

DL11-C,D,E

OFF LINE TEST
MD-11-DZDLC-A

EP DZDLC A-DL A
COPYRIGHT 1976
FICHE 1 OF 1

NOV 1976
digital
MADE IN USA

DL11-C,D,E
MD-11-DZDLC-A
NOV 1976
FICHE 1 OF 1

E01

MAINDEC-11-DZDLC-A
DZDLC.A.P11

MACY11 27(732) 20-SEP-76 09:19 PAGE 4

154
155

PDP-11 FAMILY PROCESSOR WITH 8K OF MEMORY
M7800 DL11 ASYNCHRONOUS LINE INTERFACE MODULE

156
157
158
159
160
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

BC05C SPECIAL CABLE CONNECTOR
H315A SPECIAL MODEM TEST CONNECTOR

B. SOFTWARE REQUIREMENTS

THIS PROGRAM WAS SPECIFICALLY DESIGNED FOR THE 11/40 FRONT END OF THE 1080 CONSOLE PROCESSOR SYSTEM. IN THIS ENVIRONMENT IT WOULD BE LOADED BY THE TCDP (DECTAPE) DIAGNOSTIC MONITOR. HOWEVER, ANY 11/40 USER WITH 8K OF MEMORY CAN RUN THIS PROGRAM TO TEST ONE(1) OR MULTIPLE DL11'S.

THE PROGRAM HAS THE PROPER INTERFACE CODE TO ALLOW RUNNING UNDER THE AUTOMATED MANUFACTURING TEST LINE SYSTEM - ACT11.

3.0 RELATED DOCUMENTS AND STANDARDS

- A. PROGRAMMING PRACTICES - DOCUMENT NO. 175-003-009-00
- B. PDP11/40 PROCESSOR HANDBOOK
- C. DL11 ASYNCHRONOUS LINE INTERFACE MANUAL
DOCUMENT NO. DEC-11-HDLAA
- D. PDP-11 MAINDEC SYSMAC' UTILITY PACKAGE
- E. MAINDEC-11-DZQAC-A1
APPLICABLE CIRCUIT SCHEMATIC
M7800

4.0 DIAGNOSTIC HIERARCHY PREREQUISITES

BEFORE RUNNING THIS PROGRAM, THE FOLLOWING TWO(2) DIAGNOSTIC PROGRAMS SHOULD BE RUN FOR VERIFICATION OF FUNCTIONALITY OF THE 11-INSTRUCTION SET AND MEMORY:

- 1. MAINDEC-11-DBQEA AND,
- 2. MAINDEC-11-DZQMC

5.0 LOADING AND STARTING PROCEDURE

LOAD PROGRAM IN MEMORY USING ABS LOADER
LOAD ADDRESS 200.

NOTE

IN THE CASE OF A 1080 SYSTEM ENVIRONMENT
LOAD THE PROGRAM USING THE TCDP
(DECTAPE) DIAGNOSTIC MONITOR.

PRESS START.

GO1

MAINDEC-11-DZDLC-A
DZDLC.A.P11

MACY11 27(732) 20-SEP-76 09:19 PAGE 6

212
213

A. THERE ARE ALSO THREE(3) OPTIONAL START ADDRESSES FOR THE

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

PROGRAM:

- 204 - SELECTS PROGRAM #2
- 210 - SELECTS PROGRAM #3
- 214 - SELECTS PROGRAM #4
- 220 - SELECTS PROGRAM #5

6.0 SPECIAL ENVIRONMENTS

IF THIS PROGRAM IS RUN IN QUICK VERIFY MODE UNDER ACT11 THE PROGRAM IS DONE AFTER THE FIRST PASS.

7.0 PROGRAM OPTIONS

SWITCH -----	USE ---
15=1 OR UP	HALT ON ERROR
14=1 OR UP	LOOP ON TEST
13=1 OR UP	INHIBIT ERROR TYPEOUTS
12=1 OR UP	/C OR /D MODEL BEING TESTED
11=1 OR UP	INHIBIT ITERATIONS
10=1 OR UP	BELL ON ERROR
9=1 OR UP	LOOP ON ERROR
8=1 OR UP	LOOP ON TEST IN SWR<7:0>
<7:0>	HOLDS TEST NO. OF TEST TO BE LOOPED ON. USED IN CONJUNCTION WITH SW<8>
0=1 OR UP	USED IN DEVICE TABLE CREATION (1 TO 16 DEVICES) I.E., DEFAULT DEVICE NOT DESIRED. ALSO USED FOR CHARACTER LENGTH SETTING.
	!! NOTE WELL !!

IF SW<08> IS SET THE USER CAN ONLY 'LOOP ON A TEST' OF THE DEFAULT DEVICE I.E. - DL11/E RCSR = 775610. IF THE USER DESIRES TO 'LOOP ON A TEST' OF OTHER THAN THE DEFAULT DEVICE HE MUST FIRST PATCH THE FIVE (5) LOCATIONS LABELED

DLRCSR: DLRDBR: DLXCSR: DLXDBR:
DLVECT:

THAT APPEAR UNDER 'DL11 DEFINITIONS' HEADING AT THE FRONT OF THE LISTING. I.E. - WITH SW<08> SET SW<00> IS NOT FUNCTIONAL.

MAINDEC-11-DZDLC-A
DZDLCR.F11

MACY11 27(732) 20-SEP-76 09:19 PAGE 8

270
271

8.0 EXECUTION TIMES

[Handwritten mark]

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
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327

EXECUTION TIME IS DEPENDENT ON TYPE OF MEMORY AND NUMBER OF DL11'S BEING TESTED. A REPRESENTATIVE TIME FOR 1 ERROR FREE PASS IS:

11/40 - CORE MEMORY - 1 DL11/E - 20 SECONDS

9.0 ERROR INFORMATION

9.1 ERROR REPORTING

THERE ARE A TOTAL OF SEVEN(7) TYPES OF ERROR REPORTS GENERATED BY THE PROGRAM. THE KEY COLUMN HEADINGS WILL BE DESCRIBED BELOW FOR CLARITY -

DEVADR - THIS IS THE ADDRESS OF THE RECEIVER CONTROL STATUS REGISTER FOR THE FAILING DL11

REGADR - THIS IS THE ADDRESS OF THE DL11 REGISTER ON WHICH TESTING IS BEING CONDUCTED

WAS - THIS IS WHAT THE CONTENTS OF THE REGISTER OF THE DL11 UNDERGOING TEST WAS (ADDRESS IS UNDER COLUMN '(R2)')

S/B - THIS IS WHAT THE CONTENTS OF THE REGISTER OF THE DL11 UNDERGOING TEST SHOULD BE (ADDRESS IS UNDER COLUMN '(R2)')

WASADR - THIS IS WHAT THE MEMORY ADDRESS WAS (INPUT DATA BUFFER ADDRESS)

SHBADR - THIS IS WHAT THE MEMORY ADDRESS SHOULD BE (OUTPUT DATA BUFFER ADDRESS)

(REG) - THIS IS THE CONTENTS OF THE DL11 RECEIVER DATA BUFFER IN ERROR (ADDRESS IS UNDER COLUMN '(R2)')

9.2 ERROR HALTS

WITH THE 'HALT ON ERROR' SWITCH (SW15) NOT SET THERE ARE FOUR(4) PROGRAMMED 'HALTS' IN THE PROGRAM:

A. IN THE CASE OF ERROR REPORTING AND THERE IS NO TERMINAL

K01

MAINDEC-11-DZDLC-A
DZDLC.A.P11

MACY11 27(732) 20-SEP-76 09:19 PAGE 10

328

TO ALLOW THE INFORMATION TRANSFER.

MO1

MAINDEC-11-DZDLC-A
DZDLCR.F11

MACY11 27(732) 20-SEP-76 09:19 PAGE 12

385
386

OF CURRENT TRANSMISSION OR
AN ERROR CONDITION.

- BIT11 RECEIVER ACTIVE
 1. WHEN SET INDICATES RECEIVER INTERFACE IS ACTIVE.
 2. CLEARED BY INIT OR RCVR DONE (BIT07).

- BIT10 SECONDARY RECEIVE OR SUPERVISORY RECEIVED DATA
 1. PROVIDES RECEIVE CAPABILITY, WHEN SET, FOR REVERSE CHANNEL OF REMOTE STATION. SETS WHEN BIT03 IS SET.
 2. CLEARED BY INIT

- BIT07 RECEIVER DONE
 1. SETS WHEN CHARACTER HAS BEEN RECEIVED. WILL INITI

%

167400

```
.NLIST CND,MD,MC
.LIST ME,SEQ,BIN
$SWR=167400
.MCALL .HEADER,.$SCMTAG,.$SETUP,.$ERROR
.MCALL .SWRHI,.$SWRLO,.$SCATCH,.$EQUATE,.$SETUP,.$SEOP
.MCALL .SCOPE,.$ERRTYP,.$STYPOCT,.$STYPDEC,.$STYPE,.$POWER,.$STRAP
.MCALL .SRDOCT,.$SREAD,PUSH,POP,STARS,.$SRDOEC
.ENABLE ABS
```

000001

```
.TITLE MAINDEC-11-DZDLC-A
;*COPYRIGHT (C) 1975
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY E. CROWLEY/B. BURGESS
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
;*
$TN=1
```

```
.SBTTL OPERATIONAL SWITCH SETTINGS
*
* SWITCH USE
* -----
* 15 HALT ON ERROR
* 14 LOOP ON TEST
* 13 INHIBIT ERROR TYPEOUTS
* 12 /C OR /D MODEL
* 11 INHIBIT ITERATIONS
* 10 BELL ON ERROR
* 9 LOOP ON ERROR
* 8 LOOP ON TEST IN SWR<7:0>
```

000000
 000001
 000002
 000003
 000004
 000005
 000006
 000007
 000008
 000009
 000010
 000011
 000012
 000013
 000014
 000015
 000016
 000017
 000018
 000019
 000020
 000021
 000022
 000023
 000024
 000025
 000026
 000027
 000028
 000029
 000030
 000031
 000032
 000033
 000034
 000035
 000036
 000037
 000038
 000039
 000040
 000041
 000042
 000043
 000044
 000045
 000046
 000047
 000048
 000049
 000050
 000051
 000052
 000053
 000054
 000055
 000056
 000057
 000058
 000059
 000060
 000061
 000062
 000063
 000064
 000065
 000066
 000067
 000068
 000069
 000070
 000071
 000072
 000073
 000074
 000075
 000076
 000077
 000078
 000079
 000080
 000081
 000082
 000083
 000084
 000085
 000086
 000087
 000088
 000089
 000090
 000091
 000092
 000093
 000094
 000095
 000096
 000097
 000098
 000099
 000100

635	001224	000000	\$TMP11: .WORD	0	:: USER DEFINED
636	001226	000000	\$TMP12: .WORD	0	:: USER DEFINED
637	001230	000000	\$TMP13: .WORD	0	:: USER DEFINED
638	001232	000000	\$TMP14: .WORD	0	:: USER DEFINED
639	001234	000000	\$TMP15: .WORD	0	:: USER DEFINED
640	001236	000000	\$TMP16: .WORD	0	:: USER DEFINED
641	001240	000000	\$TMP17: .WORD	0	:: USER DEFINED
642	001242	000000	\$TIMES: 0		:: MAX. NUMBER OF ITERATIONS
643	001244	000000	\$ESCAPE: 0		:: ESCAPE ON ERROR ADDRESS
644	001246	177607 000377	\$BELL: .ASCIZ	<207><377><377>	:: CODE FOR BELL
645	001252	077	\$QUES: .ASCII	/?	:: QUESTION MARK
646	001253	015	\$CRLF: .ASCII	<15>	:: CARRIAGE RETURN
647	001254	000012	\$LF: .ASCIZ	<12>	:: LINE FEED
648			:*****		
649			:THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO		
650			: 'MORETAGS' AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .SCMTAG		
651					
652					
653	001256	000000	TABFLG: .WORD	0	: AN INDICATOR TO SHOW THAT THE
654					: INFORMATION FOR MULTIPLE DEVICE
655					: TESTING HAS ALREADY TRANSPIRED
656					: & 'MAINDEC' NAME HAS BEEN PRINTED
657	001260	000000	DLBASE: .WORD	0	: STORAGE & WORKING LOCATION FOR A DEVICE
658					: RECEIVER STATUS REGISTER ADDRESS
659	001262	000000	KEEPPAD: .WORD	0	: STORAGE LOCATION FOR THE 1ST
660					: DEVICE RCSR FROM WHICH
661					: 'BASEADD' IS RESTORED AT THE
662					: END OF A COMPLETE PROGRAM PASS.
663	001264	000000	BASEADD: .WORD	0	: STORAGE LOCATION WHICH HOLDS
664					: THE RCSR ADDRESS OF THE 'NEXT'
665					: DEVICE DURING MULTIPLE TESTING
666	001266	000000	KEEPIV: .WORD	0	: STORAGE LOCATION FOR THE 1ST
667					: DEVICE RECEIVER VECTOR FROM
668					: WHICH 'BASEIV' IS RESTORED AT THE
669					: END OF A COMPLETE PROGRAM PASS
670	001270	000000	BASEIV: .WORD	0	: STORAGE LOCATION WHICH HOLDS
671					: THE VECTOR ADDRESS OF THE 'NEXT'
672					: DEVICE DURING MULTIPLE TESTING
673	001272	000000	MULTD: .WORD	0	: FLAG TO INDICATE TO 'END OF PASS'
674					: ROUTINE THAT MULTIPLE DEVICE
675					: TESTING IS BEING CONDUCTED
676					: 0=NO, 1=YES
677	001274	000000	ACTREG: .WORD	0	: THIS IS THE DEVICE ACTIVE REGISTER
678					: A BIT IS SET (STARTING AT
679					: BIT) FOR EACH CONTIGUOUS DEVICE
680					: (A MAX. OF 16) THAT IS TO UNDERGO
681					: TESTING. THIS LOCATION IS
682					: AUTOMATICALLY FILLED BASED ON
683					: USER RESPONSE TO PROGRAM QUESTIONS
684	001276	000000	ROTADD: .WORD	0	: A ROTATING POINTER TO SIGNAL
685					: THE LAST DEVICE TESTED (IF
686					: MULTIPLE DEVICE TESTING WAS BEING
687					: DONE) IF LESS THAN A FULL COMPLE-
688					: MENT OF DEVICES (16) WAS SELECTED
689	001300	000000	LASTADD: .WORD	0	: STORAGE LOCATION FOR THE
690					: RCSR ADDRESS OF THE LAST DEVICE

691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708

001302 000000

DLPRI: .WORD 0

001304 000000

LESS1: .WORD 0

001306 177740

STLMSK: 177740

:TESTED (IF MULTIPLE DEVICE
:TESTING WAS SELECTED BY USER)
:STORAGE LOCATION FOR THE DEVICE
:INTERRUPT PRIORITY LEVEL
:THE PRIORITY LEVEL THE CPU
:MUST BE AT TO ALLOW DEVICE INTERRUPTS.
:THIS WILL BE 1 LEVEL LESS THAN
:THE DEVICE LEVEL (BASED ON &
:CALCULATED FROM USER RESPONSE TO
:DEVICE PRIORITY LEVEL QUESTION)
:THIS MASK IS USED BY THE 'STALL'
:ROUTINE WHICH WAITS A RANDOM NO.
:OF MILLISECONDS. ITS' USE PREVENTS
:A STALL > 37 MSEC. THIS LOCATION
:HOWEVER, CAN BE PATCHED BY THE
:USER TO ALLOW LARGER 'STALLS'.

