

DL11-W

DL11-W DIAGNOSTIC
MD-11-DZDLD-B

EP-DZDLD-B-DL-A
COPYRIGHT © 76-77
FICHE 1 OF 1

AUG 1977
digital
MADE IN USA

This microfiche card contains a grid of frames, each displaying diagnostic data for the DL11-W system. The data is organized into columns and rows, with each frame containing a header and a list of parameters. The parameters include various system identifiers, status indicators, and numerical values. The frames are arranged in a regular grid pattern, with some frames appearing to be blank or containing minimal data. The overall layout is consistent with standard microfiche formatting for diagnostic information.

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDLD-B-D

PRODUCT NAME: DL11-W DIAGNOSTIC

DATE CREATED: JUNE, 1977

MAINTAINER: DIAGNOSTIC ENGINEERING (00301)

AUTHOR: DAN CASALETTO

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1976, 1977 BY DIGITAL EQUIPMENT CORPORATION.

1.0 GENERAL INFORMATION

1.1 ABSTRACT

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE DL11-W SERIAL LINE, REAL TIME CLOCK INTERFACE. THE PROGRAM WILL RUN WITHOUT ANY SPECIAL TEST FIXTURES BY DEFAULT. HOWEVER, A WRAP CABLE CAN BE USED AND TESTED BY OPTION (BIT7 OF SWR).

THIS TEST OPERATES ON THE CONSOLE DL11-W SERIAL LINE AND CLOCK INTERFACES AS WELL AS UP TO FIFTEEN(15) ADDITIONAL IDENTICALLY CONFIGURED DL11-W SERIAL LINE INTERFACES. THE DEFAULT ADDRESSES ARE:

A. CONSOLE - 177560 SERIAL LINE
177546 CLOCK

B. OTHER SERIAL LINE - 776500 FIRST SERIAL LINE ADDRESS
OF 15 CONSECUTIVE SERIAL
LINE ADDRESSES

THE PROGRAM IS DESIGNED TO RUN ON ANY PDP-11 WITH 8K OF MEMORY AND A DL11-W MODULE. IT CAN BE RUN UNDER XXDP APT, AND ACT MONITORS, AND ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER, SOFTWARE SWITCH REGISTER = LOCATION 176

POWER FAILURE IS SUPPORTED FOR SYSTEMS WITH CORE MEMORY.

NOTE: THIS DIAGNOSTIC WITH THE SWR = 000020' (CLOCK TESTS ONLY) SHOULD BE USED ON SWITCHLESS CPU'S TO TEST KW-11L LINE CLOCK MODULES.

1.2 SYSTEM REQUIREMENTS

1.2.1 EQUIPMENT

STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE AND 8K OF MEMORY.

1.2.2 STORAGE

THE PROGRAM USES MEMORY FROM 00000 TO 22372

1.3 ASSUMPTIONS

- A. IF THE UNIT UNDER TEST (UUT) IS THE CONSOLE, THE PROGRAM WILL ASSUME THE REAL TIME CLOCK (RTC) IS ENABLED AND WILL TEST IT UNLESS THE TESTS ARE DISABLED BY BITS OF THE SWR.
- B. IF THE UUT IS NOT THE CONSOLE, THE RTC IS NOT TESTED FOR THAT DEVICE.
- C. THE PROGRAM WILL ASSUME THE ERROR FLAG BITS AND THE BREAK FUNCTION OF THE DL11-W ARE DISABLED AND WILL NOT TEST THESE FUNCTIONS UNLESS ENABLED BY BIT10 (FOR ERROR FLAGS) AND BIT8 (FOR BREAK) OF THE SWR. THE DEFAULT CHARACTER SIZE IS 8 BITS (SEE PARA 2.3.2).

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURE

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.

2.2 STARTING PROCEDURE

LOAD THE SWITCH REGISTER WITH SETTING
(SOFTWARE SWITCH REGISTER LOCATION = 176)

- A. START AT 200.
AFTER CHECKING THE TRANSMITTER, THE PROGRAM WILL PRINT ITS IDENTIFICATION AND REPORT THE NUMBER OF DEVICES UNDER TEST (NUMBER IS OCTAL). "END PASS" IS PRINTED AFTER A FULL PASS HAS BEEN MADE ON ALL DEVICES UNDER TEST.
- B. START AT 204.
THE "ECHO" TEST WILL BE EXECUTED. AN "*" IS PRINTED AT THE BEGINNING OF THE TEST. THE ECHO TEST READS A CHARACTER FROM THE TERMINAL, WRITES THAT CHARACTER TO THE TERMINAL AND REPORTS ANY ERROR FLAGS SET IN THE RECEIVER BUFFER. A CONTROL-C HALTS THE TEST AND PRINTS "STOP" AT THE TERMINAL. CONTINUING RESTARTS THE ECHO TEST.
- C. START AT 210.
THE TERMINAL OUTPUT TEST WILL BE EXECUTED. DEPRESSING ANY CHARACTER AT THE TERMINAL, HALTS THE TEST. CONTINUING RESTARTS THE TEST. THE TEST OUTPUTS 32 CHARACTERS ON A LINE AND REPEATS THE PATTERN EVERY THREE LINES. THE PATTERN IS AS FOLLOWS (OCTAL CODE 040 -- 377):

!"#\$%&'()*+,-./0123456789:;<=?	(OCTAL CODE)
0ABCDEFGHIJKLMN0PQRSTUvwxyz{ }~	(040 -- 077)
0abcdefghijklmnopqrstuvwxyz	(100 -- 137)
	(140 -- 177) [LOWER CASE ALPHA]

THIS BOTTOM LINE COULD BE THE FOLLOWING IF THE TERMINAL DOES NOT HAVE LOWER CASE:

0ABCDEFGHIJKLMN0PQRSTUvwxyz{ }	[UPPER CASE ALPHA]
--------------------------------	--------------------

2.3 OPERATING PROCEDURE

2.3.1 OPERATIONAL SWITCH SETTINGS

NOTE: IF NO HARDWARE SWITCH REGISTER IS AVAILABLE THE PROGRAM WILL AUTOMATICALLY USE THE CONTENTS OF LOCATION 176 AS THE SOFTWARE SWITCH REGISTER. THE USER SHOULD SET THIS LOCATION BEFORE STARTING THE PROGRAM. IF A HARDWARE SWITCH IS AVAILABLE AND A SOFTWARE SWR IS DESIRED, PUT ALL SWITCHES UP.

BIT 15	- HALT ON ERROR
BIT 14	- SCOPE LOOP
BIT 13	- INHIBIT ERROR TYPEOUT
BIT 12	- UNUSED
BIT 11	- UNUSED
BIT 10	- ENABLE ERROR FLAGS TESTS
BIT 09	- LOOP ON ERROR
BIT 08	- ENABLE BREAK FUNCTION TESTS
BIT 07	- ENABLE DATA TEST WITH WRAP CABLE
BIT 06	- INHIBIT RTC TESTS (ALLOW ONLY SLU TESTS)
BIT 05	- ALLOW MANUAL SETTING OF "SDEVM" (DEVICE MAP)
BIT 04	- INHIBIT SLU TESTS (ALLOW ONLY LINE CLOCK TESTS)

BUILT INTO THE PROGRAM IS THE ABILITY TO DYNAMICALLY CHANGE THE CONTENTS OF SWREG DURING PROGRAM EXECUTION. BY STRIKING ^G (CNTRL G) ON CONSOLE TTY THE OPERATOR SETS A REQUEST FLAG TO CHANGE THE CONTENTS OF SWREG, WHICH IS PROCESSED IN KEY AREAS OF THE PROGRAM CODE (IE) ERROR ROUTINES, AFTER HALTS END OF PASS, AND OTHER APPLICABLE AREAS. BECAUSE THIS DIAGNOSTIC USES THE MAINTENANCE BIT OF THE SERIAL LINE, THE CONTROL-G SHOULD BE ISSUED DURING PROGRAM TYPEOUTS AT THAT TIME THE MAINTENANCE BIT IS SURE TO BE CLEAR.

IF A CONTROL-G IS DETECTED, UPON EXECUTION THE CONTENTS OF THE SWREG ARE DUMPED IN OCTAL ON THE CONSOLE TTY AND ANY CHANGES ARE REQUESTED

(IE) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

1.	^CR)	IF NO CHANGES ARE TO BE MADE
2.	6 DIGITS 0-7	TO REPRESENT IN OCTAL THE NEW SWITCH REGISTER VALUE ;LAST DIGIT FOLLOWED BY ^CR)
3.	^L	TO ALLOW REENTERING VALUE IF ERROR IS COMMITTED KEYING IN SWREG VALUE.

2.3.2 SETTING BITS PER CHARACTER

THIS PROGRAM DEFAULTS TO TESTING 8 BITS PER CHARACTER. IF THE SERIAL LINE IS SET FOR 5-->7 BITS PER CHARACTER, SET THE MEMORY LOCATION "\$USWR" AS FOLLOWS:

CHAR. SIZE (# OF BITS)	"\$USWR" CONTENTS	
	(BINARY)	(OCTAL)
8	10000000	400
7	01000000	200
6	00100000	100
5	00010000	40

"\$USWR" IS USED IN THE DATA PATH TESTS TO LIMIT THE BINARY COUNT TEST PATTERN TO THE NUMBER OF BITS SELECTED ON THE SERIAL LINE.

2.3.3 RUNNING UNDER APT

THE APT MAILBOX IS LOCATED AT LOCATION 500. TO ALLOW ADDITIONAL SERIAL LINE VECTOR ASSIGNMENTS TO THE 400 AREA OF MEMORY.

THE EXECUTION TIMES PROVIDED ARE FOR EXECUTION WITH AN 11/34 PROCESSOR, CORE MEMORY, AND 110 BAUD.

2.3.4 RUN WITH ALTERNATE CONSOLE ADDRESS

 TO USE A CONSOLE ADDRESS OTHER THAN 177560, OR VECTOR OTHER THAN 60, THE OPERATOR MUST SUPPLY THE PROGRAM WITH THE CORRECT ADDRESSES BY INSERTING THEM AT THE TAG LABELED "CRCSR":

CRCSR: ADDRESS OF RECEIVER RCSR
 CRBUF: ADDRESS OF RECEIVER BUFFER
 CTCRSR: ADDRESS OF TRANSMITTER CSR
 CTBUF: ADDRESS OF TRANSMITTER BUFFER
 CRVECT: ADDRESS OF RECEIVER VECTOR
 CRPSW: ADDRESS OF ASSOCIATED PSW
 CTVECT: ADDRESS OF TRANSMITTER VECTOR
 CTPSW: ADDRESS OF ASSOCIATED PSW

2.3.5 TESTING ADDITIONAL SERIAL LINES

 THIS PROGRAM WILL SUPPORT TESTING OF MULTIPLE SLU'S. IT REQUIRES THE ADDRESS OF THE FIRST ADDITIONAL RCSR (STORED AT "\$BASE") AND ITS INTERRUPT VECTOR (STORED AT "\$VECT1"); AND WILL BE ABLE TO ADDRESS ANY SLU STARTING AT THE SPECIFIED BASE ADDRESS UP TO 15 CONSECUTIVE DEVICES.

EXAMPLE: \$BASE: 776500
 \$VECT1: 300

THE PROGRAM WILL BE ABLE TO TEST THE CONSOLE PLUS ANY ADDITIONAL DL11-W SLU'S WITHIN THE RANGE 776500 --> 776650

\$BASE AND \$VECT1 DEFAULT TO 776500 AND 300 RESPECTIVELY.

THE PROGRAM ASSOCIATES UNIT NUMBERS TO DEVICES AS FOLLOWS:
 (NUMBERS IN PARENTHESIS ARE OCTAL)

UNIT# 0 --> CONSOLE (ADDRESS STORED AT "CRCSR")
 UNIT# 1 --> BASE ADDRESS STORED AT "\$BASE"
 ASSOCIATED BASE VECTOR STORED AT "\$VECT1"
 UNIT# 2 --> BASE ADDRESS + (10)
 BASE VECTOR + (10)
 UNIT# 3 --> BASE ADDRESS + (20)
 BASE VECTOR + (20)
 UNIT# 4 --> BASE ADDRESS + (30)
 BASE VECTOR + (30)
 ⋮
 UNIT# 15 --> BASE ADDRESS + (160)
 BASE VECTOR + (160)

H01

PAGE: 0007

STARTING AT LOCATION 200 AND HAVING BITS OF SWR CLEAR, THE PROGRAM WILL SELF
SIZE THE NUMBER OF DEVICES (STARTING AT THE BAS ADDRESS) AND
STORE A BIT MAP AT "\$DEVN" (DEVICE MAP) TO INDICATE WHICH UNIT NUMBERS
ARE PRESENT AND WILL BE TESTED:

```
-----  
! UNIT ! UNIT ! ..... ! UNIT ! UNIT ! CONSOLE!  
! 15 ! 14 ! ..... ! 2 ! 1 !  
-----
```

A BIT MAP CAN BE ENTERED AT "\$DEVN" PRIOR TO STARTING THE PROGRAM
SETTING BITS OF THE SWR INHIBITS THE SELF-SIZING AND DEVICE MAP
GENERATION, AND USES THE VALUE STORED BY THE OPERATOR.

EXAMPLE:

SWR = 000040 [BINARY 0 000 000 000 100 000]
\$BASE: 776500
\$VECT1: 300

\$DEVN: 13 [BINARY - 0 000 000 000 001 011]

THE PROGRAM WILL TEST -

UNIT# 0 = CONSOLE
UNIT# 1 = 776500 : 300
UNIT# 3 = 776520 : 320

2.4 EXECUTION TIMES - (110 BAUD)

 LONGEST SUBTEST TIME = 50 SECONDS
 PASS TIME = 60 SECONDS
 ADDITIONAL DEVICES = 55 SECONDS/DEVICE

3.0 ERROR REPORTING

 IF A ROUTINE FAILS AND THE INHIBIT ERROR TIMEOUT (BIT13) OF THE SWR IS NOT SET, A PRINTOUT RESULTS IN THE FORM:

"(SOME ASCII MESSAGE)"
 TEST# ERR PC RCSR [ANY APPLICABLE DAT HEADINGS]
 XXXXXX XXXXXX XXXXXX [ANY APPLICABLE DATA]

NOTE: "RCSR" IS DEPENDENT ON THE FAILURE
 & THEREFORE COULD BE TCSR, RBUF, TBUF, OR LKS

WHERE "XXXXXX" IS AN OCTAL NUMBER.
 THIS ERROR PRINTOUT OCCURS PROVIDED THE ERROR THAT EXISTS WOULD NOT HINDER THE TIMEOUT. IN CASES WHERE IT IS NOT POSSIBLE TO PRINT AN ERROR MESSAGE (I.E. FATAL CONSOLE TRANSMITTER FAILURES), A HALT OCCURS AND THE PC CAN BE EXAMINED BY THE OPERATOR TO FIND THE ERROR INFORMATION IN THE PROGRAM LISTING.

NOTE: FOR SOFTWARE SWITCH OPERATION, THE SWITCH REGISTER CAN BE CHANGED BY TYPING A CONTROL-G AT THE CONSOLE DURING ERROR PRINTOUTS. AFTER CONTINUING FROM THE ERROR HALT THE OLD SWR CONTENTS IS DISPLAYED AND THE NEW CONTENTS CAN BE ENTERED. IF ERROR HALTS ARE DISABLED, THE CONTROL-G RESPONSE OCCURS IMMEDIATELY FOLLOWING THE TIMEOUT.

4.0 SUBROUTINE ABSTRACTS

4.1 TRAPCATCHER

THIS IS A SERIES OF INSTRUCTIONS STARTING AT LOCATION 0 DESIGNED TO DETECT AND ISOLATE UNEXPECTED TRAPS TO THE TRAP AND INTERRUPT VECTOR AREA OF MEMORY.

EACH VECTOR ENTRANCE ADDRESS IS LOADED WITH THE ADDRESS OF THE NEXT LOCATION. THE NEXT LOCATION IS LOADED WITH A BREAK POINT TRAP (000003). THUS AN ILLEGAL TRAP OR INTERRUPT CAUSES A TRAP THROUGH THE BPT VECTOR (14) WHICH POINTS TO THE "CATCH" ROUTINE.

THE "CATCH" ROUTINE REPORTS THE PC THAT CAUSED THE ORIGINAL TRAP AND THE LOCATION OF THE TRAP VECTOR (IF UNDER APT, AN ERROR IS INDICATED TO APT). AFTER REPORTING THE ERROR THE PROGRAM HALTS. THE PROGRAM MUST BE RESTARTED AT THIS POINT.

4.2 WRPSW

THIS ROUTINE IS USED TO WRITE THE PSW BY POPPING VALUES FROM THE STACK. THIS METHOD IS USED TO BE COMPATIBLE WITH ALL 11 FAMILY PROCESSORS.

4.3 SCOPE

THIS ROUTINE CALL IS PLACED BETWEEN EACH SUBTEST. IT RECORDS THE STARTING ADDRESS OF EACH SUBTEST AS IT IS BEING ENTERED AND UPDATES THE TEST NUMBER. IF A SCOPE LOOP IS REQUESTED, IT WILL JUMP TO THE START OF THE SUBTEST AT WHICH THE SCOPE LOOP IS REQUESTED.

4.4
ERROR

THIS ROUTINE CALL IS PLACED WHEREEVER AN ERROR REPORT IS DESIRED. THE LOWER BYTE OF THIS CALL IS USED AS THE ERROR NUMBER AND AS A POINTER INTO THE ERROR TABLE. THIS ROUTINE REPORTS ERRORS TO APT USING "\$APTYPE" AND TYPES ERROR REPORTS TO THE CONSOLE USING "\$ERRTYPE".

4.5
\$POWER

THIS ROUTINE SAVES ALL GENERAL REGISTERS DURING POWER-DOWN AND RESTORES THEM AT POWER-UP. IF A POWER FAILURE OCCURS "POWER" IS PRINTED AT THE CONSOLE AFTER POWER IS RESTORED.

4.6
CKSWR

THIS ROUTINE CALL IS USED TO DETECT THE RECEPTION OF A CONTROL-G FROM THE CONSOLE. THE CALL USES "\$READ" TO PERFORM THE CONTROL-G SEQUENCE OF DISPLAYING THE CONTENTS OF THE SOFTWARE SWITCH REGISTER AND THE ENTERING THE NEW CONTENTS FROM THE TERMINAL.

BASIC DEFINITIONS
 ACT11 HOOKS
 APT PARAMETER BLOCK
 COMMON TAGS
 APT MAILBOX-ETABLE
 ERROR POINTER TABLE
 INITIALIZE THE COMMON TAGS
 T1 TEST ABILITY TO REFERENCE TCSR
 T2 TEST ABILITY TO REFERENCE TBUF
 T3 TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED
 T4 TEST THAT TCSR "DONE" SETS WITH RESET
 T5 TEST ABILITY TO ACCESS RCSR
 T6 TEST ABILITY TO ACCESS RBUF
 T7 TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET
 T10 TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET
 T11 TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET
 T12 TEST THAT BIT6 OF RCSR CAN BE SET & RESET
 T13 TEST ABILITY TO ACCESS LKS
 T14 TEST THAT BIT6 OF LKS CAN BE SET & RESET
 T15 TEST FOR DUAL ADDRESSING OF REGISTERS
 T16 TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED
 T17 TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY
 T20 TEST RTC FOR DOUBLE INTERRUPTS
 T21 TEST THAT RTC INTERRUPT CLEARS WITH RESET
 T22 TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS
 T23 TEST CLOCK REPEATABILITY
 T24 TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
 T25 TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
 T26 TEST TRANSMITTER FOR DOUBLE INTERRUPTS
 T27 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF
 T30 TEST THAT RCVR ACTIVE (11) & DONE (7) SET & CLEAR PROPERLY
 T31 TEST THAT RDR ENABLE CLEARS RECEIVER DONE FLAG
 T32 TEST THAT READING RBUF CLEARS RECEIVER DONE
 T33 TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
 T34 TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED
 T35 TEST RECEIVER FOR DOUBLE INTERRUPTS
 T36 TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF
 T37 TEST THAT RESET CLEARS RECEIVE INTERRUPT
 T40 TEST THAT THE "OR" ERROR (BIT14) & "ERROR" (BIT15) CAN BE SET
 T41 TEST THAT BREAK TRANSMITS ALL ZEROES
 T42 TEST THAT "FR" ERROR CAN BE SET DURING BREAK
 T43 TEST DATA PATH FROM TRANSMITTER TO RECEIVER USING MAINTENANCE WRAP
 T44 TEST DATA PATHS USING WRAP CABLE
 T45 TEST DL11-W LOGIC BY EXERCISING THE XMIT,RECEIVE, & CLOCK(IF AVAILABLE) SIMULTANEOUSLY
 END OF PASS ROUTINE
 ERROR MESSAGE TYPEOUT ROUTINE
 POWER DOWN AND UP ROUTINES
 SCOPE HANDLER ROUTINE
 APT COMMUNICATIONS ROUTINE
 TYPE ROUTINE
 BINARY TO OCTAL (ASCII) AND TYPE
 TTY INPUT ROUTINE
 TRAP DECODER
 TRAP TABLE
 ECHO TEST
 TERMINAL OUTPUT TEST

9
 154
 159
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 1
DZDLDB.P11 14-JUN-77 09:27

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

.ENABLE ABS
.LIST ME
.NLIST MC,MD,CND

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

001100

STACK= 1100
.EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

000011 HT= 11 ;:CODE FOR HORIZONTAL TAB
000012 LF= 12 ;:CODE FOR LINE FEED
000015 CR= 15 ;:CODE FOR CARRIAGE RETURN
000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS= 177776 ;:PROCESSOR STATUS WORD
177774 .EQUIV PS,PSW
177772 STKLM= 177774 ;:STACK LIMIT REGISTER
177570 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
177570 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

000000 R0= %0 ;:GENERAL REGISTER
000001 R1= %1 ;:GENERAL REGISTER
000002 R2= %2 ;:GENERAL REGISTER
000003 R3= %3 ;:GENERAL REGISTER
000004 R4= %4 ;:GENERAL REGISTER
000005 R5= %5 ;:GENERAL REGISTER
000006 R6= %6 ;:GENERAL REGISTER
000007 R7= %7 ;:GENERAL REGISTER
000006 SP= %6 ;:STACK POINTER
000007 PC= %7 ;:PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS

000000 PR0= 0 ;:PRIORITY LEVEL 0
000040 PR1= 40 ;:PRIORITY LEVEL 1
000100 PR2= 100 ;:PRIORITY LEVEL 2
000140 PR3= 140 ;:PRIORITY LEVEL 3
000200 PR4= 200 ;:PRIORITY LEVEL 4
000240 PR5= 240 ;:PRIORITY LEVEL 5
000300 PR6= 300 ;:PRIORITY LEVEL 6
000340 PR7= 340 ;:PRIORITY LEVEL 7

;*SWITCH REGISTER SWITCH DEFINITIONS

100000 SW15= 100000
040000 SW14= 40000
020000 SW13= 20000
010000 SW12= 10000
004000 SW11= 4000
002000 SW10= 2000

N01

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

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

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

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

;*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4 :: TIME OUT AND OTHER ERRORS
RESVEC= 10 :: RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC= 14 :: "T" BIT
TRTVEC= 14 :: TRACE TRAP
BPTVEC= 14 :: BREAKPOINT TRAP (BPT)
IOTVEC= 20 :: INPUT OUTPUT TRAP IOT **SCOPE**

MAIN MACY11 30(1046) 14-JUN-77 17:10 PAGE 3
CZDLDB.P11 14-JUN-77 09:27

BASIC DEFINITIONS

113	000024	PWRVEC= 24	:: POWER FAIL
114	000030	EMTVEC= 30	:: EMULATOR TRAP (EMT) **ERROR**
115	000034	TRAPVEC=34	:: "TRAP" TRAP
116	000060	TKVEC= 60	:: TTY KEYBOARD VECTOR
117	000064	TPVEC= 64	:: TTY PRINTER VECTOR
118	000240	PIRQVEC=240	:: PROGRAM INTERRUPT REQUEST VECTOR
119	176500	ABASE= 176500	
120	000300	AVECT1= 300	
121	000400	AUSWR= 400	
122	000001	\$TN= 1	
123	161000	\$SWR= 161000	
124	000003	BPT= 000003	:: THIS IS THE COMMAND FOR A TRAP

: THROUGH 14 (BPT TRAP)

125	000000	. = 0	:: *****
126			:: *ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2,BPT"
127			:: *SEQUENCE TO CATCH ILLEGAL TRAPS & INTERRUPTS
128			:: *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

129	000014	. = 14	:: THE BPT TRAP VECTOR POINTS TO THE
130	000016	.WORD CATCH	:: ILLEGAL TRAP HANDLER "CATCH"
131		.WORD 340	
132	000042	. = 42	
133	000000	.WORD J	

134	000174	. = 174	
135	000000	DISPREG: .WORD 0	
136	000000	SWREG: .WORD 0	

137	000200	. = 200	
138	000167	JMP START	:: DO INTERFACE TEST
139	000167	JMP ECHO	:: DO ECHO TEST
140	000167	JMP OUTST	:: DO OUTPUT TEST TO TERMINAL

```

000500      000500      .SBTTL      = 500
                                ACT11 HOOKS

:*****
:HOOKS REQUIRED BY ACT11
      $SVPC=.                ;SAVE PC
      =46                    ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .3ECP
000046      000046      $ENDAD
      =52                    ;;2)SET LOC.52 TO ZERO
000052      000052      .WORD      0
      = $SVPC                ;; RESTORE PC
000500      000500      .SBTTL      APT PARAMETER BLOCK

:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      .$X=.                ;;SAVE CURRENT LOCATION
      =24                    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024      000024      200      ;;FOR APT START UP
      =44                    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044      000044      $APTHDR  ;;POINT TO APT HEADER BLOCK
      =.$X                    ;;RESET LOCATION COUNTER

:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDPI! DIAGNOSTIC
:INTERFACE SPEC.

$APTHD:
$HIBTS: .WORD      0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MADDR: .WORD      $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15).
$TSTM:  .WORD      50     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD      60     ;;RUN TIME IN SECS. OF 1ST PASS ON ! UNIT (QUICK VERIFY).
$UNITM: .WORD      35     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000500      000500      $ETEND-$MAIL 2 ;;LENGTH MAILBOX-ETABLE WORDS.
000500      000500
000502      001066
000504      000050
000506      000060
000510      000055
000512      000030

```


187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243

001000
001000 000000
001002 000
001003 000
001004 000000
001006 000000
001010 000000
001012 000000
001014 000
001015 001
001016 000000
001020 000000
001022 000000
001024 000000
001026 000000
001030 000000
001032 000000
001034 000
001035 000
001036 000000
001040 177570
001042 177570
001044 177560
001046 177562
001050 177564
001052 177566
001054 000
001055 002
001056 012
001057 000
001060 000000
001062 077
001063 015
001064 000012
001066
001066 000000
001070 000000
001072 000000
001074 000000
001076 000000
001100 000000
001102 000000
001104 000000
001106

