E10006	002342	446#					
E10007	002470	477#					
E10010	002650	517#					
E10012	003134	587#					
E10013	003236	620#					
E10023	004630	869#					
E10025	005064	923#					
E10031	005626	1040#					
E10033	006124	1110#					
E1005	002226	418#					
E1011	003026	550	557#				
E20007	002502	481#					
E20010	002662	521#					
E20011	003040	563#					
E20013	003334	643#					
E20025	005100	927#					
E20031	005640	1046#					
E30013	003366	653#					
E40013	003444	669#					
F0013	003262	627#	632				
GNS =	***** U	148	1584	1585	1586	1587	1588
G0013	003276	628	631#				
H0013	003322	639#	642				

\$ENDAD	006232	160	1150#	1547															
\$ENDCT	006200	315	1139#																
\$ENDMG	006246	1141	1156#																
\$ENULL	006263	1144	1159#																
\$EOP	006144	1129#																	
\$EOPCT	006172	315*	1136#	1140															
\$ERFLG	001103	175#	1182	1211	1213	1219*	1240	1529*	1558										
\$ERMAX	001115	181#	318*	1213	1235*	1240													
\$ERROR	007634	309	1522#																
\$ERRPC	001116	182#	1474	1536*	1537*	1538	1558	1680	1696	1717	1748	1766	1782	1804					
		1825	1842																
\$ERRTB	001236	241#	1482																
\$ERRTY	007500	1467#	1541																
\$ERTTL	001112	179#	1535*	1558															
\$ESCAP	001222	219#	317*	1234*	1553	1555	1558												
\$FILLC	001154	198#	1279	1313															
\$FILLS	001153	197#	1313																
\$GDADR	001120	183#	684*	688	695*	701*	712*	719*	725*	736*	743*	749*	760*	767*					
		773*	784*	791*	797*	808*	815*	821*	834*	841*	1766								
\$GDDAT	001124	185#	357*	369*	384*	400*	427*	455*	570*	682*	710*	734*	758*	782*					
		806*	832*	855*	867	877*	890	901*	936*	950	962*	977	989*	1077*					
		1110*	1680	1804	1825														
\$GET42	006222	1145#																	
\$HD =	000000	18																	
\$ICNT	001104	176#	1226*	1227	1229*	1239													
\$ILLUP	010236	1594	1627#																
\$ITEMB	001114	180#	1471	1538*	1558														
\$LF	001232	223#	1313	1558															
\$LPADR	001106	177#	319*	342*	355*	370*	383*	401*	428*	456*	491*	531*	573*	602*					
		613*	635*	646*	656*	683*	711*	735*	759*	783*	807*	833*	856*	878*					
		902*	937*	963*	990*	1018*	1055*	1088*	1217*	1232*	1237	1239							
\$LPERR	001110	178#	320*	1217	1233*	1239	1552												
\$MAIL =	***** U	331	1232	1264	1544														
\$MXCNT	006610	1230	1239#																
\$NULL	001152	196#	1281	1313															
\$SOCNT	007474	1414*	1443*	1456#															
\$SOMODE	007476	1409*	1413*	1418	1421*	1432*	1458#												
\$OVER	006574	1194	1210	1218	1228	1236#													
\$PASS	001100	173#	1133*	1134*	1142	1156	1224	1240											
\$POWER	010244	1625	1630#																
\$PWDRN	010074	313	1594#	1622															
\$PWRMG	010232	1625#																	
\$PWUP	010162	1607	1612#																
\$QUES	001230	221#	1313	1558															
\$RDCHR=	***** U	1589																	
\$RDDEC=	***** U	1589																	
\$RDLIN=	***** U	1589																	
\$RDOCT=	***** U	1589																	
\$REGAD	001156	200#																	
\$REG0	001160	202#	1528*																
\$REG1	001162	203#	669*	1748															
\$REG2	001164	204#	670*	1748															
\$REG3	001166	205#																	
\$REG4	001170	206#																	
\$REG5	001172	207#	1526*	1527*	1680	1696	1717	1748	1766	1782	1804	1825	1842						
\$REG6	001174	208#	1523*	1524*	1680	1696	1717	1748	1766	1782	1804	1825	1842						

.SCMTA	1#	8#	164
.SDB2D	1#		
.SDB2O	1#		
.SDIV	1#		
.SEOP	1#	8#	1120
.SERRO	1#	8#	1507
.SERRT	1#	8#	1459
.SMULT	1#		
.SPOWE	1#	8#	1589
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#		
.SREAD	1#		
.SR2AZ	1#		
.\$SAVE	1#		
.\$SB2D	1#		
.\$SB2O	1#		
.\$SCOP	1#	8#	1176
.\$SIZE	1#		
.\$SUPR	1#		
.\$STRAP	1#	8#	1558
.\$STYPB	1#		
.\$STYPD	1#	8#	1313
.\$STYPE	1#	8#	1240
.\$STYPO	1#	8#	1381
.\$40CA	1#		

. ABS. 012370 000

ERRORS DETECTED: 0

CZKWAG.BIN,CZKWAG.LST/CRF/SOL/NL:TOC=DSKZ:CZKWAG.SML,DSKM:CZKWAG.P11
RUN-TIME: 16 12 1 SECONDS
RUN-TIME RATIO: 398/30=13.2
CORE USED: 28K (55 PAGES)