:END OF USER SUPPLIED TAG(S)

709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764

001310

001310 015132
001312 015201
001314 015260
001316 000000

001320 015276
001322 015322
001324 015420
001326 000000

001330 015442
001332 015472
001334 015570
001336 000000

001340 015612
001342 015664
001344 015722
001346 000000

001350 015734
001352 016011
001354 016100
001356 000000

001360 015132
001362 016120
001364 016160

.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:

;ERROR TABLE ITEM FOR ERROR MESSAGE 1

EM1 : "DL11 REGISTER REFERENCE CAUSED TIMEOUT"
DH1 : " (PC) (PS) (SP) TEST DEVADR REGADR "
DT1 : (R7) (PSW) (R6) (R0) (R1) (R2)
0 :PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 2

EM2 : " DL11 REGISTER ERROR "
DH2 : " (PC) (PS) (SP) TEST DEVADR REGADR WAS S/B "
DT2 : " (R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4) "
0 :PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 3

EM3 : " DL11 DATA COMPARE ERROR "
DH3 : " (PC) (PS) (SP) TEST WASADR SHBADR WAS S/B "
DT3 : " (R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4) "
0 :PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 4

EM4 : " UNEXPECTED TRAP TO VECTOR AT LOCATION XXX "
DH4 : " (PC) (PS) (SP) TEST "
DT4 : " (R7) (PSW) (R6) (R0) "
0 :PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 5

EM5 : " DL11 SOFT ERROR (PARITY, FRAMING, OR OVERRUN)
DH5 : " (PC) (PS) (SP) TEST DEVADR REGADR (REG) "
DT5 : " (R7) (PSW) (R6) (R0) (R1) (R2) (R3) "
0

;ERROR TABLE ITEM FOR ERROR MESSAGE 6

EM1 : "DL11 REGISTER REFERENCE CAUSED TIMEOUT"
DH6 : " (PC) (PS) (SP) REGADR"
DT6 : \$ERRPC, \$TMPO, \$REG6, \$REG2

```

765 001366 000000          0          ;PRINT ALL OCTAL
766
767          ;ERROR TABLE ITEM FOR ERROR MESSAGE 7
768
769 001370 015734          EMS          ;" DL11 SOFT ERROR (PARITY, FRAMING, OR OVERRUN) "
770 001372 016172          DH7          ;" (PC) DEVADR REGADR (REG)"
771 001374 016232          DT7          ;$ERRPC,$REG1,$REG2,$REG3
772 001376 000000          0          ;PRINT ALL OCTAL
773
774          ;ERROR TABLE ITEM FOR ERROR MESSAGE 10
775
776 001400 015442          EM3          ;" DL11 DATA COMPARE ERROR "
777 001402 016244          DH10         ;" (PC) DEVADR REGADR (REG) S/B"
778 001404 016312          DT10         ;$ERRPC,$REG1,$REG2,$REG3,$REG4
779 001406 000000          0          ;PRINT ALL OCTAL
780

```

```

;:*****
;:DL11 DEFINITIONS
;:*****

```

```

785 001410 175610          DLRCR: 175610          ;CONTAINS ADDRESS OF RCVR CSR
786 001412 175612          DLRDBR: 175612          ;CONTAINS ADDRESS OF RCVR DBR
787 001414 175614          DLXCSR: 175614          ;CONTAINS ADDRESS OF XMIT CSR
788 001416 175616          DLXDBR: 175616          ;CONTAINS ADDRESS OF XMIT DBR
789 001420 000300          DLVECT: 300          ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
790 001422 000000          XFLGO: 0          ;FLAG FOR HARD XMIT ERRORS
791 001424 000000          RFLGO: 0          ;FLAG FOR HARD RCVR ERRORS
792 001426 000000          RFLG1: 0          ;FLAG FOR SOFT RCVR ERRORS
793 001430 000000          RTRY: 0          ;COUNTS NO. OF RETRIES ON SOFT ERRORS
794 001432 000000          OPTR: 0          ;CONTAINS POINTER TO OUTPUT BUFFER
795 001434 000000          IPTR: 0          ;CONTAINS POINTER TO INPUT BUFFER
796 001436 000000          LDOUT: 0          ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
797 001440 000000          TIMR1: 0          ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
798 001442 000000          TIMR2: 0
799 001444 000000          INTFLG: 0          ;SOFTWARE INTR. FLAG
800
801
802
803

```

```

804 001446 000240          BEGIN: NOP          ;PROGRAM WILL START HERE
805          .SBTTL INITIALIZE THE COMMON TAGS
806          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
807 001450 012706 001100          MOV #CMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
808 001454 005026          CLR (R6)+          ;;CLEAR MEMORY LOCATION
809 001456 022706 001140          CMP #SWR,R6 ;;DONE?
810 001462 001374          BNE -6          ;;LOOP BACK IF NO
811 001464 012706 001100          MOV #STACK,SP          ;;SETUP THE STACK POINTER
812          ;;INITIALIZE A FEW VECTORS
813 001470 012737 010462 000020          MOV #SSCOPE,@#IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
814 001476 012737 000340 000022          MOV #340,@#IOTVEC+2 ;; LEVEL 7
815 001504 012737 010732 000030          MOV #SERROR,@#EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
816 001512 012737 000340 000032          MOV #340,@#EMTVEC+2 ;; LEVEL 7
817 001520 012737 013052 000034          MOV #STRAP,@#TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
818 001526 012737 000340 000036          MOV #340,@#TRAPVEC+2 ;; LEVEL 7
819 001534 012737 013132 000024          MOV #SPWRDN,@#PWRVEC ;; POWER FAILURE VECTOR
820 001542 012737 000340 000026          MOV #340,@#PWRVEC+2 ;; LEVEL 7

```

```

821 001550 005067 177466 CLR $TIMES ;: INITIALIZE NUMBER OF ITERATIONS
822 001554 005067 177464 CLR $ESCAPE ;: CLEAR THE ESCAPE ON ERROR ADDRESS
823 001560 112767 000001 177327 MOV#B #1,$ERMAX ;: ALLOW ONE ERROR PER TEST
824 001566 012767 001566 177312 MOV #.,$LPADR ;: INITIALIZE THE LOOP ADDRESS FOR SCOPE
825 001574 012767 001574 177305 MOV #.,$LPERR ;: SETUP THE ERROR LOOP ADDRESS
826 ;: SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
827 ;: EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
828 001602 013746 000004 MOV @#ERRVEC, -(SP) ;: SAVE ERROR VECTOR
829 001606 012737 001642 000004 MOV #64,$@#ERRVEC ;: SET UP ERROR VECTOR
830 001614 012767 177570 177316 MOV #DSWR,SWR ;: SETUP FOR A HARDWARE SWICH REGISTER
831 001622 012767 177570 177312 MOV #DDISP,DISPLAY ;: AND A HARDWARE DISPLAY REGISTER
832 001630 022777 177777 177302 CMP #-1,@SWR ;: TRY TO REFERENCE HARDWARE SWR
833 001636 001012 BNE 66$ ;: BRANCH IF NO TIMEOUT TRAP OCCURRED
834 ;: AND THE HARDWARE SWR IS NOT = -1
835 001640 000403 BR 65$ ;: BRANCH IF NO TIMEOUT
836 001642 012716 001650 64$: MOV #65$,(SP) ;: SET UP FOR TRAP RETURN
837 001646 000002 RTI
838 001650 012767 000176 177262 65$: MOV #SWREG,SWR ;: POINT TO SOFTWARE SWR
839 001656 012767 000174 177256 MOV #DISPREG,DISPLAY
840 001664 012637 000004 66$: MOV (SP)+,@#ERRVEC ;: RESTORE ERROR VECTOR
841
842 001670 005067 177376 CLR MULTD ;: CLEAR MULTIPLE DEVICE
843 ;: TESTING FLAG
844 001674 005067 177356 CLR TABFLG ;: CLEAR TABLE CREATION FLAG
845 001700 012767 000010 177326 MOV #8.,$TMP15 ;: SET CHARACTER LENGTH DESIGNATOR
846 ;: FOR 8 BITS --- THIS IS THE DEFAULT
847 ;: LENGTH ASSUMED BY THE PROGRAM
848 ;: UNLESS THE USER CHANGES IT THRU
849 ;: THE QUESTION AND ANSWER CYCLE
850 ;: INITIATED BY SETTING SW<0> TO A 1
851 001706 012767 000200 177366 MOV #200,DLPRI ;: SET STANDARD PRIORITY LEVEL
852 ;: FOR DEVICE
853 001714 032767 000400 177216 BIT #SWB,SWR ;: IS THE 'LOOP ON TEST' SWITCH SET?
854 001722 001411 BEQ 1$ ;: BRANCH IF NOT
855
856 ;: IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
857 ;: INSTRUCTION THUS BYPASSING TABLE CREATION
858
859 ;: IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
860 ;: THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
861 ;: WITH THE DESIRED DEVICE REGISTER VALUES:
862
863 *****
864 UNDER ;DL11 DEFINITIONS ABOVE
865 *****
866
867 DLRCR: PATCH THE ADDRESS OF THE RCVR CSR
868 DLRDBR: PATCH THE ADDRESS OF THE RCVR DBR
869 DLXCSR: PATCH THE ADDRESS OF THE XMIT CSR
870 DLXDBR: PATCH THE ADDRESS OF THE XMIT DBR
871 DLVECT: PATCH THE VECTOR ADDRESS OF THIS DL11
872
873 001724 104401 016635 TYPE, STMES ;: PRINT OUT 'MAINDEC' NAME
874 001730 104401 020735 TYPE, FAILSA ;: TYPE FAILSAFE MESSAGE
875 001734 104000 ERROR +0 ;: TYPE OUT THE PC VALUE
876 001736 104401 021313 TYPE, PCMSG ;: FOLLOWED BY =PC

```

```

877 001742 000000          HALT                ;WAIT FOR USER TO RESPOND
878 001744 000443          BR          ONCE          ;GO TO TEST DEVICE PATCHED IN BY USER
879 001746
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906 002026 105767 177224
907 002032 001010
908
909 002034 104401 016635
910
911 002040 105167 177212
912
913 002044 032767 000001 177066
914
915
916
917 002052 001012
918 002054 005077 177334
919 002060 005077 177324
920 002064 005777 177322
921 002070 005777 177316
922 002074 000167 000670
923
924
925
926
927
928
929
930
931 002100 104401 017217
932

```

```

          ;ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
          ;AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
          ;WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
          ;TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
          ;HE CAN RUN THE DEFAULT DEVICE (1ST RECEIVER
          ;STATUS REGISTER ADDRESS 175610)
          ;NOTE: IF THIS IS NOT SUITABLE THE USER WILL
          ;HAVE TO SET SWO=1 (OR UP) IN ORDER TO
          ;RECREATE THE TABLE HE DESTROYED FROM
          ;ABOVE
          MOV      #175610,DLBASE ;1ST POSSIBLE RECEIVER CSR
          JSR     PC,DLADDR      ;FORM DL ADDRESSES FOR
          ;1ST POSSIBLE DEVICE
          MOV      #300,DLVECT   ;1ST POSSIBLE INTERRUPT VECTOR
          CLR     TABFLG        ;CLEAR TABLE CREATION FLAG
RESTRT: MOV      #STACK,SP      ;SET UP STACK POINTER
          MOV      #BUSERR,@#ERRVEC ;SET UP BUS ERROR VECTOR
          MOV      #340,@#ERRVEC+2
          MOV      #RSVERR,@#RESVEC ;SET UP RSVD INSTR. VECTOR
          MOV      #340,@#RESVEC+2
          ;THIS NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
          ;WILL TAKE PLACE I.E.-
          ;A) HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
          ;B) IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
          TSTB   TABFLG        ;HAS TABLE CREATION BEEN PERFORMED?
          BNE    ONCE          ;BRANCH IF YES TO SKIP 'MAINDEC
          ;TITLE' MESSAGE
          TYPE   ,STMES        ;OTHERWISE, PRINT OUT 'MAINDEC'
          ;NAME
          COMB   TABFLG        ;IF TABLE CREATION HAS NOT BEEN
          ;PERFORMED, THEN SET FLAG, AND DO SO
          BIT    #SWO,SWR      ;THE PROGRAM HAS OBVIOUSLY BEEN
          ;RESTARTED - DOES USER WISH TO
          ;RESELECT VECTOR AND CONTROL REGISTER
          ;ADDRESSES I.E. - CREATE A NEW TABLE?
          BNE    GO            ;BRANCH IF YES
          ONCE: CLR     @DLXCSR  ;CLEAR OUT BOTH CSR'S
          CLR     @DLRCSR
          TST    @DLRDBR      ;FLUSH RCVR "DONE" BIT
          TST    @DLRDBR
          JMP    TST1         ;OTHERWISE, GO WITH EXISTING
          ;TABLE OR NOT USE ANY TABLE AT
          ;ALL WHICHEVER THE CASE MAY BE
          ;DEFAULT CASE IS 1ST POSSIBLE
          ;DEVICE)
          ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES:
          ;A) TO RUN MULTIPLE DEVICES
          ;B) TO CREATE A NEW TABLE TO RUN FROM, OR
          ;C) TO CHANGE THE CHARACTER LENGTH
          GO:   TYPE,   LENGTH ;ASK USER FOR THE CHARACTER LENGTH
          ;FOR WHICH HIS DEVICE IS SET

```



```

933 002104 104411          RUDEC          ;ACCEPT THE ANSWER TYPED BY USER
934          ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
935 002106 012600          MOV          (SP)+,RO          ;GET THE ANSWER TYPED
936 002110 020027 000010  CMP          RO,#8.          ;IS THE NUMBER TOO HIGH?
937 002114 101114          BHI          RETRY          ;IF YES - GO TO RETRY SITUATION
938 002116 020027 000005  CMP          RO,#5.          ;IS THE NUMBER TOO LOW?
939 002122 103511          BLO          RETRY          ;IF YES - GO TO RETRY SITUATION
940 002124 010067 177104  MOV          RO,$TMP15       ;THE VALUE TYPED IS OK
941          ;STORE FOR FUTURE USE
942 002130 104401 017301  TYPE,        DEFAULT       ;ASK USER IF HE WISHES TO TEST OTHER
943          ;THAN THE DEFAULT DEVICE
944 002134 104410          RDOCT          ;ACCEPT THE ANSWER TYPED BY USER
945 002136 005726          TST          (SP)+          ;LOOK AT THE ANSWER
946 002140 001002          BNE          1$            ;BRANCH IF REPLY WAS YES
947 002142 000137 002724  JMP          2$#FLUSH       ;OTHERWISE, SKIP REST OF INTERROGATION
948 002146 012700 000300  1$:          MOV          #300,RO       ;START RESTORATION OF TRAPCATCHER
949 002152 012701 000302  MOV          #302,R1        ;AREA FROM LOCATIONS 300 TO 776
950 002156 012702 000004  MOV          #4,R2          ;SO THAT WE CREATE THE MULTIPLE
951 002162 010110 2$:          MOV          R1,(RO)        ;DEVICES TABLE WITH A CLEAN SLATE
952 002164 005011          CLR          (R1)          ;
953 002166 060200          ADD          R2,RO         ;
954 002170 50201          ADD          R2,R1         ;
955 002172 022701 001000  CMP          #1000,R1       ;
956 002176 002771          BLT          2$            ;
957          ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
958          ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
959          ;ADDRESS
960 002200 104401 017406  FIRSTD: TYPE ,MFIRSTD      ;ASK USER FOR THE RECEIVER CONTROL
961          ;REGISTER ADDRESS OF HIS FIRST
962          ;DEVICE
963 002204 104410          RDOCT          ;ACCEPT THE ANSWER TYPED BY USER
964          ;AND STORE ON TOP OF STACK
965          ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
966 002206 012600          MOV          (SP)+,RO       ;GET THE ANSWER TYPED
967 002210 020027 176170  CMP          RO,#176170     ;IS THE NUMBER TOO HIGH?
968 002214 101060          BHI          RETRY         ;IF YES-GO TO RETRY SITUATION
969 002216 020027 175610  CMP          RO,#175610     ;IS THE NUMBER TOO LOW?
970 002222 103455          BLO          RETRY         ;IF YES - GO TO RETRY SITUATION
971 002224 132700 000001  BITB        #BIT0,RO       ;NUMBER IS IN RANGE BUT IS IT
972          ;ON AN EVEN BOUNDARY?
973 002230 001052          BNE          RETRY         ;IF NO - GO TO RETRY SITUATION
974          ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
975 002232 032700 000007  BIT          #7,RO         ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
976          ;USER RESPONSE EQUAL TO A ZERO?
977 002236 001047          BNE          RETRY         ;BRANCH IF NOT
978 002240 010067 177014  MOV          RO,DLBASE      ;THE 1ST ADDRESS VALUE TYPED IS OK
979          ;STORE FOR FUTURE USE
980          ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
981 002244 016767 177010 177010  MOV          DLBASE,KEEPADD ;GET 1ST ADDRESS VALUE
982 002252 004767 005576  JSR          PC,DLADDR     ;GO FORM DL ADDRESSES FOR
983          ;1ST DEVICE SELECTED
984 002256 016767 177000 177000  MOV          KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
985 002264 104401 017474  VECT:       TYPE ,MVECT    ;ASK USER FOR A VECTOR ADDRESS
986 002270 104410          RDOCT          ;ACCEPT THE ANSWER TYPED BY USER
987          ;AND STORE ON TOP OF STACK
988          ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS

```

989	002272	012600			MOV	(SP)+,RO		;GET THE ANSWER TYPED
990	002274	020027	000776		CMP	RO,#776		;IS THE NUMBER TOO HIGH?
991	002300	101032			BHI	RETRY1		;IF YES - GO TO RETRY SITUATION
992	002302	020027	000300		CMP	RO,#300		;IS THE NUMBER TOO LOW?
993	002306	103427			BLO	RETRY1		;IF YES - GO TO RETRY SITUATION
994	002310	132700	000001		BITB	#BIT0,RO		;NUMBER IS IN RANGE BUT IS IT
995								;ON AN EVEN BOUNDARY?
996	002314	001024			BNE	RETRY1		;IF NO - GO TO RETRY SITUATION
997								;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
998	002316	032700	000007		BIT	#7,RO		;WAS THE LEAST SIGNIFICANT DIGIT OF THE
999								;USER RESPONSE EQUAL TO A ZERO?
1000	002322	001021			BNE	RETRY1		;BRANCH IF NOT
1001	002324	010067	177070		MOV	RO,DLVECT		;THE VECTOR VALUE TYPED IS OK
1002								;STORE FOR FUTURE USE
1003	002330	016767	177064	176730	MOV	DLVECT,KEEP1V		;GET THE FIRST VECTOR VALUE
1004	002336	016767	177056	176724	MOV	DLVECT,BASE1V		;SAVE FIRST VECTOR VALUE
1005	002344	000414			BR	HOWMANY		;GO TO SEE IF USER WANTS MORE
1006								;THAN 1 DEVICE
1007	002346	104401	001252		RETRY:	TYPE, \$QUES		;TYPE '?' INDICATING USER TYPED
1008								;SOMETHING WRONG FOR CHARACTER LENGTH
1009	002352	000167	177522		JMP	GO		;GO BACK TO REISSUE QUESTION
1010	002356	104401	001252		RETRY0:	TYPE , \$QUES		;TYPE '?' INDICATING USER TYPE
1011								;SOMETHING WRONG FOR 1ST ADDRESS
1012	002362	000167	177612		JMP	FIRSTD		;GO BACK TO REISSUE QUESTION
1013	002366	104401	001252		RETRY1:	TYPE , \$QUES		;TYPE '?' INDICATING USER TYPED
1014								;SOMETHING WRONG FOR VECTOR
1015	002372	000167	177666		JMP	VECT		;GO BACK TO REISSUE QUESTION
1016	002376	104401	017552		HOWMANY:	TYPE ,MULDEV		;ASK USER IF HE WISHES TO RUN
1017								;MULTIPLE DEVICES
1018	002402	104410			RDOCT			;ACCEPT THE ANSWER TYPED BY USER
1019								;AND STORE ON TOP OF STACK
1020	002404	012600			MOV	(SP)+,RO		;GET THE ANSWER TYPED
1021	002406	005700			TST	RO		;WAS THE ANSWER YES?
1022	002410	001003			BNE	1\$;BRANCH IF IT WAS
1023	002412	005067	176654		CLR	MULTD		;OTHERWISE, INITIALIZE FLAG TO
1024								;INDICATE NON-MULTIPLE DEVICES
1025	002416	000402			BR	2\$;SKIP NEXT INSTRUCTION
1026	002420	105167	176646		1\$:	COMB MULTD		;INITIALIZE FLAG TO INDICATE
1027								;RUNNING OF MULTIPLE DEVICES
1028	002424				2\$:			
1029	002424	105767	176642		TSTB	MULTD		;ARE THERE MULTIPLE DEVICES ON
1030								;THE SYSTEM?
1031	002430	100406			BMI	LASTD		;IF SO, GO TO ASK NEXT QUESTION
1032	002432	005067	176636		CLR	ACTREG		;CLEAR DEVICE ACTIVE FLAG TO
1033								;INDICATE NO RUNNING OF MULTIPLE
1034								;DEVICES
1035	002436	005067	176634		CLR	ROTADD		;CLEAR DEVICE ADDRESS POINTER IN
1036								;USE WHEN RUNNING MULTIPLE DEVICES
1037	002442	000167	000160		JMP	CONQUES		;SKIP ASKING NEXT QUESTION
1038	002446							
1039					LASTD:			;WE WILL NOW BEGIN TO SET UP THE DEVICE ACTIVE REGISTER FOR RUNNING
1040								;MULTIPLE DL11 DEVICES
1041	002446	104401	017637		TYPE	,MLASTD		;ASK USER FOR THE RECEIVER
1042								;CONTROL REGISTER ADDRESS OF
1043								;HIS LAST DEVICE
1044	002452	104410			RDOCT			;ACCEPT THE ANSWER TYPED BY

```

1045                                     ;USER AND STORE ON TOP OF STACK
1046                                     ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1047 002454 012600 1$: MOV (SP)+,RO ;GET THE ANSWER TYPED
1048 002456 020027 176170 CMP RO,#176170 ;IS THE NUMBER TOO HIGH?
1049 002462 101132 BHI RETRY2 ;IF YES - GO TO RETRY SITUATION
1050 002464 020027 175610 CMP RO,#175610 ;IS THE NUMBER TOO LOW?
1051 002470 103527 SLO RETRY2 ;IF YES - GO TO RETRY SITUATION
1052 002472 132700 000001 BITB #BIT0,RO ;NUMBER IS IN RANGE BUT IS IT
1053                                     ;ON AN EVEN BOUNDARY?
1054 002476 001124 BNE RETRY2 ;IF NOT - GO TO RETRY SITUATION
1055                                     ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1056 002500 032700 000007 BIT #7,RO ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1057                                     ;USER RESPONSE EQUAL TO A ZERO?
1058                                     BNE RETRY2 ;BRANCH IF NOT
1059 002506 010067 176566 MOV RO, LASTADD ;THE LAST ADDRESS VALUE TYPED IS OK
1060                                     ;STORE FOR FUTURE USE
1061                                     ;NOW WE BEGIN TO ACTUALLY INITIALIZE THE DEVICE ACTIVE REGISTER
1062                                     ;FROM WHICH THE PROGRAM WILL CYCLE UNTIL ALL DEVICES HAVE BEEN TESTED
1063 002512 012767 000001 176556 MOV #1, ROTADD ;SET UP POINTER FOR 'ACTREG'
1064 002520 005067 176550 CLR ACTREG ;CLEAR DEVICE ACTIVE REGISTER
1065 002524 056767 176546 176542 2$: BIS ROTADD, ACTREG ;MAKE 1ST DEVICE ACTIVE
1066 002532 000241 CLC ;CLEAR CARRY BIT FOR POINTER
1067                                     ;ROTATION
1068 002534 006167 176536 ROL ROTADD ;ARE WE PAST 16 LINE RANGE?
1069 002540 103422 BCS 3$ ;BRANCH IF YES
1070 002542 062767 000010 176514 ADD #10, BASEADD ;STEP UP BASE ADDRESS
1071 002550 026767 176524 176506 CMP LASTADD, BASEADD ;IS THIS THE LAST DEVICE?
1072 002556 101362 BHI 2$ ;BRANCH IF NOT
1073                                     ;NOTE: IF THIS PATH IS TAKEN IT IS ASSUMED THAT AT LEAST 2 DEVICES
1074                                     ;EXIST AND THAT ALL ADDRESSING IS CONTIGUOUS
1075 002560 056767 176512 176506 BIS ROTADD, ACTREG ;INDICATE NEXT DEVICE ACTIVE
1076 002566 012767 000001 176502 MOV #1, ROTADD ;RESET POINTER FOR 'ACTREG' FOR
1077                                     ;LATER USE IN END OF PASS ROUTINE
1078 002574 016767 176462 176462 MOV KEEPADD, BASEADD ;RESET 1ST DEVICE RECEIVER
1079                                     ;CONTROLLER REGISTER ADDRESS FOR
1080                                     ;LATER USE IN END OF PASS ROUTINE
1081 002602 000167 000020 JMP CONQUES ;GO TO CONTINUE QUESTIONING OF USER
1082 002605
1083                                     ;IF WE TAKE THIS PATH IT APPEARS THAT THERE ARE NOT AT LEAST
1084                                     ;TWO DEVICES PRESENT - IN RESPONSE TO USER TYPING 'YES' TO MULTIPLE
1085                                     ;DEVICES QUESTION
1086 002606 016767 176450 176450 MOV KEEPADD, BASEADD ;RESET 1ST DEVICE RECEIVER
1087                                     ;CONTROLLER REGISTER ADDRESS
1088 002614 104401 017735 TYPE ,MRANGE ;INFORM USER TO CHECK AND RETYPE
1089                                     ;THE LAST DEVICE RCSR ADDRESS
1090 002620 104410 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1091                                     ;AND STORE ON TOP OF STACK
1092 002622 000167 177626 JMP 1$
1093 002626
1094 CONQUES:
1095 ;IF WE HAVE REACHED THIS PORTION WE KNOW:
1096 ; A) THE 'RXCSR' ADDRESS OF THE 1ST DEVICE
1097 ; B) THE 'RXCSR' ADDRESS OF THE LAST DEVICE, SAND
1098 ; C) THE INTERRUPT VECTOR OF THE 1ST DEVICE
1099 ;NOW LET'S FIND THE PRIORITY LEVEL
1100 002626 104401 020021 TYPE ,PLEVEL ;ASK USER FOR PRIORITY LEVEL
1101 002632 104410 RDOCT ;ACCEPT ANSWER TYPED BY USER AND
  
```



```

11101 003044 012767 003062 174732      MOV      #1$,ERRVEC      :GO TO 1$ IF TIMEOUT
11102 003052 016702 176336      MOV      DLXCSR,R2      :REGADR = XCSR ADR
11103 003056 005712      TST      (R2)           :USE REGADR ON BUS
11104 003060 000407      BR       3$            :<GO TO NEXT TEST IF NO TIMEOUT>
11105 003062 004767 011030 1$:      JSR      PC,SUERT1      :GO SET UP ERROR INFO
11106 003066 012767 003076 176150      MOV      #2$, $ESCAPE   :RETURN TO 2$ AFTER ERROR PRINT
11107 003074 104001      ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
11108 003076 022626      2$:      CMP      (SP)+,(SP)+   :CLEAN STACK FROM TIMEOUT
11109 003100 012667 174700      3$:      MOV      (SP)+,ERRVEC  :RESTORE TIMEOUT VECTOR

:*****
:*TEST 3      TEST THAT REFERENCE TO RDBR DOES NOT CAUSE TIMEOUT
:*****
11110 003104 000004      *TST3:  SCOPE
11111 003106 016746 174672      MOV      ERRVEC,-(SP)   :SAVE THE TIMEOUT VECTOR
11112 003110 012767 003130 174664      MOV      #1$,ERRVEC    :GO TO 1$ IF TIMEOUT
11113 003120 016702 176266      MOV      DLADR,R2      :REGADR = RDBR ADR
11114 003124 005712      TST      (R2)           :USE REGADR ON BUS
11115 003126 000407      BR       3$            :<GO TO NEXT TEST IF NO TIMEOUT>
11116 003130 004767 010762 1$:      JSR      PC,SUERT1      :GO SET UP ERROR INFO
11117 003134 012767 003144 176102      MOV      #2$, $ESCAPE   :RETURN TO 2$ AFTER ERROR PRINT
11118 003142 104001      ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
11119 003144 022626      2$:      CMP      (SP)+,(SP)+   :CLEAN STACK FROM TIMEOUT
11120 003146 012667 174632      3$:      MOV      (SP)+,ERRVEC  :RESTORE TIMEOUT VECTOR

:*****
:*TEST 4      TEST THAT REFERENCE TO XDBR DOES NOT CAUSE TIMEOUT
:*****
11121 003152 000004      *TST4:  SCOPE
11122 003154 016746 174624      MOV      ERRVEC,-(SP)   :SAVE THE TIMEOUT VECTOR
11123 003160 012767 003176 174616      MOV      #1$,ERRVEC    :GO TO 1$ IF TIMEOUT
11124 003166 016702 176224      MOV      DLXDBR,R2     :REGADR = XDBR ADR
11125 003172 005712      TST      (R2)           :USE REGADR ON BUS
11126 003174 000407      BR       3$            :<GO TO NEXT TEST IF NO TIMEOUT>
11127 003176 004767 010714 1$:      JSR      PC,SUERT1      :GO SET UP ERROR INFO
11128 003202 012767 003212 176034      MOV      #2$, $ESCAPE   :RETURN TO 2$ AFTER ERROR PRINT
11129 003210 104001      ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
11130 003212 022626      2$:      CMP      (SP)+,(SP)+   :CLEAN STACK FROM TIMEOUT
11131 003214 012667 174564      3$:      MOV      (SP)+,ERRVEC  :RESTORE TIMEOUT VECTOR

:*****
:*TEST 5      TEST THAT RCSR IS ALL ZEROES ON ENTRY
:*****
12001 003220 000004      *TST5:  SCOPE
12002 003222 005004      CLR      R4             :RESULT IN RCSR S/B = 0
12003 003224 016702 176160      MOV      DLRCSR,R2     :REGADR = RCSR ADR
12004 003230 020412      CMP      R4,(R2)       :[RCSR]=000000 ??
12005 003232 001403      BEQ     TST6           :<BR IF YES>
12006 003234 004767 010576      JSR      PC,SUER2      :GO SET UP ERROR INFO
12007 003240 104002      ERROR+2 :RCSR NOT CLEAR ON START UP

:*****
:*TEST 6      TEST THAT "READY" BIT IS ONLY BIT SET IN XCSR
:*****
12110 003242 000004      *TST6:  SCOPE
12111 003244 012704 000200      MOV      #200,R4       :RESULT IN XCSR S/B = 000200
12112 003250 016702 176140      MOV      DLXCSR,R2     :REGADR = XCSR ADR

```

D03

MAINDEC-11-DZDLC-A
DZDLC.A.P11 T6

MACY11 27(732) 20-SEP-76 09:19 PAGE 29
TEST THAT "READY" BIT IS ONLY BIT SET IN XCSR

```

1213 003254 020412          CMP      R4,(R2)          ;[XCSR]=000200 ??
1214 003256 001403          BEQ      TST7            ;;<BR IF YES>
1215 003260 004767 010552     JSR      PC,SUER2        ;GO SETUP ERROR INFO
1216 003264 104002          ERROR+2          ;[XCSR] INCORRECT ON START UP
1217
1218 ::*****
1219 :*TEST 7          TEST THAT "MAINT" BIT CAN BE SET AND CLEARED
1220 ::*****
1221 TST7:  SCOPE
1222 003266 000004          MOV      #204,R4         ;RESULT IN XCSR S/B = 000204
1223 003270 012704 000204     MOV      DLXCSR,R2       ;REGADR = XCSR ADR
1224 003274 016702 176114     BIS      #BIT2,(R2)      ;SET THE "MAINT" BIT
1225 003300 052712 000004     CMP      R4,(R2) ;RESULT IN XCSR OK ??
1226 003304 020412          BEQ      1$             ;;<BR IF YES>
1227 003306 001403          JSR      PC,SUER2        ;GO SET UP ERROR INFO
1228 003310 004767 010522     ERROR+2          ;MAINT. BIT FAILED TO SET PROPERLY
1229 003314 104002          1$:  MOV      #200,R4        ;RESULT IN XCSR S/B = 000200
1230 003316 012704 000200     BIC      #BIT2,(R2)      ;NOW CLEAR THE "MAINT" BIT
1231 003322 042712 000004     CMP      R4,(R2)        ;RESULT IN XCSR OK ??
1232 003326 020412          BEQ      TST10          ;;<BR IF YES>
1233 003330 001403          JSR      PC,SUER2        ;GO SET UP ERROR INFO
1234 003332 004767 010500     ERROR+2          ;MAINT BIT FAILED TO CLEAR PROPERLY
1235
1236 ::*****
1237 :*TEST 10         TEST THAT XMIT I.E. CAN CAUSE AN INTR
1238 ::*****
1239 TST10: SCOPE
1240 003340 000004          CLR      INTFLG          ;INIT SOFTWARE INTR FLAG
1241 003342 005067 176076     MOV      DLVECT,R5       ;GET VECTOR ADDRESS
1242 003346 016705 176046     MOV      #2$,4(R5)       ;GO TO 4$ ON INTR
1243 003352 012765 003440 000004     MOV      DLPRI,6(R5)     ;PRIORITY LEVEL 4
1244 003360 016765 175716 000006     CLR      R5              ;INIT INTR. TIMER
1245 003366 005005          MOV      #200,R4        ;RESULT IN XCSR S/B = 000200
1246 003370 012704 000200     MOV      DLXCSR,R2       ;REGADR = XCSR ADR
1247 003374 016702 176014     BIS      #100,(R2)       ;SET INTR. ENABLE BIT 06
1248 003400 052712 000100     TST     INTFLG          ;DID INTR OCCUR YET ??
1249 003404 005767 176034     BNE     3$             ;BR IF IT DID
1250 003410 001020          DEC     R5              ;COUNT THE TIMER
1251 003412 005305          BNE     1$             ;BR IF NO TIMEOUT
1252 003414 001373          MOV      #300,R4        ;RESULT IN XCSR S/B = 000300
1253 003416 012704 000300     JSR      PC,SUER2        ;GO SETUP ERROR INFO
1254 003422 004767 010410     MOV      #4$,SESCAPE     ;RETURN TO 4$ AFTER ERROR PRINT
1255 003426 012767 003436 175610     ERROR+2          ;INTR. FAILED
1256 003434 104002          4$:  BR      TST11          ;;<GO TO NEXT TEST>
1257 003436 000412          COM     INTFLG          ;SET THE SOFTWARE FLAG
1258 003440 005167 176000     BIC     #100,(R2)       ;TURN OFF I.E. BIT
1259 003444 042712 000100     RTI                    ;RETURN CONTROL TO INTR. ROUTINE
1260 003450 000002          CMP     R4,(R2)        ;RESULT IN XCSR OK ??
1261 003452 020412          BEQ     TST11          ;;<BR IF YES>
1262 003454 001403          JSR     PC,SUER2        ;GO SET UP ERROR INFO.
1263 003456 004767 010354     ERROR+2          ;XMIT INTR. NOT SERVICED PROPERLY
1264
1265 ::*****
1266 :*TEST 11         TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
1267 ::*****
1268 TST11: SCOPE
1269 003464 000004          MOV     #100,R4         ;RESULT IN RCSR S/B = 000100
1270 003466 012704 000100     MOV     DLRCR,R2        ;REGADR = RCSR ADR
1271 003472 016702 175712