.SBTTL COMMON TAGS
:*****
:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
:*USED IN THE PROGRAM.
.=1000
\$CMTAG: ::START OF COMMON TAGS
\$STNM: .WORD 0 ::CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 ::CONTAINS ERROR FLAG
\$ICNT: .WORD 0 ::CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 ::CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 ::CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 ::CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 ::CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 ::CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 ::CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 ::CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 ::CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT: .WORD 0 ::CONTAINS 'GOOD' DATA
\$BDDAT: .WORD 0 ::CONTAINS 'BAD' DATA
::RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 ::AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 ::INTERRUPT MODE INDICATOR
\$SWR: .WORD 0 DSWR ::ADDRESS OF SWITCH REGISTER
\$DISPLAY: .WORD 0 DDISP ::ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 ::TTY KBD STATUS
\$TKB: 177562 ::TTY KBD BUFFER
\$TPS: 177564 ::TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 ::TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILES
\$FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 ::INSERT FILL CHARS. AFTER A "LINE FEED"
\$TPFLG: .BYTE 0 ::"TERMINAL AVAILABLE" FLAG (BIT 0) =0=55
\$ESCAPE: 0 ::ESCAPE ON ERROR ADDRESS
\$QUES: .ASCII '?' ::QUESTION MARK
\$CRLF: .ASCII '<15>' ::CARRIAGE RETURN
\$LF: .ASCII '<12>' ::LINE FEED
:*****
.SBTTL APT MAILBOX-ETABLE
:*****
.EVEN
\$MAIL: ::APT MAILBOX
\$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ::TEST NUMBER
\$PASS: .WORD APASS ::PASS COUNT
\$DEVCT: .WORD ADEVCT ::DEVICE COUNT
\$UNIT: .WORD AUNIT ::I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
\$ETABLE: ::APT ENVIRONMENT TABLE

001106 000
001107 000
001110 000000
001112 000400
001114 000000

001116 000
001117 000

001120 000000

001122 000
001123 000
001124 000000
001126 000
001127 000
001130 000000
001132 000
001133 000
001134 000000
001136 000300
001140 000000
001142 175500
001144 000000
001146 000000

\$ENV: .BYTE AENV :: ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM :: ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG :: APT SWITCH REGISTER
\$USWR: .WORD AUSWR :: USER SWITCHES
\$CPUOP: .WORD ACPUOP :: CPU TYPE, OPTIONS
*
* BIT 15-11=CPU TYPE
* 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
* 11/70=06, PDQ=07, Q=10
* BIT 10=REAL TIME CLOCK
* BIT 9=FLOATING POINT PROCESSOR
* BIT 8=MEMORY MANAGEMENT
\$MAMS1: .BYTE AMAMS1 :: HIGH ADDRESS, M.S. BYTE
\$MTYP1: .BYTE AMTYP1 :: MEM. TYPE, BLK#1
* MEM. TYPE BYTE -- (HIGH BYTE)
* 900 NSEC CORE=001
* 300 NSEC BIPOLAR=002
* 500 NSEC MOS=003
\$MADR1: .WORD AMADR1 :: HIGH ADDRESS, BLK#1
* MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
\$MAMS2: .BYTE AMAMS2 :: HIGH ADDRESS, M.S. BYTE
\$MTYP2: .BYTE AMTYP2 :: MEM. TYPE, BLK#2
\$MADR2: .WORD AMADR2 :: MEM. LAST ADDRESS, BLK#2
\$MAMS3: .BYTE AMAMS3 :: HIGH ADDRESS, M.S. BYTE
\$MTYP3: .BYTE AMTYP3 :: MEM. TYPE, BLK#3
\$MADR3: .WORD AMADR3 :: MEM. LAST ADDRESS, BLK#3
\$MAMS4: .BYTE AMAMS4 :: HIGH ADDRESS, M.S. BYTE
\$MTYP4: .BYTE AMTYP4 :: MEM. TYPE, BLK#4
\$MADR4: .WORD AMADR4 :: MEM. LAST ADDRESS, BLK#4
\$VECT1: .WORD AVECT1 :: INTERRUPT VECTOR#1, BUS PRIORITY#1
\$VECT2: .WORD AVECT2 :: INTERRUPT VECTOR#2, BUS PRIORITY#2
\$BASE: .WORD ABASE :: BASE ADDRESS OF EQUIPMENT UNDER TEST
\$DEVN: .WORD ADEVN :: DEVICE MAP
\$ETEND:
\$NEXT

ERROR POINTER TABLE

.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:

. \$ERRTB:

001146
001146 015266
001150 021451
001152 022214
001154 000000

001156 015312
001150 021476
001152 022224
001164 000000

001166 015336
001170 021451
001172 022214
001174 000000

001176 015403
001200 021451
001202 022214
001204 000000

001206 015425
001210 021451
001212 022214
001214 000000

001216 015462
001220 021523
001222 022234
001224 000000

001226 015506
001230 021550
001232 022244
001234 000000

001236 015532
001240 021575
001242 022254
001244 000000

EM1 ::"CAN NOT ACCESS TCSR"
DH1 ::"TEST# ERR PC TCSR"
DT1 ::\$TESTN,\$ERRPC,TCSR
0

EM2 ::"CAN NOT ACCESS TBUF"
DH2 ::"TEST# ERR PC TBUF"
DT2 ::\$TESTN,\$ERRPC,TBUF
0

EM3 ::"TCSR DONE NOT CLEARED WITH TBUF FULL"
DH1 ::"TEST# ERR PC TCSR"
DT1 ::\$TESTN,\$ERRPC,TCSR
0

EM4 ::"TCSR DONE NOT SET"
DH1 ::"TEST# ERR PC TCSR"
DT1 ::\$TESTN,\$ERRPC,TCSR
0

EM5 ::"TCSR DONE NOT SET WITH RESE"
DH1 ::"TEST# ERR PC TCSR"
DT1 ::\$TESTN,\$ERRPC,TCSR
0

EM6 ::"CAN NOT ACCESS RCSR"
DH6 ::"TEST# ERR PC RCSR"
DT6 ::\$TESTN,\$ERRPC,RCSR
0

EM7 ::"CAN NOT ACCESS RBUF"
DH7 ::"TEST# ERR PC RBUF"
DT7 ::\$TESTN,\$ERRPC,RBUF
0

EM10 ::"CAN NOT ACCESS LRS"
DH10 ::"TEST# ERR PC LRS"
DT10 ::\$TESTN,\$ERRPC,LRS
0

333				
334	001246	015555	EM11	:"BIT0 OF TCSR NOT CLEAR AFTER RESET"
335	001250	021451	DH1	:"TEST# ERR PC TCSR"
336	001252	022214	DT1	:\$TESTN,\$ERRPC,TCSR
337	001254	000000	0	
338				
339	001256	015620	EM12	:"CAN NOT SET BIT0 OF TCSR"
340	001260	021451	DH1	:"TEST# ERR PC TCSR"
341	001262	022214	DT1	:\$TESTN,\$ERRPC,TCSR
342	001264	000000	0	
343				
344	001266	015651	EM13	:"CAN NOT CLEAR BIT0 OF TCSR"
345	001270	021451	DH1	:"TEST# ERR PC TCSR"
346	001272	022214	DT1	:\$TESTN,\$ERRPC,TCSR
347	001274	000000	0	
348				
349	001276	015704	EM14	:"RESET DID NOT CLEAR BIT0 OF TCSR"
350	001300	021451	DH1	:"TEST# ERR PC TCSR"
351	001302	022214	DT1	:\$TESTN,\$ERRPC,TCSR
352	001304	000000	0	
353				
354	001306	015745	EM15	:"BIT2 OF TCSR NOT CLEAR AFTER RESET"
355	001310	021451	DH1	:"TEST# ERR PC TCSR"
356	001312	022214	DT1	:\$TESTN,\$ERRPC,TCSR
357	001314	000000	0	
358				
359	001316	016010	EM16	:"CAN NOT SET BIT2 OF TCSR"
360	001320	021451	DH1	:"TEST# ERR PC TCSR"
361	001322	022214	DT1	:\$TESTN,\$ERRPC,TCSR
362	001324	000000	0	
363				
364	001326	016041	EM17	:"CAN NOT CLEAR BIT2 OF TCSR"
365	001330	021451	DH1	:"TEST# ERR PC TCSR"
366	001332	022214	DT1	:\$TESTN,\$ERRPC,TCSR
367	001334	000000	0	
368				
369	001336	016074	EM20	:"RESET DID NOT CLEAR BIT2 OF TCSR"
370	001340	021451	DH1	:"TEST# ERR PC TCSR"
371	001342	022214	DT1	:\$TESTN,\$ERRPC,TCSR
372	001344	000000	0	
373				
374	001346	016135	EM21	:"BIT6 OF TCSR NOT CLEAR AFTER RESET"
375	001350	021451	DH1	:"TEST# ERR PC TCSR"
376	001352	022214	DT1	:\$TESTN,\$ERRPC,TCSR
377	001354	000000	0	
378				
379	001356	016201	EM22	:"XMIT INTERRUPT WITH PRIORITY"
380	001360	021451	DH1	:"TEST# ERR PC TCSR"
381	001362	022214	DT1	:\$TESTN,\$ERRPC,TCSR
382	001364	000000	0	
383				
384	001366	016236	EM23	:"CAN NOT SET BIT6 OF TCSR"
385	001370	021451	DH1	:"TEST# ERR PC TCSR"
386	001372	022214	DT1	:\$TESTN,\$ERRPC,TCSR
387	001374	000000	0	
388				

MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 9
 SDCLOB.P11 14-JUN-77 09:27 ERROR POINTER TABLE

389	001376	016267	EM24	:"CAN NOT CLEAR BIT6 OF TCSR"
390	001400	021451	DH1	:"TEST# ERR PC TCSR"
391	001402	022214	DT1	:\$TESTN,\$ERRPC,TCSR
392	001404	000000	0	
393				
394	001406	016322	EM25	:"RESET DID NOT CLEAR BIT6 OF TCSR"
395	001410	021451	DH1	:"TEST# ERR PC TCSR"
396	001412	022214	DT1	:\$TESTN,\$ERRPC,TCSR
397	001414	000000	0	
398				
399	001416	016363	EM26	:"BIT6 OF RCSR NOT CLEAR AFTER RESET"
400	001420	021523	DH6	:"TEST# ERR PC RCSR"
401	001422	022234	DT6	:\$TESTN,\$ERRPC,RCSR
402	001424	000000	0	
403				
404	001426	016426	EM27	:"RCVR INTERRUPT WITH PRIORITY 7"
405	001430	021523	DH6	:"TEST# ERR PC RCSR"
406	001432	022234	DT6	:\$TESTN,\$ERRPC,RCSR
407	001434	000000	0	
408				
409	001436	016465	EM30	:"CAN NOT SET BIT6 OF RCSR"
410	001440	021523	DH6	:"TEST# ERR PC RCSR"
411	001442	022234	DT6	:\$TESTN,\$ERRPC,RCSR
412	001444	000000	0	
413				
414	001446	016516	EM31	:"CAN NOT CLEAR BIT6 OF RCSR"
415	001450	021523	DH6	:"TEST# ERR PC RCSR"
416	001452	022234	DT6	:\$TESTN,\$ERRPC,RCSR
417	001454	000000	0	
418				
419	001456	016551	EM32	:"CAN NOT CLEAR BIT6 OF RCSR WITH RESET2"
420	001460	021523	DH6	:"TEST# ERR PC RCSR"
421	001462	022234	DT6	:\$TESTN,\$ERRPC,RCSR
422	001464	000000	0	
423				
424	001466	016617	EM33	:"BIT6 OF LKS NOT CLEAR AFTER RESET"
425	001470	021575	DH10	:"TEST# ERR PC LKS"
426	001472	022254	DT10	:\$TESTN,\$ERRPC,LKS
427	001474	000000	0	
428				
429	001476	016661	EM34	:"LKS INTERRUPT WITH PRIORITY 7"
430	001500	021575	DH10	:"TEST# ERR PC LKS"
431	001502	022254	DT10	:\$TESTN,\$ERRPC,LKS
432	001504	000000	0	
433				
434	001506	016717	EM35	:"CAN NOT SET BIT6 OF LKS"
435	001510	021575	DH10	:"TEST# ERR PC LKS"
436	001512	022254	DT10	:\$TESTN,\$ERRPC,LKS
437	001514	000000	0	
438				
439	001516	016747	EM36	:"CAN NOT CLEAR BIT6 OF LKS"
440	001520	021575	DH10	:"TEST# ERR PC LKS"
441	001522	022254	DT10	:\$TESTN,\$ERRPC,LKS
442	001524	000000	0	
443				
444	001526	017001	EM37	:"RESET DID NOT CLEAR BIT6 OF LKS"

MAIN. MACY11 30,1046) 14-JUN-77 17:10 PAGE 10
 022LOB.P11 14-JUN-77 09:27 ERROR POINTER TABLE

445	001530	021575	DH10	:"TEST# ERR PC LKS"
446	001532	022254	DT10	:"TESTN, \$ERRPC, LKS"
447	001534	000000	0	
448				
449	001536	017041	EM40	:"DUAL ADDRESSING ERROR"
450	001540	021621	DH40	:"TEST# ERR PC GOOD BAC"
451	001542	022264	DT40	:"TESTN, \$ERRPC, \$GDADR, \$BDCSP"
452	001544	000000	0	
453				
454	001546	017067	EM41	:"BIT7 OF LKS NOT SET AFTER RESET"
455	001550	021575	DH10	:"TEST# ERR PC LKS"
456	001552	022254	DT10	:"TESTN, \$ERRPC, LKS"
457	001554	000000	0	
458				
459	001556	017127	EM42	:"CAN NOT CLEAR BIT7 OF LKS"
460	001560	021575	DH10	:"TEST# ERR PC LKS"
461	001562	022254	DT10	:"TESTN, \$ERRPC, LKS"
462	001564	000000	0	
463				
464	001566	017161	EM43	:"BIT7 OF LKS DOES NOT SET"
465	001570	021575	DH10	:"TEST# ERR PC LKS"
466	001572	022254	DT10	:"TESTN, \$ERRPC, LKS"
467	001574	000000	0	
468				
469	001576	017212	EM44	:"RTC INTERRUPT AT PRIORITY"
470	001600	021575	DH10	:"TEST# ERR PC LKS"
471	001602	022254	DT10	:"TESTN, \$ERRPC, LKS"
472	001604	000000	0	
473				
474	001606	017246	EM45	:"RTC INTERRUPTS WHEN DISABLED"
475	001610	021575	DH10	:"TEST# ERR PC LKS"
476	001612	022254	DT10	:"TESTN, \$ERRPC, LKS"
477	001614	000000	0	
478				
479	001616	017303	EM46	:"RTC INTERRUPT DID NOT OCCUR"
480	001620	021575	DH10	:"TEST# ERR PC LKS"
481	001622	022254	DT10	:"TESTN, \$ERRPC, LKS"
482	001624	000000	0	
483				
484	001626	017303	EM47	:"RTC INTERRUPT DID NOT OCCUR"
485	001630	021575	DH10	:"TEST# ERR PC LKS"
486	001632	022254	DT10	:"TESTN, \$ERRPC, LKS"
487	001634	000000	0	
488				
489	001636	017337	EM50	:"RTC DOUBLE INTERRUPT"
490	001640	021575	DH10	:"TEST# ERR PC LKS"
491	001642	022254	DT10	:"TESTN, \$ERRPC, LKS"
492	001644	000000	0	
493				
494	001646	017364	EM51	:"RESET DID NOT CLEAR RTC INTERRUPT"
495	001650	021575	DH10	:"TEST# ERR PC LKS"
496	001652	022254	DT10	:"TESTN, \$ERRPC, LKS"
497	001654	000000	0	
498				
499	001656	017414	EM52	:"RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS"
500	001660	021575	DH10	:"TEST# ERR PC LKS"

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 11
 DDLDLDB.P11 14-JUN-77 09:27 ERROR POINTER TABLE

501	001662	022254	DT10	:STESTN,SERRPC,LKS
502	001664	000000	0	
503				
504	001666	017471	EM53	
505	001670	021655	DH53	: "TEST# ERR PC LKS CNT1 CNT2"
506	001672	022276	DT53	:STESTN,SERRPC,LKS,FIRST,SECND
507	001674	000000	0	
508				
509	001676	017515	EM54	: "XMIT INTERRUPTS WHEN DISABLED"
510	001700	021451	DH1	: "TEST# ERR PC TCSR"
511	001702	022214	DT1	:STESTN,SERRPC,TCSR
512	001704	000000	0	
513				
514	001706	017653	EM55	: "XMIT DID NOT INTERRUPT"
515	001710	021451	DH1	: "TEST# ERR PC TCSR"
516	001712	022214	DT1	:STESTN,SERRPC,TCSR
517	001714	000000	0	
518				
519	001716	017553	EM56	: "XMIT INTERRUPT AT PRIORITY 7"
520	001720	021451	DH1	: "TEST# ERR PC TCSR"
521	001722	022214	DT1	:STESTN,SERRPC,TCSR
522	001724	000000	0	
523				
524	001726	017611	EM57	: "XMIT INTERRUPTS WITH ENABLE CLEAR"
525	001730	021451	DH1	: "TEST# ERR PC TCSR"
526	001732	022214	DT1	:STESTN,SERRPC,TCSR
527	001734	000000	0	
528				
529	001736	017653	EM60	: "XMIT DID NOT INTERRUPT"
530	001740	021451	DH1	: "TEST# ERR PC TCSR"
531	001742	022214	DT1	:STESTN,SERRPC,TCSR
532	001744	000000	0	
533				
534	001746	017702	EM61	: "XMIT RE-INTERRUPTED"
535	001750	021451	DH1	: "TEST# ERR PC TCSR"
536	001752	022214	DT1	:STESTN,SERRPC,TCSR
537	001754	000000	0	
538				
539	001756	017726	EM62	: "LOADING TBUF DID NOT CLEAR INTERRUPT"
540	001760	021451	DH1	: "TEST# ERR PC TCSR"
541	001762	022214	DT1	:STESTN,SERRPC,TCSR
542	001764	000000	0	
543				
544	001766	017773	EM63	: "RCVR ACTIVE NOT SET"
545	001770	021523	DH6	: "TEST# ERR PC RCSR"
546	001772	022234	DT6	:STESTN,SERRPC,RCSR
547	001774	000000	0	
548				
549	001776	020017	EM64	: "RECEIVER DONE NEVER SET"
550	002000	021523	DH6	: "TEST# ERR PC RCSR"
551	002002	022234	DT6	:STESTN,SERRPC,RCSR
552	002004	000000	0	
553				
554	002006	020043	EM65	: "RCVR ACTIVE NOT CLEARED WITH DONE SET"
555	002010	021523	DH6	: "TEST# ERR PC RCSR"
556	002012	022234	DT6	:STESTN,SERRPC,RCSR

557	002014	000000	0	
558				
559	002016	020111	EM66	:"RESET DID NOT CLEAR RCVR DONE"
560	002020	021523	DH6	:"TEST# ERR PC RCSR"
561	002022	022234	DT6	:\$TESTN,\$ERRPC,RCSR
562	002024	000000	0	
563				
564	002026	020147	EM67	:"RDR ENABLE SET DID NOT CLEAR RCVR DONE"
565	002030	021523	DH6	:"TEST# ERR PC RCSR"
566	002032	022234	DT6	:\$TESTN,\$ERRPC,RCSR
567	002034	000000	0	
568				
569	002036	020212	EM70	:"READING RBUF DID NOT CLEAR RCVR DONE"
570	002040	021523	DH6	:"TEST# ERR PC RCSR"
571	002042	022234	DT6	:\$TESTN,\$ERRPC,RCSR
572	002044	000000	0	
573				
574	002046	020257	EM71	:"RCVR INTERRUPTS WITH ENABLE CLEAR"
575	002050	021523	DH6	:"TEST# ERR PC RCSR"
576	002052	022234	DT6	:\$TESTN,\$ERRPC,RCSR
577	002054	000000	0	
578				
579	002056	020426	EM72	:"RCVR DID NOT INTERRUPT"
580	002060	021523	DH6	:"TEST# ERR PC RCSR"
581	002062	022234	DT6	:\$TESTN,\$ERRPC,RCSR
582	002064	000000	0	
583				
584	002066	020321	EM73	:"RCVR INTERRUPTS AT PRIORITY 7"
585	002070	021523	DH6	:"TEST# ERR PC RCSR"
586	002072	022234	DT6	:\$TESTN,\$ERRPC,RCSR
587	002074	000000	0	
588				
589	002076	020357	EM74	:"RCVR INTERRUPT REQUEST PASSED WITH ENABLE CLEAR"
590	002100	021523	DH6	:"TEST# ERR PC RCSR"
591	002102	022234	DT6	:\$TESTN,\$ERRPC,RCSR
592	002104	000000	0	
593				
594	002106	020426	EM75	:"RCVR DID NOT INTERRUPT"
595	002110	021523	DH6	:"TEST# ERR PC RCSR"
596	002112	022234	DT6	:\$TESTN,\$ERRPC,RCSR
597	002114	000000	0	
598				
599	002116	020455	EM76	:"RECEIVER RE-INTERRUPTED"
600	002120	021523	DH6	:"TEST# ERR PC RCSR"
601	002122	022234	DT6	:\$TESTN,\$ERRPC,RCSR
602	002124	000000	0	
603				
604	002126	020501	EM77	:"READING RBUF DID NOT CLEAR INTERRUPT"
605	002130	021523	DH6	:"TEST# ERR PC RCSR"
606	002132	022234	DT6	:\$TESTN,\$ERRPC,RCSR
607	002134	000000	0	
608				
609	002136	020546	EM100	:"RESET DID NOT CLEAR RCVR INTERRUPT"
610	002140	021523	DH6	:"TEST# ERR PC RCSR"
611	002142	022234	DT6	:\$TESTN,\$ERRPC,RCSR
612	002144	000000	0	

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 13
 DZDLDB.P11 14-JUN-77 09:27 ERROR POINTER TABLE

613					
614					
615	002146	020611	EM101		:"'OR' FLAG DID NOT SET"
616	002150	021523	DH6		
617	002152	022234	DT6		;\$TESTN,\$ERRPC,RCSR
618	002154	000000	0		
619					
620	002156	020637	EM102		:"'ERROR' NOT SET WITH 'OR' FLAG"
621	002160	021523	DH6		:"TEST# ERR PC RCSR"
622	002162	022234	DT6		;\$TESTN,\$ERRPC,RCSR
623	002164	000000	0		
624					
625	002166	020676	EM103		:"BREAK DID NOT TRANSMIT ALL ZEROES"
626	002170	021722	DH103		:"TEST# ERR PC RCSR DATA"
627	002172	022312	DT103		;\$TESTN,\$ERRPC,RCSR,\$BDDAT
628	002174	000000	0		
629					
630	002176	020734	EM104		:"BREAK DID NOT SET FRAMING ERROR"
631	002200	021523	DH6		:"TEST# ERR PC RCSR"
632	002202	022234	DT6		;\$TESTN,\$ERRPC,RCSR
633	002204	000000	0		
634					
635	002206	020771	EM105		:"DATA COMPARE ERROR"
636	002210	021757	DH105		:"TEST# ERR PC RCSR GOOD BAD"
637	002212	022324	DT105		;\$TESTN,\$ERRPC,RCSR,GD,BD
638	002214	000000	0		
639					
640	002216	021014	EM106		:"DATA COMPARE ERROR WITH CABLE"
641	002220	021757	DH105		:"TEST# ERR PC RCSR GOOD BAD"
642	002222	022324	DT105		;\$TESTN,\$ERRPC,RCSR,GD,BD
643	002224	000000	0		
644					
645	002226	021052	EM107		:"TIMEOUT IN EXERCISER TEST"
646	002230	021523	DH6		:"TEST# ERR PC RCSR"
647	002232	022234	DT6		;\$TESTN,\$ERRPC,RCSR
648	002234	000000	0		
649					
650	002236	021104	EM110		:"INCORRECT RECEIVE COUNT"
651	002240	022023	DH110		:"TEST# ERR PC RCSR TRANS RCV"
652	002242	022340	DT110		;\$TESTN,\$ERRPC,RCSR,XMTCNT,RCVONT
653	002244	000000	0		
654					
655	002246	021134	EM111		:"DATA COMPARE ERROR IN EXERCISER"
656	002250	021757	DH105		:"TEST# ERR PC RCSR GOOD BAD"
657	002252	022324	DT105		;\$TESTN,\$ERRPC,RCSR,GD,BD
658	002254	000000	0		
659					
660	002256	021174	EM112		:"TRAP CATCHER"
661	002260	022067	DH112		:"TEST# ERR PC RCSR OLDFC TRAP ACR"
662	002262	022354	DT112		;\$TESTN,\$ERRPC,RCSR,OLDFC,BDVECT
663	002264	000000	0		
664					
665	002266	021211	EM113		:"NO CLK INTERRUPT IN EXERCISER"
666	002270	021575	DH10		:"TEST# ERR PC LKS"
667	002272	022254	DT10		;\$TESTN,\$ERRPC,LKS
668	002274	000000	0		

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 14
 DZC_DB.P11 14-JUN-77 09:27 ERROR POINTER TABLE

669					
670	002276	021247	EM114		;'ERROR' NOT SET WITH 'FR' FLAG"
671	002300	021523	DH6		;"TEST# ERR PC RCSR"
672	002302	022234	DT6		;\$TESTN,\$ERRPC,RCSR
673	002304	000000	0		
674					
675	002306	021306	EM115		;RCV ACTVIE NOT CLEAR WITH INIT
676	002310	021523	DH6		;"TEST# ERR PC RCSR"
677	002312	022234	DT6		;\$TESTN,\$ERRPC,RCSR
678	002314	000000	0		
679					
680	002316	021345	EM116		;RCV ACTIVE WITHOUT "START" BIT
681	002320	021523	DH6		;"TEST# ERR PC RCSR"
682	002322	022234	DT6		;\$TESTN,\$ERRPC,RCSR
683	002324	000000	0		
684					
685	002326	021404	EM117		;RDR ENABLE NOT CLEAR WITH RCV ACTIVE
686	002330	021523	DH6		;"TEST# ERR PC RCSR"
687	002332	022234	DT6		;\$TESTN,\$ERRPC,RCSR
688	002334	000000	0		
689					
690	002336	000000	CTSTFL: .WORD	0	;CONSOLE UNDER TEST FLAG
691	002340	000000	TMP1: .WORD	0	;TEMP LOCATION FOR TABLE OFFSETS
692	002342	000000	TMP2: .WORD	0	;TEMP LOCATION FOR DEVICE COUNT
693	002344	000000	TMP3: .WORD	0	;LOCATION FOR DEVICE MAP BIT TEST MASK
694			;REGISTER AND VECTOR ADDRESSES FOR THE DL-11W UNDER TEST		
695					
696	002346	000000	RCSR: .WORD	0	
697	002350	000000	RBUF: .WORD	0	
698	002352	000000	TCSR: .WORD	0	
699	002354	000000	TBUF: .WORD	0	
700	002356	000000	RVECT: .WORD	0	
701	002360	000000	RPSW: .WORD	0	
702	002362	000000	TVECT: .WORD	0	
703	002364	000000	TPSW: .WORD	0	
704					
705			;CONSOLE REGISTER AND VECTOR ADDRESSES FOR THE DL-11W		
706					
707	002366	177560	CRCSR: 177560		;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
708	002370	177562	CRBUF: 177562		;ADDRESS OF RECEIVER BUFFER
709	002372	177564	CTCSR: 177564		;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
710	002374	177566	CTBUF: 177566		;ADDRESS OF TRANSMITTER BUFFER
711	002376	000060	CRVECT: 60		;RECEIVER INTERRUPT VECTOR
712	002400	000062	CRPSW: 62		
713	002402	000064	CTVECT: 64		;TRANSMITTER INTERRUPT VECTOR
714	002404	000066	CTPSW: 66		
715					
716			;REAL TIME CLOCK REGISTER AND VECTOR ADDRESSES		
717	002406	177546	LKS: .WORD	177546	
718	002410	000100	RTCVT: .WORD	100	
719	002412	000102	RTCPSW: .WORD	102	
720					
721	002414	000020	ADRTBL: .BLKW	20	
722	002454	000020	VCTTBL: .BLKW	20	
723					
724					

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 15
DZDLDB.P11 14-JUN-77 09:27 ERROR POINTER TABLE

```

725      :SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE
726
727 002514 012702 002414  DEVADR: MOV      #ADRTBL,R2      ;POINT R2 TO THE DEVICE ADDRESS TABLE
728 002520 016700 176416      MOV      $BASE,R0      ;LOAD BASE DEVICE ADDRESS IN R0
729 002524 C10001      MOV      R0,R1      ;
730 002526 062701 000170      ADD      #170,R1      ;POINT R1 TO LAST DEVICE ADDRESS
731 002532 010022 1$:      MOV      R0,(R2)+      ;MOVE DEVICE ADDRESS TO TABLE
732 002534 062700 000010      ADD      #10,R0      ;POINT R0 TO NEXT DEVICE ADDRESS
733 002540 020001      CMP      R0,R1      ;FINISHED GENERATING TABLE?
734 002542 003773      BLE     1$          ;BR, IF LAST DEVICE ADDRESS NOT LOADED
735 002544 000207      RTS     PC
736
737
738
739 002546 005067 176316  START: CLR      $FATAL      ;CLEAR ERROR NO.
740 002552 005067 176310      CLR      $MSGTYP      ;CLEAR MESSAGE TYPE
741 002556 005067 176310      CLR      $TESTN      ;CLEAR TEST NO.
742 002562 005067 177550      CLR      CTSTFL      ;CLEAR CONSOLE UNDER TEST FLAG
743 002566 005067 176304      CLR      $DEVCT      ;CLEAR DEVICE COUNT
744 002572 005067 176302      CLR      $UNIT      ;CLEAR UNIT NUMBER
745 002576 005767 176310      TST     $USWR      ;IS $USWR LOADED?
746 002602 001003      BNE     1$          ;BR IF YES
747 002604 012767 000400 176300  MOV     #400,$USWR      ;ELSE, DEFAULT TO $USWR=400
748 002612 012737 000006 000004 1$:      MOV     #6,@#4      ;INITIALIZE TIMEOUT VECTORS TO TRAP
749 002620 012737 000003 000006      MOV     #3,@#6      ; CATCHER ROUTINE
750
751 .SBTTL INITIALIZE THE COMMON TAGS
752 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
753 002626 012706 001000      MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
754 002632 005026      CLR     (R6)+          ;;CLEAR MEMORY LOCATION
755 002634 022706 001040      CMP     #SWR,R6 ;;DONE?
756 002640 001374      BNE     -6            ;;LOOP BACK IF NO
757 002642 012706 001000      MOV     #1000,SP      ;;SETUP THE STACK POINTER
758
759 002646 012737 013000 000020  ::INITIALIZE A FEW VECTORS
760 002654 012737 000340 000022      MOV     $$SCOPE,@#IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
761 002662 012737 012304 000030      MOV     #340,@#IOTVEC+2 ;; LEVEL 7
762 002670 012737 000340 000032      MOV     $ERROR,@#EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
763 002676 012737 014704 000034      MOV     #340,@#EMTVEC+2 ;; LEVEL 7
764 002704 012737 000340 000036      MOV     $TRAP,@#TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
765 002712 012737 012622 000024      MOV     #340,@#TRAPVEC+2; LEVEL 7
766 002720 012737 000340 000026      MOV     $PWARN,@#PWRVEC ;; POWER FAILURE VECTOR
767 002726 016767 007200 007170      MOV     #340,@#PWRVEC+2 ;; LEVEL 7
768 002734 005067 176120      MOV     $ENDCT,$EOPCT ;; SETUP END-OF-PROGRAM COUNTER
769 002740 112767 000001 176047      CLR     $ESCAPE      ;; CLEAR THE ESCAPE ON ERROR ADDRESS
770 002746 012767 002746 176032      MOV     #1,$ERMAX      ;; ALLOW ONE ERROR PER TEST
771 002754 012767 002754 176026      MOV     #,$SLPADR      ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
772      MOV     #,$SLPERR      ;; SETUP THE ERROR LOOP ADDRESS
773      ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
774      ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
775 002762 013746 000004      MOV     @#ERRVEC,-(SP) ;; SAVE ERROR VECTOR
776 002766 012737 003022 000004      MOV     #E4$,@#ERRVEC ;; SET UP ERROR VECTOR
777 002774 012767 177570 176036      MOV     #DSWR,SWR      ;; SETUP FOR A HARDWARE SWITCH REGISTER
778 003002 012767 177570 176032      MOV     #DDISP,DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
779 003010 022777 177777 176022      CMP     #-1,$SWR      ;; TRY TO REFERENCE HARDWARE SWR
780 003016 001012      BNE     66$          ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
781      ;; AND THE HARDWARE SWR IS NOT = -1

```


INITIALIZE THE COMMON TAGS

```

781 003020 000403          BR      65$      ;; BRANCH IF NO TIMEOUT
782 003022 012716 003030 64$:  MOV      #65$, (SP)  ;; SET UP FOR TRAP RETURN
783 003026 000002          RTI
784 003030 012767 000176 176002 65$:  MOV      #SWREG, SWR  ;; POINT TO SOFTWARE SWR
785 003036 012767 000174 175776  MOV      #DISPRÉG, DISPLAY
786 003044 012637 000004 66$:  MOV      (SP)+, @#ERRVEC  ;; RESTORE ERROR VECTOR
787
788 003050 005067 176020          CLR      $PASS      ;; CLEAR PASS COUNT
789 003054 132767 000200 176025  BITB     #APTSIZE, $ENVM  ;; TEST USER SIZE UNDER APT
790 003062 001403          BEQ      67$      ;; YES, USE NON-APT SWITCH
791 003054 012767 001110 175746  MOV      #SSWREG, SWR  ;; NO, USE APT SWITCH REGISTER
792 003072
793 003072 032777 000020 175740 67$:  BIT      #BIT4, @SWR  ;; TEST CLOCK ONLY?
794 003100 001404          BEQ      INIT      ;; BR IF NOT
795 003102 005267 177230          INC      CTSTFL     ;; ELSE, SET CONSOLE TEST FLAG TO ENABLE CLOCK TESTS
796 003106 000167 000744          JMP      ID        ;; AND JUMP TO TYPE PROGRAM ID
797 003112 132767 000001 175766 INIT:  BITB     #BIT0, $ENV  ;; CHECK IF ON APT
798 003120 001404          BEQ      MANL     ;; BR IF NOT APT
799 003122 132767 000200 175757  BITB     #BIT7, $ENVM  ;; DID APT SIZE
800 003130 001056          BNE      APTSZD   ;; BR, IF APT SIZED
801 003132 032777 000040 175700 MANL:  BIT      #BITS, @SWR  ;; WAS "$DEVN" MANUALLY SET?
802 003140 001052          BNE      APTSZD   ;; IF YES, SKIP SELF-SIZING
803
804 003142 004767 177346  SIZE:  JSR      PC, DEVADR  ;; GENERATE DEVICE ADDRESS TABLE
805 003146 005067 177170          CLR      TMP2      ;; CLR TEMP LOCATION TO KEEP DEVICE COUNT
806 003152 005067 175766          CLR      $DEVN     ;; CLEAR DEVICE MAP
807 003156 013703 000004          MOV      @#4, R3   ;; SAVE TIMEOUT VECTOR
808 003162 012737 003212 000004  MOV      #4$, @#4   ;; SET TIMEOUT POINTER
809 003170 016700 175746          MOV      $BASE, R0  ;; LOAD BASE ADDRESS
810 003174 062700 000160          ADD      #160, R0   ;; POINT R0 TO UNIT #15 (UNIT#0=CONSOLE)
811 003200 005710          TST      (R0)      ;; CHECK FOR DEVICE EXISTANCE
812 003202 005267 175736          INC      $DEVN     ;; INDICATE DEVICE EXISTANCE IN DEVICE MAP
813 003206 005267 177130          INC      TMP2      ;; INCREMENT DEVICE COUNT
814 003212 012706 001000 4$:  MOV      #1000, SP  ;; RESET STACK POINTER
815 003216 006367 175722          ASL      $DEVN     ;; ADJUST DEVICE MAP FOR NEXT UNIT CHECK
816 003222 162700 000010          SUB      #10, R0   ;; POINT R0 TO NEXT DEVICE NUMBER
817 003226 026700 175710          CMP      $BASE, R0  ;; FINISHED SIZING?
818 003232 003762          BLE      3$      ;; BR, IF BASE ADDRESS HAS NOT BEEN CHECKED
819 003234 016700 177126          MOV      CCRSR, R0  ;; LOAD CONSOLE DEVICE ADDRESS
820 003240 012737 003260 000004  MOV      #5$, @#4   ;; SET UP TIMEOUT POINTER
821 003246 005710          TST      (R0)      ;; TEST FOR CONSOLE EXISTANCE
822 003250 005267 175670          INC      $DEVN     ;; INDICATE CONSOLE EXISTANCE IN DEVICE MAP
823 003254 005267 177062          INC      TMP2      ;; INCREMENT DEVICE COUNT
824 003260 010337 000004 5$:  MOV      R3, @#4   ;; RESTORE TIMEOUT VECTOR
825 003264 000415          BR      VCTADR    ;; BR TO GENERATE VECTOR ADDRESS TABLE
826
827 003266 005067 177050  APTSZD: CLR      TMP2      ;; CLEAR TEMP LOCATION TO KEEP DEVICE CNT
828 003272 016702 175646          MOV      $DEVN, R2  ;; MOVE DEVICE MAP TO R2
829 003276 005702  TSTDVM: TST      R2      ;; TEST MSB OF DEVICE MAP
830 003300 100002          BPL      1$      ;; BR, IF MSB IS ZERO
831 003302 005267 177034          INC      TMP2      ;; INCREMENT DEVICE COUNT, IF MSB=1
832 003306 006302 1$:  ASL      R2        ;; SHIFT NEXT BIT INTO MSB POSITION
833 003310 001401          BEQ      DVACT    ;; BR, IF NO OTHER BITS ARE SET IN $DEVN
834 003312 000771          BR      TSTDVM   ;; CONTINUE CHECKING $DEVN, IF MORE BITS SET
835 003314 004767 177174  DVACT: JSR      PC, DEVADR  ;; GENERATE DEVICE ADDRESS TABLE
    
```

INITIALIZE THE COMMON TAGS

```

837      :GENERATE VECTOR ADDRESS TABLE
838
839      003320 012702 002454 VCTADR: MOV      #VCTTBL,R2      ;GET LOCATION OF VECTOR TABLE
840      003324 113700 001136      MOVB     @#$VECT1,R0    ;COPY BASE VECTOR
841      003330 042700 177400      BIC     #177400,R0     ;CLEAR BYTE SIGN EXTENSION
842      003334 010001          MOV     R0,R1         ;
843      003336 062701 000170      ADD     #170,R1       ;POINT R1 TO LAST DEVICE VECTOR
844      003342 010022          IS:  MOV     R0,(R2)+    ;PUT VECTOR ADDRESS IN TABLE
845      003344 062700 000010      ADD     #10,R0        ;POINT R0 TO NEXT VECTOR ADDRESS
846      003350 020001          CMP     R0,R1         ;FINISHED GENERATING VECTOR TABLE?
847      003352 003773          BLE     IS           ;BR, IF LAST VECTOR IS NOT LOADED
848
849      :MOVE DEVICE COUNT INTO DEVICE COUNT MESSAGE
850
851      003354 016700 176762          MOV     TMP2,R0       ;COPY DEVICE COUNT INTO R0
852      003360 005001          CLR     R1            ;CLEAR AUXILIARY REGISTER
853      003362 000300          SWAB   R0            ;PUT DEVICE COUNT IN UPPER BYTE OF R0
854      003364 006300          ASL    R0            ;MOVE MSB OF COUNT INTO
855      003366 006300          ASL    R0            ;MSB OF R0
856      003370 006300          SHIFT: ASL    R0       ;PUT MSB OF COUNT INTO CARRY
857      003372 106101          ROLB   R1            ;MOVE MSB OF COUNT INTO R1
858      003374 006300          ASL    R0            ;MOVE NEXT BIT TO CARRY
859      003376 106101          ROLB   R1            ;MOVE INTO R1
860      003400 006300          ASL    R0            ;MOVE LAST BIT OF DIGIT
861      003402 106101          ROLB   R1            ;INTO R1
862      003404 062701 000060          ADD     #60,R1       ;CONVERT DIGIT TO ASCII
863      003410 000301          SWAB   R1            ;MOVE DIGIT TO UPPER BYTE
864      003412 032701 000020          BIT     #BIT4,R1     ;HAVE BOTH DIGITS BEEN MOVED TO R1?
865      003416 001764          BEQ    SHIFT        ;BR, IF NOT
866      003420 010167 016540          MOV     R1,M2A       ;MOVE DEVICE COUNT TO OUTPUT MESSAGE
867
868
869      003424 052767 000002 176712 BEGIN: BIS     #BIT1,TMP3    ;SET UP BIT MASK TO TEST $DEVN FOR DEVICES EXCEPT CONSOL
870      003432 005067 176702          CLR     TMP1         ;CLEAR LOCATION TO STORE TABLE OFFSETS
871      003436 032767 000001 175500          BIT     #BIT0,$DEVN  ;IS CONSOLE TO BE TESTED?
872      003444 001001          BNE    TCONS        ;BR, IF CONSOLE IS TO BE TESTED
873      003446 000414          BR     TSTDEV       ;BR, TO TEST OTHER DEVICES
874      003450 005267 176662          TCONS: INC    CTSTFL  ;INDICATE CONSOLE UNDER TEST
875      003454 012700 002366          MOV     #CRCSR,R0    ;SET UP CONSOLE DEVICE ADDRESSES
876      003460 012701 002346          MOV     #RCSR,R1     ;POINT R1 TO UUT ADDRESS TABLE
877      003464 012021          IS:  MOV     (R0)+,(R1)+ ;TRANSFER CONSOLE ADDRESSES
878      003466 022701 002364          CMP     #TPSW,R1     ;FINISHED TRANSFER?
879      003472 002374          BGE    IS           ;BR, IF NOT
880      003474 000167 000122          JMP     TST1        ;GO TEST CONSOLE INTERFACE
881
882      :PREPARE ADDRESSES AND VECTORS FOR UUT
883      003500 036767 176640 175436 TSTDEV: BIT     TMP3,$DEVN ;CHECK TO SEE IF DEVICE IS TO BE TESTED
884      003506 001010          BNE    SETADR       ;BR, IF YES
885      003510 006367 176630          ASL    TMP3         ;SHIFT MASK TO CHECK NEXT $DEVN BIT
886      003514 062767 000002 176616          ADD     #2,TMP1      ;INCREMENT TABLE INDEX
887      003522 005267 175352          INC    $UNIT        ;INCREMENT UNIT NUMBER
888      003526 000764          BR     TSTDEV       ;GO TEST NEXT BIT OF DEVICE MAP
889
890      003530 005267 175344          SETADR: INC    $UNIT  ;UPDATE UNIT NUMBER
891      003534 006367 176604          ASL    TMP3         ;UPDATE DEVICE MAP TEST MASK
892      003540 016702 176574          MOV     TMP1,R2     ;MOVE TABLE OFFSET TO R2

```

INITIALIZE THE COMMON TAGS

```

893 003544 062767 000002 176566      ADD    #2,TMP1      :UPDATE TABLE OFFSET FOR NEXT DEVICE
894 003552 016200 002414      MOV    ADRTBL(R2),RO :PUT UUT ADDRESS INTO RO
895 003556 012701 002346      MOV    #RCSR,R1    :POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
896 003562 010021      ADR:  MOV    RO,(R1)+ :TRANSFER UUT ADDRESS
897 003564 062700 000002      ADD    #2,RO       :POINT TO NEXT UUT REGISTER
898 003570 030027 000006      BIT    RO,#6       :FINISHED TRANSFER?
899 003574 001372      BNE    ADR         :BR, IF NOT
900
901 003576 016200 002454      MOV    VCTTBL(R2),RO :PUT UUT VECTOR INTO RO
902 003602 010021      VECT: MOV    RO,(R1)+ :TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
903 003604 062700 000002      ADD    #2,RO       :POINT TO NEXT VECTOR
904 003610 030027 000006      BIT    RO,#6       :FINISHED TRANSFER?
905 003614 001372      BNE    VECT       :BR, IF NOT
906 003616 000167 000000      JMP    TST1       :GC TEST DEVICE

```

```

909
910
911
912
913
914 003622 000004
915 003624 013703 000004
916 003630 012737 003644 000004
917 003636 005777 176510
918 003642 000412
919
920 003644 022626
921 003646 005767 176464
922 003652 001002
923 003654 104001
924 003656 000404
925 003660
926 003660 004767 007304
927 003664 000001
928 003666 000000
929 003670 010337 000004
930
931
932
933
934
935
936 003674 000004
937 003676 013703 000004
938 003702 012737 003716 000004
939 003710 005777 176440
940 003714 000412
941
942 003716 022626
943 003720 005767 176412
944 003724 001002
945 003726 104002
946 003730 000404
947 003732
948 003732 004767 007232
949 003736 000002
950 003740 000000
951 003742 010337 000004

::*****
:*TEST 1 TEST ABILITY TO REFERENCE TCSR
::*****
↑ST1: SCOPE
MOV R3,R3 ;SAVE TIMEOUT VECTOR
MOV R1,R3 ;SET UP TIMEOUT VECTOR
TST TCSR ;REFERENCE THE XMIT COMMAND/STATUS REG.
BR 4$ ;GO TO END OF TEST

1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 2$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 1 ;REPORT ERROR TO APT & TTY
BR 4$ ;BR TO END OF TEST

2$: JSR PC,SATY4 ;:ONLY REPORT A FATAL ERROR
;:THE ERROR NUMBER FROM APT LIST

3$: HALT
4$: MOV R3,R3 ;RESTORE TIMEOUT VECTOR

::*****
:*TEST 2 TEST ABILITY TO REFERENCE TBUF
::*****
↑ST2: SCOPE
MOV R3,R3 ;SAVE TIMEOUT VECTOR
MOV R1,R3 ;SET UP TIMEOUT VECTOR
TST TBUF ;REFERENCE THE XMIT BUFFER
BR 4$ ;GO TO END OF TEST

1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 2$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 2 ;REPORT ERROR TO APT & TTY
BR 4$ ;BR TO END OF TEST

2$: JSR PC,SATY4 ;:ONLY REPORT A FATAL ERROR
;:THE ERROR NUMBER FROM APT LIST

3$: HALT
4$: MOV R3,R3 ;RESTORE TIMEOUT VECTOR

```

TEST ABILITY TO REFERENCE TBUF

```

952
953
954
955
956
957 003746 000004
958 003750 005077 176400
959 003754 105777 176372
960 003760 100016
961
962
963 003762 005077 176366
964 003766 105777 176360
965 003772 100011
966
967 003774 005767 176336
968 004000 001002
969 004002 104003
970 004004 000404
971 004006
972 004006 004767 007156
973 004012 000003
974 004014 000000
975 004016 005000
976 004020 105777 176326
977 004024 100414
978 004026 005200
979 004030 001373
980
981 004032 005767 176300
982 004036 001002
983 004040 104004
984 004042 000405
985 004044
986 004044 004767 007120
987 004050 000004
988 004052 000000
989 004054 000424
990
991
992 004056 023737 000042 000046 10: CMP 0#42,0#46
993 004064 001412 66 BEQ 66
994 004066 005767 175002 66 TST $PASS
995 004072 001007 66 BNE 66
996 004074 005767 174776 66 TST $DEVCT
997 004100 001004 66 BNE 66
998 004102 104401 TYPE
999 004104 022140 M1
1000 004106 104401 TYPE
1001 004110 022162 M2
1002 004112 032777 000020 174720 66: BIT #BIT4,$SWP
1003 004120 001402 TST
1004 004122 000167 000730 JMP TOLCOP

```

```

*****
*TEST 3 TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED
*****
TST3: SCOPE
CLR QTBUF :LOAD XBUF
TSTB QTCSR :CHECK DONE
BPL 35 :BR IF CLEAR
:FILL SECOND BUFFER, BECAUSE REFRESH COULD CAUSE
: FIRST TEST TO FAIL
CLR QTBUF :FILL DOUBLE BUFFER
TSTB QTCSR :CHECK DONE
BPL 35 :BR IF CLEAR

TST CTSTFL :CHECK IF DEVICE IS CONSOLE
BNE 15 :IF YES, SKIP ERROR TYPEOUT
ERROR 3 :DONE NOT CLEARED WITH TBUF FULL
BR 35 :BR TO END OF TEST

15: JSR PC,$ATY4 :ONLY REPORT A FATAL ERROR
3 :THE ERROR NUMBER FROM APT LIST
25: HALT :TCSR "DONE" NOT CLEARED WITH TBUF FULL
35: CLR RD :CLEAR TIMER
45: TSTB QTCSR :CHECK FOR XMIT DONE
BMI 10 :IF DONE SETS, BR TO END OF TEST
INC RD :INCREMENT TIMER
BNE 45 :BR IF TIMER NOT DONE

TST CTSTFL :CHECK IF DEVICE IS CONSOLE
BNE 55
ERROR 4 :TCSR "DONE" DOES NOT SET
BR 10 :BR TO END OF TEST

55: JSR PC,$ATY4 :ONLY REPORT A FATAL ERROR
4 :THE ERROR NUMBER FROM APT LIST
HALT
BR TST4 :BR TO NEXT TEST, AND SKIP THE TYPEOUT THAT FOLLOWS
: BECAUSE OF THIS FAILURE

10: CMP 0#42,0#46 :UNDER ACT11?
66 BEQ 66 :IF YES, SKIP IDENT. TYPEOUT
66 TST $PASS :IS THIS THE FIRST PASS?
66 BNE 66 :IF NOT BR TO NEXT TEST & SKIP THE IDENTIFICATION TYPEOUT
66 TST $DEVCT :IS THIS THE FIRST SUBPASS?
66 BNE 66 :IF NOT, BR TO NEXT TEST
TYPE :TYPE PROGRAM IDENTIFICATION

M1 :TYPE NUMBER OF DEVICES UNDER TEST
M2
66: BIT #BIT4,$SWP :CLOCK TEST ONLY?
TST :BR IF NOT
JMP TOLCOP :ELSE, JUMP TO TEST CLOCK

```

TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED

```

1005
1006
1007
1008
1009
1010 004126 000004
1011 004130 005077 176220
1012 004134 105777 176212
1013 004140 100375
1014 004142 005077 176206
1015 004146 000240
1016 004150 000005
1017 004152 105777 176174
1018 004156 100401
1019
1020 004160 104005
1021
1022
1023
1024
1025
1026
1027 004162 000004
1028 004164 013703 000004
1029 004170 012737 004204 000004
1030 004176 005777 176144
1031 004202 000402
1032
1033 004204 022626
1034 004206 104002
1035 004210 010002 000004

```

```

:*****
:*TEST 4 TEST THAT TCSR "DONE" SETS WITH RESET
:*****
↑ST4: SCOPE
CLR @TBUF ;LOAD TRANSMIT BUFFER
1$: TSTB @TCSR ;WAIT FOR DONE
BPL 1$
CLR @TBUF ;LOAD SECOND BUFFER
NOP
RESET ;CLEAR DONE WITH RESET
TSTB @TCSR ;CHECK FOR DONE SET
BMI TST5 ;BR TO NEXT TEST IF DONE SET
ERROR 5 ;TCSR "DONE" DOES NOT SET WITH RESET

```

```

:*****
:*TEST 5 TEST ABILITY TO ACCESS RCSR
:*****
↑T5: SCOPE
MOV @#4,R3 ;SAVE TIMEOUT VECTOR
MOV @1$,@#4 ;SET UP TIMEOUT VECTOR
TST @RCSR ;ACCESS RCSR
BR 2$ ;BR TO END OF TEST
1$: CMP (SP)+,SP,+ ;RESTORE SP AFTER TIMEOUT
ERROR 5 ;CAN NOT ACCESS RCSR
2$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR

```

.MAIN. MACY11 30.10461 14-JUN-77 17:10 PAGE 22
DZDLDB.P11 14-JUN-77 09:27 TS TEST ABILITY TO ACCESS RCSR

10036
10037
10038
10039
10040
10041
10042
10043
10044
10045
10046
10047
10048
10049
10050
10051
10052
10053
10054
10055
10056
10057
10058
10059
10060
10061
10062
10063
10064
10065
10066
10067
10068
10069
10070
10071
10072
10073
10074
10075
10076
10077
10078
10079
10080
10081
10082
10083
10084
10085
10086
10087
10088
10089
10090
10091
10092
10093
10094
10095
10096
10097
10098
10099
10100

004214 000004
004216 013703 000004
004222 012737 004236 000004
004230 005777 176114
004234 000402

004236 022626
004240 104007
004242 010337 000004

```
*****  
: *TEST 6 TEST ABILITY TO ACCESS RBUF  
: *****  
TST6: SCOPE  
MOV R4,R3 ;SAVE TIMEOUT VECTOR  
MOV R15,R4 ;SET UP TIMEOUT VECTOR  
TST RBUF ;ACCESS RBUF  
BR R5 ;BR TO END OF TEST  
  
18: CMP ,SP+, ,SP,+ ;RESTORE SP AFTER TIMEOUT  
ERROR ;CAN NOT ACCESS RBUF  
28: MOV R3,R4 ;RESTORE TIMEOUT VECTOR
```



```

1054
1055
1056
1057
1058
1059 004246 000004
1060 004250 032777 000400 174562
1061 004256 001545
1062 004260 000005
1063 004262 032777 000001 176062
1064 004270 001411
1065 004272 005767 176040
1066 004276 001002
1067 004300 104011
1068 004302 000404
1069
1070 004304 004767 006660
1071 004310 000011
1072 004312 000000
1073
1074 004314 052777 000001 176030
1075 004322 032777 000001 176022
1076 004330 001001
1077
1078 004332 104012
1079
1080 004334 042777 000001 176010
1081 004342 032777 000001 176002
1082 004350 001411
1083 004352 005767 175760
1084 004356 001002
1085 004360 104013
1086 004362 000404
1087 004364
1088 004364 004767 006600
1089 004370 000013
1090 004372 000000
1091
1092 004374 052777 000001 175750
1093 004402 000005
1094 004404 032777 000001 175740
1095 004412 001467
1096 004414 042777 000001 175730
1097 004422 104014