```

E03

MAINDEC-11-DZDLC-A
DZDLC.A.P11

MACY11 27(732) 20-SEP-76 09:19 PAGE 30
TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED

```

1269 003476 052712 000100      BIS      #BIT6,(R2)      ;SET I.E. BIT
1270 003502 020412      CMP      R4,(R2)       ;DID IT SET PROPERLY ??
1271 003504 001403      BEQ     1$             ;<BR IF YES>
1272 003506 004767 010324      JSR     PC,SUER2      ;GO SET UP ERROR INFO.
1273 003512 104002      ERROR+2 ;RCVR I.E. BIT FAILED TO SET PROPERLY
1274 003514 005004      1$:    CLR      R4          ;RESULT IN RCSR S/B = 0000C0
1275 003516 042712 000100      BIC     #BIT6,(R2)    ;CLEAR THE I.E. BIT
1276 003522 020412      CMP      R4,(R2)       ;DID IT CLEAR PROPERLY ??
1277 003524 001403      BEQ     TST12         ;<BR IF YES>
1278 003526 004767 010304      JSR     PC,SUER2      ;GO SET UP ERROR INFO
1279 003532 104002      ERROR+2 ;RCVR I.E. BIT FAILED TO CLEAR PROPERLY
1280
1281
1282
1283 003534 000004      ;*****
1284 003536 016705 175656      ;*TEST 12      TEST THAT RCVR "DONE" CAN GENERATE AN INTR.
1285 003542 012725 003702      ;*****
1286 003546 016715 175530      ;*ST12:  SCOPE
1287 003552 005067 175666      MOV     DLVECT,R5      ;GET THE VECTOR ADDRESS
1288 003556 005005      MOV     #3$(R5)+      ;GO TO 3$ ON RCVR INTR.
1289 003560 105067 175420      MOV     DLPRI,(R5)    ;AT LEVEL 4
1290 003564 016702 175620      CLR     INTFLG        ;INIT THE SOFTWARE FLAG
1291 003570 005012      CLR     R5            ;INIT INTR. TIMER
1292 003572 052712 000100      CLRB   $TMP1         ;INIT WHERE DATA WILL BE STORED
1293 003576 052762 000004 000004      MOV     DLRCSR,R2     ;REGADR = RCSR ADR
1294 003604 112762 000252 000006      CLR     (R2)          ;INIT THE RCSR TO 0000C0
1295 003612 005767 175626      BIS     #BIT6,(R2)    ;ENABLE RCVR INTERRUPTS
1296 003616 001044      BIS     #BIT2,4(R2)   ;NOW TURN ON MAINT MODE
1297 003620 005305      MOVVB  #252,6(R2)     ;LOAD XMIT BUFFER REG.
1298 003622 001373      1$:    TST     INTFLG      ;DID RCVR INTR. YET ??
1299 003624 013767 177776 175350      BNE    4$            ;BR IF IT DID
1300 003632 042762 000004 000004      DEC    R5            ;COUNT THE TIMER
1301 003640 042712 000100      BNE    1$            ;BR IF NO TIMEOUT
1302 003644 010667 175326      MOV     @#PSW,$TMP0   ;SAVE ERROR PSW
1303 003650 010201      BIC     #BIT2,4(R2)   ;DISABLE MAINT MODE
1304 003652 011203      BIC     #100,(R2)     ;DISABLE RCVR INTR.
1305 003654 012704 000200      MOV     SP,$REG6     ;SAVE THE ERROR SP
1306 003660 004767 010200      MOV     R2,R1        ;DEVADR = RCSR ADR
1307 003664 012767 003674 175352      MOV     (R2),R3      ;GET THE WAS DATA
1308 003672 104002      MOV     #200,R4      ;[RCSR] S/B = 000200
1309 003674 005762 000002 2$:    JSR     PC,SUERR1    ;GO SET UP ERROR INFO.
1310      ERROR+2 ;RETURN TO 2$ AFTER ERROR ALWAYS
1311      TST     2(R2)      ;RCVR INTERRUPT FAILED
1312      BR     TST13     ;REFERENCE RCVR DATA BUFFER
1313      BIC     #BIT2,4(R2) ;TO CLEAR RCSR IN CASE RCVR
1314      MOVVB  2(R2),$TMP1 ;INTERRUPTS COULD NOT BE ENABLED
1315      BIC     #BIT6,(R2) ;<GO TO NEXT TEST>
1316      COM    INTFLG    ;DISABLE THE MAINT MODE
1317      RTI                    ;GET THE RECEIVED DATA
1318      CLR     R4          ;TURN OFF RCVR INTR. ENAB
1319      TST     (R2)       ;SET THE SOFTWARE FLAG
1320      BEQ     5$        ;RETURN TO MAINLINE
1321      JSR     PC,SUER2   ;[RCSR] S/B=0
1322      ERROR+2 ;IS IT ALL ZEROES ??
1323      MOV     DLRDBR,R1 ;<BR IF YES>
1324      MOV     DLXDBR,R2 ;GO SET UP ERROR INFO
                        ;RCVR INTR NOT SERVICED PROPERLY
                        ;SAVE WAS ADDRESS
                        ;SAVE THE S/B ADDRESS

```

```

1325 003754 016703 175224      MOV      $TMP1,R3      ;GET THE WAS DATA
1326 003760 042703 177400      BIC      #177400,R3    ;REMOVE JUNK FROM HI BYTE
1327 003764 012704 000252      MOV      #252,R4      ;DATA S/B = 252
1328 003770 020403          CMP      R4,R3        ;WAS = S/B ??
1329 003772 001403          BEQ      TST13        ;<BR IF YES>
1330 003774 004767 010064      JSR      PC,SUERR1    ;GO SET UP THE ERROR INFO
1331 004000 104003          ERROR+3             ;DATA COMPARE ERROR
1332
1333      ;*****
1334      ;*TEST 13      TEST THAT "REQ TO SEND" ASSERTS "RING"
1335      ;*****
1335 004002 000004          TST13: SCOPE
1336 004004 032767 010000 175126      BIT      #SW12,SWR    ;ARE WE TESTING /C OR /D MODEL?
1337 004012 001047          BNE     TST14        ;<BRANCH IF YES>
1338 004014 012704 140004          MOV      #140004,R4  ;RESULT IN RCSR S/B = 140004
1339 004020 016702 175364          MOV      DLRCR,R2    ;REGADR = RCSR ADR
1340 004024 005012          CLR     (R2)         ;INIT THE RCSR TO 000000
1341 004026 052712 000004          BIS     #BIT2,(R2)   ;SET "REQ TO SEND"
1342 004032 032777 100000 175350      BIT      #BIT15,DLRCR ;DID "RING" SET "DATA SET INT" ?
1343 004040 001003          BNE     1$          ;<BR IF YES>
1344 004042 004767 007770          JSR      PC,SUER2    ;GO SET UP ERROR INFO.
1345 004046 104002          ERROR+2             ;"RING" TRANSITION FAILED TO SET "DATA SET INT"
1346
1347 004050 012704 040004          1$: MOV      #40004,R4  ;RESULT IN RCSR S/B = 40004
1348 004054 020412          CMP     R4,(R2)     ;BOTH "RING" AND "REQ TO SEND" ASSERTED ?
1349 004056 001403          BEQ     2$          ;<BR IF YES>
1350 004060 004767 007752          JSR      PC,SUER2    ;GO SET UP ERROR INFO.
1351 004064 104002          ERROR+2             ;"RING" OR "REQ TO SEND" FAILED TO SET
1352 004066 005004          2$: CLR     R4        ;RESULT IN RCSR S/B = 000000
1353 004070 042712 000004          BIC     #BIT2,(R2)   ;CLEAR "REQ TO SEND"
1354 004074 032777 100000 175306      BIT      #BIT15,DLRCR ;DID "DATA SET INT" GET SET ??
1355 004102 001403          BEQ     3$          ;<BR IF NOT>
1356 004104 004767 007726          JSR      PC,SUER2    ;GO SET UP ERROR INFO
1357 004110 104002          ERROR+2             ;CLEARING "RING" SET "DATA SET INT"
1358 004112 020412          3$: CMP     R4,(R2)   ;RCSR CONTAIN ALL ZEROES ??
1359 004114 001406          BEQ     TST14        ;<BR IF YES>
1360 004116 004767 007714          JSR      PC,SUER2    ;GO SET UP ERROR INFO.
1361 004122 016767 000002 175050      MOV      .+6,$REG7   ;SAVE THE ERROR PC
1362 004130 104002          ERROR+2             ;CLEARING "REQ TO SEND" FAILED TO CLEAR "RING"
1363
1364      ;*****
1365      ;*TEST 14      TEST THAT "SEC XMIT" ASSERTS "SEC REC" AND "DATA SET INT"
1366      ;*****
1366 004132 000004          TST14: SCOPE
1367 004134 032767 010000 174776      BIT      #SW12,SWR    ;ARE WE TESTING /C OR /D MODEL?
1368 004142 001046          BNE     TST15        ;<BRANCH IF YES>
1369 004144 016702 175240          MOV      DLRCR,R2    ;REGADR = RCSR ADR
1370 004150 005012          CLR     (R2)         ;INIT RCSR TO 000000
1371 004152 012704 102010          MOV      #102010,R4  ;CONTENTS OF RCSR S/B = 102010
1372 004156 052712 000010          BIS     #BIT3,(R2)   ;SET "SEC XMIT" BIT
1373 004162 032777 100000 175220      BIT      #BIT15,DLRCR ;DID "DATA SET INT" SET ??
1374 004170 001003          BNE     1$          ;<BR IF YES>
1375 004172 004767 007640          JSR      PC,SUER2    ;GO SET UP ERROR INFO
1376 004176 104002          ERROR+2             ;"DATA SET INT" FAILED TO SET-NOTE THAT
1377
1378 004200 012704 002010          1$: MOV      #2010,R4  ;"BIT #BIT15,(R2)" RESETS BIT15
1379 004204 020412          CMP     R4,(R2)     ;RESULT IN RCSR S/B = 2010
1380 004206 001403          BEQ     2$          ;ARE "SEC XMIT" AND "SEC REC" BOTH SET ?
1380
1380

```



```

1381 004210 004767 007622 JSR PC,SUER2 ;GO SET UP ERROR INFO
1382 004214 104002 ERROR+2 ;"SEC XMIT" OR "SEC REC" FAILED TO SET
1383 ;OR "DATA SET INT" FAILED TO BE CLEARED
1384 ;WHEN REFERENCING RCSR
1395 004216 012704 100000 2$: MOV #BIT15,R4 ;RESULT IN RCSR S/B = 100000
1396 004222 042712 000010 BIC #BIT3,(R2) ;CLEAR "SEC XMIT" BIT
1387 004226 032777 100000 175154 BIT #BIT15,DLRCSR ;DID CLEARING IT SET "DATA SET INT"??
1398 004234 001003 BNE 3$ ;<BR IF YES>
1389 004236 004767 007574 JSR PC,SUER2 ;GO SET UP ERROR INFO.
1390 004242 104002 ERROR+2 ;CLEARING "SEC XMIT" FAILED TO SET "DATA
1391 ;SET INT. (NOTE THAT REFERENCING RCSR
1392 ;CLEARS "DATA SET INT"
1393 004244 005004 3$: CLR R4 ;RESULT IN RCSR S/B = 000000
1394 004246 020412 CMP R4,(R2) ;"SEC XMIT" AND "SEC REC" CLEAR ?
1395 004250 001403 BEQ TST15 ;<BR IF YES>
1396 004252 004767 007560 JSR PC,SUER2 ;GO SETUP ERROR INFO
1397 004256 104002 ERROR+2 ;"SEC XMIT" OR "SEC REC" FAILED TO CLEAR
1398 ;OR REFERENCING RCSR FAILED TO CLEAR "DATA SET INT"
1399 ;*****
1400 ;*TEST 15 TEST THAT "DTR" CAN ASSERT "CLR TO SEND" AND "CAR DET"
1401 ;*****
1402 004260 000004 TST15: SCOPE
1403 004262 032767 010000 174650 BIT #SW12,SWR ;ARE WE TESTING /C OR /D MODEL?
1404 004270 001046 BNE TST16 ;<BRANCH IF YES>
1405 004272 016702 175112 MOV DLRCSR,R2 ;REGADR = RCSR ADR
1406 004276 005012 CLR (R2) ;INIT RCSR TO 000000
1407 004300 012704 130002 MOV #130002,R4 ;RESULT IN RCSR S/B = 130002
1408 004304 052712 000002 BIS #BIT1,(R2) ;SET "DTR" BIT
1409 004310 032777 100000 175072 BIT #BIT15,DLRCSR ;DID "DATA SET INT" SET ??
1410 004316 001003 BNE 1$ ;<BR IF YES>
1411 004320 004767 007512 JSR PC,SUER2 ;GO SET UP ERROR INFO.
1412 004324 104002 ERROR+2 ;"DATA SET INT" FAILED TO SET -
1413 ;NOTE: THE REFERENCE TO RCSR ABOVE WILL
1414 ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.
1415 004326 012704 030002 1$: MOV #30002,R4 ;RESULT IN RCSR S/B = 30002
1416 004332 020412 CMP R4,(R2) ;"DTR" "CLR TO SEND", AND "CAR DET" ALLSET
1417 004334 001403 BEQ 2$ ;<BR IF ALL SET>
1418 004336 004767 007474 JSR PC,SUER2 ;GO SET UP ERROR INFO
1419 004342 104002 ERROR+2 ;"DTR" "CLR TO SEND" OR "CAR DET" FAILED
1420 ;TO SET OR "DATA SET INT" FAILED TO CLEAR
1421 004344 012704 100000 2$: MOV #BIT15,R4 ;RESULT IN RCSR S/B = 100000
1422 004350 042712 000002 BIC #BIT1,(R2) ;NOW CLEAR "DTR"
1423 004354 032777 100000 175026 BIT #BIT15,DLRCSR ;DID "DATA SET INT" SET ??
1424 004362 001003 BNE 3$ ;<BR IF YES>
1425 004364 004767 007446 JSR PC,SUER2 ;GO SETUP ERROR INFO
1426 004370 104002 ERROR+2 ;"DATA SET INT" FAILED TO SET WHEN "DTR"
1427 ;WENT TO A ZERO.
1428 004372 005004 3$: CLR R4 ;RESULT IN RCSR S/B = 000000
1429 004374 020412 CMP R4,(R2) ;DID ALL BITS CLEAR??
1430 004376 001403 BEQ TST16 ;<BR IF YES>
1431 004400 004767 007432 JSR PC,SUER2 ;GO SET UP ERROR INFO
1432 004404 104002 ERROR+2 ;"DTR" "CLR TO SEND" OR "CAR DET" FAILED
1433 ;TO CLEAR PROPERLY
1434 ;*****
1435 ;*TEST 16 TEST THAT "DATA SET INT ENAB" CAN SET AND CLEAR
1436 ;*****

```

H03

MAINDEC-11-DZDLC-A
DZDLC.A.P11 T16

MACY11 27(732) 20-SEP-76 09:19 PAGE 33
TEST THAT "DATA SET INT ENAB" CAN SET AND CLEAR

```

1437 004406 000004          TST16: SCOPE
1438 004410 032767 010000 174522 BIT      #SW12,SWR      ;ARE WE TESTING /C OR /D MODEL?
1439 004416 001023          BNE     TST17      ;<BRANCH IF YES>
1440 004420 016702 174764 MOV     DLRCR,R2   ;REGADR = RCSR ADR
1441 004424 012704 000040 MOV     #40,R4     ;RESULT IN RCSR S/B = 000040
1442 004430 052712 000040 BIS     #BITS,(R2) ;SET THE "DATA SET I.E." BIT
1443 004434 020412          CMP     R4,(R2)   ;DID IT SET OK ??
1444 004436 001403          BEQ     1$        ;<BR IF YES>
1445 004440 004767 007372 JSR     PC,SUER2  ;GO SET UP ERROR INFO
1446 004444 104002          ERROR+2 ;"DAT SET I. E." FAILED TO SET
1447 004446 005004          1$: CLR     R4       ;MAKE S/B DATA = 000000
1448 004450 042712 000040 BIC     #BITS,(R2) ;NOW CLEAR THE "DATA SET I.E." BIT
1449 004454 020412          CMP     R4,(R2)   ;DID IT CLEAR OK ??
1450 004456 001403          BEQ     TST17     ;<BR IF YES>
1451 004460 004767 007352 JSR     PC,SUER2  ;GO SET UP ERROR INFO.
1452 004464 104002          ERROR+2 ;"DATA SET I.E." FAILED TO CLEAR
1453          ;*****
1454          ;*TEST 17      TEST THE "DATA SET I.E." CAN CAUSE A RCVR INTR
1455          ;*****
1456 004466 000004          TST17: SCOPE
1457 004470 032767 010000 174442 BIT      #SW12,SWR      ;ARE WE TESTING A /C OR /D MODEL?
1458 004476 001054          BNE     TST20     ;<BRANCH IF YES>
1459 004500 016705 174714 MOV     DLVECT,R5 ;GET THE VECTOR ADDR
1460 004504 012725 004572 MOV     #3$,R5+    ;GO TO 3$ ON RCVR INTR.
1461 004510 016715 174566 MOV     DLPRI,(R5) ;AT LEVEL 4
1462 004514 005005          CLR     R5       ;INIT INTR. TIMER
1463 004516 005067 174722 CLR     INTFLG    ;INIT SOFTWARE FLAG
1464 004522 005004          CLR     R4       ;RESULT IN RCSR S/B = 0 AFTER INTR.
1465 004524 016702 174660 MOV     DLRCR,R2   ;REGADR = RCSR ADR
1466 004530 052712 000040 BIS     #BITS,(R2) ;SET THE "DATA SET I.E." BIT
1467 004534 052712 000002 BIS     #BIT1,(R2) ;NOW SET "DTR" TO GEN INTR.
1468 004540 005767 174700 1$: TST     INTFLG   ;DID INTR OCCUR YET ??
1469 004544 001016          BNE     4$       ;BR IF YES
1470 004546 005305          DEC     R5       ;COUNT THE TIMER
1471 004550 001377          BNE     1$       ;BR IF NO TIMEOUT
1472 004552 004767 007260 JSR     PC,SUER2  ;GO SET UP ERROR INFO
1473 004556 005012          CLR     (R2)     ;TURN IT ALL OFF
1474 004560 012767 004570 174456 MOV     #2$, $ESCAPE ;COME BACK TO 2$ IN ALL CASES
1475 004566 104002          ERROR+2 ;"DATA SET" INTR FAILED TO OCCUR
1476 004570          2$:
1477 004570 000417          BR     TST20     ;<GO TO NEXT TEST>
1478 004572 005012          3$: CLR     (R2)   ;ZERO THE RCSR
1479 004574 005167 174644 COM     INTFLG    ;SET THE SOFTWARE FLAG
1480 004600 000002          RTI                    ;RETURN TO SENDER
1481 004602 032712 100000 4$: BIT     #BIT15,(R2) ;DID "DATA SET INT" GET SET BY INTR. SERVICE ??
1482 004606 001003          BNE     5$       ;<BR IF YES>
1483 004610 004767 007222 JSR     PC,SUER2  ;GO SET UP ERROR INFO
1484 004614 104002          ERROR+2 ;DATA SET INTR. NOT SERVICED PROPERLY
1485 004616 020412          5$: CMP     R4,(R2) ;ALL BITS IN RCSR CLEAR ??
1486 004620 001403          BEQ     TST20     ;<BR IF YES>
1487 004622 004767 007210 JSR     PC,SUER2  ;GO SET UP ERROR INFO
1488 004626 104002          ERROR+2 ;INTR. SERVICE FAILED TO CLEAR RCSR
1489          ;*****
1490          ;*TEST 20      TEST THAT THE "BREAK" BIT CAN BE SET AND CLEARED
1491          ;*****
1492 004630 000004          TST20: SCOPE

```

MAINDEC-11-DZDLC-A
DZDLC.A.P11 T20

MACY11 27(732) 20-SEP-76 09:19 PAGE 34
TEST THAT THE "BREAK" BIT CAN BE SET AND CLEARED

```

1493 004632 032767 010000 174300      BIT      #SW12,SWR      ;ARE WE TESTING /C OR /D MODEL?
1494 004640 001024                    BNE      TST21      ;:<BRANCH IF YES>
1495 004642 012704 000201              MOV      #201,R4    ;RESULT S/B = 201 IN XCSR
1496 004646 016702 174542              MOV      DLXCSR,R2 ;SET UP REGADR
1497 004652 052712 000001              BIS      #BIT0,(R2) ;SET THE "BREAK" BIT
1498 004656 020412                    CMP      R4,(R2)    ;DID IT SET PROPERLY ??
1499 004660 001403                    BEQ      1$         ;:<BR IF YES>
1500 004662 004767 007150              JSR      PC,SUER2  ;GO SET UP ERROR INFO.
1501 004666 104002                    ERROR+2 ;"BREAK" BIT FAILED TO SET PROPERLY
1502 004670 012704 000200      1$: MOV      #200,R4    ;RESULT S/B = 200 IN XCSR
1503 004674 042712 000001              BIC      #BIT0,(R2) ;CLEAR THE "BREAK" BIT
1504 004700 020412                    CMP      R4,(R2)    ;DID IT CLEAR PROPERLY ??
1505 004702 001403                    BEQ      TST21     ;:<BR IF YES>
1506 004704 004767 007126              JSR      PC,SUER2  ;GO SET UP ERROR INFO
1507 004710 104002                    ERROR+2 ;"BREAK" FAILED TO CLEAR PROPERLY
1508
1509
1510
1511
1512
1513 004712 000004                    ;:*****
1514 004714 012704 000200      *TEST 21      TEST THAT A "RESET" CLEARS THE "BREAK" BIT
1515 004720 016702 174470                    ;:*****
1516 004724 052712 000001      TST21: SCOPE
1517 004730 000005                    MOV      #200,R4    ;RESULT S/B = 200
1518 004732 020412                    MOV      DLXCSR,R2 ;SET UP REGADR
1519 004734 001403                    BIS      #BIT0,(R2) ;SET THE "BREAK" BIT
1520 004736 004767 007074              RESET    ;CLEAR IT WITH A "RESET"
1521 004742 104002                    CMP      R4,(R2)    ;DID IT CLEAR ??
1522                                BEQ      TST22     ;:<BR IF YES>
                                JSR      PC,SUER2  ;GO SET UP ERROR INFO.
                                ERROR+2 ;RESET INSTR. FAILED TO CLEAR "BREAK"

```

```

1523
1524
1525
1526 004744 000004
1527 004746 012767 000001 174266
1528 004754 004767 007212
1529 004760 005067 174444
1530 004764 012767 014346 174444 1$:
1531 004772 004767 007222
1532 004776 005767 174420 2$:
1533 005002 001040
1534 005004 005767 174414
1535 005010 001053
1536 005012 005767 174410
1537 005016 001065
1538 005020 022767 022322 174406
1539 005026 001003
1540 005030 004767 007456
1541 005034 000500
1542 005036 005367 174376 3$:
1543 005042 001355
1544 005044 005367 174372
1545 005050 001352
1546 005052 042777 000100 174330
1547 005060 042777 000104 174326
1548 005066 104401 016326
1549 005072 012767 005102 174144
1550 005100 104000
1551 005102
1552 005102 000455 4$:
1553 005104 016701 174300 5$:
1554 005110 016702 174300
1555 005114 011203
1556 005116 012704 000204
1557 005122 004767 006736
1558 005126 012767 005136 174110
1559 005134 104002
1560 005136
1561 005136 000437 6$:
1562 005140 016701 174244 7$:
1563 005144 010102
1564 005146 011203
1565 005150 012704 000200
1566 005154 004767 006704
1567 005160 012767 005170 174056
1568 005166 104002
1569 005170
1570 005170 000422 8$:
1571 005172 016701 174212 9$:
1572 005176 016702 174210
1573 005202 016703 173776
1574 005206 004767 006652
1575 005212 012767 005222 174024
1576 005220 04005
1577 005222 005267 174202 10$:
1578 005226 022767 000003 174174

```

```

*****
*TEST 22 TEST TO TURN AROUND NULL-DEL-NULL PATTERN
*****
TST22: SCOPE
MOV #1,$TIMES ;: 1 ITERATION
JSR PC,SUVEC ;GO SET UP VECTORS
CLR RTRY ;INITIALIZE RETRY FLAG
MOV #LDOUT1,LDOUT ;SET POINTER TO LOAD ROUTINE
JSR PC,PRIME ;GO SET UP BUFFERS AND DEVICE
TST XFLGO ;ANY HARD XMIT ERRORS ??
BNE 5$ ;BR IF YES
TST RFLGO ;ANY HARD RECEIVER ERROR ??
BNE 7$ ;BR IF YES
TST RFLG1 ;ANY SOFT RECEIVER ERRORS ??
BNE 9$ ;BR IF YES
CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??
BNE 3$ ;BR IF NOT
JSR PC,CHKDAT ;GO CHECK THE DATA BUFFERS
BR TST23 ;<GO TO NEXT TEST>
DEC TIMR1 ;DEC TIMEOUT COUNTER 1
BNE 2$ ;BR IF NO TIMEOUT
DEC TIMR2 ;DEC TIMEOUT COUNTER 2
BNE 2$ ;BR IF NO TIMEOUT
BIC #100,$DLRCSR ;TURN OFF THE INTRs.
BIC #104,$DLXCSR
TYPE XMSG1 ;GO TYPE TIMEOUT MESSAGE
MOV #4,$$ESCAPE ;GO TO 4$ AFTER ERROR PRINT
ERROR ;PRINT ERROR PC
4$:
BR TST23 ;<GO TO NEXT TEST>
5$:
MOV DLRCSR,R1 ;PUT DEVADR IN R1
MOV DLXCSR,R2 ;PUT REGADR IN R2
MOV (R2),R3 ;GET THE WAS DATA
MOV #204,R4 ;PUT S/B DATA IN R4
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #6,$$ESCAPE ;GO TO 6$ AFTER PRINTING ERROR
ERROR+2 ;TRANSMITTER FALSE INTERRUPT
6$:
BR TST23 ;<GO TO NEXT TEST>
7$:
MOV DLRCSR,R1 ;SAVE THE DEVADR
MOV R1,R2 ;SAVE THE REGADR
MOV (R2),R3 ;GET THE WAS DATA
MOV #200,R4 ;RESULT S/B = 200
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #8,$$ESCAPE ;GO TO 8$ AFTER ERROR PRINT
ERROR+2 ;RECEIVER FALSE INTERRUPT
8$:
BR TST23 ;<GO TO NEXT TEST>
9$:
MOV DLRCSR,R1 ;SAVE THE DEVADR
MOV DLRDBR,R2 ;SAVE REGADR
MOV $TMP1,R3 ;GET CONTENTS OF ERROR RDBR
JSR PC,SUERR1 ;GO SETUP ERROR INFO
MOV #10,$$ESCAPE ;GO TO 10$ AFTER ERROR PRINT
ERROR+5 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN)
10$:
INC RTRY ;COUNT ONE TRY
CMP #3,RTRY ;TRIED THREE TIMES

```

K03

MAINDEC-11-DZDLC-A
DZDLC.A.P11 T22

MACY11 27(732) 20-SEP-76 09:19 PAGE 36
TEST TO TURN AROUND NULL-DEL-NULL PATTERN

1579 005234 001253

BNE 1\$

;BR IF NOT

```

1580          ;:*****
1581          ;*TEST 23      TEST TO TURN AROUND BINARY UP COUNT PATTERN
1582          ;:*****
1583          TST23:  SCOPE
1584          MOV      #1,$TIMES      ;:DO 1 ITERATION
1585          JSR      PC,SUVEC      ;:GO SET UP VECTORS
1586          CLR      RTRY          ;:INITIALIZE RETRY FLAG
1587          MOV      #LDOUT2,LDOUT ;:SET POINTER TO LOAD ROUTINE
1588          JSR      PC,PRIME      ;:GO SET UP BUFFERS AND DEVICE
1589          TST      XF LGD        ;:ANY HARD XMIT ERRORS ??
1590          BNE      5$           ;:BR IF YES
1591          TST      RFLGD         ;:ANY HARD RECEIVER ERROR ??
1592          BNE      7$           ;:BR IF YES
1593          TST      RFLG1        ;:ANY SOFT RECEIVER ERRORS ??
1594          BNE      9$           ;:BR IF YES
1595          CMP      #BUFEND,IPTR  ;:RECEIVED 256. BYTES ??
1596          BNE      3$           ;:BR IF NOT
1597          JSR      PC,CHKDAT     ;:GO CHECK THE DATA BUFFERS
1598          BR       TST24        ;: <GO TO NEXT TEST>
1599          DEC      TIMR1        ;:DEC TIMEOUT COUNTER 1
1600          BNE      2$           ;:BR IF NO TIMEOUT
1601          DEC      TIMR2        ;:DEC TIMEOUT COUNTER 2
1602          BNE      2$           ;:BR IF NO TIMEOUT
1603          BIC      #100,$DLRCSR ;:TURN OFF THE INTRs.
1604          BIC      #104,$DLXCSR
1605          TYPE     ,XMSG2       ;:GO TYPE TIMEOUT MESSAGE
1606          MOV      #4$,$ESCAPE  ;:GO TO 4$ AFTER ERROR PRINT
1607          ERROR   ;:PRINT ERROR PC
1608          4$:
1609          BR       TST24        ;: <GO TO NEXT TEST>
1610          MOV      DLRCSR,R1     ;:PUT DEVADR IN R1
1611          MOV      DLXCSR,R2     ;:PUT REGADR IN R2
1612          MOV      (R2),R3      ;:GET THE WAS DATA
1613          MOV      #204,R4      ;:PUT S/B DATA IN R4
1614          JSR      PC,SUERR1    ;:GO SET UP ERROR INFO
1615          MOV      #6$,$ESCAPE  ;:GO TO 6$ AFTER PRINTING ERROR
1616          ERROR+2 ;:TRANSMITTER FALSE INTERRUPT
1617          6$:
1618          BR       TST24        ;: <GO TO NEXT TEST>
1619          MOV      DLRCSR,R1     ;:SAVE THE DEVADR
1620          MOV      R1,R2        ;:SAVE THE REGADR
1621          MOV      (R2),R3      ;:GET THE WAS DATA
1622          MOV      #200,R4      ;:RESULT S/B = 200
1623          JSR      PC,SUERR1    ;:GO SET UP ERROR INFO
1624          MOV      #8$,$ESCAPE  ;:GO TO 8$ AFTER ERROR PRINT
1625          ERROR+2 ;:RECEIVER FALSE INTERRUPT
1626          8$:
1627          BR       TST24        ;: <GO TO NEXT TEST>
1628          MOV      DLRCSR,R1     ;:SAVE THE DEVADR
1629          MOV      DLRDBR,R2     ;:SAVE REGADR
1630          MOV      $TMP1,R3     ;:GET CONTENTS OF ERROR RDBR
1631          JSR      PC,SUERR1    ;:GO SETUP ERROR INFO
1632          MOV      #10$,$ESCAPE ;:GO TO 10$ AFTER ERROR PRINT
1633          ERROR+5 ;:REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN)
1634          INC      RTRY          ;:COUNT ONE TRY
1635          CMP      #3,RTRY       ;:TRIED THREE TIMES

```

M03

MAINDEC-11-DZDLC-A
DZDLCR.F11 T23

MACY11 27(732) 20-SEP-76 09:19 PAGE 39
TEST TO TURN AROUND BINARY UP COUNT PATTERN

1636 005526 001253

BNE 1\$

;BR IF NOT

```

1637
1638
1639
1640 005530 000004
1641 005532 012767 000001 173502
1642 005540 004767 006426
1643 005544 005067 173660
1644 005550 012767 014410 173660 1$:
1645 005556 004767 006436
1646 005562 005767 173634 2$:
1647 005566 001040
1648 005570 005767 173630
1649 005574 001053
1650 005576 005767 173624
1651 005602 001065
1652 005604 022767 022322 173622
1653 005612 001003
1654 005614 004767 006672
1655 005620 000500
1656 005622 005367 173612 3$:
1657 005626 001355
1658 005630 005367 173606
1659 005634 001352
1660 005636 042777 000100 173544
1661 005644 042777 000104 173542
1662 005652 104401 016466
1663 005656 012767 005666 173360
1664 005664 104000
1665 005666
1666 005666 000455 4$:
1667 005670 016701 173514 5$:
1668 005674 016702 173514
1669 005700 011203
1670 005702 012704 000204
1671 005706 004767 006152
1672 005712 012767 005722 173324
1673 005720 104002
1674 005722
1675 005722 000437
1676 005724 016701 173460 7$:
1677 005730 010102
1678 005732 011203
1679 005734 012704 000200
1680 005740 004767 006120
1681 005744 012767 005754 173272
1682 005752 104002
1683 005754 8$:
1684 005754 000422
1685 005756 016701 173426 9$:
1686 005762 016702 173424
1687 005766 016703 173212
1688 005772 004767 006066
1689 005776 012767 006006 173240
1690 006004 104005
1691 006006 005267 173416 10$:
1692 006012 022767 000003 173410

```

```

*****
*TEST 24 TEST TO TURN AROUND BINARY DOWN COUNT PATTERN
*****
TST24: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
JSR PC,SUVEC ;GO SET UP VECTORS
CLR RTRY ;INITIALIZE RETRY FLAG
MOV #LDOUT3,LDOUT ;SET POINTER TO LOAD ROUTINE
JSR PC,PRIME ;GO SET UP BUFFERS AND DEVICE
TST XFLGO ;ANY HARD XMIT ERRORS ??
BNE 5$ ;BR IF YES
TST RFLGO ;ANY HARD RECEIVER ERROR ??
BNE 7$ ;BR IF YES
TST RFLG1 ;ANY SOFT RECEIVER ERRORS ??
BNE 9$ ;BR IF YES
CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??
BNE 3$ ;BR IF NOT
JSR PC,CHKDAT ;GO CHECK THE DATA BUFFERS
BR TST25 ;<GO TO NEXT TEST>
DEC TIMR1 ;DEC TIMEOUT COUNTER 1
BNE 2$ ;BR IF NO TIMEOUT
DEC TIMR2 ;DEC TIMEOUT COUNTER 2
BNE 2$ ;BR IF NO TIMEOUT
BIC #100,$DLRCSR ;TURN OFF THE INTRS.
BIC #104,$DLXCSR
TYPE ,XMSG3 ;GO TYPE TIMEOUT MESSAGE
MOV #4$,$ESCAPE ;GO TO 4$ AFTER ERROR PRINT
ERROR ;PRINT ERROR PC
4$: BR TST25 ;<GO TO NEXT TEST>
5$: MOV DLRCSR,R1 ;PUT DEVADR IN R1
MOV DLXCSR,R2 ;PUT REGADR IN R2
MOV (R2),R3 ;GET THE WAS DATA
MOV #204,R4 ;PUT S/B DATA IN R4
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #6$,$ESCAPE ;GO TO 6$ AFTER PRINTING ERROR
ERROR+2 ;TRANSMITTER FALSE INTERRUPT
6$: BR TST25 ;<GO TO NEXT TEST>
7$: MOV DLRCSR,R1 ;SAVE THE DEVADR
MOV R1,R2 ;SAVE THE REGADR
MOV (R2),R3 ;GET THE WAS DATA
MOV #200,R4 ;RESULT S/B = 200
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #8$,$ESCAPE ;GO TO 8$ AFTER ERROR PRINT
ERROR+2 ;RECEIVER FALSE INTERRUPT
8$: BR TST25 ;<GO TO NEXT TEST>
9$: MOV DLRCSR,R1 ;SAVE THE DEVADR
MOV DLRDBR,R2 ;SAVE REGADR
MOV $TMP1,R3 ;GET CONTENTS OF ERROR RDBR
JSR PC,SUERR1 ;GO SETUP ERROR INFO
MOV #10$,$ESCAPE ;GO TO 10$ AFTER ERROR PRINT
ERROR+5 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
10$: INC RTRY ;COUNT ONE TRY
CMP #3,RTRY ;TRIED THREE TIMES