```

```

*****
*TEST 7 TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET
*****
TST7: SCOPE
BIT #BIT0,%SWR ;IS BREAK FUNCTION ENABLED?
BEQ TST11 ;BR TO NEXT TEST, IF NOT
RESET ;CLEAR EVERYTHING
BIT #BIT0,%TCSR ;CHECK BIT0 OF TCSR CLEAR
BEQ 3$ ;BR IF CLEAR
TST CTSTFL
BNE 1$
ERROR 11 ;BIT0 WAS NOT CLEAR AFTER RESET
BR 3$
1$: JSR PC,%SATY4 ;: ONLY REPORT A FATAL ERROR
!! ;: THE ERROR NUMBER (FROM APT LIST)
2$: HALT
3$: BIS #BIT0,%TCSR ;SET BIT0 IN TCSR
BIT #BIT0,%TCSR ;TEST BIT0 OF TCSR
BNE 4$ ;BR IF SET
4$: ERROR 12 ;BIT0 OF TCSR WILL NOT SET
5$: BIC #BIT0,%TCSR ;CLEAR BIT0 OF TCSR
BIT #BIT0,%TCSR ;TEST BIT0 OF TCSR
BEQ 7$
TST CTSTFL
BNE 5$
ERROR 13 ;BIT0 OF TCSR WILL NOT CLEAR
BR 7$
6$: JSR PC,%SATY4 ;: ONLY REPORT A FATAL ERROR
13 ;: THE ERROR NUMBER (FROM APT LIST)
7$: HALT
8$: BIS #BIT0,%TCSR ;SET BIT0 IN TCSR
RESET ;CLEAR BIT0 WITH RESET
BIT #BIT0,%TCSR ;TEST BIT0 CLEAR
BEQ TST11 ;BR IF CLEAR
BIC #BIT0,%TCSR ;CLEAR BIT0, TO PRINT ERROR
ERROR 14 ;RESET DID NOT CLEAR BIT0 OF TCSR

```

TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET

```

1098
1099
1100
1101
1102
1103 004424 000004
1104 004426 000005
1105 004430 032777 000004 175714
1106 004436 001411
1107
1108 004440 005767 175672
1109 004444 001002
1110 004446 104015
1111 004450 000404
1112
1113 004452
1114 004452 004767 006512
1115 004456 000015
1116 004460 000000
1117
1118 004462 052777 000004 175662
1119 004470 032777 000004 175654
1120 004476 001001
1121
1122 004500 104016
1123
1124 004502 042777 000004 175642
1125 004510 032777 000004 175634
1126 004516 001411
1127
1128 004520 005767 175612
1129 004524 001002
1130 004526 104017
1131 004530 000404
1132
1133 004532 004767 006432
1134 004536 000017
1135 004540 000000
1136
1137 004542 052777 000004 175602
1138 004550 000005
1139 004552 032777 000004 175572
1140 004560 001461
1141
1142 004562 042777 000004 175562
1143 004570 104020
1144

```

```

*****
*TEST 10 TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET
*****
TST10: SCOPE
RESET ;CLEAR EVERYTHING
BIT #BIT2,@TCSR ;TEST FOR BIT2 OF TCSR CLEAR
BEQ 3$ ;BR IF CLEAR
TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 1$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 15 ;BIT2 OF TCSR NOT CLEAR AFTER RESET
BR 3$
1$: JSR PC,$ATY4 ;: ONLY REPORT A FATAL ERROR
;: THE ERROR NUMBER (FROM APT LIST)
2$: HALT
3$: BIS #BIT2,@TCSR ;SET BIT2 OF TCSR
BIT #BIT2,@TCSR ;TEST FOR BIT2 SET
BNE 4$ ;BR IF SET
ERROR 16 ;BIT2 OF TCSR WILL NOT SET
4$: BIC #BIT2,@TCSR ;CLEAR BIT2 OF TCSR
BIT #BIT2,@TCSR ;TEST BIT2 CLEAR
BEQ 7$ ;BR IF CLEAR
TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 5$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 17
BR 7$
5$: JSR PC,$ATY4 ;: ONLY REPORT A FATAL ERROR
;: THE ERROR NUMBER (FROM APT LIST)
6$: HALT ;BIT0 OF TCSR WILL NOT CLEAR
7$: BIS #BIT2,@TCSR ;SET BIT2 OF TCSR
RESET ;CLEAR BIT2 WITH RESET
BIT #BIT2,@TCSR ;TEST FOR BIT2 CLEAR
BEQ TST12 ;IF CLEAR, GO TO NEXT TEST
BIC #BIT2,@TCSR ;CLEAR BIT2, TO PRINT ERROR
ERROR 20 ;RESET DID NOT CLEAR BIT2 OF TCSR

```

TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET

```

1145
1146
1147
1148
1149
1150 004572 000004
1151 004574 000005
1152 004576 017703 175560
1153 004602 012777 004632 175552
1154 004610 004767 005426
1155 004614 000340
1156 004616 032777 000100 175526
1157 004624 001404
1158 004626 104021
1159
1160 004630 000402
1161
1162 004632 022626 1$: CMP (SP)+, (SP)+
1163 004634 104022 ERROR 22 ;RESTORE SP AFTER INTERRUPT
1164
1165
1166 004636 052777 000100 175506 2$: BIS #BIT6, @TCSR ;SET BIT6 OF TCSR
1167 004644 032777 000100 175500 BIT #BIT6, @TCSR ;TEST BIT6 OF TCSR
1168 004652 001001 BNE 3$ ;BR, IF SET
1169
1170 004654 104023 ERROR 23 ;CANNOT SET BIT6 OF TCSR
1171
1172
1173 004656 042777 000100 175466 3$: BIC #BIT6, @TCSR ;CLEAR BIT6 OF TCSR
1174 004664 032777 000100 175460 BIT #BIT6, @TCSR ;TEST BIT6 OF TCSR
1175 004672 001401 BEQ 4$ ;BR IF CLEAR
1176 004674 104024 ERROR 24 ;CANNOT CLEAR BIT6 OF TCSR
1177
1178
1179 004676 052777 000100 175446 4$: BIS #BIT6, @TCSR ;SET BIT6 OF TCSR
1180 004704 000005 RESET ;CLEAR BIT6 WITH RESET
1181 004706 032777 000100 175436 BIT #BIT6, @TCSR ;TEST BIT6 OF TCSR
1182 004714 001401 BEQ 5$ ;BR IF CLEAR
1183
1184 004716 104025 ERROR 25 ;CANNOT CLEAR BIT6 OF TCSR WITH RESET
1185
1186 004720 010377 175436 5$: MOV R3, @TVECTOR ;RESTORE XMIT VECTOR

```

TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET

```

1197
1198
1199
1190
1191
1192 004724 000004
1193 004726 000005
1194 004730 017703 175422
1195 004734 012777 004764 175414
1196 004742 004767 005274
1197 004746 000340
1198 004750 032777 000100 175370
1199 004756 001404
1200 004760 104026
1201
1202 004762 000402
1203
1204 004764 022626 1$: CMP (SP)+,(SP)+
1205 004766 104027 ERROR 27
1206
1207
1208 004770 052777 000100 175350 2$: BIS #BIT6,@RCSR
1209 004776 032777 000100 175342 BIT #BIT6,@RCSR
1210 005004 001001 BNE 3$
1211
1212 005006 104030 ERROR 30
1213
1214
1215 005010 042777 000100 175330 3$: BIC #BIT6,@RCSR
1216 005016 032777 000100 175322 BIT #BIT6,@RCSR
1217 005024 001401 BEQ 4$
1218
1219 005026 104031 ERROR 31
1220
1221
1222 005030 052777 000100 175310 4$: BIS #BIT6,@RCSR
1223 005036 000005 RESET
1224 005040 032777 000100 175300 BIT #BIT6,@RCSR
1225 005046 001401 BEQ 5$
1226
1227 005050 104032 ERROR 32
1228
1229 005052 010377 175300 5$: MOV R3,@RVECT
1230

```

```

*****
*TEST 12 TEST THAT BIT6 OF RCSR CAN BE SET & PESET
*****
TST12: SCOPE
RESET ;CLEAR EVERYTHING
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV #15,@RVECT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
JSR PC,@RPSW ;SET PSW TO PRIORITY=7
.WORD 340
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR
BEQ 2$
ERROR 26
;BIT6 OF RCSR NOT CLEAR AFTER RESET
BR 2$
1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 27 ;RCVR INTERRUPT WITH PRIORITY=7
2$: BIS #BIT6,@RCSR ;SET BIT6 OF RCSR
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR
BNE 3$ ;BR. IF SET
ERROR 30 ;CANNOT SET BIT6 OF RCSR
3$: BIC #BIT6,@RCSR ;CLEAR BIT6 OF RCSR
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR
BEQ 4$ ;BR. IF CLEAR
ERROR 31 ;CANNOT CLEAR BIT6 OF RCSR
4$: BIS #BIT6,@RCSR ;SET BIT6 OF RCSR
RESET ;CLEAR BIT6 OF RCSR WITH RESET
BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR
BEQ 5$ ;BR. IF CLEAR
ERROR 32 ;CANNOT CLEAR BIT6 OF RCSR WITH RESET
5$: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

```

```

1231
1232 005056 012767 000012 173716 TCLOCK: MOV #12,STSTNM ;ADJUST TEST NUMBER TO (NEXT TEST - 1)
1233 ;*****
1234 ;*TEST 13 TEST ABILITY TO ACCESS LKS
1235 ;*****
1236 005064 000004 †ST13: SCOPE
1237 005066 005767 175244 TST CTSTFL ;IS CONSOLE UNDER TEST?
1238 005072 001420 BEQ TST14 ;IF NOT, SKIP THIS TEST
1239 005074 032777 000100 173736 BIT #BIT6, QSWR ;ARE LINE CLOCK TESTS INHIBITED?
1240 005102 001014 BNE TST14 ;IF YES, SKIP THIS TEST
1241 005104 013703 000004 MOV Q#4, R3 ;SAVE TIMEOUT VECTOR
1242 005110 012737 005124 000004 MOV #1$, Q#4 ;SET UP TIMEOUT VECTOR
1243 005116 005777 175264 TST QLKS ;ACCESS LKS
1244 005122 000402 BR 2$ ;NO TIMEOUT - BR TO END OF TEST
1245
1246 005124 022626 1$: CMP (SP)+, (SP)+ ;RESTORE SP AFTER TIMEOUT
1247 005126 104010 ERROR 10 ;CAN NOT ACCESS LKS
1248
1249 005130 010337 000004 2$: MOV R3, Q#4 ;RESTORE TIMEOUT VECTOR
1250
1251 ;*****
1252 ;*TEST 14 TEST THAT BIT6 OF LKS CAN BE SET & RESET
1253 ;*****
1254 005134 000004 †ST14: SCOPE
1255 005136 005767 175174 TST CTSTFL ;IS CONSOLE UNDER TEST?
1256 005142 001460 BEQ TST15 ;IF NOT, SKIP THIS TEST
1257 005144 032777 000100 173666 BIT #BIT6, QSWR ;ARE LINE CLOCK TESTS INHIBITED?
1258 005152 001054 BNE TST15 ;IF YES, SKIP THIS TEST
1259 005154 000005 RESET
1260 005156 017703 175226 MOV QRTCVT, R3 ;SAVE LINE CLOCK VECTOR
1261 005162 012777 005212 175220 MOV #1$, QRTCVT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
1262 005170 004767 005046 JSR PC, WRPSW ;SET PSW TO PRIORITY 7
1263 005174 000340 .WORD 340
1264 005176 032777 000100 175202 BIT #BIT6, QLKS ;TEST BIT6 OF LKS
1265 005204 001404 BEQ 2$
1266 005206 104033 ERROR 33
1267 ;BIT6 OF LKS NOT CLEAR AFTER RESET
1268 005210 000402 BR 2$
1269
1270 005212 022626 1$: CMP (SP)+, (SP)+ ;RESTORE SP AFTER INTERRUPT
1271 005214 104034 ERROR 34
1272 ;LKS INTERRUPT WITH PRIORITY=7
1273
1274 005216 052777 000100 175162 2$: BIS #BIT6, QLKS ;SET BIT6 OF LKS
1275 005224 032777 000100 175154 BIT #BIT6, QLKS ;TEST BIT6 OF LKS
1276 005232 001001 BNE 3$ ;BR IF SET
1277
1278 005234 104035 ERROR 35
1279 ;CANNOT SET BIT6 OF LKS
1280
1281 005236 042777 000100 175142 3$: BIC #BIT6, QLKS ;CLEAR BIT6 OF LKS
1282 005244 032777 000100 175134 BIT #BIT6, QLKS ;TEST BIT6 OF LK
1283 005252 001401 BEQ 4$
1284 005254 104036 ERROR 36
1285 ;CANNOT CLEAR BIT6 OF LKS
1286 005256 052777 000100 175122 4$: BIS #BIT6, QLKS ;SET BIT6 OF LKS
  
```

N03

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 28
DZDLDB.P11 14-JUN-77 09:27 T14

TEST THAT BIT6 OF LKS CAN BE SET & RESET

1287	005264	000005			RESET			;CLEAR BIT6 OF LKS WITH RESET
1288	005266	032777	000100	175112	BIT	#BIT6,2LKS		;TEST BIT6 OF LKS
1289	005274	001401			BEQ	55		;BR IF CLEAR
1290								
1291	005276	104037			ERROR	37		
1292								;CANNOT CLEAR BIT6 OF LKS WITH RESET
1293	005300	010377	175104	55:	MOV	R3,2RTCVT		;RESTORE LINE CLOCK VECTOR

```

1294
1295
1296
1297
1298
1299 005304 000004
1300 005306 013703 000004
1301 005312 013704 000006
1302 005316 012737 005450 000004
1303 005324 012737 000340 000006
1304 005332 000005
1305 005334 012700 000002
1306 005340 032777 000020 173472
1307 005346 001404
1308 005350 016767 175032 173442
1309 005356 000403
1310 005360 016767 174762 173432 1$: MOV RCSR,$GDADR ;MOVE GOOD RCSR ADDRESS INTO $GDADR
1311 005366 016767 173426 173426 2$: MOV $GDADR,$BDADR ;MOVE GOOD ADDRESS INTO TEST ADDRESS LOCATION
1312 005374 040067 173422
1313 005400 026767 173414 173414
1314 005406 001002
1315 005410 050067 173406
1316 005414 017767 173402 173402 3$: MOV 2$BDADR,$DDAT ;SAVE CONTENTS OF BAD ADDRESS IF IT EXISTS
1317 005422 052777 000100 173372
1318 005430 032777 000100 173362
1319 005436 001011
1320 005440 016777 173360 173354
1321 005446 000401
1322 005450 022626 4$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
1323 005452 006300 5$: ASL R0 ;SHIFT BIT MASK TO NEXT POSITION
1324 005454 005700
1325 005456 001343
1326 005460 000401
1327
1328 005462 104040 6$: ERROR 40 ;DUAL ADDRESSING ERROR
1329
1330
1331
1332 005464 010337 000004 7$: MOV R3,2$4 ;RESTORE TIMEOUT VECTOR
1333 005470 010437 000006 MOV R4,2$6 ;RESTORE TIMEOUT PSW

```



```

1334
1335
1336
1337
1338
1339 005474 000004
1340 005476 005767 174634
1341 005502 001437
1342 005504 032777 000100 173326
1343 005512 001033
1344 005514 000005
1345 005516 105777 174664
1346 005522 100401
1347
1348 005524 104041
1349
1350 005526 042777 000200 174652
1351 005534 032777 000200 174644
1352 005542 001410
1353 005544 042777 000200 174634
1354 005552 032777 000200 174626
1355 005560 001401
1356
1357 005562 104042
1358
1359 005564 005000
1360 005566 105777 174614
1361 005572 100403
1362 005574 005200
1363 005576 001373
1364
1365 005600 104043
1366

```

```

:*****
:*TEST 16 TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED
:*****
TST16: SCOPE
TST CTSTFL :IS CONSOLE UNDER TEST?
BEQ TST17 :IF NOT, SKIP THIS TEST
BIT #BIT6,#SWR :ARE LINE CLOCK TESTS INHIBITED?
BNE TST17 :IF YES, SKIP THIS TEST
RESET :CLEAR EVERYTHING & SET BIT7 OF LKS
15: TSTB #LKS :TEST FOR BIT7 OF LKS
BMI 25 :BR IF SET
ERROR 41 ;BIT7 OF LKS DID NOT SET WITH RESET
25: BIC #BIT7,#LKS :CLEAR BIT7 OF LKS
BIT #BIT7,#LKS :TEST BIT7 OF LKS
BEQ 35
BIC #BIT7,#LKS :TRY ONE MORE TIME BECAUSE THE CLOCK
BIT #BIT7,#LKS : MAY HAVE SET IMMEDIATELY AFTER THE FIRST CLEAR
BEQ 35
ERROR 42 ;CAN NOT CLEAR BIT7 OF LKS
35: CLR R0 :CLEAR TIMER
CONT: TSTB #LKS :TEST FOR BIT7 OF LKS
BMI TST17 :BR, IF SET
INC R0 :INCREMENT TIMER
BNE CONT :CONTINUE UNTIL TIME EXPIRES
ERROR 43 ;BIT7 OF LKS DOES NOT SET

```

TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED

```

1367
1368
1369
1370
1371 005602 000004
1372 005604 005767 174526
1373 005610 001503
1374 005612 032777 000100 173220
1375 005620 001077
1376 005622 004767 004414
1377 005626 000340
1378 005630 017703 174554
1379 005634 017704 174552
1380 005640 012777 005702 174542
1381 005646 012777 000340 174536
1382 005654 042777 000200 174524
1383 005662 052777 000100 174516
1384 005670 105777 174512
1385 005674 100375
1386 005676 000240
1387 005700 000402
1388
1389 005702 022626
1390 005704 104044
1391
1392 005706 005077 174474
1393 005712 012777 005740 174470
1394 005720 004767 004316
1395 005724 000240
1396 005726 105777 174454
1397 005732 100375
1398 005734 000240
1399 005736 000402
1400
1401 005740 022626
1402 005742 104045
1403
1404 005744 012777 006000 174436
1405 005752 042777 000200 174426
1406 005760 052777 000100 174420
1407 005766 105777 174414
1408 005772 100375
1409 005774 000240
1410
1411 005776 104046
1412
1413 006000 022626
1414 006002 042777 000100 174376
1415 006010 010377 174374
1416 006014 010477 174372
1417
1418

```

```

*****
*TEST 17 TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY
*****
TST17: SCOPE
TST CTSTFL :IS CONSOLE UNDER TEST?
BEQ TST20 :IF NOT, SKIP THIS TEST
BIT #BIT6,2SWR :ARE LINE CLOCK TESTS INHIBITED?
BNE TST20 :IF YES, SKIP THIS TEST
JSR PC,WRPSW :SET PSW TO PRIORITY 7
      .WORD 340
MOV @RTCVT,R3 :SAVE LINE CLOCK VECTOR
MOV @RTCP SW,R4 :SAVE LINE CLOCK PSW VECTOR
MOV #25,@RTCVT :SET RTC INTERRUPT VECTOR TO ERROR REPORT
MOV #340,@RTCP SW :KEEP PRIORITY AT 7
BIC #BIT7,@LKS :CLEAR CLOCK DONE FLAG
BIS #BIT6,@LKS :SET INTERRUPT ENABLE
1$: TSTB @LKS :WAIT FOR RTC DONE (INTERRUPT REQUEST)
    BPL 1$
    NOP
    BR 3$ :GIVE TIME FOR ANY INTERRUPTS
    :BR, IF NO INTERRUPT OCCURS
2$: CMP (SP)+,(SP)+ :RESTORE SP AFTER INTERRUPT
    ERROR 44 :RTC INTERRUPTS AT PRIORITY 7
3$: CLR @LKS :DISABLE RTC INTERRUPTS & CLEAR DONE
    MOV #45,@RTCVT :SET RTC INTERRUPT VECTOR FOR ERROR
    JSR PC,WRPSW :CHANGE PSW TO PRIORITY 5
      .WORD 240
20$: TSTB @LKS :WAIT FOR DONE (INTERRUPT REQUEST)
     BPL 20$
     NOP
     BR 5$ :GIVE TIME FOR ANY INTERRUPT
     :IF NO INTERRUPT - BR TO CONTINUE TEST
4$: CMP (SP)+,(SP)+ :RESTORE SP AFTER INTERRUPT
    ERROR 45 :RTC INTERRUPTS WITH INTERRUPTS DISABLED
5$: MOV #75,@RTCVT :POINT RTC VECTOR TO END OF TEST
    BIC #BIT7,@LKS :CLEAR CLOCK DONE FLAG
    BIS #BIT6,@LKS :ALLOW INTERRUPTS
6$: TSTB @LKS :WAIT FOR RTC DONE
     BPL 6$
     NOP
     :GIVE TIME FOR INTERRUPT
ERROR 46 :RTC INTERRUPT DID NOT OCCUR
7$: CMP (SP)+,(SP)+ :RESTORE SP AFTER INTERRUPT
    BIC #BIT6,@LKS :DISABLE INTERRUPTS
    MOV R3,@RTCVT :RESTORE LINE CLOCK VECTOR
    MOV R4,@RTCP SW :RESTORE LINE CLOCK PSW VECTOR

```

TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPEPLY

```

1419
1420
1421
1422
1423
1424
1425 006020 000004
1426 006022 005767 174310
1427 006026 001457
1428 006030 032777 000100 173002
1429 006036 001053
1430 006040 000005
1431 006042 017703 174342
1432 006046 017704 174340
1433 006052 012777 006122 174330
1434 006060 012777 000340 174324
1435 006066 004767 004150
1436 006072 000240
1437 006074 042777 000200 174304
1438 006102 052777 000100 174276
1439 006110 105777 174272 1S:
1440 006114 100375
1441 006116 000240
1442
1443 006120 104047 ERROR 47 ;RTC INTERRUPT DID NOT OCCUR
1444
1445 006122 022626 2S:
1446 006124 012777 006144 174256 CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1447 006132 004767 004104 MOV #3S,RTCVT ;POINT RTC VECTOR TO ERROR REPORT
1448 006136 000240 JSR PC,WRPSW ;SET PSW TO PRIORITY 5
1449 006140 000240 .WORD 240
1450 006142 000402 NOP ;GIVE SOME TIME FOR AN INTERRUPT
1451 BR 4S ;NO INTERRUPT - BR TO END OF TEST
1452 006144 022626 3S:
1453 006146 104050 CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1454 ERROR 50 ;INTERRUPT SEQUENCE DID NOT CLEAR
1455 ;INTERRUPT REQUEST
1456 006150 042777 000100 174230 4S:
1457 006156 010377 174226 BIC #BIT6,CLKS ;DISABLE CLOCK INTERRUPTS
1458 006162 010477 174224 MOV R3,RTCVT ;RESTORE LINE CLOCK VECTOR
MOV R4,RTCPSW ;RESTORE LINE CLOCK PSW VECTOR

```

```

1459
1460
1461
1462
1463
1464 006166 000004
1465 006170 005767 174142
1466 006174 001442
1467 006176 032777 000100 172634
1468 006204 001036
1469 006206 004767 004030
1470 006212 000340
1471 006214 017703 174170
1472 006220 012777 006272 174162
1473 006226 042777 000200 174152
1474 006234 052777 000100 174144
1475 006242 105777 174140 1S:
1476 006246 100375
1477 006250 000005
1478 006252 004767 003764
1479 006256 000240
1480 006260 000240
1481 006262 042777 000100 174116
1482 006270 000402
1483
1484 006272 022626 2S:
1485 006274 104051
1486
1487
1488 006276 010377 174106 3S:

```

```

*****
*TEST 21 TEST THAT RTC INTERRUPT CLEARS WITH RESET
*****
↑ST21: SCOPE
TST CTSTFL ;IS CONSOLE UNDER TEST?
BEQ TST22 ;IF NOT, SKIP THIS TEST
BIT #BIT6, QSWR ;ARE LINE CLOCK TESTS INHIBITED?
BNE TST22 ;IF YES, SKIP THIS TEST
JSR PC, WRPSW ;SET PRIORITY TO 7
.WORD 340
MOV @RTCVT, R3 ;SAVE LINE CLOCK VECTOR
MOV #25, @RTCVT ;POINT RTC VECTOR TO ERROR REPORT
BIC #BIT7, QLKS ;CLEAR CLOCK DONE FLAG
BIS #BIT6, QLKS ;ENABLE CLOCK INTERRUPTS
1S: TSTB QLKS ;WAIT FOR DONE (INTERRUPT REQUEST)
BPL 1S
RESET ;CLEAR PENDING INTERRUPT WITH RESET
JSR PC, WRPSW ;SET PRIORITY TO 5
.WORD 240
NOP ;GIVE TIME FOR ANY INTERRUPT
BIC #BIT6, QLKS ;DISALLOW INTERRUPTS
BR 3S ;BR TO END OF TEST

2S: JMP (SP)+, (SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR S1 ;RESET DID NOT CLEAR INTERRUPT

3S: MOV R3, @RTCVT ;RESTORE LINE CLOCK VECTOR

```

TEST THAT RTC INTERRUPT CLEARS WITH RESET

1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

006302 000004
006304 005767 174026
006310 001447
006312 032777 000100 172520
006320 001043
006322 004767 003714
006326 000340
006330 017703 174054
006334 012777 006412 174046
006342 042777 000200 174036
006350 052777 000100 174030
006356 105777 174024
006362 100375
006364 042777 000200 174014
006372 004767 003644
006376 000240
006400 000240
006402 042777 000100 173776
006410 000402