```


B04

MMN 20-01-20-01-01
000000000000000000
000000000000000000
000000000000000000

MAY 11 21:28:21 20-SEP-76 09:19 PAGE 40
TEST TO TURN AROUND BINARY DOWN COUNT PATTERN

0000 000000 001250

BNE 15

:BR IF NOT

```

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

```

006022	000004	000001	173210						
006024	012767	006134							
006032	004767	173366							
006036	005067	014444	173366	1\$:					
006042	012767	006144							
006050	004767	173342							
006054	005767			2\$:					
006060	001042								
006062	005767	173336							
006066	001056								
006070	005767	173332							
006074	001071								
006076	022767	022322	173330						
006104	001004								
006106	004767	006400							
006112	000167	002012							
006116	005367	173316		3\$:					
006122	001354								
006124	005367	173312							
006130	001351								
006132	042777	000100	173250						
006140	042777	000104	173246						
006146	104401	016551							
006152	012767	006162	173064						
006160	104000								
006162	000167	001742		4\$:					
006166	016701	173216		5\$:					
006172	016702	173216							
006176	011203								
006200	012704	000204							
006204	004767	005654							
006210	012767	006220	173026						
006216	104002								
006220	000167	001704		6\$:					
006224	016701	173160		7\$:					
006230	010102								
006232	011203								
006234	012704	000200							
006240	004767	005620							
006244	012767	006254	172772						
006252	104002								
006254	000167	001650		8\$:					
006260	016701	173124		9\$:					
006264	016702	173122							
006270	016703	172710							
006274	004767	005564							
006300	012767	006310	172736						
006306	104005								
006310	005267	173114		10\$:					
006314	022767	000003	173106						
006322	001247								
006324	000167	001600							

```

*****
*TEST 25 TEST TO TURN AROUND WORST CASE PATTERN
*****
TST25: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
JSR PC,SUVEC ;GO SET UP VECTORS
CLR RTRY ;INITIALIZE RETRY FLAG
MOV #LDOUT4,LDOUT ;SET POINTER TO LOAD ROUTINE
JSR PC,PRIME ;GO SET UP BUFFERS AND DEVICE
TST XFLGO ;ANY HARD XMIT ERRORS ??
BNE 5$ ;BR IF YES
TST RFLGO ;ANY HARD RECEIVER ERROR ??
BNE 7$ ;BR IF YES
TST RFLG1 ;ANY SOFT RECEIVER ERRORS ??
BNE 9$ ;BR IF YES
CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??
BNE 3$ ;BR IF NOT
JSR PC,CHKDAT ;GO CHECK THE DATA BUFFERS
JMP $EOP ;GO TO NEXT TEST
DEC TIMR1 ;DEC TIMEOUT COUNTER 1
BNE 2$ ;BR IF NO TIMEOUT
DEC TIMR2 ;DEC TIMEOUT COUNTER 2
BNE 2$ ;BR IF NO TIMEOUT
BIC #100,$DLRCSR ;TURN OFF THE INTRS.
BIC #104,$DLXCSR
TYPE .XMSG4 ;GO TYPE TIMEOUT MESSAGE
MOV #4$,$ESCAPE ;GO TO 4$ AFTER ERROR PRINT
ERROR ;PRINT ERROR PC
JMP $EOP ;GO TO NEXT TEST
MOV $DLRCSR,R1 ;PUT DEVADR IN R1
MOV $DLXCSR,R2 ;PUT REGADR IN R2
MOV (R2),R3 ;GET THE WAS DATA
MOV #204,R4 ;PUT S/B DATA IN R4
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #6$,$ESCAPE ;GO TO 6$ AFTER PRINTING ERROR
ERROR+2 ;TRANSMITTER FALSE INTERRUPT
JMP $EOP ;GO TO NEXT TEST
MOV $DLRCSR,R1 ;SAVE THE DEVADR
MOV R1,R2 ;SAVE THE REGADR
MOV (R2),R3 ;GET THE WAS DATA
MOV #200,R4 ;RESULT S/B = 200
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #8$,$ESCAPE ;GO TO 8$ AFTER ERROR PRINT
ERROR+2 ;RECEIVER FALSE INTERRUPT
JMP $EOP ;GO TO NEXT TEST
MOV $DLRCSR,R1 ;SAVE THE DEVADR
MOV $DLRDBR,R2 ;SAVE REGADR
MOV $STMP1,R3 ;GET CONTENTS OF ERROR RDBR
JSR PC,SUERR1 ;GO SET UP ERROR INFO
MOV #10$,$ESCAPE ;GO TO 10$ AFTER ERROR PRINT
ERROR+5 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN)
INC RTRY ;COUNT ONE TRY
CMP #3,RTRY ;TRIED THREE TIMES
BNE 1$ ;BR IF NOT
JMP $EOP ;GO TO END OF PASS ROUTINE

```

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

006330 012706 001100
006334 012737 013052 000034
006342 012737 000340 000036
006350 012737 010732 000030
006356 012737 000340 000032
006364 104401 016664
006370 104401 020306
006374 104410
006376 012602
006400 020227 176176
006404 101065
006406 020227 175616
006412 103462
006414 132702 000001
006420 001057
006422 010203
006424 142703 000370
006430 122703 000006
006434 001051
006436 010267 172540
006442 016746 171336
006446 012767 006460 171330
006454 005712
006456 000412
006460 004767 005460
006464 012767 006474 172552
006472 104006
006474 022626
006476 012667 171302
006502 000426
006504 012667 171274

```
: THIS IS PROGRAM #2
: THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
: A) SELECTION OF A TRANSMITTER DATA BUFFER
: B) SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
: C) SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
:   BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
: D) A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
:
PRG2:  MOV     #STACK,SP           ;INITIALIZE THE STACK POINTER
      MOV     #STRAP,2#TRAPVEC    ;TRAP VECTOR FOR TRAP CALLS
      MOV     #340,3#TRAPVEC+2    ;LEVEL 7
      MOV     #ERROR,2#EMTVEC     ;EMT VECTOR FOR ERROR ROUTINE
      MOV     #340,3#EMTVEC+2     ;LEVEL 7
      TYPE    .PRG2M              ;INDICATE THAT USER SELECTED
: PROGRAM #2
PRG2A: TYPE    .LINTAD           ;ASK USER FOR THE TRANSMITTER
: DATA BUFFER ADDRESS OF THE DEVICE
: HE WISHES TO TEST
      RDOCT                       ;ACCEPT THE ANSWER TYPED BY USER
: AND STORE ON TOP OF STACK
: CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
      MOV     (SP)+,R2            ;GET THE ANSWER TYPED
      CMP     R2,#176176         ;IS THE NUMBER TOO HIGH?
      BHI     REDO1              ;IF YES - GO TO RETRY SITUATION
      CMP     R2,#175616         ;IS THE NUMBER TOO LOW?
      BLO     REDO1              ;IF YES - GO TO RETRY SITUATION
      BITB   #BIT0,R2           ;NUMBER IS IN RANGE BUT IS IT
: ON AN EVEN BOUNDARY?
      BNE     REDO1              ;IF NOT GO TO RETRY SITUATION
: CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
      MOV     R2,R3              ;GET THE USER RESPONSE
      BICB   #370,R3            ;MASK OFF LOWER BYTE EXCEPT FOR
: LEAST SIGNIFICANT DIGIT
      CMPB   #6,R3              ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
: USER RESPONSE EQUAL TO A SIX?
      BNE     REDO1              ;BRANCH IF NOT
      MOV     R2,$TMP0          ;THE TRANSMITTER ADDRESS
: TYPED IS OK - STORE FOR
: FUTURE USE
: NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
      MOV     ERRVEC,-(SP)       ;SAVE THE TIMEOUT VECTOR
      MOV     #2$,ERRVEC        ;SET UP TIMEOUT SERVICE ADDRESS
      TST    (R2)               ;IF PRESENT WE WILL EXECUTE THE
: NEXT INSTRUCTION - IF NOT
: WE GO TO 2$:
      BR     4$                 ;BRANCH IF PRESENT
2$:   JSR    PC,SUERT2          ;GO SET UP FOR ERROR INFORMATION
      MOV     #3$,SESCAPE       ;POINT OF RETURN AFTER ERROR REPORT
      ERROR  +6                 ;XDBR REFERENCE CAUSED TIMEOUT
3$:   CMP    (SP)+,(SP)+        ;CLEAN STACK FROM TIMEOUT
      MOV     (SP)+,ERRVEC      ;RESTORE TIMEOUT VECTOR
      BR     REDO1              ;GO TO RETRY SITUATION
4$:   MOV    (SP)+,ERRVEC       ;DEVICE REGISTER IS PRESENT!
: RESTORE TIMEOUT VECTOR
: WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
```

MAINDEC-11-DZDLC-A
DZDLC.A.P11 T25

MACY11 27(732) 20-SEP-76 09:19 PAGE 43
TEST TO TURN AROUND WORST CASE PATTERN

```

1806      :DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN
1807      :SUCCESSIVE CHARACTER TRANSFERS
1808 006510 104401 020367      PRG2B: TYPE      .SELCAR      ;ASK USER FOR THE CHARACTER HE
1809      ;WISHES TO TRANSFER
1810 006514 104410      RDOCT      ;ACCEPT THE ANSWER TYPED BY
1811      ;USER AND STORE ON TOP OF STACK
1812 006516 012667 172462      MOV      (SP)+,$TMP1      ;GET THE ANSWER TYPED
1813      ;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
1814      ;OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
1815 006522 104401 020475      TYPE      .SELDLY      ;ASK THE USER FOR THE DELAY
1816      ;IN MSEC (OCTAL NO.) BETWEEN
1817      ;CHARACTER TRANSFERS
1818 006526 104410      RDOCT      ;ACCEPT THE ANSWER TYPED BY
1819      ;USER AND STORE ON TOP OF STACK
1820 006530 012667 172452      MOV      (SP)+,$TMP2      ;GET THE ANSWER TYPED
1821 006534 116767 172446 000012 1$:  MOVB     $TMP2,2$      ;SET THE DELAY COUNT ARGUMENT
1822      ;FOR TIMER ROUTINE
1823 006542 116777 172436 172432  MOVB     $TMP1,2$TMP0      ;LOAD THE TRANSMITTER DATA
1824      ;BUFFER WITH THE CHARACTER
1825 006550 004767 004750      JSR      PC,DELAY      ;GO OFF TO WAIT THE SPECIFIED

```

```

1826                                     ;NO. OF MSEC. BEFORE ISSUING
1827                                     ;ANOTHER CHARACTER
1828 006554 000000 2$: .WORD 0 ;THIS IS WHERE THE DELAY COUNT RESIDES
1829 006556 000766 BR 1$ ;GO BACK TO ISSUE ANOTHER CHARACTER
1830 006560 104401 001252 RED01: TYPE $QUES ;TYPE A QUESTION MARK(?)
1831 006564 000167 177600 JMP PRG2A ;REITERATE THE XDBR QUESTION TO USER
1832                                     ;THIS IS PROGRAM #3
1833                                     ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
1834                                     A) SELECTION OF A RECEIVER DATA BUFFER
1835                                     B) SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
1836                                     C) SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
1837                                     BETWEEN EACH RECEIVER DATA BUFFER CHARACTER TRANSFER
1838                                     D) A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
1839
1840
1841 006570 012706 001100 PRG3: MOV #STACK,SP ;INITIALIZE THE STACK POINTER
1842 006574 012737 013052 000034 MOV #STRAP,#TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
1843 006602 012737 000340 000036 MOV #340,#TRAPVEC+2 ;LEVEL 7
1844 006610 012737 010732 000030 MOV #ERROR,#EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
1845 006616 012737 000340 000032 MOV #340,#EMTVEC+2 ;LEVEL 7
1846 006624 104401 016730 TYPE ,PRG3M ;INDICATE THAT USER SELECTED
1847                                     PROGRAM #3
1848 006630 104401 020570 PRG3A: TYPE ,LINRAD ;ASK USER FOR THE RECEIVER DATA
1849                                     BUFFER ADDRESS OF THE DEVICE
1850                                     HE WISHES TO TEST
1851 006634 104410 RDOCT ;ACCEPT THE ANSWER TYPED BY
1852                                     USER AND STORE ON TOP OF STACK
1853 ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1854 006636 012602 MOV (SP)+,R2 ;GET THE ANSWER TYPED
1855 006640 020227 176172 CMP R2,#176172 ;IS THE NUMBER TOO HIGH?
1856 006644 101071 BHI RED02 ;IF YES - GO TO RETRY SITUATION
1857 006646 020227 175612 CMP R2,#175612 ;IS THE NUMBER TOO LOW?
1858 006652 103466 BLO RED02 ;IF YES - GO TO RETRY SITUATION
1859 006654 132702 000001 BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
1860                                     ON AN EVEN BOUNDARY?
1861 006660 001063 BNE RED02 ;IF NOT - GO TO RETRY SITUATION
1862 ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR DBR ADDRESS
1863 006662 010203 MOV R2,R3 ;GET THE USER RESPONSE
1864 006664 142703 000370 BICB #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
1865                                     LEAST SIGNIFICANT DIGIT
1866 006670 122703 000002 CMPB #2,R3 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1867                                     USER RESPONSE EQUAL TO A TWO?
1868 006674 001055 BNE RED02 ;BRANCH IF NOT
1869 006676 010267 172300 MOV R2,$TMP0 ;THE RECEIVER ADDRESS TYPED IS
1870                                     OK - STORE FOR FUTURE USE
1871 ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1872 006702 016746 171076 MOV ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
1873 006706 012767 006720 171070 MOV #2$,ERRVEC ;SET UP TIMEOUT SERVICE ADDRESS
1874 006714 005712 TST (R2) ;IF PRESENT WE WILL EXECUTE THE
1875                                     NEXT INSTRUCTION - IF NOT WE
1876                                     GO TO 2$:
1877 006716 000412 BR 4$ ;BRANCH IF PRESENT
1878 006720 004767 005220 2$: JSR PC,SUERT2 ;GO SET UP FOR ERROR INFORMATION
1879 006724 012767 005734 172312 MOV #3$, $ESCAPE ;POINT OF RETURN AFTER ERROR REPORT
1880 006732 104006 ERROR +6 ;RDBR REFERENCE CAUSED TIMEOUT
1881 006734 022626 3$: CMP (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT

```



```

1938                                     ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1939 007106 012602                       MOV (SP)+,R2 ;GET THE ANSWER TYPED
1940 007110 020227 176176                CMP R2,#176176 ;IS THE NUMBER TOO HIGH?
1941 007114 101136                        BHI RED03 ;IF YES - GO TO RETRY SITUATION
1942 007116 020227 175616                CMP R2,#175616 ;IS THE NUMBER TOO LOW?
1943 007122 103533                        BLO RED03 ;IF YES - GO TO RETRY SITUATION
1944 007124 132702 000001                BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
1945                                     ;ON AN EVEN BOUNDARY?
1946 007130 001130                        BNE RED03 ;IF NO - GO TO RETRY SITUATION
1947                                     ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
1948 007132 010203                        MOV R2,R3 ;GET THE USER RESPONSE
1949 007134 142703 000370                BICB #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
1950                                     ;LEAST SIGNIFICANT DIGIT
1951 007140 122703 000006                CMPB #6,R3 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1952                                     ;USER RESPONSE EQUAL TO A SIX?
1953 007144 001122                        BNE RED03 ;BRANCH IF NOT
1954 007146 010267 172030                MOV R2,$TMP0 ;THE TRANSMITTER ADDRESS TYPED
1955                                     ;IS OK - STORE FOR FUTURE USE
1956                                     ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1957 007152 016746 170626                MOV ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
1958 007156 012767 007170 170620        MOV #2$,ERRVEC ;SET UP TIMEOUT SERVICE ADDRESS
1959 007164 005712                        TST (R2) ;IF PRESENT WE WILL EXECUTE THE
1960                                     ;NEXT INSTRUCTION - IF NOT WE
1961                                     ;GO TO 2$:
1962 007166 000412                        BR 4$ ;BRANCH IF PRESENT
1963 007170 004767 004750 2$:           JSR PC,SUERT2 ;GO SET UP FOR ERROR INFORMATION
1964 007174 012767 007204 172042        MOV #3$, $ESCAPE ;POINT OF RETURN AFTER ERROR REPORT
1965 007202 104006                        ERROR +6 ;XDBR REFERENCE CAUSED TIMEOUT
1966 007204 022626 3$:           CMP (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
1967 007206 012667 170572                MOV (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
1968 007212 000477                        BR RED03 ;GO TO RETRY SITUATION
1969 007214 012667 170564 4$:           MOV (SP)+,ERRVEC ;DEVICE REGISTER IS PRESENT!
1970                                     ;RESTORE TIMEOUT VECTOR
1971 007220 104401 020646                TYPE ,RSTALL ;ASK THE USER IF HE DESIRES SOME
1972                                     ;RANDOM NO. OF MSEC. WAIT TIME
1973                                     ;BEFORE CHECKING FOR XCSR DONE
1974                                     ;FLAG
1975 007224 104410                        RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1976                                     ;AND STORE ON TOP OF STACK
1977 007226 012667 171754                MOV (SP)+,$TMP2 ;GET THE ANSWER TYPED
1978                                     ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED
1979 007232 104401 020367  PRG4B:      TYPE ,SELCAR ;ASK USER FOR THE CHARACTER HE
1980                                     ;WISHES TO TRANSFER
1981 007236 104410                        RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1982                                     ;AND STORE ON TOP OF STACK
1983 007240 012667 171740                MOV (SP)+,$TMP1 ;GET THE ANSWER TYPED
1984                                     ;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE OCTAL
1985                                     ;ASCII EQUIVALENT OF THE CHARACTER E.G. C=103
1986                                     ;
1987 007244 104401 017217  PRG4C:      TYPE, LENGTH ;ASK USER FOR THE CHARACTER LENGTH
1988                                     ;FOR WHICH HIS DEVICE IS SET
1989 007250 104411                        RDECT ;ACCEPT THE ANSWER TYPED BY USER
1990                                     ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1991 007252 012600                        MOV (SP)+,R0 ;GET THE ANSWER TYPED
1992 007254 020027 000010                CMP R0,#8 ;IS THE NUMBER TOO HIGH?
1993 007260 101060                        BHI RED03A ;IF YES - GO TO RETRY SITUATION

```

MAINDEC-11-DZDLC-A
DZDLC.A.P11 T25

MACY11 27(732) 20-SEP-76 09:19 PAGE 47
TEST TO TURN AROUND WORST CASE PATTERN

```

1994 007262 020027 000005          CMP      RO,#5          ;IS THE NUMBER TOO LOW?
1995 007266 103455          BLO     RED03A         ;IF YES - GO TO RETRY SITUATION
1996 007270 010067 171740          MOV     RO,$TMP15      ;THE VALUE TYPED IS OK
1997                                     ;STORE FOR FUTURE USE
1998 007274 016767 171702 171706          MOV     $TMP0,$TMP3   ;GET THE XDBR ADDRESS
1999 007302 162767 000002 171700          SUB     #2,$TMP3      ;FORM THE XCSR ADDRESS
2000 007310 005767 171672          1$:    TST     $TMP2     ;DO WE RANDOM STALL?
2001 007314 001402          BEQ     2$            ;BRANCH IF IT WASN'T DESIRED
2002 007316 004767 004246          JSR     PC,$STALL     ;GO STALL RANDOM VALUE OF MSEC.
2003 007322 004767 004352          2$:    JSR     PC,$TIMETX ;GO WAIT FOR TRANSMITTER DONE
2004                                     ;BIT TO SET
2005 007326 104401 017104          TYPE   ,XDB          ;TYPE TRANSMITTER DONE BIT MESSAGE
2006 007332 104000          ERROR  +0           ;XCSR DONE BIT NEVER SET
2007 007334 052777 000004 171646          BIS    #BIT2,$TMP3   ;SET THE MAINTENANCE BIT IN THE
2008                                     ;TRANSMITTER CONTROL STATUS REGISTER
2009 007342 016777 171636 171632          MOV     $TMP1,$TMP0   ;LOAD TRANSMITTER DATA BUFFER
2010                                     ;WITH SELECTED CHARACTER
2011 007350 004767 004306          JSR     PC,$TIMERX    ;GO WAIT FOR RECEIVER DONE BIT
2012                                     ;TO SET
2013 007354 104401 017153          TYPE   ,RDB          ;TYPE RECEIVER DONE BIT MESSAGE
2014 007360 104000          ERROR  +0           ;RCSR DONE BIT NEVER SET
2015 007362 016767 171622 171622          MOV     $TMP3,$TMP4   ;GET THE TRANSMITTER CONTROL
2016                                     ;STATUS REGISTER ADDRESS
2017 007370 162767 000002 171614          SUB     #2,$TMP4     ;FORM THE RECEIVER DATA BUFFER
2018                                     ;ADDRESS
2019 007376 017767 171610 171610          MOV     $TMP4,$TMP5   ;STORE THE CHARACTER FROM THE
2020                                     ;RECEIVER BUFFER + REST OF CONTENTS
2021 007404 004767 004326          JSR     PC,$DATCHK    ;GO TO COMPARE EXPECTED & RECEIVED
2022                                     ;DATA
2023 007410 000737          BR     1$            ;GO BACK TO ISSUE ANOTHER CHARACTER
2024 007412 104401 001252          RED03: TYPE ,SQUES   ;TYPE A QUESTION MARK(?)
2025 007416 000167 177456          JMP     PRG4A         ;REITERATE THE XDBR QUESTION TO USER
2026 007422 104401 001252          RED03A: TYPE, SQUES  ;TYPE '?' INDICATING USER TYPED
2027                                     ;SOMETHING WRONG FOR CHARACTER LENGTH
2028 007426 000167 177612          JMP     PRG4C         ;GO BACK TO REISSUE QUESTION
2029
2030                                     ;THIS IS PROGRAM #5
2031                                     ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
2032                                     ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
2033
2034
2035 007432 012706 001100          PRG5:  MOV     #STACK,SP ;INITIALIZE THE STACK POINTER
2036 007436 012737 013052 000034          MOV     #STRAP,$TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
2037 007444 012707 000340 000036          MOV     #340,$TRAPVEC+2 ;LEVEL 7
2038 007452 012737 010732 000030          MOV     #ERROR,$EMTVEC  ;EMT VECTOR FOR ERROR ROUTINE
2039 007460 012737 000340 000032          MOV     #340,$EMTVEC+2  ;LEVEL 7
2040 007466 104401 017040          TYPE   ,PRG5M        ;INDICATE THAT USER SELECTED
2041                                     ;PROGRAM #5
2042 007472 104401 020306          PRG5A: TYPE ,LINTAD   ;ASK USER FOR THE TRANSMITTER DATA
2043                                     ;BUFFER ADDRESS OF THE DEVICE
2044                                     ;HE WISHES TO TEST
2045 007476 104410          RDOCT                ;ACCEPT THE ANSWER TYPED BY USER
2046                                     ;AND STORE ON TOP OF STACK
2047                                     ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
2048 007500 012602          MOV     (SP)+,R2     ;GET THE ANSWER TYPED
2049 007502 020227 176176          CMP     R2,#176176   ;IS THE NUMBER TOO HIGH?

```



```

2050 007506 101152          BHI    RED04          ; IF YES - GO TO RETRY SITUATION
2051 007510 020227 175616  CMP    R2,#175616    ; IS THE NUMBER TOO LOW?
2052 007514 103547          BLO    RED04          ; IF YES - GO TO RETRY SITUATION
2053 007516 132702 000001  BITB   #BIT0,R2     ; NUMBER IS IN RANGE BUT IS IT
2054                                     ; ON AN EVEN BOUNDARY?
2055 007522 001144          BNE    RED04          ; IF NOT - GO TO RETRY SITUATION
2056                                     ; CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
2057 007524 010203          MOV    R2,R3         ; GET THE USER RESPONSE
2058 007526 142703 000370  BICB   #370,R3 ; MASK OFF LOWER BYTE EXCEPT FOR
2059                                     ; LEAST SIGNIFICANT DIGIT
2060 007532 122703 000006  CMPB   #6,R3        ; WAS THE LEAST SIGNIFICANT DIGIT OF THE
2061                                     ; USER RESPONSE EQUAL TO A SIX?
2062 007536 001136          BNE    RED04          ; BRANCH IF NOT
2063 007540 010267 171436  MOV    R2,$TMP0     ; THE TRANSMITTER ADDRESS TYPED
2064                                     ; IS OK - STORE FOR FUTURE USE
2065                                     ; NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2066 007544 016746 170234  MOV    ERRVEC,-(SP)  ; SAVE THE TIMEOUT VECTOR
2067 007550 012767 007562 170226  MOV    #2,$ERRVEC   ; SET UP TIMEOUT SERVICE ADDRESS
2068 007556 005712          TST    (R2)          ; IF PRESENT WE WILL EXECUTE THE
2069                                     ; NEXT INSTRUCTION - IF NOT WE
2070                                     ; GO TO 2$:
2071 007560 000412          BR     4$            ; BRANCH IF PRESENT
2072 007562 004767 004356 2$: JSR    PC,SUERT2     ; GO SETUP FOR ERROR INFORMATION
2073 007566 012767 007576 171450  MOV    #3,$$ESCAPE  ; POINT OF RETURN AFTER ERROR REPORT
2074 007574 104006          ERROR  +6           ; XDR REFERENCE CAUSED TIMEOUT
2075 007576 022626 3$:  CMP    (SP)+,(SP)+  ; CLEAN STACK FROM TIMEOUT
2076 007600 012667 170200  MOV    (SP)+,ERRVEC ; RESTORE TIMEOUT VECTOR
2077 007604 000513          BR     RED04         ; GO TO RETRY SITUATION
2078 007606 012667 170172 4$:  MOV    (SP)+,ERRVEC ; DEVICE REGISTER IS PRESENT!
2079                                     ; ASK THE USER IF HE DESIRES SOME
2080                                     ; RANDOM NO. OF MSEC. WAIT TIME
2081                                     ; BEFORE CHECKING XCSR DONE FLAG
2082 007612 104401 017217  PRG5C: TYPE, LENGTH ; ASK USER FOR THE CHARACTER LENGTH
2083                                     ; FOR WHICH HIS DEVICE IS SET
2084 007616 104411          RDEC                                     ; ACCEPT THE ANSWER TYPED BY USER
2085                                     ; CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2086 007620 012600          MOV    (SP)+,R0     ; GET THE ANSWER TYPED
2087 007622 020027 000010  CMP    R0,#8        ; IS THE NUMBER TOO HIGH?
2088 007626 101106          BHT   RED04A        ; IF YES - GO TO RETRY SITUATION
2089 007630 020027 000005  CMP    R0,#5        ; IS THE NUMBER TOO LOW?
2090 007634 103503          BLO   RED04A        ; IF YES - GO TO RETRY SITUATION
2091 007636 010067 171372  MOV    R0,$TMP15    ; THE VALUE TYPED IS OK
2092                                     ; STORE FOR FUTURE USE
2093 007642 104401 020646          TYPE  ,RSTALL      ; RANDOM NO. OF MSEC. WAIT TIME
2094 007646 104410          RDOCT                                     ; ACCEPT THE ANSWER TYPED BY USER
2095                                     ; AND STORE ON TOP OF STACK
2096 007650 012667 171332  MOV    (SP)+,$TMP2  ; GET THE ANSWER TYPED
2097                                     ; WE ARE NOW READY TO INITIALIZE THE BINARY COUNT AND GET
2098                                     ; THE BINARY CHARACTER
2099                                     ;
2100 007654 012767 177777 171342  MOV    #-1,$TMP11   ; SET LEAD IN VARIABLE TO -1
2101 007662 016767 171336 171336  PRG5B: MOV    $TMP11,$TMP12 ; STORE PREVIOUS BINARY CHARACTER
2102 007670 005267 171332          INC    $TMP12       ; FLIP BINARY CHARACTER AGAIN
2103 007674 042767 177400 171324  BIC    #177400,$TMP12 ; MASK TO 2 BITS
2104 007702 016767 171320 171314  MOV    $TMP12,$TMP11 ; STORE BINARY CHARACTER
2105 007710 016767 171312 171266  MOV    $TMP12,$TMP1 ; STORE BINARY CHARACTER

```

```

2106                                     ;FOR FUTURE USE
2107 007716 016767 171260 171264      MOV   $TMP0,$TMP3      ;GET THE XDBR ADDRESS
2108 007724 000002 171256              SUB   #2,$TMP3       ;FORM THE XCSR ADDRESS
2109 007732 005767 171250              1$:  TST   $TMP2       ;DO WE RANDOM STALL?
2110 007736 001402                      BEQ   2$            ;BRANCH IF IT WASN'T DESIRED
2111 007740 004767 003624              JSR   PC,$STALL     ;GO STALL RANDOM VALUE OF M:SEC.
2112 007744 004767 003730              2$:  JSR   PC,$TIMETX  ;GO WAIT FOR TRANSMITTER DONE
2113                                     ;BIT TO SET
2114 007750 104401 017104              TYPE  ,XDB          ;TYPE TRANSMITTER DONE BIT MESSAGE
2115 007754 104000                      ERROR +0            ;XCSR DONE BIT NEVER SET
2116 007756 052777 000004 171224      BIS   #BIT2,$TMP3   ;SET THE MAINTENANCE BIT IN THE
2117                                     ;TRANSMITTER CONTROL STATUS REGISTER
2118 007764 016777 171214 171210      MOV   $TMP1,$TMP0   ;LOAD TRANSMITTER DATA BUFFER
2119                                     ;WITH SELECTED CHARACTER
2120 007772 004767 003664              JSR   PC,$TIMERX    ;GO WAIT FOR RECEIVER DONE BIT TO SET
2121 007776 104401 017153              TYPE  ,RDB          ;TYPE RECEIVER DONE BIT MESSAGE
2122 010002 104000                      ERROR +0            ;RCSR DONE BIT NEVER SET
2123 010004 016767 171200 171200      MOV   $TMP3,$TMP4   ;GET THE TRANSMITTER CONTROL
2124                                     ;STATUS REGISTER ADDRESS
2125 010012 162767 000002 171172      SUB   #2,$TMP4     ;FORM THE RECEIVER DATA BUFFER
2126                                     ;ADDRESS
2127 010020 017767 171166 171166      MOV   $TMP4,$TMP5   ;STORE THE CHARACTER FROM THE
2128                                     ;RECEIVER BUFFER + REST OF CONTENTS
2129 010026 004767 003704              JSR   PC,$DATCHK    ;GO TO COMPARE EXPECTED & RECEIVED
2130                                     ;DATA
2131 010032 000713                      BR    PRG5B         ;GO BACK TO ISSUE ANOTHER BINARY
2132                                     ;CHARACTER
2133 010034 104401 001252              RED04: TYPE  $QUES   ;TYPE A QUESTION MARK(?)
2134 010040 000167 177426              JMP    PRG5A        ;GO BACK TO REITERATE XDBR QUESTION
2135 010044 104401 001252              RED04A: TYPE, $QUES ;TYPE '?' INDICATING USER TYPED
2136                                     ;SOMETHING WRONG FOR CHARACTER LENGTH
2137 010050 000167 177536              JMP    PRG5C        ;GO BACK TO REISSUE QUESTION
2138
2139                                     ; THIS ROUTINE WILL SET UP:
2140                                     ; RCSR - RECEIVER STATUS REGISTER
2141                                     ; RBUF - RECEIVER BUFFER REGISTER
2142                                     ; XCSR - TRANSMITTER STATUS REGISTER
2143                                     ; XBUF - TRANSMITTER BUFFER REGISTER
2144                                     ; INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS
2145                                     ; TESTED, AND THEREAFTER, AT THE END OF A PROGRAM TO CYCLE THRU
2146                                     ; ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED)
2147 010054 016757 171200 171326      DLADDR: MOV   DLBASE,DLRCSR ;STORE RECEIVER STATUS REGISTER
2148                                     ; OF CURRENT DEVICE
2149 010062 062767 000002 171170      ADD   #2,DLBASE     ;FORM RECEIVER BUFFER REGISTER
2150                                     ; OF CURRENT DEVICE
2151 010070 016767 171164 171314      MOV   DLBASE,DLRDBR ;STORE RECEIVER BUFFER REGISTER
2152                                     ; OF CURRENT DEVICE
2153 010076 062767 000002 171154      ADD   #2,DLBASE     ;FORM TRANSMITTER STATUS REGISTER
2154                                     ; OF OF CURRENT DEVICE
2155 010104 016767 171150 171302      MOV   DLBASE,DLXCSR ;STORE TRANSMITTER STATUS REGISTER
2156                                     ; OF CURRENT DEVICE
2157 010112 062767 000002 171140      ADD   #2,DLBASE     ;FOR TRANSMITTER BUFFER REGISTER
2158                                     ; OF CURRENT DEVICE
2159 010120 016767 171134 171270      MOV   DLBASE,DLXDBR ;STORE TRANSMITTER BUFFER REGISTER
2160                                     ; OF CURRENT DEVICE
2161 010126 000207                      RTS   PC            ;RETURN

```

```

2162
2163      .SBTTL  END OF PASS ROUTINE
2164
2165      ;*****
2166      ;*INCREMENT THE PASS NUMBER ($PASS)
2167      ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
2168      ;*IF THERES A MONITOR GO TO IT
2169      ;*IF THERE ISN'T JUMP TO RESTRT
2170
2171 01013C
2172
2173      ;SEOP:
2174      ;THIS NEXT SECTION UP TO THE NEXT LINE OF ASTERISKS WAS
2175      ;SUPPLIED BY THE MACRO 'EOPBEG'. THE MACRO NAME APPEARS IN
2176      ;THE SOURCE PROGRAM AS ONE OF THE ARGUMENTS TO THE .SEOP
2177      ;SYSMAC UTILITY ROUTINE CALL.
2178
2179      SCOPE
2180      TSTB  MULTD      ;ARE WE RUNNING MULTIPLE DEVICES?
2181      BEQ   3$        ;BRANCH IF NOT FOR NORMAL
2182
2183      TST  ACTREG     ;'END PASS # XX' TYPEOUT
2184      BNE  1$        ;ARE ANY DEVICES ACTIVE?
2185      TYPE ,FOULUP   ;BRANCH IF YES
2186
2187      HALT
2188      CLR  MULTD     ;INDICATE SOMETHING WRONG!
2189      CLR  TABFLG    ;MULTIPLE DEVICES ARE BEING
2190      JMP  RESTRT    ;RUN SUPPOSEDLY, BUT NONE ARE
2191
2192      ADD  #10,BASEADD ;SHOWN ACTIVE
2193
2194      ADD  #10,BASEIV ;WAIT FOR A USER RESPONSE
2195
2196      CLC
2197      ROL  ROTADD    ;CLEAR MULTIPLE DEVICE FLAG
2198
2199      BCS  2$        ;CLEAR TABLE CREATION FLAG
2200
2201      BIT  ROTADD,ACTREG ;CLEAR LAST DEVICE INDICATOR
2202
2203      BEQ  1$        ;UPDATE NEXT POSSIBLE DEVICE ACTIVE
2204
2205      MOV  BASEADD,DLBASE ;POINTER
2206
2207      MOV  BASEIV,DLVECT ;BRANCH IF THIS WAS THE
2208
2209      NOP
2210      JSR  PC,DLADDR  ;LAST DEVICE TO BE TESTED ON
2211
2212      CLR  $TSTNM    ;THIS PASS
2213
2214      CLR  $DLXCSR   ;IS THIS DEVICE TRULY ACTIVE
2215
2216      CLR  $DLRCSR   ;(AS PER USER)
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237

```

```

2218 010262 005777 171124          TST    @DLRDBR      ;FLUSH RCVR "DONE" BIT
2219 010266 005777 171120          TST    @DLRDBR
2220 010272 000167 172472          JMP    TST1        ;START TESTING THIS DEVICE
2221 010276
2222
2223
2224
2225 010276 012767 000001 170772    MOV    #1,ROTADD   ;SET UP ROTATING POINTER FOR NEXT
2226                                     ;MULTIPLE PASS
2227 010304 016767 170752 170752    MOV    KEEPADD,BASEADD ;RESTORE BASE ADDRESS
2228 010312 016767 170750 170750    MOV    KEEPIV,BASEIV  ;RESTORE BASE INTERRUPT VECTOR
2229 010320 016767 170740 170732    MOV    BASEADD,DLBASE ;RESTORE 1ST DEVICE BASE ADDRESS
2230 010326 016767 170736 171064    MOV    BASEIV,DLVECT ;RESTORE 1ST DEVICE VECTOR ADDRESS
2231 010334 000240
2232 010336 004767 177512          JSR    PC,DLADDR   ;FORM ADDRESSES FOR 1ST DEVICE
2233 010342
2234
2235 010342 005067 170534          CLR    $STNM       ;;ZERO THE TEST NUMBER
2236 010346 005067 170670          CLR    $TIMES     ;;ZERO THE NUMBER OF ITERATIONS
2237 010352 005267 170522          INC    $PASS      ;;INCREMENT THE PASS NUMBER
2238 010356 042757 100000 170514    BIC    #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
2239 010364 000001          DEC    (PC)+      ;;LOOP?
2240 010366 000001          $EOPCT: .WORD    1
2241 010370 003022          BGT    $DOAGN     ;;YES
2242 010372 012737          MOV    (PC)+,@(PC)+ ;RESTORE COUNTER
2243 010374 000001          $ENDCT: .WORD    1
2244 010376 010366          $EOPCT
2245 010400 104401 010445          TYPE   $SENDMG   ;;TYPE "END PASS #"
2246 010404 016746 170470          MOV    $PASS,-(SP) ;SAVE $PASS FOR TYPEOUT
2247 010410 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
2248 010412 104401 010442          TYPE   $ENULL    ;;TYPE A NULL CHARACTER
2249 010416 013700 000042          $GET42: MOV    @#42,R0 ;GET MONITOR ADDRESS
2250 010422 001405          BEQ    $DOAGN    ;;BRANCH IF NO MONITOR
2251 010424 000005          RESET          ;;CLEAR THE WORLD
2252 010426 004710          $ENDAD: JSR    PC,(R0) ;GO TO MONITOR
2253 010430 000240          NOP
2254 010432 000240          NOP
2255 010434 000240          NOP
2256 010436          $DOAGN:
2257 010436 000137          JMP    @(PC)+    ;;RETURN
2258 010440 001772          $RTNAD: .WORD   RESTR
2259 010442 377 377 000          $ENULL: .BYTE   -1,-1,0 ;NULL CHARACTER STRING
2260 010445 015 042412 042116          $ENDMG: .ASCIZ  <15><12>/END PASS #/
2261 010452 050040 051501 020123
2262 010460 000043