006412 022626
006414 104052

006416 010377 173766
006418 004767 003614
006420 000340

```
*****  
*TEST 22 TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS  
*****  
TST22: SCOPE  
TST CTSTFL ;IS CONSOLE UNDER TEST?  
BEQ TST23 ;IF NOT, SKIP THIS TEST  
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?  
BNE TST23 ;IF YES, SKIP THIS TEST  
JSR PC,WPSW ;SET PRIORITY TO 7  
 .WORD 340  
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR  
MOV #25,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT  
BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG  
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS  
15: TSTB @LKS ;WAIT FOR DONE (INTERRUPT REG'EST)  
BPL 15  
BIC #BIT7,@LKS ;CLEAR DONE & INTERRUPT  
JSR PC,WPSW ;ALLOW INTERRUPTS  
 .WORD 240  
NOP ;GIVE TIME FOR ANY INTERRUPT  
BIC #BIT6,@LKS ;DISALLOW INTERRUPTS  
BR 35 ;BR TO END OF TEST  
  
25: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
ERROR 52 ;CLEARING BIT7 OF LKS DID NOT CLEAR INTERRUPT  
  
35: MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR  
JSR PC,WPSW ;RESTORE PRIORITY TO 7  
 .WORD 340
```

TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS

```

1520
1521
1522
1523
1524
1525 006430 000004
1526 006432 005767 173730
1527 006436 001462
1528 006440 032777 000100 172372
1529 006446 001056
1530 006450 042777 000100 173730
1531
1532 006456 005000
1533 006460 012701 177777
1534 006464 005002 1$: CLR R2
1535 006466 005077 173714 2$: CLR R2
1536 006472 105777 173710 3$: CLR R2
1537 006476 100375 4$: CLR R2
1538 006500 005077 173702 5$: CLR R2
1539 006504 105777 173676 6$: CLR R2
1540 006510 100003
1541 006512 005202
1542 006514 005077 173666 7$: CLR R2
1543 006520 005200 8$: INC R2
1544 006522 001370 9$: INC R2
1545 006524 005201 10$: INC R2
1546 006526 001003 11$: INC R2
1547 006530 010267 000044 12$: INC R2
1548 006534 000753 13$: INC R2
1549 006536 016701 000036 14$: INC R2
1550 006542 160201 15$: INC R2
1551 006544 100001 16$: INC R2
1552 006546 005401 17$: INC R2
1553 006550 020127 000001 18$: INC R2
1554 006554 003403 19$: INC R2
1555
1556 006556 010267 000020 20$: INC R2
1557 006552 104053 21$: INC R2
1558
1559 006564 032777 000020 172246 22$: INC R2
1560 006572 001404 23$: INC R2
1561 006574 000167 003302 24$: INC R2
1562
1563 006600 000000 25$: INC R2
1564 006602 000000 26$: INC R2

```

*TEST 23 TEST CLOCK REPEATABILITY

```

↑ST23: SCOPE
TST CTSTFL ; IS CONSOLE UNDER TEST?
BEQ TST24 ; IF NOT, SKIP THIS TEST
BIT #BIT6, QSWR ; ARE LINE CLOCK TESTS INHIBITED?
BNE TST24 ; IF YES, SKIP THIS TEST
BIC #BIT6, QLKS ; DISALLOW INTERRUPTS

CLR R0 ; CLEAR A TIMER
MOV #1, R1 ; SET A FLAG INDICATING FIRST PASS THRU THIS LOOP
1$: CLR R2 ; CLEAR CLOCK COUNTER
CLR R2 ; CLEAR DONE
TSB QLKS ; SYNC ON DONE
2$: BPL R2
CLR R2 ; CLEAR DONE
3$: TSTB QLKS ; IS CLOCK DONE?
BR IF NOT, TO INCREMENT TIMER
; IF DONE, INCREMENT CLOCK COUNT
4$: CLR R2 ; CLEAR DONE
INC R0 ; INCREMENT TIMER
BNE Q3 ; BR IF TIME REMAINS
INC R1 ; INCREMENT LOOP PASS FLAG
BNE COMPARE ; BR IF TWO PASSES HAVE BEEN MADE
MOV R2, FIRST ; IF NOT, STORE FIRST CLOCK COUNT
BR 1$ ; DO LOOP AGAIN
COMPARE: MOV FIRST, R1 ; RECALL FIRST CLOCK COUNT
SUB R2, R1 ; CALCULATE DIFFERENCE OF TWO COUNTS
BPL TOLER ; IF POSITIVE, SKIP NEGATION OF DIFFERENCE
NEG R1 ; MAKE DIFFERENCE A POSITIVE NUMBER
TOLER: CMP R1, #1 ; COMPARE DIFFERENCE WITH DESIRED TOLERANCE
BLE S5 ; BR, IF LOWER/EQUAL TO TOLERANCE

MOV R2, SECOND ; STORE SECOND COUNT
ERROR S3 ; CLOCK REPEATABILITY ERROR

5$: BIT #BIT4, QSWR ; CLOCK TESTS ONLY?
BEQ TST24 ; BR IF NOT
JMP $EOP ; ELSE, JUMP TO END OF PASS ROUTINE

FIRST: .WORD 0
SECOND: .WORD 0

```

```

1565
1566
1567
1568
1569
1570 006604 000004
1571 006606 042777 000100 173536
1572 006614 017703 173542
1573 006620 012777 006644 173534
1574 006626 105777 173520 1$:
1575 006632 100375
1576 006634 004767 003402
1577 006640 000140
1578 006642 000402
1579
1580 006644 022626 2$:
1581 006646 104054
1582
1583 006650 012777 006670 173504 3$:
1584 006656 052777 000100 173466
1585 006664 000240
1586
1587 006666 104055 ERROR 55 ;XMIT DID NOT INTERRUPT
1588
1589 006670 042777 000100 173454 4$:
1590 006676 022626
1591 006700 010377 173456

```

```

*****
*TEST 24 TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
*****
TST24: SCOPE
BIC #BIT6,@TCR ;CLEAR TRANSMIT INTERRUPT ENABLE
MOV @TVECT,R3 ;SAVE XMIT VECTOR
MOV #25,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT
1$: TSTB @TCR ;WAIT FOR DONE
BPL 1$
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
BR 3$

2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 54

3$: MOV #45,@TVECT ;XMIT INTERRUPTS WITH INTERRUPT ENABLE CLEAR
BIS #BIT6,@TCR ;SET XMIT VECTOR TO END OF TEST
NOP ;ENABLE INTERRUPTS

4$: BIC #BIT6,@TCR ;DISABLE INTERRUPTS
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
MOV R3,@TVECT ;RESTORE XMIT VECTOR

```


TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED

```

1593
1594
1595      ;*****
1596      ;*TEST 25      TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
1597      ;*****
1597      006704 000004      †ST25: SCOPE
1598      006706 042777 000100 173436      BIC      #BIT6,@TCSR      ;DISABLE INTERRUPTS
1599      006714 004767 003322      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 7
1600      006720 000340      .WORD    340
1601      006722 017703 173434      MOV      @TVECT,R3      ;SAVE XMIT VECTOR
1602      006726 012777 006754 173426      MOV      #25,@TVECT      ;POINT XMIT VECTOR TO ERROR REPORT
1603      006734 105777 173412      1$:     TSTB     @TCSR      ;WAIT FOR DONE
1604      006740 100375      BPL      1$
1605      006742 052777 000100 173432      BIS      #BIT6,@TCSR      ;ENABLE INTERRUPT
1606      006750 000240      NOP
1607      006752 000402      BR      3$      ;CONTINUE TEST
1608
1609      006754 022626      2$:     CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
1610      006756 104056      ERROR   56
1611
1612      006760 042777 000100 173364      3$:     BIC      #BIT6,@TCSR      ;XMIT INTERRUPTS AT PRIORITY=7
1613      006766 012777 007036 173366      MOV      #45,@TVECT      ;CLEAR INTERRUPT ENABLE
1614      006774 004767 003242      JSR      PC,WRPSW      ;POINT XMIT VECTOR TO ERROR REPORT
1615      007000 000140      .WORD    140      ;SET PSW TO PRIORITY 3
1616      007002 000240      NOP
1617      007004 000402      BR      5$      ;BR TO END OF TEST-NO INTERRUPT
1618
1619      007006 022626      4$:     CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
1620      007010 104057      ERROR   57
1621
1622      007012 010377 173344      5$:     MOV      R3,@TVECT      ;XMIT INTERRUPT OCCURS WITH BIT6 CLEAR
1623      ;RESTORE XMIT VECTOR

```

TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED

```

1623
1624
1625
1626
1627
1628 007016 000004
1629 007020 042777 000100 173324
1630 007026 017703 173330
1631 007032 017704 173326
1632 007036 012777 007100 173316
1633 007044 012777 000340 173312
1634 007052 004767 003164
1635 007056 000140
1636 007060 105777 173266 1$: TSTB 2TCSR ;WAIT FOR DONE
1637 007064 100375
1638 007066 052777 000100 173256
1639 007074 000240
1640
1641 007076 104060
1642
1643 007100 022626
1644 007102 012777 007130 173252 2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1645 007110 004767 003126
1646 007114 000140
1647 007116 000240
1648 007120 042777 000100 173224
1649 007126 000402
1650
1651 007130 022626
1652 007132 104061 4$: CMP ERROR (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1653
1654 007134 010377 173222
1655 007140 010477 173220 5$: MOV R3,2TVECT ;XMIT RE-INTERRUPTED
;RESTORE XMIT VECTOR
MOV R4,2TPSW ;RESTORE XMIT PSW VECTOR
1656
1657
1658
1659
1660 007144 000004
1661 007146 042777 000100 173176
1662 007154 004767 003062
1663 007160 000340
1664 007162 017703 173174
1665 007166 012777 007240 173166
1666 007174 052777 000100 173150
1667 007202 005077 173146
1668 007206 105777 173140
1669 007212 100375 1$: TSTB 2TCSR ;WAIT FOR DONE (INTERRUPT)
1670 007214 005077 173134
1671 007220 004767 003016
1672 007224 000140
1673 007226 000240
1674 007230 042777 000100 173114
1675 007236 000402
1676
1677 007240 022626 2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1678 007242 104062

```

L04

PAGE: 0050

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 39
020LDB.P11 14-JUN-77 09:27 T27 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF

1679
1680 007244 010377 173112 38: MOV R3, @TVECT :LOADING TBUF DID NOT CLEAR INTEPRUPT.
1681 :RESTORE XMIT VECTOR

```

1682
1683
1684
1685
1686 007250 000004
1687 007252 000005
1688 007254 052777 000004 173070
1689 007262 005000
1690 007264 005077 173064
1691 007270 032777 004000 173050 WACTV:
1692 007276 001006
1693 007300 005200
1694 007302 001372
1695 007304 042777 000004 173040
1696
1697 007312 104063 ERROR 63 ;RCVR ACTIVE DID NOT SET WHILE RECEIVING
1698
1699 007314 000005 2$: RESET
1700 007316 032777 004000 173022 BIT #BIT11, @RCSR ;VERIFY "INIT" CLEARS RCV ACTIVE
1701 007324 001401 BEQ 3$
1702
1703 007326 104115 ERROR 115 ;INIT DID NOT CLEAR RCV ACTIVE
1704
1705 007330 005000 3$: CLR RO ;CLEAR A TIMER
1706 007332 052777 000004 173012 BIS #BIT2, @TCSR ;SET MAINTENANCE WRAP
1707 007340 062700 000000 WT: ADD #0, RO ;WAIT AT LEAST ONE BIT TIME
1708 007344 005200 INC RO
1709 007346 001374 BNE WT
1710 007350 036777 174424 172770 BIT BIT11, @RCSR ;VERIFY RCV ACTIVE STILL CLEAR
1711 007356 001404 BEQ 4$ ;BR IF CLEAR
1712
1713 007360 042777 000004 172764 BIC #BIT2, @TCSR ;CLEAR MAINTENANCE WRAP
1714 007366 104116 ERROR 116 ;RCV ACTIVE WITHOUT "START" BIT
1715
1716 007370 005000 4$: CLR RO ;CLEAR TIMER
1717 007372 005077 172756 CLR @TBUF ;LOAD TRANSMIT BUFFER
1718 007376 105777 172744 WDONE: TSTB @RCSR ;CHECK FOR RECEIVER DONE
1719 007402 100406 BMI 5$ ;BR, IF DONE
1720 007404 005200 INC RO ;INCREMENT TIMER, IF NOT DONE
1721 007406 001373 BNE WDONE ;CONTINUE WAIT IF TIME REMAINS
1722 007410 042777 000004 172734 EOC #BIT2, @TCSR ;CLEAR MAINTENANCE BIT
1723 007416 104064 ERROR 64 ;RECEIVER DONE NEVER SET
1724
1725
1726 007420 032777 004000 172720 5$: BIT #BIT11, @RCSR ;CHECK FOR RCVR ACTIVE CLEAR
1727 007426 001404 BEQ 6$ ;BR, IF CLEAR
1728 007430 042777 000004 172714 BIC #BIT2, @TCSR ;CLEAR MAINTENANCE BIT
1729 007436 104065 ERROR 65 ;RCVR ACTIVE DID NOT CLEAR WITH RCVR DONE
1730
1731
1732 007440 000005 6$: RESET ;CLEAR DONE WITH RESET
1733 007442 105777 172700 TSTB @RCSR ;CHECK FOR DONE CLEAR
1734 007446 001404 BEQ 7$
1735
1736 007450 042777 000004 172674 BIC #BIT2, @TCSR ;CLEAR MAINTENANCE BIT
1737 007456 104066 ERROR 66

```

N04

PAGE: 0052

M
02

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 41
DZDLDB.P11 14-JUN-77 09:27 T30 TEST THAT RCVR ACTIVE (11) & DONE (7) SET & CLEAR PROPERLY

1738 ;RESET DID NOT CLEAR RCVR DONE
1739
1740 007460 042777 000004 172660 7\$: BIC #BIT2,RCVR ;CLEAR MAINTENANCE BIT
1741 007466 000400 BR TST31 ;BR TO NEXT TEST

1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782

007470 000004
007472 000005
007474 052777 000001 172644
007502 052777 000004 172642
007510 005077 172640
007514 105777 172626
007520 100375
007522 032777 000001 172616
007530 001401

007532 104117

007534 052777 000001 172604
007542 105777 172600
007546 001404
007550 042777 000004 172574
007556 104067

007560 000004
007562 000005
007564 052777 000004 172560
007572 005077 172556
007576 105777 172544
007602 100375
007604 017700 172540
007610 042777 000004 172534
007616 105777 172524
007622 001401
007624 104070

```
::*****  
:*TEST 31 TEST THAT RDR ENABLE CLEARS RECEIVER DONE FLAG  
:*****  
↑ST31: SCOPE  
RESET ;CLEAR EVERYTHING  
BIS #BIT0, @RCSR ;SET RDR ENABLE  
BIS #BIT2, @TCSR ;SET MAINTENANCE WRAP  
CLR @TBUF ;LOAD TRANSMITTER  
1$: TSTB @RCSR ;WAIT FOR RECEIVER DONE  
BPL 1$  
BIT #BIT0, @RCSR ;VERIFY RCVR ACTIVE-CLEARED RDR ENABLE  
BEQ 2$ ;BR IF CLEAR  
  
ERROR 117 ;RDR ENABLE NOT CLEARED WITH RCVR ACTIVE  
  
2$: BIS #BIT0, @RCSR ;CLEAR DONE BY SETTING RDR ENABLE  
TSTB @RCSR ;CHECK FOR DONE CLEAR  
BEQ TST32 ;BR IF CLEAR TO NEXT TEST  
BIC #BIT2, @TCSR ;CLEAR MAINTENANCE BIT  
ERROR 67 ;SETTING RDR ENABLE DID NOT CLEAR RCVR DONE
```

```
::*****  
:*TEST 32 TEST THAT READING RBUF CLEARS RECEIVER DONE  
:*****  
↑ST32: SCOPE  
RESET ;CLEAR EVERYTHING  
BIS #BIT2, @TCSR ;SET MAINTENANCE WRAP  
CLR @TBUF ;LOAD TRANSMITTER  
1$: TSTB @RCSR ;WAIT FOR RECEIVER DONE  
BPL 1$  
MOV @RBUF, R0 ;READ RECEIVE BUFFER  
BIC #BIT2, @TCSR ;CLEAR MAINTENANCE BIT  
TSTB @RCSR ;CHECK FOR RECEIVE DONE CLEAR  
BEQ TST33 ;BR IF CLEAR TO NEXT TEST  
ERROR 70 ;READING RBUF DID NOT CLEAR RCVR DONE
```

```

1783
1784
1785
1786
1787
1788 007626 000004
1789 007630 042777 00010C 172514
1790 007636 042777 000100 172502
1791 007644 052777 000004 172500
1792 007652 017703 172500
1793 007656 012777 007714 172472
1794 007664 004767 002352
1795 007670 000140
1796 007672 005077 172456
1797 007676 105777 172444 1S:
1798 007702 100375
1799 007704 042777 000004 172440
1800 007712 000405
1801
1802 007714 042777 000004 172430 2S:
1803 007722 022626
1804 007724 104071
1805
1806
1807 007726 012777 007754 172422 3S:
1808 007734 052777 00010C 172404
1809 007742 000240
1810 007744 042777 000004 172400
1811 007752 104072
1812
1813
1814 007754 042777 00010C 172364 4S:
1815 007762 042777 000004 172362
1816 007770 022626
1817 007772 010377 172360

```

```

*****
*TEST 33 TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
*****
TST33: SCOPE
      BIC      #BIT6, @TCSR      ;DISABLE TRANSMIT INTERRUPTS
      BIC      #BIT6, @RCR      ;DISABLE RECEIVER INTERRUPTS
      BIS      #BIT2, @TCSR      ;SET MAINTENANCE WRAP
      MOV      @RVECT, R3        ;SAVE RECEIVE VECTOR
      MOV      #25, @RVECT      ;POINT RCV VECTOR TO ERROR REPORT
      JSR      PC, WRPSW        ;SET PSW TO PRIORITY 3
      .WORD    140
      CLR      @TBUF            ;SEND A CHARACTER
      TSTB     @RCR            ;WAIT FOR RECEIVER DONE
      BPL      1S
      BIC      #BIT2, @TCSR      ;CLEAR MAINTENANCE BIT
      BR       3S              ;CONTINUE TEST
      BIC      #BIT2, @TCSR      ;CLEAR MAINTENANCE BIT
      CMP      (SP)+, (SP)+      ;RESTORE SP AFTER INTERRUPT
      ERROR    71
      ;RECEIVER INTERRUPTS WITH INT. ENABLE CLEAR
      MOV      #45, @RVECT      ;POINT RCV VECTOR TO END OF TEST
      BIS      #BIT6, @RCR      ;ENABLE RCV INTERRUPTS
      NOP
      BIC      #BIT2, @TCSR      ;GIVE ANY INTERRUPTS TIME
      ERROR    72              ;CLEAR MAINTENANCE BIT
      ;RCVR DID NOT INTERRUPT
      BIC      #BIT6, @RCR      ;DISABLE INTERRUPTS
      BIC      #BIT2, @TCSR      ;CLEAR MAINTENANCE BIT
      CMP      (SP)+, (SP)+      ;RESTORE SP AFTER INTERRUPT
      MOV      R3, @RVECT      ;RESTORE RECEIVE VECTOR

```

TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED

```

1818
1819
1820
1821
1822
1823 007776 000004
1824 010000 000005
1825 010002 004767 002234
1826 010006 000340
1827 010010 017703 172342
1828 010014 012777 010054 172334
1829 010022 052777 000004 172322
1830 010030 005077 172320
1831 010034 105777 172306 1$: TSTB
1832 010040 100375 BPL 1$
1833 010042 052777 000100 172276 BIS #BIT6,@RCR
1834 010050 000240 NOP
1835 010052 000405 BR 3$
1836 010054 042777 000004 172270 2$: BIC #BIT2,@TCR
1837 010062 022626 CMP (SP)+,(SP)+
1838 010064 104073 ERROR 73
1839
1840
1841 010066 042777 000100 172252 3$: BIC #BIT6,@RCR
1842 010074 012777 010122 172254 MOV #4$,@RVECT
1843 010102 004767 002134 JSR PC,WRPSW
1844 010106 000140 .WORD 140
1845 010110 000240 NOP
1846 010112 042777 000004 172232 BIC #BIT2,@TCR
1847 010120 000405 BR 5$
1848
1849 010122 042777 000004 172222 4$: BIC #BIT2,@TCR
1850 010130 022626 CMP (SP)+,(SP)+
1851 010132 104074 ERROR 74
1852
1853 010134 010377 172216 5$: MOV R3,@RVECT

```

*TEST 34 TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED

↑ST34: SCOPE
RESET ;CLEAR EVERYTHING
JSR PC,WRPSW ;SET PSW TO PRIORITY 7
.WORD 340
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV #2\$,@RVECT ;POINT RCVR VECTOR TO ERROR REPORT
BIS #BIT2,@TCR ;SET MAINTENANCE WRAP
CLR @TBUF ;SEND A CHARACTER
1\$: TSTB @RCR ;WAIT FOR RECEIVER DONE
BPL 1\$
BIS #BIT6,@RCR ;ENABLE INTERRUPTS
NOP ;GIVE TIME FOR INTERRUPT
BR 3\$;CONTINUE TEST
2\$: BIC #BIT2,@TCR ;CLEAR MAINTENANCE BIT
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 73
;RCVR INTERRUPTS AT PRIORITY 7
3\$: BIC #BIT6,@RCR ;CLEAR INTERRUPT ENABLE
MOV #4\$,@RVECT ;POINT RCVR VECTOR TO ERROR REPORT
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
;GIVE TIME FOR ANY INTERRUPT
;CLEAR MAINTENANCE BIT
BR 5\$;BR TO END OF TEST, IF NO INTERRUPT
4\$: BIC #BIT2,@TCR ;CLEAR MAINTENANCE BIT
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 74
;RCVR INTERRUPT REQUEST PASSED WITH BITS CLEAR
5\$: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED

```

1854
1855
1856
1857
1858
1859 010140 000004
1860 010142 000005
1861 010144 017703 172206
1862 010150 017704 172204
1863 010154 012777 010236 172174
1864 010162 012777 000340 172170
1865 010170 004767 002046
1866 010174 000140
1867 010176 052777 000004 172146
1868 010204 005077 172144
1869 010210 105777 172132
1870 010214 100375
1871 010216 042777 000004 172126
1872 010224 052777 000100 172114
1873 010232 000240
1874
1875 010234 104075
1876
1877
1878 010236 022626
1879 010240 012777 010276 172110
1880 010246 004767 001770
1881 010252 000140
1882 010254 000240
1883 010256 042777 000100 172062
1884 010264 010377 172066
1885 010270 010477 172064
1886 010274 000402
1887
1888 010276 022626
1889 010300 104076
1890
1891 010302 010377 172050

```

```

*****
*TEST 35 TEST RECEIVER FOR DOUBLE INTERRUPTS
*****
†ST35: SCOPE
RESET ;CLEAR EVERYTHING
MOV R3,ARVECT ;SAVE RECEIVE VECTOR
MOV R4,ARPSW ;SAVE RECEIVE PSW VECTOR
MOV R2,ARVECT ;POINT RCV VECTOR TO CONTINUE TEST
MOV R3,ARPSW ;SET PRIORITY TO 7 AFTER INTERRUPT
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
BIS #BIT2,ATCSR ;SET MAINTENANCE WRAP
CLR ATBUF ;SEND A CHARACTER
1$: TSTB ARCSR ;WAIT FOR RCVR DONE
BPL 1$
BIC #BIT2,ATCSR ;CLEAR MAINTENANCE BIT
BIS #BIT6,ARCSR ;ENABLE RCV INTERRUPTS
NOP ;GIVE SOME TIME
ERROR 75 ;RCVR INTERRUPT DID NOT OCCUR
2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
MOV R3,ARVECT ;POINT RCV VECTOR TO ERROR REPORT
JSR PC,WRPSW ;RESET PSW TO PRIORITY 3
.WORD #140
NOP ;GIVE SOME TIME
BIC #BIT6,ARCSR ;CLEAR INTERRUPT ENABLE
MOV R3,ARVECT ;RESTORE RECEIVE VECTOR
MOV R4,ARPSW ;RESTORE RECEIVE PSW VECTOR
BR 4$ ;BR TO END OF TEST
3$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 76
;RECEIVER RE-INTERRUPTED
4$: MOV R3,ARVECT ;RESTORE RECEIVE VECTOR

```

```

1892
1893
1894
1895
1896
1897 010306 000004
1898 010310 000005
1899 010312 004767 001724
1900 010316 000340
1901 010320 017703 172032
1902 010324 012777 010412 172024
1903 010332 052777 000100 172006
1904 010340 052777 000004 172004
1905 010346 005077 172002
1906 010352 105777 171770 1$:
1907 010356 100375
1908 010360 042777 000004 171764
1909 010366 005077 171756
1910 010372 004767 001644
1911 010376 000140
1912 010400 000240
1913 010402 042777 000100 171736
1914 010410 000402
1915
1916 010412 022626 2$:
1917 010414 104077
1918
1919 010416 010377 171734 3$:
1920
    
```

```

:*****
:*TEST 36 TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF
:*****
1$T36: SCOPE
      RESET          :CLEAR EVERYTHING
      JSR            :SET PSW PRIORITY TO 7
      PC,WRPSW
      .WORD 340
      @RVECT,R3     :SAVE RECEIVE VECTOR
      #2$,@RVECT    :POINT RCVR VECTOR TO ERROR REPORT
      #BIT6,@RCSR   :SET RCVR INTERRUPT ENABLE
      #BIT2,@TCSR   :SET MAINTENANCE WRAP
      CLR @TBUF     :SEND A CHARACTER
      TSTB @RCSR    :WAIT FOR DONE (INTERRUPT)
      BPL 1$
      BIC #BIT2,@TCSR :CLEAR MAINTENANCE BIT
      CLR @RBUF     :READ RBUF TO CLEAR PENDING INTERRUPT
      JSR PC,WRPSW  :SET PSW TO PRIORITY 3
      .WORD 140
      NOP
      BIC #BIT6,@RCSR :NO INTERRUPT-CLEAR INT. ENABLE
      BR 3$
2$:  CMP (SP)+,(SP)+ :RESTORE SP AFTER INTERRUPT
      ERROR 77       :READING RBUF DID NOT CLEAR INTERRUPT
3$:  MOV R3,@RVECT   :RESTORE RECEIVE VECTOR
    
```

TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF

```

1921
1922
1923
1924
1925
1926 010422 000004
1927 010424 000005
1928 010426 004767 001610
1929 010432 000340
1930 010434 017703 171716
1931 010440 012777 010520 171710
1932 010446 052777 000100 171672
1933 010454 052777 000004 171670
1934 010462 012777 000377 171664
1935 010470 105777 171652 15:
1936 010474 100375
1937 010476 000005
1938 010500 004767 001536
1939 010504 000140
1940 010506 000240
1941 010510 042777 000100 171630
1942 010516 000402
1943
1944
1945 010520 022626 25:
1946 010522 104100 ERROR 100
1947
1948 010524 010377 171626 35:

```

```

*****
*TEST 37 TEST THAT RESET CLEARS RECEIVE INTERRUPT
*****
TST37: SCOPE
RESET ;CLEAR EVERYTHING
JSR PC,WRPSW ;SET PSW TO PRIORITY 7
.WORD 340
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV #25,@RVECT ;POINT RCV VECTOR TO ERROR REPORT
BIS #BIT6,@RCSR ;SET RCV INTERRUPT ENABLE
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
MOV #377,@TBUF ;SEND AN ALL 1'S CHARACTER
TSTB @RCSR ;WAIT FOR RCV DONE
BPL 15
RESET ;CLEAR RCV INTERRUPT & RBUF
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
NOP ;ALLOW TIME FOR AN ERRONEOUS INTERRUPT
BIC #BIT6,@RCSR ;NO INTERRUPT-CLEAR INT. ENABLE
BR 35 ;CONTINUE TEST

25: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 100 ;RESET DID NOT CLEAR RCVR INTERRUPT

35: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

```

TEST THAT RESET CLEARS RECEIVE INTERRUPT

1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975

010530 000004
010532 032777 002000 170300
010540 001432
010542 000005
010544 052777 000004 171600
010552 012700 000003
010556 005077 171572
010562 105777 171564
010566 100375
010570 005300
010572 001371
010574 042777 000004 171550
010602 032777 040000 171540
010610 001001
010612 104101

010614 032777 100000 171526 3\$:
010622 001001
010624 104102

010626 4\$:

```
*****  
: *TEST 40 TEST THAT THE "OR" ERROR (BIT14) & "ERROR" (BIT15) CAN BE SET  
*****  
↑ST40: SCOPE  
BIT #BIT10,@SWR ;IS THIS TEST ENABLED  
BEQ TST41 ;IF NOT ENABLED, BR TO NEXT TEST  
RESET ;CLEAR EVERYTHING  
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP  
MOV #3,R0 ;SET CHARACTER COUNT TO SEND 3 CHAR.  
1$: CLR @TBUF ;LOAD TRANSMIT BUFFER  
2$: TSTB @TCSR ;WAIT FOR TRANSMIT DONE  
BPL 2$  
DEC R0 ;DECREMENT CHARACTER COUNT  
BNE 1$ ;BR IF ALL CHARACTERS NOT TRANSMITTED  
BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT  
BIT #BIT14,@RBUF ;TEST FOR "OR" ERROR FLAG  
BNE 3$ ;BR, IF SET  
ERROR 101 ;"OR" ERROR FLAG DID NOT SET  
  
3$: BIT #BIT15,@RBUF ;TEST "ERROR" FLAG  
BNE 4$ ;BR, IF SET  
ERROR 102 ;"ERROR" FLAG DID NOT SET WITH "OR" FLAG  
  
4$:
```

```

1976
1977
1978
1979
1980
1981 010626 000004
1982 010630 032777 000400 170202
1983 010636 001444
1984 010640 000005
1985 010642 052777 000004 171502
1986 010650 012777 177777 171475
1987 010656 105777 171464 1$: TSTB 2RC5R ;WAIT FOR RCVR DONE
1988 010662 100375
1989 010664 005077 171460
1990 010670 052777 000001 171454
1991 010676 005000
1992 010700 117767 171442 170120 2$: MOVB 2RC5R,$BDDAT ;WAIT FOR RCVR DONE
1993 010706 100406
1994 010710 005200
1995 010712 001372
1996
1997 010714 042777 000305 171430
1998 010722 104103
1999
2000 010724 105777 171420
2001 010730 001404
2002 010732 042777 000005 171412
2003
2004 010740 104103
2005
2006 010742 042777 000005 171402 3$: BIC #BIT0!BIT2,2TCSR ;CLEAR MAINTENANCE & BREAK BITS

```

```

:*****
:*TEST 41 TEST THAT BREAK TRANSMITS ALL ZEROES
:*****
↑TST41: SCOPE
BIT #BIT8,2SWR ;IS BREAK FUNCTION TEST ENABLED?
BEQ TST42 ;BR TO NEXT TEST, IF NOT ENABLED
RESET ;CLEAR EVERYTHING
BIS #BIT2,2TCSR ;SET MAINTENANCE WRAP
MOV #-1,2RBUF ;TRANSMIT ALL ONES TO RCVR
1$: TSTB 2RC5R ;WAIT FOR RCVR DONE
BPL 1$
CLR 2RBUF ;CLEAR DONE (LEAVING ALL ONES IN RBUF)
BIS #BIT0,2TCSR ;TRANSMIT BREAK
CLR RO ;CLEAR A TIMER
2$: MOVB 2RC5R,$BDDAT ;WAIT FOR RCVR DONE
BMI CONT41 ;BR IF DONE
INC RO ;IF NOT, INCREMENT TIMER
BNE 2$ ;BR IF TIME REMAINS

BIC #BIT0!BIT2,2TCSR ;CLEAR MAINTENANCE & BREAK BITS
ERROR 103 ;BREAK DID NOT TRANSMIT ANYTHING

CONT41: TSTB 2RBUF ;CHECK RECEIVE BUFFER FOR ZERO
BEQ 3$ ;BR IF ZERO
BIC #BIT0!BIT2,2TCSR ;CLEAR MAINTENANCE & BREAK BITS

ERROR 103 ;BREAK DID NOT TRANSMIT ALL ZEROES

3$: BIC #BIT0!BIT2,2TCSR ;CLEAR MAINTENANCE & BREAK BITS

```

2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034

010750 000004
010752 032777 002000 170060
010760 001435
010762 032777 000400 170050
010770 001431
010772 000005
010774 052777 000004 171350
011002 052777 000001 171342
011010 005077 171340
011014 105777 171326
011020 100375
011022 042777 000005 171322
011030 032777 020000 171312
011036 001001
011040 104104
011042 032777 100000 171300
011050 001001
011052 104114
011054

```
*****
: *TEST 42 TEST THAT "FR" ERROR CAN BE SET DURING BREAK
: *****
TST42: SCOPE
BIT #BIT10,@SWR :IS THE "TEST ERROR FLAGS" BIT SET
BEQ TST43 :BR TO NEXT TEST, IF NOT SET
BIT #BIT8,@SWR :IS BREAK FUNCTION ENABLED
BEQ TST43 :BR TO NEXT TEST, IF NOT SET
RESET :CLEAR EVERYTHING
BIS #BIT2,@TCSR :SET MAINTENANCE WRAP
BIS #BIT0,@TCSR :SEND BREAK
CLR @TBUF :TRANSMIT A CHARACTER TO TIME BREAK
1$: TSTB @RCSR :WAIT FOR RCVR DONE
BPL 1$
BIC #BIT0!BIT2,@TCSR :CLEAR MAINTENANCE & BREAK BITS
BIT #BIT13,@RBUF :CHECK FOR FRAMING ERROR FLAG
BNE 2$ :BR, IF SET

ERROR 104
:BREAK DID NOT SET FRAMING ERROR
2$: BIT #BIT15,@RBUF :TEST "ERROR" FLAG
BNE 3$ :BR, IF SET

ERROR 114
:"ERROR" FLAG DID NOT SET WITH "CR" FLAG
3$:
```

MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 51
J2DLDB.P11 14-JUN-77 09:27 T42

TEST THAT "FR" ERROR CAN BE SET DURING BREAK

2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090

011054 000004
011056 000005
011060 005001
011062 052777 000004 171262
011070 005201
011072 010177 171256
011076 105777 171244
011102 100375
011104 017702 171240
011110 043701 001112
011114 020102
011116 001003
011120 105701
011122 001411
011124 000761
011126 010167 167672
011132 010267 167670
011136 042777 000004 171206
011144 104105
011146 042777 000004 171176

```
::*****  
:*TEST 43 TEST DATA PATH FROM TRANSMITTER TO RECEIVER USING MAINTENANCE WRAP  
:*****  
TST43: SCOPE  
RESET ;CLEAR EVERYTHING  
CLR R1 ;CLEAR REGISTER FOR TEST DATA  
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP  
;TRANSMIT A BINARY COUNT PATTERN - JP  
;TO THE BIT POSITION INDICATED BY THE  
;CONTENTS OF LOCATION "$USWR"  
1$: INC R1 ;INCREMENT THE TEST DATA  
MOV R1,@TBUF ;XMIT A CHARACTER  
2$: TSTB @RCSR ;WAIT FOR RECEIVER DONE  
BPL 2$  
MOV @RBUF,R2 ;GET RECEIVED CHARACTER  
BIC @#$USWR,R1 ;CLEAR LOWEST UNUSED DATA BIT POSITION IN TEST DATA  
CMP R1,R2 ;COMPARE DATA  
BNE 3$ ;BR, IF NON-COMPARE  
TSTB R1 ;TEST XMIT DATA FOR ZERO  
BEQ 4$ ;BR, IF FINISHED  
BR 1$ ;CONTINUE IF NOT  
3$: MOV R1,$GDDAT ;STORE THE EXPECTED DATA  
MOV R2,$BDDAT ;STORE RECEIVED DATA  
BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT  
ERROR 105 ;DATA COMPARE DATA  
4$: BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT  
:  
:  
:*****  
:*TEST 44 TEST DATA PATHS USING WRAP CABLE  
:*****  
TST44: SCOPE  
BIT #BIT7,@SWR ;IS THIS TEST ENABLED  
BEQ TST45 ;BR, IF NOT  
RESET ;CLEAR EVERYTHING  
CLR R1 ;CLEAR REGISTER FOR TEST DATA  
;TRANSMIT A BINARY COUNT PATTERN - JP  
;TO THE BIT POSITION INDICATED BY THE  
;CONTENTS OF LOCATION "$USWR"  
1$: INCB R1 ;INCREMENT THE TEST DATA  
MOV R1,@TBUF ;XMIT A CHARACTER  
CLR R0 ;CLEAR A TIMER  
2$: TSTB @RCSR ;WAIT FOR RECEIVER DONE  
BMT 3$ ;BR IF DONE  
INC R0 ;INCREMENT TIMER IF NOT  
BNE 2$ ;BR IF TIME REMAINS  
ERROR 64 ;RECEIVER DONE NOT SET  
3$: MOV @RBUF,R2 ;GET RECEIVED CHARACTER  
BIC @#$USWR,R1 ;CLEAR LOWEST UNUSED DATA BIT POSITION IN TEST DATA
```

MAIN. MACY11 30,1046) 14-JUN-77 17:10 PAGE 52
02DLDB.P11 14-JUN-77 09:27 T44

TEST DATA PATHS USING WRAP CABLE

2091	011226	020102			CMP	R1,R2	:COMPARE DATA
2092	011230	001003			BNE	4\$:BR, IF NON-COMPARE
2093	011232	105701			TSTB	R1	:TEST XMIT DATA FOR ZERO
2094	011234	001406			BEQ	TST45	:BR, IF FINISHED
2095	011236	000755			BR	1\$:CONTINUE IF NOT
2096	011240	010167	167560		MOV	R1,\$GDDAT	:STORE EXPECTED DATA
2097	011244	010267	167556	4\$:	MOV	R2,\$BDDAT	:STORE RECEIVED DATA
2098							
2099	011250	104106			ERROR	106	:DATA COMPARE ERROR WITH WRAP CABLE

2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155

011252 000004
011254 000005
011256 004767 000760
011262 000340
011264 017703 171072
011270 017704 171062
011274 017705 171110
011300 017767 171060 000426
011306 017767 171046 000422
011314 017767 171072 000416
011322 012777 011650 171032
011330 005077 171030
011334 012777 011704 171014
011342 005077 171012
011346 005767 170764
011352 001414
011354 032777 000100 167456
011362 001010
011364 012777 011720 171016
011372 005077 171014
011376 052777 000100 171002
011404 052777 000004 170740
011412 052777 000100 170732
011420 052777 000100 170720
011426 005067 000276
011432 005067 000270
011436 005001
011440 005000
011442 012702 011742
011446 005077 170702
011452 004767 000564
011456 000140
011460 062700 000000
011464 062700 000001
011470 001373
011472 032777 000100 170652
011500 001402
011502 000005
011504 104107
011506 026767 000216 000212
011514 001402
011516 000005
011520 104110

```
*****  
; *TEST 45 TEST DL11-W LOGIC BY EXERCISING THE XMIT, RECEIVE, & CLOCK (IF AVAILABLE)  
*****  
†ST45: SCOPE  
RESET ; CLEAR EVERYTHING  
JSR PC, WRPSW ; SET PRIORITY TO 7  
.WORD 340  
MOV @TVECT, R3 ; SAVE XMIT VECTOR  
MOV @RVECT, R4 ; SAVE RECEIVE VECTOR  
MOV @RTCVT, R5 ; SAVE CLOCK VECTOR  
MOV @TPSW, STPSW ; SAVE XMIT PSW VECTOR  
MOV @RPSW, SRPSW ; SAVE RECEIVE PSW VECTOR  
MOV @RTCP$W, SCPSW ; SAVE CLOCK PSW VECTOR  
MOV #XMIT, @TVECT ; POINT TRANSMIT VECTOR TO TRANSMIT ROUTINE  
CLR @TPSW ; ALLOW INTERRUPTS AFTER XMIT INTERRUPT  
MOV #RCV, @RVECT ; POINT RECEIVE VECTOR TO RECEIVE ROUTINE  
CLR @RPSW ; ALLOW INTERRUPTS AFTER RCVR INTERRUPT  
TST CTSTFL ; IS CONSOLE UNDER TEST?  
BEQ 1$ ; IF NOT SKIP CLOCK SET UP  
BIT #BIT6, @SWR ; IF YES, ARE CLOCK TEST DISABLED?  
BNE 1$ ; IF YES, SKIP CLOCK SET UP  
MOV #CLK, @RTCVT ; POINT VECTOR TO CLOCK INTERRUPT ROUTINE  
CLR @RTCP$W ; ALLOW INTERRUPTS AFTER CLOCK INTERRUPT  
BIS #BIT6, @LKS ; ENABLE CLOCK INTERRUPTS  
BIS #BIT2, @TCSR ; SET MAINTENANCE WRAP  
BIS #BIT6, @TCSR ; ENABLE TRANSMIT INTERRUPTS  
BIS #BIT6, @RCSR ; ENABLE RECEIVE INTERRUPTS  
CLR XMTCNT ; CLEAR XMIT INTERRUPT COUNTER  
CLR RCVCNT ; CLEAR RCV INTERRUPT COUNTER  
CLR R1 ; CLEAR A REGISTER FOR TEST DATA USE  
CLR R0 ; CLEAR TIMER  
MOV #BUF, R2 ; POINT R2 TO RECEIVE DATA STORAGE  
CLR @TBUF ; SEND FIRST CHARACTER  
JSR PC, WRPSW ; SET PSW TO PRIORITY 3  
.WORD 140  
; WAIT FOR INTERRUPTS  
2$: ADD #0, R0 ; ADD INSTRUCTIONS ARE USED TO LENGTHEN LOOP TIME  
ADD #1, R0 ; TO COVER THE SLOWEST BAUD RATE ON THE FASTEST CPU  
BNE 2$  
BIT #BIT6, @TCSR ; FINISHED ENTIRE TRANSMISSION  
BEQ 3$ ; BR, IF INTERRUPTS ARE DISABLED (FINISHED)  
RESET ; CLEAR EVERYTHING  
ERROR 107 ; TRANSMIT INTERRUPT TIMEOUT IN MAIN. DATA TEST  
3$: CMP XMTCNT, RCVCNT ; COMPARE THE NUMBER OF INTERRUPTS  
BEQ 4$ ; BR, IF EQUAL  
RESET ; CLEAR EVERYTHING  
ERROR 110 ; RECEIVER DID NOT GET FULL TRANSMISSION  
; IF RCVCNT=0, NO DATA RECEIVED  
; IF RCVCNT=?, THEN (XMTCNT-RCVCNT,  
; EQUALS THE NO. OF INTERRUPTS LOST.
```

N05

2156									
2157									
2158	011522	005767	170610	4\$:	TST	CTSTFL			: IS CONSOLE UNDER TEST?
2159	011526	001410			BEQ	5\$: IF NOT, SKIP CLOCK COUNT CHECK
2160	011530	032777	000100 167302		BIT	#BIT6, @SWR			: IF YES, ARE CLOCK TESTS DISABLED?
2161	011536	001004			BNE	5\$: IF YES, SKIP CLOCK COUNT CHECK
2162	011540	005767	000166		TST	CLKCNT			: CHECK FOR AT LEAST ONE CLOCK INTERRUPT
2163	011544	001001			BNE	5\$: BR IF INTERRUPTS OCCURRED
2164									
2165	011546	104113			ERROR	113			: NO CLOCK INTERRUPTS IN EXERCISER
2166									
2167	011550	000005		5\$:	RESET				: CLEAR EVERYTHING
2168	011552	012700	011742		MOV	#BUF, R0			: LOAD RECEIVED DATA POINTER TO R0
2169	011556	005001			CLR	R1			: SET UP REGISTER FOR COMPARISSN
2170	011560	022001		COMP:	CMP	(R0)+, R1			: COMPARE XMIT & RCV DATA
2171	011562	001005			BNE	6\$: BR, IF NOT EQUAL
2172	011564	105201			INCB	R1			: INCREMENT COMPARE DATA
2173	011566	032701	000040		BIT	#BIT5, R1			: FINISHED CHECKING RECEIVED DATA?
2174	011572	001772			BEQ	COMP			: BR, IF NOT FINISHED
2175	011574	000405			BR	7\$: BR TO END OF TEST
2176									
2177	011576	014067	167224	6\$:	MOV	-(R0), \$BDDAT			: STORE BAD DATA FOR ERROR REPORT
2178	011602	010167	167216		MOV	R1, \$GDDAT			: STORE GOOD DATA FOR ERROR REPORT
2179	011606	104111			ERROR	111			: DATA COMPARE ERROR IN EXERCISER
2180									
2181	011610	010377	170546	7\$:	MOV	R3, @TVECT			: RESTORE XMIT VECTOR
2182	011614	010477	170536		MOV	R4, @RVECT			: RESTORE RECEIVE VECTOR
2183	011620	010577	170564		MOV	R5, @RTCVT			: RESTORE CLOCK VECTOR
2184	011624	016777	000104 170532		MOV	STPSW, @TPSW			: RESTORE XMIT PSW VECTOR
2185	011632	016777	000100 170520		MOV	SRPSW, @RPSW			: RESTORE RECEIVE PSW VECTOR
2186	011640	016777	000074 170544		MOV	SCPSW, @RTCPSW			: RESTORE CLOCK PSW VECTOR
2187	011646	000501			BR	ENDEV			: BR TO END OF DEVICE TEST ROUTINE
2188									
2189	011650	005267	000054	XMIT:	INC	XMTCNT			: INCREMENT XMIT INTERRUPT COUNTER
2190	011654	105201			INCB	R1			: INCREMENT TEST DATA
2191	011656	032701	000040		BIT	#BIT5, R1			: SEND DATA PATTERN FROM 00 --> 37
2192	011662	001404			BEQ	XCONT			: BR, IF MORE DATA TO BE SENT
2193	011664	042777	000100 170460		BIC	#BIT6, @TCSR			: CLEAR XMIT INTERRUPT ENABLE
2194	011672	000402			BR	XRET			: RETURN, WITHOUT SENDING ANY MORE DATA
2195	011674	110177	170454	XCONT:	MOVB	R1, @TBUF			: SEND NEW CHARACTER
2196	011700	005000		XRET:	CLR	R0			: CLEAR TIMER
2197	011702	000002			RTI				: RETURN
2198									
2199	011704	017722	170440	RCV:	MOV	@RBUF, (R2)+			: STORE RECEIVED DATA
2200	011710	005267	000012		INC	RCVCNT			: INCREMENT RCV INTERRUPT COUNTER
2201	011714	005000			CLR	R0			: CLEAR TIMER
2202	011716	000002			RTI				: RETURN
2203									
2204	011720	005267	000006	CLK:	INC	CLKCNT			: INCREMENT CLOCK INTERRUPT COUNT
2205	011724	000002			RTI				: RETURN
2206									
2207	011726	000000		RCVCNT:	.WORD	0			
2208	011730	000000		XMTCNT:	.WORD	0			
2209	011732	000000		CLKCNT:	.WORD	0			
2210	011734	000000		STPSW:	.WORD	0			
2211	011736	000000		SRPSW:	.WORD	0			

806

MACY11 30(1046) 14-JUN-77 17:10 PAGE 55
14-JUN-77 09:27

T45 TEST DL11-W LOGIC BY EXERCISING THE XMIT, RECEIVE, & CLOCK IF AVAILABLE, SIMULTAN PAGE: 0066

011740 000000
011742 000044

SCPSW: .WORD 0
BUF: .BLKW 44

:END OF DEVICE PASS ROUTINE

012052	005067	166724		ENDEV:	CLR	\$TSTNM	:CLEAR TEST NO. COUNT FOR SCOPE ROUTINE
012056	005267	167014			INC	\$DEVCT	:INCREMENT DEVICE COUNTER
012062	026767	170254	167006		CMP	TMP2,\$DEVCT	:ALL DEVICES TESTED
012070	001404				BEG	\$EOP	:BR. IF YES
012072	005067	170240			CLR	CTSTFL	:CLEAR CONSOLE UNDER TEST FLAG
012076	000167	171376			JMP	TSTDEV	:GO TEST NEXT DEVICE

.SBTTL END OF PASS ROUTINE

```

:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO GOAGIN

```

012102	000004			\$EOP:	SCOPE		
012102	005067	166672			CLR	\$TSTNM	::ZERO THE TEST NUMBER
012104	005267	166760			INC	\$PASS	::INCREMENT THE PASS NUMBER
012110	042767	100000	166752		BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
012114	005327				DEC	(PC)+	::LOOP?
012122	000001			\$EOPCT:	.WORD	1	
012124	003015				BGT	\$DOAGN	::YES
012126	012737				MOV	(PC)+,2(PC)+	::RESTORE COUNTER
012130	000001			\$ENDCT:	.WORD	1	
012132	012124				\$EOPCT		
012134	104401	012172			TYPE	ENDMG	::TYPE "END PASS"
012136	013700	000042		\$GET42:	MOV	#42,R0	::GET MONITOR ADDRESS
012142	001405				BEG	\$DOAGN	::BRANCH IF NO MONITOR
012146	000005				RESET		::CLEAR THE WORLD
012150	004710			\$ENDAD:	JSR	PC,(R0)	::GO TO MONITOR
012152	000240				NOP		::SAVE ROOM
012154	000240				NOP		::FOR
012156	000240				NOP		::ACT11
012160	000137			\$DOAGN:	JMP	2(PC)+	::RETURN
012162	012206			\$RTNAD:	.WORD	GOAGIN	
012164	377	377	000	\$ENULL:	.BYTE	-1,-1,0	::NULL CHARACTER STRING
012166					.EVEN		
012172	047105	020104		ENDMG:	.ASCIIZ	OF LEF. END PASS	

MACY11 30.1046) 14-JUN-77 17:10 PAGE 57
 C23L08.P11 14-JUN-77 09:27 END OF PASS ROUTINE

PAGE: 0068

```

2260
2261 012206 005067 166664      GOAGIN: CLR      $DEVCT      :CLEAR DEVICE COUNT
2262 012212 022767 000001 170122  CMP      #1,TMP2      :IS THERE ONLY ONE DEVICE UNDER TEST?
2263 012220 001004                BNE      RSTRT        :BR, IF NOT
2264 012222 012706 001000                MOV      #1000,SP     :RESET STACK PCINTER
2265 012226 000167 171370                JMP      TST1         :GO DO ANOTHER PASS
2266
2267 012232 005067 166642      RSTRT: CLR      $UNIT      :CLEAR UNIT NUMBER
2268 012236 000167 171152                JMP      BEGIN
2269
2270 012242 011646                WRPSW: MOV(SP),-(SP)    :COPY RETURN PC
2271 012244 013616                MOV      2(SP)+,(SP)  :MOVE NEW PSW TO STACK
2272 012246 062746 000002                ADD      #2,-(SP)     :ADJUST JSR RETURN
2273 012252 000002                RTI                    :POP RETURN PC & NEW PSW
2274
2275                :SUBROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
2276
2277 012254 012600                CATCH: MOV      (SP)+,R0 :GET ADDRESS OF TRAP VECTOR + 4
2278 012256 162700 000004                SUB      #4,R0        :ADJUST TO POINT TO TRAP ADDRESS
2279 012262 010067 000014                MOV      R0,$DEVCT    :STORE TRAP OR INTERRUPT ADDRESS
2280 012266 016667 000002 000004                MOV      2(SP),OLDPC  :GET PC WHERE TRAP OR INTERRUPT OCCURRED
2281 012274 104112                ERROR  112           :REPORT ERROR
2282
2283 012276 000000                HALT                    :PROGRAM MUST BE RESTARTED AT THIS POINT
2284 012300 000000                OLDPC: .WORD 0
2285 012302 000000                $DEVCT: .WORD 0
2286

```

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND ADDRESS OF THE ERROR CALL
*AND GO TO SERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW09=1      LOOP IN ERROR
*CALL
*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
*****

```

```

$ERROR:
7$:      INCB      $ERFLG      ;SET THE ERROR FLAG
          BEQ      7$          ;DON'T LET FLAG GO TO ZERO
          MOV      $STNM,$DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
          INC      $ERTL      ;INCREMENT ERROR COUNT
          MOV      (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
          SUB      #2,$ERRPC
          MOVB    @($ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
          BIT      #BIT13,$SWR   ;SKIP TYPEOUT IF SET
          BNE     20$          ;SKIP TYPEOUTS
          JSR     PC,$SERRTYP    ;GO TO USER ERROR ROUTINE
          TYPE    ,SCLF

20$:
          CMPB    #APTENV,$ENV   ;RUNNING IN APT MODE
          BNE     2$            ;NO, SKIP APT ERROR REPORT
          MOVE    $ITEMB,21$     ;SET ITEM NUMBER AS ERROR NUMBER
          JSR     PC,$ATY4       ;REPORT FATAL ERROR TO APT

21$:      .BYTE    0
          .BYTE    0

22$:      BR      22$          ;APT ERROR LOOP
23$:      TST     @SWR          ;HALT ON ERROR
          BPL     3$            ;SKIP IF CONTINUE
          HALT    ON ERROR!     ;HALT ON ERROR!
          CKSWR   ;TEST FOR CHANGE IN SOFT-SWR
          BIT     #BIT09,$SWR    ;LOOP ON ERROR SWITCH SET?
          BEQ     4$            ;BR IF NO
          MOV     $LPERR,(SP)    ;FUDGE RETURN FOR LOOPING
          TST     $ESCAPE        ;CHECK FOR AN ESCAPE ADDRESS
          BEQ     5$            ;BR IF NONE
          MOV     $ESCAPE,(SP)   ;FUDGE RETURN ADDRESS FOR ESCAPE

5$:
          CMP     #SENDAD,$#42   ;ACT-11 AUTO-ACCEPT?
          BNE     6$            ;BR IF NO
          HALT

6$:      RTI                    ;RETURN

```

```

012304
012304 105267 166473
012310 001775
012312 016777 166464 166522
012320 005267 166466
012324 011667 166466
012330 162767 000002 166460
012336 117767 166454 166450
012344 032777 020000 166466
012352 001004
012354 004767 000106
012360 104401 001063
012364
012364 122767 000001 166514
012372 001007
012374 116767 166414 000004
012402 004767 000562
012406 000
012407 000
012410 000777
012412 005777 166422
012416 100001
012420 000000
012422 104406
012424 032777 001000 166406
012432 001402
012434 016716 166350
012440 005767 166414
012444 001402
012446 016716 166406
012452
012452 022737 012152 000042
012460 001001
012462 000000
012464
012464 000002

```

.SBTTL ERROR MESSAGE TIMEOUT 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      $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
        MOV      R0, -(SP)    ;; SAVE R0
        CLR      R0          ;; PICKUP THE ITEM INDEX
        BISB     @#$ITEMB,R0
        SNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                          ;; TYPE THE PC OF THE ERROR
        MOV      $ERRPC, -(SP) ;; SAVE $ERRPC FOR TIMEOUT
                          ;; ERROR ADDRESS
        TYFPC     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        BR      6$          ;; GET OUT
1$:     DEC      R0          ;; ADJUST THE INDEX SO THAT IT WILL
        ASL     R0          ;; WORK FOR THE ERROR TABLE
        ASL     R0
        ASL     R0
        ADD     @#$ERRTB,R0  ;; FORM TABLE POINTER
        MOV     (R0)+, 2$    ;; PICKUP "ERROR MESSAGE" POINTER
        BEQ    3$          ;; SKIP TIMEOUT IF NO POINTER
        TYPE   0            ;; TYPE THE "ERROR MESSAGE"
2$:     .WORD   0           ;; "ERROR MESSAGE" POINTER GOES HERE
        TYPE   $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
3$:     MOV     (R0)+, 4$    ;; PICKUP "DATA HEADER" POINTER
        BEQ    5$          ;; SKIP TIMEOUT IF 0
        TYPE   0            ;; TYPE THE "DATA HEADER"
4$:     .WORD   0           ;; "DATA HEADER" POINTER GOES HERE
        TYPE   $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
5$:     MOV     (R0), R0     ;; PICKUP "DATA TABLE" POINTER
        BNE   7$          ;; GO TYPE THE DATA
6$:     MOV     (SP)+, R0    ;; RESTORE R0
        TYPE   $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
        RTS     PC         ;; RETURN
7$:     MOV     @ (R0)+, -(SP) ;; SAVE @ (R0)+ FOR TIMEOUT
        TYFPC     ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
        TST     (R0)        ;; IS THERE ANOTHER NUMBER?
        BEQ    6$          ;; BR IF NO
        TYPE   2$          ;; TYPE TWO(2) SPACES
        BR     7$          ;; LOOP
8$:     .ASCII  "          " ;; TWO(2) SPACES
        .EVEN

```

```

2337
2338
2339
2340
2341
2342
2343
2344
2345 012466
2346 012466 104401 001063
2347 012472 010046
2348 012474 005000
2349 012476 153700 001014
2350 012502 001004
2351
2352 012504 016746 156306
2353
2354 012510 104402
2355 012512 000426
2356 012514 005300
2357 012516 006300
2358 012520 006300
2359 012522 006300
2360 012524 062700 001146
2361 012530 012067 000004
2362 012534 001404
2363 012536 104401
2364 012540 000000
2365 012542 104401 001063
2366 012546 012067 000004
2367 012552 001404
2368 012554 104401
2369 012556 000000
2370 012560 104401 001063
2371 012564 011000
2372 012566 001004
2373 012570 012600
2374 012572 104401 001063
2375 012576 000207
2376 012600
2377 012600 013046
2378 012602 104402
2379 012604 005710
2380 012606 001770
2381 012610 104401 012616
2382 012614 000771
2383 012616 020040 000
2384 012622

```

.SBTTL POWER DOWN AND UP ROUTINES

::*****

;*POWER DOWN ROUTINE

::*****

012622 012737 012762 000024
012630 012737 000340 000026
012636 010046
012640 010146
012642 010246
012644 010346
012646 010446
012650 010546
012652 017746 166162
012656 010667 000104
012662 012737 012574 000024
012670 000000
012672 000776

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

::*****

;*POWER UP ROUTINE

::*****

012674 012737 012762 000024
012702 016706 000060
012706 012677 166126
012712 012605
012714 012604
012716 012603
012720 012602
012722 012601
012724 012600
012726 012737 012622 000024
012734 012737 000340 000026
012742 005067 000020
012746 005267 000014
012752 001375
012754 104401
012756 012770
012760 000002
012762 000000
012764 000776
012766 000000
012770 005016 047520 042527
012772 000122

```
$PWRUP: MOV $SILLUP,@PWRVEC ;SET FOR FAST DOWN
MOV $$SAVR6,SP ;GET SP
MOV (SP)+,@SWR ;POP STACK INTO @SWR
MOV (SP)+,R5
MOV (SP)+,R4 ;POP STACK INTO R4
MOV (SP)+,R3 ;POP STACK INTO R3
MOV (SP)+,R2 ;POP STACK INTO R2
MOV (SP)+,R1 ;POP STACK INTO R1
MOV (SP)+,R0 ;POP STACK INTO R0
MOV $PWRDN,@PWRVEC ;SET UP THE POWER DOWN VECTOR
MOV #340,@PWRVEC+2 ;PRIO:7
CLR $$SAVR6 ;WAIT LOOP FOR THE TTY
IS: INC $$SAVR6 ;WAIT FOR THE INC
BNE IS ;OF WORD
TYPE ;REPORT THE POWER FAILURE
$PWRMG: .WORD $POWER ;POWER FAIL MESSAGE POINTER
RTI
$SILLUP: HALT ;THE POWER UP SEQUENCE WAS STARTED
BR -2 ;BEFORE THE POWER DOWN WAS COMPLETE
$$SAVR6: 0 ;PUT THE SP HERE
$POWER: .ASCII 15,12,"POWER"
```


.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>),
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW09=1      LOOP ON ERROR
*CALL
*          SCOPE          ;;SCOPE=10T

```

```

$SCOPE:
1$:      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
         BIT          #BIT14,$SWR      ;;LOOP ON PRESENT TEST?
         BNE          $OVER          ;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XSTR:  BR          6$
         MOV          2,$ERRVEC,-(SP)  ;;IF RUNNING ON THE "XOR" TESTER CHANGE
         MOV          #5,$ERRVEC      ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
         TST          2,$177060      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
         MOV          (SP)+,2,$ERRVEC  ;;SET FOR TIMEOUT
         BR          $SVLAD          ;;TIME OUT ON XOR?
4$:      CMP          (SP)+,(SP)+      ;;RESTORE THE ERROR VECTOR
         MOV          (SP)+,2,$ERRVEC  ;;GO TO THE NEXT TEST
         BR          7$              ;;CLEAR THE STACK AFTER A TIME OUT
5$:      MOV          (SP)+,2,$ERRVEC  ;;RESTORE THE ERROR VECTOR
         BR          7$              ;;LOOP ON THE PRESENT TEST
6$:      *****END OF CODE FOR THE XOR TESTER*****
2$:      TSTB         $ERFLG          ;;HAS AN ERROR OCCURRED?
         BEQ          $SVLAD          ;;BR IF NO
         BIT          #BIT09,$SWR      ;;LOOP ON ERROR?
         BEQ          4$              ;;BR IF NO
7$:      MOV          $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
         BR          $OVER
4$:      CLRB         $ERFLG          ;;ZERO THE ERROR FLAG
$SVLAD:  INCB         $STNM           ;;COUNT TEST NUMBERS
         MOVB        $STNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
         MOV          (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
         MOV          (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
         CLR          $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
         MOVB        #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
$OVER:   MOV          $STNM,$DISPLAY  ;;DISPLAY TEST NUMBER
         MOV          $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
         RTI

```

```

2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478

```

```

013000
013000 104406
013002 032777 040000 166030
013010 001052
013012 000416
013014 013746 000004
013020 012737 013040 000004
013026 005737 177060
013032 012637 000004
013036 000421
013040 022626
013042 012637 000004
013046 000407
013050
013050 105767 165727
013054 001412
013056 032777 001000 165754
013064 001404
013066 016767 165716 165712
013074 000420
013076 105067 165701
013102 105267 165674
013106 116767 165670 165756
013114 011667 165666
013120 011667 165664
013124 005067 165730
013130 112767 000001 165657
013136 016777 165640 165676
013144 016716 165636
013150 000002

```

```

2478
2479
2480
2481
2482
2483 013152 112767 000001 000236 $ATY1: MOV  #1,$FFLG ;TO REPORT FATAL ERROR
2484 013160 112767 000001 000226 $ATY3: MOV  #1,$MFLG ;TO TYPE A MESSAGE
2485 013166 000403          BR      $ATYC
2486 013170 112767 000001 000220 $ATY4: MOV  #1,$FFLG ;TO ONLY REPORT FATAL ERROR
2487 013176          $ATYC:
2488 013176 010046          MOV  R0,-(SP) ;PUSH R0 ON STACK
2489 013200 010146          MOV  R1,-(SP) ;PUSH R1 ON STACK
2490 013202 105767 000206          TSTB $MFLG ;SHOULD TYPE A MESSAGE?
2491 013206 001450          BEQ  5$ ;IF NOT: BR
2492 013210 122767 000001 165670          CMPB #APTENV,$ENV ;OPERATING UNDER APT?
2493 013216 001031          BNE  3$ ;IF NOT: BR
2494 013220 132767 000100 165661          BITB #APTPOOL,$ENVM ;SHOULD SPOOL MESSAGE?
2495 013226 001425          BEQ  3$ ;IF NOT: BR
2496 013230 017600 000004          MOV  @4(SP),R0 ;GET MESSAGE ADDRESS
2497 013234 062766 000002 000004          ADD  #2,4(SP) ;BUMP RETURN ADDRESS
2498 013242 005767 165620          1$: TST  $MSGTYPE ;SEE IF DONE W/ LAST XMISSION?
2499 013246 001375          BNE  1$ ;IF NOT: WAIT
2500 013250 010067 165626          MOV  R0,$MSGAD ;PUT ADDRESS IN MAILBOX
2501 013254 105720          2$: TSTB (R0)+ ;FIND END OF MESSAGE
2502 013256 001376          BNE  2$
2503 013260 166700 165616          SUB  $MSGAD,R0 ;SUB START OF MESSAGE
2504 013264 006200          ASR  R0 ;GET MESSAGE LENGTH IN WORDS
2505 013266 010067 165612          MOV  R0,$MSGGLT ;PUT LENGTH IN MAILBOX
2506 013272 012767 000004 165566          MOV  #4,$MSGTYPE ;TELL APT TO TAKE MESSAGE
2507 013300 000413          BR   5$
2508 013302 017667 000004 000016 3$: MOV  @4(SP),4$ ;PUT MSG ADDR IN JSR LINKAGE
2509 013310 062766 000002 000004          ADD  #2,4(SP) ;BUMP RETURN ADDRESS
2510 013316 016746 164454          MOV  177776,-(SP) ;PUSH 177776 ON STACK
2511 013322 004767 000072          JSR  PC,$TYPE ;CALL TYPE MACRO
2512 013326 000000          4$: .WORD 0
2513 013330          5$:
2514 013330 105767 000062          10$: TSTB $FFLG ;SHOULD REPORT FATAL ERROR?
2515 013334 001413          BEQ  12$ ;IF NOT: BR
2516 013336 005767 165544          TST  $ENV ;RUNNING UNDER APT?
2517 013342 001410          BEQ  12$ ;IF NOT: BR
2518 013344 005767 165516          11$: TST  $MSGTYPE ;FINISHED LAST MESSAGE?
2519 013350 001375          BNE  11$ ;IF NOT: WAIT
2520 013352 017667 000004 165510          MOV  @4(SP),$FATAL ;GET ERROR #
2521 013360 005267 165502          INC  $MSGTYPE ;TELL APT TO TAKE ERROR
2522 013364 062766 000002 000004 12$: ADD  #2,4(SP) ;BUMP RETURN ADDRESS
2523 013372 105067 000020          CLRB $FFLG ;CLEAR FATAL FLAG
2524 013376 105067 000013          CLRB $LFLG ;CLEAR LOG FLAG
2525 013402 105067 000006          CLRB $MFLG ;CLEAR MESSAGE FLAG
2526 013406 012601          MOV  (SP)+,R1 ;POP STACK INTO R1
2527 013410 012600          MOV  (SP)+,R0 ;POP STACK INTO R1
2528 013412 000207          RTS  PC ;RETURN
2529 013414          $MFLG: .BYTE 0
2530 013416          $LFLG: .BYTE 0 ;LOG FLAG
2531 013416          $FFLG: .BYTE 0 ;FATAL FLAG
2532
2533
2534          .EVEN

```

J06

PAGE: 0074

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 63
020208.P11 14-JUN-77 09:27 APT COMMUNICATIONS ROUTINE

U
U
U
U
U
U
U
U
U
U

003200
000001
000100
000040

APTSIZE=200
APTELV=001
APTSPOOL=100
APTC SUP=040

.SBTTL TYPE ROUTINE

```

*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.

```

```

*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*

```

2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594

```

013420 105767 165433 $TYPE: TSTB $TPFLG ;: IS THERE A TERMINAL?
013424 100002 BPL 1$ ;: BR IF YES
013426 000000 HALT ;: HALT HERE IF NO TERMINAL
013430 000430 BR 3$ ;: LEAVE
013432 010046 1$: MOV RO, -(SP) ;: SAVE RO
013434 017600 000002 MOV @2(SP), RO ;: GET ADDRESS OF ASCIZ STRING
013440 122767 000001 165440 CMPB #APTENV, $ENV ;: RUNNING IN APT MODE
013446 001011 BNE 62$ ;: NO, GO CHECK FOR APT CONSOLE
013450 132767 000100 165431 BITB #APTPOOL, $ENVM ;: SPOOL MESSAGE TO APT
013456 001405 BEQ 62$ ;: NO, GO CHECK FOR CONSOLE
013460 010067 000004 MOV RO, 61$ ;: SETUP MESSAGE ADDRESS FOR APT
013464 004767 177470 JSR PC, $ATY3 ;: SPOOL MESSAGE TO APT
013470 000000 61$: .WORD 0 ;: MESSAGE ADDRESS
013472 132767 000040 165407 62$: BITB #APTCSUP, $ENVM ;: APT CONSOLE SUPPRESSED
013500 001003 BNE 60$ ;: YES, SKIP TYPE OUT
013502 112046 2$: MOVB (RO)+, -(SF) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
013504 001005 BNE 4$ ;: BR IF IT ISN'T THE TERMINATOR
013506 005726 TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
013510 012600 60$: MOV (SP)+, RO ;: RESTORE RO
013512 062716 000002 3$: ADD #2, (SP) ;: ADJUST RETURN PC
013516 000002 RTI ;: RETURN
013520 122716 000011 4$: CMPB #HT, (SP) ;: BRANCH IF <HT>
013524 001430 BEQ 8$ ;:
013526 122716 000200 CMPB #CRLF, (SP) ;: BRANCH IF NOT <CRLF>
013532 001006 BNE 5$ ;:
013534 005726 TST (SP)+ ;: POP <CR><LF> EQUIV
013536 104401 TYPE ;: TYPE A CR AND LF
013540 001063 $CRLF ;:
013542 105067 000130 CLRB $CHARCNT ;: CLEAR CHARACTER COUNT
013546 000755 BR 2$ ;: GET NEXT CHARACTER
013550 004767 000056 5$: JSR PC, $TYPEC ;: GO TYPE THIS CHARACTER
013554 126726 165276 6$: CMPB $FILLC, (SP)+ ;: IS IT TIME FOR FILLER CHARS.?
013560 001350 BNE 2$ ;: IF NO GO GET NEXT CHAR.
013562 016746 165266 MOV $NULL, -(SP) ;: GET # OF FILLER CHARS. NEEDED
;: AND THE NULL CHAR.
013566 105366 000001 7$: DECB 1(SP) ;: DOES A NULL NEED TO BE TYPED?
013572 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
013574 004767 000032 JSR PC, $TYPEC ;: GO TYPE A NULL

```

.MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 65
DZDLDB.P11 14-JUN-77 09:27 TYPE ROUTINE

```

2595 013600 105367 000072          DECB  $CHARCNT      ;;DC NOT COUNT AS A COUNT
2596 013604 000770          BR      7$          ;;LOOP
2597
2598          ;HORIZONTAL TAB PROCESSOR
2599
2600 013606 112716 000040      8$:   MOVB  #' (SP)      ;; REPLACE TAB WITH SPACE
2601 013612 004767 000014      9$:   JSR   PC,$TYPEC    ;; TYPE A SPACE
2602 013616 132767 000007 000052  BITB  #7,$CHARCNT    ;; BRANCH IF NOT AT
2603 013624 001372          BNE   9$          ;; TAB STOP
2604 013626 005726          TST   (SP)+        ;; POP SPACE OFF STACK
2605 013630 000724          BR    2$          ;; GET NEXT CHARACTER
2606 013632 105777 165212  $TYPEC: TSTB  @2$TPS    ;; WAIT UNTIL PRINTER IS READY
2607 013636 100375          BPL   $TYPEC
2608 013640 116677 000002 165204  MOVB  2(SP),@2$TPB   ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2609 013646 122766 000015 000002  CMPB  #CR,2(SP)     ;; IS CHARACTER A CARRIAGE RETURN?
2610 013654 001003          BNE   1$          ;; BRANCH IF NO
2611 013656 105067 000014          CLRB  $CHARCNT     ;; YES--CLEAR CHARACTER COUNT
2612 013662 000406          BR    $TYPEX      ;; EXIT
2613 013664 122766 000012 000002  1$:   CMPB  #LF,2(SP)   ;; IS CHARACTER A LINE FEED?
2614 013672 001402          BEQ   $TYPEX      ;; BRANCH IF YES
2615 013674 105227          INCB (PC)+        ;; COUNT THE CHARACTER
2616 013676 000000          $CHARCNT: .WORD  0 ;; CHARACTER COUNT STORAGE
2617 013700 000207          $TYPEX: RTS      PC
2618
2619          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2620
2621          ;*****
2622          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2623          ;OCTAL (ASCII) NUMBER AND TYPE IT.
2624          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2625          ;*CALL:
2626          ;*   MOV    NUM,-(SP)      ;; NUMBER TO BE TYPED
2627          ;*   TYPOS      ;; CALL FOR TYPEOUT
2628          ;*   .BYTE  N          ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2629          ;*   .BYTE  M          ;; M=1 OR 0
2630          ;*                                     ;; 1=TYPE LEADING ZEROS
2631          ;*                                     ;; 0=SUPPRESS LEADING ZEROS
2632          ;*
2633          ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2634          ;*$TYPOS OR $TYPOC
2635          ;*CALL:
2636          ;*   MOV    NUM,-(SP)      ;; NUMBER TO BE TYPED
2637          ;*   TYPON      ;; CALL FOR TYPEOUT
2638          ;*
2639          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2640          ;*CALL:
2641          ;*   MOV    NUM,-(SP)      ;; NUMBER TO BE TYPED
2642          ;*   TYPOC      ;; CALL FOR TYPEOUT
2643          ;*
2644 013702 017646 000000          $TYPOS: MOV   @2(SP),-(SP)  ;; PICKUP THE MODE
2645 013706 116667 000001 000211  MOVB  1(SP),$OFILL  ;; LOAD ZERO FILL SWITCH
2646 013714 112667 000207          MOVB  (SP)+,$OMODE+1 ;; NUMBER OF DIGITS TO TYPE
2647 013720 062716 000002          ADD   #2,(SP)      ;; ADJUST RETURN ADDRESS
2648 013724 000406          BR    $TYPON
2649 013726 112767 000001 000171  $TYPOC: MOVB  #1,$OFILL  ;; SET THE ZERO FILL SWITCH
2650 013734 112767 000006 000165  MOVB  #6,$OMODE+1  ;; SET FOR SIX(6) DIGITS

```

2651	013742	112767	000005	000154	\$TYPON: MOV	#5,\$OCNT	::SET THE ITERATION COUNT
2652	013750	010346			MOV	R3,-(SP)	::SAVE R3
2653	013752	010446			MOV	R4,-(SP)	::SAVE R4
2654	013754	010546			MOV	R5,-(SP)	::SAVE R5
2655	013756	116704	000145		MOV	\$OMODE+1,R4	::GET THE NUMBER OF DIGITS TO TYPE
2656	013762	005404			NEG	R4	
2657	013764	062704	000006		ADD	#6,R4	::SUBTRACT IT FOR MAX. ALLOWED
2658	013770	110467	000132		MOV	R4,\$OMODE	::SAVE IT FOR USE
2659	013774	116704	000125		MOV	\$OFILL,R4	::GET THE ZERO FILL SWITCH
2660	014000	016605	000012		MOV	12(SP),R5	::PICKUP THE INPUT NUMBER
2661	014004	005003			CLR	R3	::CLEAR THE OUTPUT WORD
2662	014006	006105		1\$:	ROL	R5	::ROTATE MSB INTO "C"
2663	014010	000404			BR	3\$::GO DO MSB
2664	014012	006105		2\$:	ROL	R5	::FORM THIS DIGIT
2665	014014	006105			ROL	R5	
2666	014016	006105			ROL	R5	
2667	014020	010503			MOV	R5,R3	
2668	014022	006103		3\$:	ROL	R3	::GET LSB OF THIS DIGIT
2669	014024	105367	000076		DECB	\$OMODE	::TYPE THIS DIGIT?
2670	014030	100016			BPL	7\$::BR IF NO
2671	014032	042703	177770		BIC	#177770,R3	::GET RID OF JUNK
2672	014036	001002			BNE	4\$::TEST FOR 0
2673	014040	005704			TST	R4	::SUPPRESS THIS 0?
2674	014042	001403			BEQ	5\$::BR IF YES
2675	014044	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
2676	014046	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
2677	014052	052703	000040		BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
2678	014056	110367	000040		MOV	R3,8\$::SAVE FOR TYPING
2679	014062	104401	014122		TYPE	8\$::GO TYPE THIS DIGIT
2680	014066	105367	000032		DECB	\$OCNT	::COUNT BY 1
2681	014072	003347			BGT	2\$::BR IF MORE TO DO
2682	014074	002402			BLT	6\$::BR IF DONE
2683	014076	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
2684	014100	000744			BR	2\$::GO DO THE LAST DIGIT
2685	014102	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
2686	014104	012604			MOV	(SP)+,R4	::RESTORE R4
2687	014106	012603			MOV	(SP)+,R3	::RESTORE R3
2688	014110	016666	000002	000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
2689	014116	012616			MOV	(SP)+,(SP)	
2690	014120	000002			RTI		::RETURN
2691	014122	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
2692	014123	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
2693	014124	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
2694	014125	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
2695	014126	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE
2696							

2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752

.SBTTL TTY INPUT ROUTINE

::*****

.ENABL LSB

::*****

*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.

*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL

*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL

*WHEN OPERATING IN TTY FLAG MODE.

\$CKSWR: CMP #SWREG,SWR :: IS THE SOFT-SWR SELECTED?

BNE 15\$:: BRANCH IF NO

TSTB @STKS :: CHAR THERE?

BPL 15\$:: IF NO, DON'T WAIT AROUND

MOVB @STKB, -(SP) :: SAVE THE CHAR

BIC #1C177, (SP) :: STRIP-OFF THE ASCII

CMF #7, (SP)+ :: IS IT A CONTROL G?

BNE 15\$:: NO, RETURN TO USER

CMPB \$AUTOB, #1 :: ARE WE RUNNING IN AUTO-MODE?

BEQ 15\$:: BRANCH IF YES

\$GTSWR: TYPE . \$CNTLG :: ECHO THE CONTROL-G (↑G)

TYPE \$MSWR :: TYPE CURRENT CONTENTS

MOV \$WREG, -(SP) :: SAVE SWREG FOR TYPEOUT

TYPOC :: GO TYPE--OCTAL ASCII(ALL DIGITS)

TYPE . \$MNEW :: PROMPT FOR NEW SWR

19\$: CLR -(SP) :: CLEAR COUNTER

CLR -(SP) :: THE NEW SWR

7\$: TSTB @STKS :: CHAR THERE?

BPL 7\$:: IF NOT TRY AGAIN

MOVB @STKB, -(SP) :: PICK UP CHAR

BIC #1C177, (SP) :: MAKE IT 7-BIT ASCII

9\$: CMP (SP), #25 :: IS IT A CONTROL-U?

BNE 10\$:: BRANCH IF NOT

TYPE . \$CNTLU :: YES, ECHO CONTROL-U (↑U)

20\$: ADD #6, SP :: IGNORE PREVIOUS INPUT

BR 19\$:: LET'S TRY IT AGAIN

10\$: CMP (SP), #15 :: IS IT A <CR>?

BNE 16\$:: BRANCH IF NO

TST 4(SP) :: YES, IS IT THE FIRST CHAR?

BEQ 11\$:: BRANCH IF YES

MOV 2(SP), @SWR :: SAVE NEW SWR

11\$: ADD #6, SP :: CLEAR UP STACK

14\$: TYPE . \$CRLF :: ECHO <CR> AND <LF>

CMPB \$INTAG, #1 :: RE-ENABLE TTY KBD INTERRUPTS?

BNE 15\$:: BRANCH IF NOT

15\$: MOV #100, @STKS :: RE-ENABLE TTY KBD INTERRUPTS

RTI :: RETURN

16\$: JSR PC, \$TYPEC :: ECHO CHAR

```

2753 014336 021627 000060          CMP      (SP),#60      ;;CHAR < 0?
2754 014342 002420          BLT      18$          ;;BRANCH IF YES
2755 014344 021627 000067          CMP      (SP),#67      ;;CHAR > 7?
2756 014350 003015          BGT      18$          ;;BRANCH IF YES
2757 014352 042726 000060          BIC      #60,(SP)+    ;;STRIP-OFF ASCII
2758 014356 005766 000002          TST      2,(SP)      ;;IS THIS THE FIRST CHAR
2759 014362 001403          BEQ      17$          ;;BRANCH IF YES
2760 014364 006316          ASL      (SP)        ;;NO, SHIFT PRESENT
2761 014366 006316          ASL      (SP)        ;;CHAR OVER TO MAKE
2762 014370 006316          ASL      (SP)        ;;ROOM FOR NEW ONE.
2763 014372 005266 000002 17$: INC      2,(SP)      ;;KEEP COUNT OF CHAR
2764 014376 056616 177776          BIS      -2,(SP),(SP) ;;SET IN NEW CHAR
2765 014402 000707          BR       7$          ;;GET THE NEXT ONE
2766 014404 104401 001062 19$: TYPE  $QUES      ;;TYPE ?(CR)(LF)
2767 014410 000720          BR       20$        ;;SIMULATE CONTROL-U
                .DSABL  LSB

```

THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

*CALL:

```

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

```

```

2778          $RDCHR: MOV      (SP),-(SP)    ;;PUSH DOWN THE PC
2779 014412 011646          MOV      4(SP),2(SP)  ;;SAVE THE PS
2780 014414 016666 000004 000002 1$: TSTB   2$TKS      ;;WAIT FOR
2781 014422 105777 164416          BPL      1$          ;;A CHARACTER
2782 014426 100375          MOVB    2$TKB,4(SP)  ;;READ THE TTY
2783 014430 117766 164412 000004          BIC      #1C(177),4(SP) ;;GET RID OF JUNK IF ANY
2784 014436 042766 177600 000004          CMP      4(SP),#23   ;;IS IT A CONTROL-S?
2785 014444 025627 000004 000023          BNE      3$          ;;BRANCH IF NO
2786 014452 001013          TSTB   2$TKS      ;;WAIT FOR A CHARACTER
2787 014454 105777 164364 2$: BPL      2$          ;;LOOP UNTIL ITS THERE
2788 014460 100375          MOVB    2$TKB,-(SP)  ;;GET CHARACTER
2789 014462 117746 164360          BIC      #1C177,(SP) ;;MAKE IT 7-BIT ASCII
2790 014466 042716 177600          CMP      (SP)+,#21   ;;IS IT A CONTROL-G?
2791 014472 022627 000021          BNE      2$          ;;IF NOT DISCARD IT
2792 014476 001366          BR      1$          ;;YES, RESUME
2793 014500 000750          CMP      4(SP),#140  ;;IS IT UPPER CASE?
2794 014502 026627 000004 000140 3$: BLT      4$          ;;BRANCH IF YES
2795 014510 002407          CMP      4(SP),#175  ;;IS IT A SPECIAL CHAR?
2796 014512 026627 000004 000175          BGT      4$          ;;BRANCH IF YES
2797 014520 003003          BIC      #40,4(SP)   ;;MAKE IT UPPER CASE
2798 014522 042766 000040 000004 4$: RTI          ;;GO BACK TO USER
2799 014530 000002

```

THIS ROUTINE WILL INPUT A STRING FROM THE TTY

*CALL:

```

*      RDLIN         ;;INPUT A STRING FROM THE TTY
*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*                  ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

```

```

2800          $RDLIN: MOV      R3, -(SP)    ;;SAVE R3
2801 014532 010346          1$: MOV      #5+TYIN,R3 ;;GET ADDRESS
2802 014534 012703 014640

```


2809	014540	022703	014650	25:	CMP	#STTYIN+8.,R3	:: BUFFER FULL?	
2810	014544	101405			BLOS	45	:: BR IF YES	
2811	014546	104407			RDCHR		:: GO READ ONE CHARACTER FROM THE TTY	
2812	014550	112613			MOVB	(SP)+,(R3)	:: GET CHARACTER	
2813	014552	122713	000177	105:	CMPB	#177,(R3)	:: IS IT A RUBOUT	
2814	014556	001003			BNE	35	:: SKIP IF NOT	
2815	014560	104401	001062	45:	TYPE	5QUES	:: TYPE A '?'	
2816	014564	000763			BR	15	:: CLEAR THE BUFFER AND LOOP	
2817	014566	111367	000044	35:	MOVB	(R3),95	:: ECHO THE CHARACTER	
2818	014572	104401	014636		TYPE	95		
2819	014576	122723	000015		CMPB	#15,(R3)+	:: CHECK FOR RETURN	
2820	014602	001356			BNE	25	:: LOOP IF NOT RETURN	
2821	014604	105063	177777		CLRB	-1(R3)	:: CLEAR RETURN (THE 15)	
2822	014610	104401	001064		TYPE	5LF	:: TYPE A LINE FEED	
2823	014614	012603			MOV	(SP)+,R3	:: RESTORE R3	
2824	014616	011646			MOV	(SP)-,(SP)	:: ADJUST THE STACK AND PUT ADDRESS OF THE	
2825	014620	016666	000004 000002		MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON ?	
2826	014626	012766	014640 000004		MOV	#STTYIN,4(SP)		
2827	014634	000002			RTI		:: RETURN	
2828	014636	000		95:	.BYTE	0	:: STORAGE FOR ASCII CHAR. TO TYPE	
2829	014637	000			.BYTE	0	:: TERMINATOR	
2830	014640	000010			STTYIN:	.BLKB	8	:: RESERVE 8 BYTES FOR TTY INPUT
2831	014650	052536	005015 000		SCNTLU:	.ASCIZ	<10><15><12>	:: CONTROL "J"
2832	014655	136	006507 000012		SCNTLG:	.ASCIZ	<10><15><12>	:: CONTROL "G"
2833	014662	005015	053523 020122		SMSWR:	.ASCIZ	<15><12>SWR =	
2834	014670	020075	000					
2835	014673	040	047040 053505		SMNEW:	.ASCIZ	NEW =	
2836	014700	036440	000040					

.SBTTL TRAP DECODER

::*****
:*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
:*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
:*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
:*GO TO THAT ROUTINE.

2838									
2839									
2840									
2841									
2842									
2843									
2844									
2845									
2846									
2847									
2848	014704	010046							
2849	014706	016600	000002						
2850	014712	005740							
2851	014714	111000							
2852	014716	006300							
2853	014720	016000	014740						
2854	014724	000200							

::THIS IS USE TO HANDLE THE "GETPRI" MACRO

2855									
2856									
2857									
2858									
2859	014726	011646							
2860	014730	016666	000004	000002					
2861	014736	000002							

.SBTTL TRAP TABLE

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

2862									
2863									
2864									
2865									
2866									
2867									
2868									
2869									
2870	014740	014726							
2871	014742	013420							
2872	014744	013726							
2873	014746	013702							
2874	014750	013742							
2875									
2876	014752	014200							
2877									
2878	014754	014130							
2879	014756	014412							
2880	014760	014502							

```

2882 .SBTTL ECHO TEST
2883 ;*****
2884 ;*THIS ROUTINE WILL ECHO ANY CHARACTER TYPED
2885 ;*AT THE CONSOLE
2886 ;*THE TEST IS HALTED BY TYPING A CONTROL-C
2887 ;*TEST CAN BE RESTARTED AFTER HALTING BY JUST CONTINUING
2888 ;*****
2889 ECHO:
2890 014762 000005 RESET ;CLEAR EVERYTHING
2891 014762 112777 000052 165402 MOVB #*,QCTBUF ;TRANSMIT PROMPT "*"
2892 014772 105777 165370 2$: TSTB QCTCSR ;WAIT FOR INPUT
2893 014776 100375 BPL 2$
2894 015000 117777 165364 165366 MOVB QCRBUF,QCTBUF ;ECHO INPUT
2895 015006 017700 165356 MOV QCRBUF,RO ;STORE INPUT
2896 015012 100023 BPL 5$ ;BR IF "ERROR" NOT SET
2897 015014 052701 010000 BIS #BIT12,R1 ;SET PARITY ERROR TEST MASK
2898 015020 030100 BIT R1,RO ;CHECK FOR PARITY ERROR FLAG
2899 015022 001403 BEQ 3$ ;BR IF NOT SET
2900 015024 004767 000056 JSR PC,MSG ;REPORT PARITY ERROR
2901 015030 015134 MPAR
2902 015032 006301 3$: ASL R1 ;SHIFT MASK TO TEST "FR" FLAG
2903 015034 030100 BIT R1,RO ;TEST FOR FRAMING ERROR FLAG
2904 015036 001403 BEQ 4$ ;BR IF NOT SET
2905 015040 004767 000042 JSR PC,MSG ;REPSORT FRAMING ERROR
2906 015044 015145 MFR
2907 015046 006301 4$: ASL R1 ;SHIFT MASK TO TEST "OR" FLAG
2908 015050 030100 BIT R1,RO ;TEST FOR OVERFLOW ERROR
2909 015052 001403 BEQ 5$ ;BR IF NOT SET
2910 015054 004767 000026 JSR PC,MSG ;REPORT OVERFLOW ERROR
2911 015060 015157 MOR
2912 015062 042700 000200 5$: BIC #BIT7,RO ;CLEAR ANY PARITY BIT
2913 015066 022700 000003 CMP #3,RO ;WAS INPUT CONTROL-C
2914 015072 001337 BNE 2$ ;BR IF NOT
2915 015074 004767 000006 JSR PC,MSG ;REPORT PROGRAM STOP
2916 015100 015172 MSTOP
2917 015102 000000 HALT ;END OF TEST HALT
2918 015104 000726 BR ECHO ;AFTER END OF TEST HALT
2919 ; PRESS CONTINUE TO RESTART ECHO TEST
2920
2921 015106 013600 MSG: MOV Q(SP),RO ;PICK UP MESSAGE POINTER
2922 015110 062746 000002 ADD #2,-(SP) ;ADJUST RETURN PC
2923 015114 105777 165252 WAIT: TSTB QCTCSR ;WAIT FOR XMIT DONE
2924 015120 100375 BPL WAIT
2925 015122 112077 165246 MOVB (RO)+,QCTBUF ;SEND CHARACTER
2926 015126 105710 TSTB (RO) ;IS THIS END OF MESSAGE?
2927 015130 001371 BNE WAIT ;BR IF NOT
2928 015132 000207 RTS PC ;RETURN
2929
2930 015134 005015 040520 044522 MPAR: .ASCIZ <CR><LF>/PARITY/
2931 015142 054524 000
2932 015145 015 043012 040522 MFR: .ASCIZ <CR><LF>/FRAMING/
2933 015152 044515 043516 000
2934 015157 015 047412 042526 MOR: .ASCIZ <CR><LF>/OVERFLOW/
2935 015164 043122 047514 000127
2936 015172 005015 052123 050117 MSTOP: .ASCIZ <CR><LF>/STOP/
2937 015200 000

```

2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971

015202

.EVEN

.SBTTL TERMINAL OUTPUT TEST

::*****
: *THIS ROUTINE WILL OUTPUT ALL WRITABLE CHARACTERS FOR THE
: *THE OCTAL CODE 040 --> 377
: *32 CHARACTERS ARE PRINTED ON EACH LINE
: *THE PATTERN IS REPEATED EVERY THREE LINES
: *
::*****

015202 012701 000040
015206 012700 000040
015212 105777 165154
015216 100375
015220 010177 165150
015224 105201
015226 005300
015230 001370
015232 004767 177650

015236 001063
015240 105777 165122
015244 100404
015246 032701 000200
015252 001353
015254 000754

015256 005077 165106
015262 000000
015264 000746

OUTTST: MOV #40,R1 ;LOAD FIRST WRITABLE CHARACTER
1\$: MOV #40,R0 ;LOAD CHAR COUNT PER LINE
2\$: TSTB @CTCSR ;WAIT FOR DONE
BPL 2\$
MOV R1,@CTBUF ;TRANSMIT A CHARACTER
INCB R1 ;INCREMENT CHARACTER CODE
DEC RC ;DECREMENT CHAR COUNT
BNE 2\$;BR IF LINE NOT COMPLETE
JSR PC,MSG ;ISSUE CR,LINE FEED

\$CRLF
TSTB @CRCSR ;ANY CHARACTER RECEIVED?
BNI 3\$;BR IF YES
BIT #BIT7,R1 ;FINISHED ONE PASS OF WRITABLE CHARACTERS?
BNE OUTTS+ ;BR IF YES
BR 1\$;IF NOT WRITE NEXT LINE

3\$: CLR @CRBUF ;CLEAR RECEIVER
HALT ;STOP TEST
BR OUTTST ;RESTART TEST IF CONTINUED

3028	015720	020124	046103	040505		
3029	015726	020122	044502	030124		
3030	015734	047440	020106	041524		
3031	015742	051123	000			
3032	015745	102	052111	020062	EM15:	.ASCIZ /BIT2 OF TCSR NOT CLEAR AFTER RESET/
3033	015752	043117	052040	051503		
3034	015760	020122	047516	020124		
3035	015766	046103	040505	020122		
3036	015774	043101	042524	020122		
3037	016002	042522	042523	000124		
3038	016010	040503	020116	047516	EM16:	.ASCIZ /CAN NOT SET BIT2 OF TCSR/
3039	016016	020124	042523	020124		
3040	016024	044502	031124	047440		
3041	016032	020106	041524	051123		
3042	016040	000				
3043	016041	103	047101	047040	EM17:	.ASCIZ /CAN NOT CLEAR BIT2 OF TCSR/
3044	016046	052117	041440	042514		
3045	016054	051101	041040	052111		
3046	016062	020062	043117	052040		
3047	016070	051503	000122			
3048	016074	042522	042523	020124	EM20:	.ASCIZ /RESET DID NOT CLEAR BIT2 OF TCSR/
3049	016102	044504	020104	047516		
3050	016110	020124	046103	040505		
3051	016116	020122	044502	031124		
3052	016124	047440	020106	041524		
3053	016132	051123	000			
3054	016135	102	052111	020066	EM21:	.ASCIZ /BIT6 OF TCSR NOT CLEAR AFTER RESET/
3055	016142	043117	052040	051503		
3056	016150	020122	047516	020124		
3057	016156	046103	040505	020122		
3058	016164	043101	042524	020122		
3059	016172	042522	042523	027524		
3060	016200	000				
3061	016201	130	044515	020124	EM22:	.ASCIZ /XMIT INTERRUPT AT PRIORITY 7/
3062	016206	047111	042524	051122		
3063	016214	050125	020124	052101		
3064	016222	050040	044522	051117		
3065	016230	052111	020131	000067		
3066	016236	040503	020116	047516	EM23:	.ASCIZ /CAN NOT SET BIT6 OF TCSR/
3067	016244	020124	042523	020124		
3068	016252	044502	033124	047440		
3069	016260	020106	041524	051123		
3070	016266	000				
3071	016267	103	047101	047040	EM24:	.ASCIZ /CAN NOT CLEAR BIT6 OF TCSR/
3072	016274	052117	041440	042514		
3073	016302	051101	041040	052111		
3074	016310	020066	043117	052040		
3075	016316	051503	000122			
3076	016322	042522	042523	020124	EM25:	.ASCIZ /RESET DID NOT CLEAR BIT6 OF TCSR/
3077	016330	044504	020104	047516		
3078	016336	020124	046103	040505		
3079	016344	020122	044502	033124		
3080	016352	047440	020106	041524		
3081	016360	051123	000			
3082	016363	102	052111	020066	EM26:	.ASCIZ /BIT6 OF TCSR NOT CLEAR AFTER RESET/
3083	016370	043117	052040	051503		

3084	016376	020122	047516	020124	
3085	016404	046103	040505	020122	
3086	016412	043101	042524	020122	
3087	016420	042522	042523	000124	
3088	016426	041522	051126	044440	EM27: .ASCIZ /RCVR INTERRUPT WITH PRIORITY 7/
3089	016434	052116	051105	052522	
3090	016442	052120	053440	052111	
3091	016450	020110	051120	047511	
3092	016456	044522	054524	033440	
3093	016464	000			
3094	016465	103	047101	047040	EM30: .ASCIZ /CAN NOT SET BIT6 OF RCSR/
3095	016472	052117	051440	052105	
3096	016500	041040	052111	020066	
3097	016506	043117	051040	051503	
3098	016514	000122			
3099	016516	040503	020116	047516	EM31: .ASCIZ /CAN NOT CLEAR BIT6 OF RCSR/
3100	016524	020124	046103	040505	
3101	016532	020122	044502	033124	
3102	016540	047440	020106	041522	
3103	016546	051123	000		
3104	016551	103	047101	047040	EM32: .ASCIZ /CAN NOT CLEAR BIT6 OF RCSR WITH RESET/
3105	016556	052117	041440	042514	
3106	016564	051101	041040	052111	
3107	016572	020066	043117	051040	
3108	016600	051503	020122	044527	
3109	016606	044124	051040	051505	
3110	016614	052105	000		
3111	016617	102	052111	020066	EM33: .ASCIZ /BIT6 OF LKS NOT CLEAR AFTER RESET/
3112	016624	043117	046040	051513	
3113	016632	047040	052117	041440	
3114	016640	042514	051101	040440	
3115	016646	052106	051105	051040	
3116	016654	051505	052105	000	
3117	016661	114	051513	044440	EM34: .ASCIZ /LKS INTERRUPT WITH PRIORITY 7/
3118	016666	052116	051105	052522	
3119	016674	052120	053440	052111	
3120	016702	020110	051120	047511	
3121	016710	044522	054524	033440	
3122	016716	000			
3123	016717	103	047101	047040	EM35: .ASCIZ /CAN NOT SET BIT6 OF LKS/
3124	016724	052117	051440	052105	
3125	016732	041040	052111	020066	
3126	016740	043117	046040	051513	
3127	016746	000			
3128	016747	103	047101	047040	EM36: .ASCIZ /CAN NOT CLEAR BIT6 OF LKS/
3129	016754	052117	041440	042514	
3130	016762	051101	041040	052111	
3131	016770	020066	043117	046040	
3132	016776	051513	000		
3133	017001	122	051505	052105	EM37: .ASCIZ /RESET DID NOT CLEAR BIT6 OF LKS/
3134	017006	042040	042111	047040	
3135	017014	052117	041440	042514	
3136	017022	051101	041040	052111	
3137	017030	020066	043117	046040	
3138	017036	051513	000		
3139	017041	104	040525	020114	EM40: .ASCIZ /LOCAL ADDRESSING ERROR/

MAIN. MACY11 30(1046) 14-JUN-77 17:10 PAGE 78
 22DL08.P11 14-JUN-77 09:27 TERMINAL OUTPUT TEST

PAGE: 0089

3252	020162	044504	020104	047516	
3253	020170	020124	046103	040505	
3254	020176	020122	041522	051126	
3255	020204	042040	047117	000105	
3256	020212	042522	042101	047111	EM70: .ASCIZ /READING RBUF DID NOT CLEAR RCVR DONE/
3257	020220	020107	041122	043125	
3258	020226	042040	042111	047040	
3259	020234	052117	041440	042514	
3260	020242	051101	051040	053103	
3261	020250	020122	047504	042516	
3262	020256	000			
3263	020257	122	053103	020122	EM71: .ASCIZ /RCVR INTERRUPTS WITH ENABLE CLEAR/
3264	020264	047111	042524	051122	
3265	020272	050125	051524	053440	
3266	020300	052111	030110	047105	
3267	020306	041101	042514	041440	
3268	020314	042514	051101	000	
3269	020321	122	053103	020122	EM73: .ASCIZ /RCVR INTERRUPTS AT PRIORITY 7/
3270	020326	047111	042524	051122	
3271	020334	050125	051524	040440	
3272	020342	020124	051120	047511	
3273	020350	044522	054524	033440	
3274	020356	000			
3275	020357	122	053103	020122	EM74: .ASCIZ /RCVR INT ROST PASSED WITH ENABLE CLEAR/
3276	020364	047111	020124	050522	
3277	020372	052123	050040	051501	
3278	020400	042523	020104	044527	
3279	020406	044124	042440	040516	
3280	020414	046102	020105	046103	
3281	020422	040505	000122		
3282	020426				EM72:
3283	020426	041522	051126	042040	EM75: .ASCIZ /RCVR DID NOT INTERRUPT/
3284	020434	042111	047040	052117	
3285	020442	044440	052116	051105	
3286	020450	052522	052120	000	
3287	020455	122	053103	020122	EM76: .ASCIZ /RCVR RE-INTERRUPTED/
3288	020462	042522	044455	052116	
3289	020470	051105	052522	052120	
3290	020476	042105	000		
3291	020501	122	040505	044504	EM77: .ASCIZ /READING RBUF DID NOT CLEAR INTERRUPT/
3292	020506	043516	051040	052502	
3293	020514	020106	044504	020104	
3294	020522	047516	020124	046103	
3295	020530	040505	020122	047111	
3296	020536	042524	051122	050125	
3297	020544	000124			
3298	020546	042522	042523	020124	EM100: .ASCIZ /RESET DID NOT CLEAR RCVR INTERRUPT/
3299	020554	044504	020104	047516	
3300	020562	020124	046103	040505	
3301	020570	020122	041522	051126	
3302	020576	044440	052116	051105	
3303	020604	052522	052120	000	
3304	020611	047	051117	020047	EM101: .ASCIZ /'OR' FLAG DID NOT SET/
3305	020616	046106	043501	042040	
3306	020624	042111	047040	052117	
3307	020632	051440	052105	000	

.M
 02
 SR
 SR
 SR
 SR
 SR
 SR
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 SS
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 ST
 SU
 SU
 SU
 SY
 SY
 SY
 SY
 SY
 SY
 SC

3308	020637	047	051105	047522	EM102: .ASCIZ /'ERROR' NOT SET WITH 'OR' FLAG/
3309	020644	023522	047040	052117	
3310	020652	051440	052105	053440	
3311	020660	052111	020110	047447	
3312	020666	023522	043040	040514	
3313	020674	000107			
3314	020676	051102	040505	020113	EM103: .ASCIZ /BREAK DID NOT XMIT ALL ZEROES/
3315	020704	044504	020104	047516	
3316	020712	020124	046530	052111	
3317	020720	040440	046114	055040	
3318	020726	051105	042517	000123	
3319	020734	051102	040505	020113	EM104: .ASCIZ /BREAK DID NOT SET 'FR' ERROR/
3320	020742	044504	020104	047516	
3321	020750	020124	042523	020124	
3322	020756	043047	023522	042440	
3323	020764	051122	051117	000	
3324	020771	104	052101	020101	EM105: .ASCIZ /DATA COMPARE ERROR/
3325	020776	047503	050115	051101	
3326	021004	020105	051105	047522	
3327	021012	000122			
3328	021014	040504	040524	041440	EM106: .ASCIZ /DATA COMPARE ERROR WITH CABLE/
3329	021022	046517	040520	042522	
3330	021030	042440	051122	051117	
3331	021036	053440	052111	020110	
3332	021044	040503	046102	000105	
3333	021052	044524	042515	052517	EM107: .ASCIZ /TIMEOUT IN EXERCISER TEST/
3334	021060	020124	047111	042440	
3335	021066	042530	041522	051511	
3336	021074	051105	052040	051505	
3337	021102	000124			
3338	021104	047111	047503	051122	EM110: .ASCIZ /INCORRECT RECEIVE COUNT/
3339	021112	041505	020124	042522	
3340	021120	042503	053111	020105	
3341	021126	047503	047125	000124	
3342	021134	040504	040524	041440	EM111: .ASCIZ /DATA COMPARE ERROR IN EXERCISER/
3343	021142	046517	040520	042522	
3344	021150	042440	051122	051117	
3345	021156	044440	020116	054105	
3346	021164	051105	044503	042523	
3347	021172	000122			
3348	021174	051124	050101	041440	EM112: .ASCIZ /TRAP CATCHER/
3349	021202	052101	044103	051105	
3350	021210	000			
3351	021211	116	020117	046103	EM113: .ASCIZ /NO CLK INTERRUPT IN EXERCISER/
3352	021216	020113	047111	042524	
3353	021224	051122	050125	020124	
3354	021232	047111	042440	042530	
3355	021240	041522	051511	051105	
3356	021246	000			
3357	021247	042	051105	047522	EM114: .ASCIZ /"ERRCR" NOT SET WITH "FR" FLAG/
3358	021254	021122	047040	052117	
3359	021262	051440	052105	053440	
3360	021270	052111	020110	043042	
3361	021276	021122	043040	040514	
3362	021304	000107			
3363	021306	041522	020126	041501	EM115: .ASCIZ /RCV ACTIVE NOT CLEAR WITH INIT/

ABASE =	176500	119#	232	273
ACDW1 =	000000	232		
ACDW2 =	000000	232		
ACPUOP =	000000	232	247	
ADDW0 =	000000	232		
ADDW1 =	000000	232		
ADDW10 =	000000	232		
ADDW11 =	000000	232		
ADDW12 =	000000	232		
ADDW13 =	000000	232		
ADDW14 =	000000	232		
ADDW15 =	000000	232		
ADDW2 =	000000	232		
ADDW3 =	000000	232		
ADDW4 =	000000	232		
ADDW5 =	000000	232		
ADDW6 =	000000	232		
ADDW7 =	000000	232		
ADDW8 =	000000	232		
ADDW9 =	000000	232		
ACEVCT =	000000	232		
ACEVM =	000000	232		
ADR =	003562	232		
ADRTBL =	002414	232		
AENV =	000000	232		
AENVM =	000000	232		
AFATAL =	000000	232		
AMADR1 =	000000	232		
AMADR2 =	000000	232		
AMADR3 =	000000	232		
AMADR4 =	000000	232		
AMAMS1 =	000000	232		
AMAMS2 =	000000	232		
AMAMS3 =	000000	232		
AMAMS4 =	000000	232		
AMSGAO =	000000	232		
AMSGLG =	000000	232		
AMSGTY =	000000	232		
AMTYP1 =	000000	232		
AMTYP2 =	000000	232		
AMTYP3 =	000000	232		
AMTYP4 =	000000	232		
APASS =	000000	232		
APRIOR =	000000	232		
APYCSU =	000040	2537#		
APYENV =	000001	2313		
APYSIZ =	000200	789	2535#	2563
APYSPO =	000100	2494		
APYSZO =	003266	800	2565	827#
ASWREG =	000000	232		
ATESTN =	000000	232		
AUNIT =	000000	232		
AUSWP =	000400	1#		
DEJECT1 =	000300	1#		
DEJECT2 =	000000	1#		
DEJECT3 =	000000	1#		

894

2535# 2563

2565 827#

249

2470

CROSS REFERENCE TABLE -- USER SYMBOLS

EM113	021211	665	3351#
EM114	021247	670	3357#
EM115	021306	675	3363#
EM116	021345	680	3369#
EM117	021404	685	3375#
EM118	015620	339	3016#
EM119	015651	344	3021#
EM120	015704	349	3026#
EM121	015745	354	3032#
EM122	016010	359	3038#
EM123	016041	364	3043#
EM124	016312	369	2978#
EM125	016074	369	3048#
EM126	016135	374	3054#
EM127	016201	379	3061#
EM128	016236	384	3066#
EM129	016267	389	3071#
EM130	016322	394	3076#
EM131	016363	399	3082#
EM132	016426	404	3088#
EM133	015336	304	2982#
EM134	016465	409	3094#
EM135	016516	414	3099#
EM136	016551	419	3104#
EM137	016617	424	3111#
EM138	016661	429	3117#
EM139	016717	434	3123#
EM140	016747	439	3128#
EM141	017001	444	3133#
EM142	015403	309	2989#
EM143	017041	449	3139#
EM144	017057	454	3143#
EM145	017127	459	3149#
EM146	017161	464	3154#
EM147	017212	469	3159#
EM148	017246	474	3164#
EM149	017303	479	3170#
EM150	017303	484	3169#
EM151	015425	314	2993#
EM152	017337	489	3175#
EM153	017364	494	3179#
EM154	017414	499	3183#
EM155	017471	504	3191#
EM156	017515	509	3195#
EM157	017653	514	3213#
EM158	017553	519	3201#
EM159	017611	524	3207#
EM160	015462	319	2998#
EM161	017653	529	3214#
EM162	017702	534	3218#
EM163	017726	539	3222#
EM164	017773	544	3229#
EM165	020017	549	3233#
EM166	020043	554	3237#
EM167	020111	559	3244#
EM168	020147	564	3250#