```

.SBTTL SCOPE HANDLER ROUTINE

```

2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273

```

```

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

```

```

2274 ;*CALL
2275 ;* SCOPE ;;SCOPE=IOT
2276
2277 010462 $SCOPE:
2278 010462 032777 040000 170450 1$: BIT #BIT14, @SWR ;; LOOP ON PRESENT TEST?
2279 010470 001111 BNE $OVER ;; YES IF SW14=1
2280 ;*****START OF CODE FOR THE XOR TESTER*****
2281 010472 000416 $XTSTR: BR 6$ ;; IF RUNNING ON THE "XOR" TESTER CHANGE
2282 ; THIS INSTRUCTION TO A "NOP" (NOP=240)
2283 010474 013746 000004 MOV @#ERRVEC, -(SP) ;; SAVE THE CONTENTS OF THE ERROR VECTOR
2284 010500 012737 010520 000004 MOV #5$, @#ERRVEC ;; SET FOR TIMEOUT
2285 010506 005737 177060 TST @#177060 ;; TIME OUT ON XOR?
2286 010512 012637 000004 MOV (SP)+, @#ERRVEC ;; RESTORE THE ERROR VECTOR
2287 010516 000463 BR $SVLAD ;; GO TO THE NEXT TEST
2288 010520 022626 5$: CMP (SP)+, (SP)+ ;; CLEAR THE STACK AFTER A TIME OUT
2289 010522 012637 000004 MOV (SP)+, @#ERRVEC ;; RESTORE THE ERROR VECTOR
2290 010526 000423 BR 7$ ;; LOOP ON THE PRESENT TEST
2291 010530 6$: ;*****END OF CODE FOR THE XOR TESTER*****
2292 010530 032777 000400 170402 BIT #BIT08, @SWR ;; LOOP ON SPEC. TEST?
2293 010536 001404 BEQ 2$ ;; BR IF NO
2294 010540 127767 170374 170334 CMPB @SWR, $STNM ;; ON THE RIGHT TEST? SWR<7:0>
2295 010546 001462 BEQ $OVER ;; BR IF YES
2296 010550 105767 170327 2$: TSTB $ERFLG ;; HAS AN ERROR OCCURRED?
2297 010554 001421 BEQ 3$ ;; BR IF NO
2298 010556 126767 170333 170317 CMPB $ERMAX, $ERFLG ;; MAX. ERRORS FOR THIS TEST OCCURRED?
2299 010564 101015 BHI 3$ ;; BR IF NO
2300 010566 032777 001000 170344 BIT #BIT09, @SWR ;; LOOP ON ERROR?
2301 010574 001404 BEQ 4$ ;; BR IF NO
2302 010576 016767 170306 170302 7$: MOV $LPERR, $LPADR ;; SET LOOP ADDRESS TO LAST SCOPE
2303 010604 000443 BR $OVER
2304 010606 105067 170271 4$: CLRB $ERFLG ;; ZERO THE ERROR FLAG
2305 010612 005067 170424 CLR $TIMES ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
2306 010616 000415 BR 1$ ;; ESCAPE TO THE NEXT TEST
2307 010620 032777 004000 170312 3$: BIT #BIT11, @SWR ;; INHIBIT ITERATIONS?
2308 010626 001011 BNE 1$ ;; BR IF YES
2309 010630 005767 170244 TST $PASS ;; IF FIRST PASS OF PROGRAM
2310 010634 001406 BEQ 1$ ;; INHIBIT ITERATIONS
2311 010636 005267 170242 INC $ICNT ;; INCREMENT ITERATION COUNT
2312 010642 026767 170374 170234 CMP $TIMES, $ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
2313 010650 002021 BGE $OVER ;; BR IF MORE ITERATION REQUIRED
2314 010652 012767 000001 170224 1$: MOV #1, $ICNT ;; REINITIALIZE THE ITERATION COUNTER
2315 010660 016767 000044 170354 MOV $MXCNT, $TIMES ;; SET NUMBER OF ITERATIONS TO DO
2316 010666 105267 170210 $SVLAD: INCB $STNM ;; COUNT TEST NUMBERS
2317 010672 011667 170210 MOV (SP), $LPADR ;; SAVE SCOPE LOOP ADDRESS
2318 010676 011667 170206 MOV (SP), $LPERR ;; SAVE ERROR LOOP ADDRESS
2319 010702 005067 170336 CLR $ESCAPE ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2320 010706 112767 000001 170201 MOVB #1, $ERMAX ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2321 010714 016777 170162 170220 $OVER: MOV $STNM, @DISPLAY ;; DISPLAY TEST NUMBER
2322 010722 016716 170160 MOV $LPADR, (SP) ;; FUDGE RETURN ADDRESS
2323 010726 000002 RTI ;; FIXES PS
2324 010730 000100 $MXCNT: 100 ;; MAX. NUMBER OF ITERATIONS
2325
2326 .SBTTL ERROR HANDLER ROUTINE
2327
2328 ;*****
2329 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.

```



```

011114 005300
011116 006300
011120 006300
011122 006300
011124 062700 001310
011130 012067 000004
011134 001404
011136 104401
011140 000000
011142 104401 001253
011144 012067 000004
011146 001404
011152 104401
011154 000000
011156 104401 001253
011160 104401
011164 011000
011166 001004
011170 012600
011172 104401 001253
011176 000207
011200
011202 013046
011204 104402
011206 005710
011210 001770
011214 104401 011216
011216 000771
011222 020040 000
011222 011222

```

```

1$: DEC RO ::ADJUST THE INDEX SO THAT IT WILL
ASL RO :: WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD $ERRTB,RO ::FORM TABLE POINTER
MOV (RO)+,2$ ::PICKUP "ERROR MESSAGE" POINTER
BEG 3$ ::SKIP TYPEOUT IF NO POINTER
TYPE ::TYPE THE "ERROR MESSAGE"
WORD 0 ::"ERROR MESSAGE" POINTER GOES HERE
TYPE $SCALF ::"CARRIAGE RETURN" & "LINE FEED"
MOV (RO)+,4$ ::PICKUP "DATA HEADER" POINTER
BEG 5$ ::SKIP TYPEOUT IF 0
TYPE ::TYPE THE "DATA HEADER"
WORD 0 ::"DATA HEADER" POINTER GOES HERE
TYPE $SCALF ::"CARRIAGE RETURN" & "LINE FEED"
MOV (RO),RO ::PICKUP "DATA TABLE" POINTER
BNE 7$ ::GO TYPE THE DATA
MOV (SP)+,RO ::RESTORE RO
TYPE $SCALF ::"CARRIAGE RETURN" & "LINE FEED"
RTS PC ::RETURN

2$:
3$:
4$:
5$:
6$:
7$:
8$:
MOV 2(RO)+,-(SP) ::SAVE 2(RO)+ FOR TYPEOUT
TYPOC ::GO TYPE--OCTAL ASCII(ALL DIGITS)
TST (RO) ::IS THERE ANOTHER NUMBER?
BEG 6$ ::BR IF NO
TYPE 2$ ::TYPE TWO(2) SPACES
BR 7$ ::LOOP
ASCIZ / / ::TWO(2) SPACES
.EVEN

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ::NUMBER TO BE TYPED
*   TYPOS  ::CALL FOR TYPEOUT
*   .BYTE  N              ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ::M=1 OR 0
*                               ::1=TYPE LEADING ZEROS
*                               ::0=SUPPRESS LEADING ZEROS
*
*STYPON---ENTER HERE TO T.PE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ::NUMBER TO BE TYPED
*   TYPON  ::CALL FOR TYPEOUT
*
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ::NUMBER TO BE TYPED
*   TYPOC  ::CALL FOR TYPEOUT
*
011222 017646 000000 $TYPOS: MOV 2(SP),-(SP) ::PICKUP THE MODE

```

```

2442 011226 116667 000001 000211 MOVB 1(SP),SOFILL ::LOAD ZERO FILL SWITCH
2443 011234 112667 000207 MOVB (SP)+,SOMODE+1 ::NUMBER OF DIGITS TO TYPE
2444 011240 062716 000002 ADD #2,(SP) ::ADJUST RETURN ADDRESS
2445 011244 000406 BR $TYPON
2446 011246 112767 000001 000171 $TYP0C: MOVB #1,SOFILL ::SET THE ZERO FILL SWITCH
2447 011254 112767 000006 000165 MOVB #6,SOMODE+1 ::SET FOR SIX(6) DIGITS
2448 011262 112767 000005 000154 $TYPON: MOVB #5,$OCNT ::SET THE ITERATION COUNT
2449 011270 010346 MOV R3,-(SP) ::SAVE R3
2450 011272 010446 MOV R4,-(SP) ::SAVE R4
2451 011274 010546 MOV R5,-(SP) ::SAVE R5
2452 011276 116704 000145 MOVB $OMODE+1,R4 ::GET THE NUMBER OF DIGITS TO TYPE
2453 011302 005404 NEG R4
2454 011304 062704 000006 ADD #6,R4 ::SUBTRACT IT FOR MAX. ALLOWED
2455 011310 110467 000132 MOVB R4,SOMODE ::SAVE IT FOR USE
2456 011314 116704 000125 MOVB $OFILL,R4 ::GET THE ZERO FILL SWITCH
2457 011320 016605 000012 MOV 1,(SP),R5 ::PICKUP THE INPUT NUMBER
2458 011324 075003 CLR R3 ::CLEAR THE OUTPUT WORD
2459 011326 006105 1$: ROL R5 ::ROTATE MSB INTO "C"
2460 011330 000404 BR 3$ ::GO DO MSB
2461 011332 006105 2$: ROL R5 ::FORM THIS DIGIT
2462 011334 006105 ROL R5
2463 011336 006105 ROL R5
2464 011340 010503 MOV R5,R3
2465 011342 006103 3$: ROL R3 ::GET LSB OF THIS DIGIT
2466 011344 105367 000076 DECB $OMODE ::TYPE THIS DIGIT?
2467 011350 100016 BPL 7$ ::BR IF NO
2468 011352 042703 177770 BIC #177770,R3 ::GET RID OF JUNK
2469 011356 001002 BNE 4$ ::TEST FOR 0
2470 011360 005704 TST R4 ::SUPPRESS THIS 0?
2471 011362 001403 BEQ 5$ ::BR IF YES
2472 011364 005204 4$: INC R4 ::DON'T SUPPRESS ANYMORE 0'S
2473 011366 052703 000060 BIS #'0,R3 ::MAKE THIS DIGIT ASCII
2474 011372 052703 000040 5$: BIS #' ,R3 ::MAKE ASCII IF NOT ALREADY
2475 011376 110367 000040 MOVB R3,$S ::SAVE FOR TYPING
2476 011402 104401 011442 TYPE #S ::GO TYPE THIS DIGIT
2477 011406 105367 000032 7$: DECB $OCNT ::COUNT BY 1
2478 011412 003347 BGT 2$ ::BR IF MORE TO DO
2479 011414 002402 BLT 6$ ::BR IF DONE
2480 011416 005204 INC R4 ::INSURE LAST DIGIT ISN'T A BLANK
2481 011420 000744 BR 2$ ::GO DO THE LAST DIGIT
2482 011422 012605 6$: MOV (SP)+,R5 ::RESTORE R5
2483 011424 012604 MOV (SP)+,R4 ::RESTORE R4
2484 011426 012603 MOV (SP)+,R3 ::RESTORE R3
2485 011430 016666 000002 000004 MOV 2(SP),4(SP) ::SET THE STACK FOR RETURNING
2486 011436 012616 MOV (SP)+,(SP)
2487 011440 000002 RTI ::RETURN
2488 011442 000 8$: .BYTE 0 ::STORAGE FOR ASCII DIGIT
2489 011443 000 .BYTE 0 ::TERMINATOR FOR TYPE ROUTINE
2490 011444 000 $OCNT: .BYTE 0 ::OCTAL DIGIT COUNTER
2491 011445 000 $OFILL: .BYTE 0 ::ZERO FILL SWITCH
2492 011446 000000 $OMODE: .WORD 0 ::NUMBER OF DIGITS TO TYPE

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT


```

2498
2499
2500
2501
2502
2503
2504
2505
2506 011450
2507 011450 010046
2508 011452 010146
2509 011454 010246
2510 011456 010346
2511 011460 010546
2512 011462 012746 020200
2513 011466 016605 000020
2514 011472 100004
2515 011474 005405
2516 011476 112766 000055 000001
2517 011504 005000 1$:
2518 011506 012703 011664
2519 011512 112723 000040
2520 011516 005002 2$:
2521 011520 016001 011654
2522 011524 160105 3$:
2523 011526 002402
2524 011530 005202
2525 011532 000774
2526 011534 060105 4$:
2527 011536 005702
2528 011540 001002
2529 011542 105716
2530 011544 100407
2531 011546 106316 5$:
2532 011550 103003
2533 011552 116663 000001 177777
2534 011560 052702 000060 6$:
2535 011564 052702 000040 7$:
2536 011570 110223
2537 011572 005720
2538 011574 020027 000010
2539 011600 002746
2540 011602 003002
2541 011604 010502
2542 011606 000764
2543 011610 105726 8$:
2544 011612 100003
2545 011614 116663 177777 177776 9$:
2546 011622 105013
2547 011624 012605
2548 011626 012603
2549 011630 012602
2550 011632 012601
2551 011634 012600
2552 011636 104401 011664
2553 011642 016666 000002 000004
    
```

```

:*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
:*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
:*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
:*REPLACED WITH SPACES.
:*CALL:
:*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
:*      TYPDS                    ;;GO TO THE ROUTINE
$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      1$             ;;BR IF INPUT IS POS.
NEG      R5             ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
3$:      SUB      R1,R5    ;;FORM THIS BCD DIGIT
BLT      4$             ;;BR IF DONE
INC      R2             ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:      ADD      R1,R5    ;;ADD BACK THE CONSTANT
TST      R2             ;;CHECK IF BCD DIGIT=0
BNE      5$             ;;FALL THROUGH IF 0
TSTB     (SP)           ;;STILL DOING LEADING 0'S?
BMI      7$             ;;BR IF YES
ASLB     (SP)           ;;MSD?
BCC      6$             ;;BR IF NO
MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN
6$:      BIS      #'0,R2    ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST      (R0)+          ;;JUST INCREMENTING
CMP      R0,#10         ;;CHECK THE TABLE INDEX
BLT      2$             ;;GO DO THE NEXT DIGIT
BGT      8$             ;;GO TO EXIT
MOV      R5,R2          ;;GET THE LSD
BR       6$             ;;GO CHANGE TO ASCII
8$:      TSTB     (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
BPL      9$             ;;BR IF NO
MOVB     -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
9$:      CLRB     (R3)     ;;SET THE TERMINATOR
MOV      (SP)+,R5      ;;POP STACK INTO R5
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
TYPE     $SDBLK        ;;NOW TYPE THE NUMBER
MOV      2(SP),4(SP)    ;;ADJUST THE STACK
    
```

```

2554 011650 012616          MOV      (SP)+,(SP)
2555 011652 000002          RTI              ;;RETURN TO USER
2556 011654 023420          $DTBL: 10000.
2557 011656 001750          1000.
2558 011660 000144          100.
2559 011662 000012          10.
2560 011664 000004          $DBLK: .BLKW 4
2561
2562          .SBTTL TTY INPUT ROUTINE
2563
2564          ;*****
2565          .ENABL LSB
2566
2567          .DSABL LSB
2568
2569
2570          ;*****
2571          *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2572          *CALL:
2573          *      RDCHR              ;; INPUT A SINGLE CHARACTER FROM THE TTY
2574          *      RETURN HERE        ;; CHARACTER IS ON THE STACK
2575          *                          ;; WITH PARITY BIT STRIPPED OFF
2576
2577
2578 011674 011646          $RDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC
2579 011676 016666 000004 000002  MOV      4(SP),2(SP)      ;; SAVE THE PS
2580 011704 105777 167234 1$:  TSTB    2$TKS          ;; WAIT FOR
2581 011710 100375          BPL     1$              ;; A CHARACTER
2582 011712 117766 167230 000004  MOVB   2$TKB,4(SP)      ;; READ THE TTY
2583 011720 042766 177600 000004  BIC    #1C(177),4(SP)   ;; GET RID OF JUNK IF ANY
2584 011726 026627 000004 000023  CMP    4(SP),#23       ;; IS IT A CONTROL-S?
2585 011734 001013          BNE    3$              ;; BRANCH IF NO
2586 011736 105777 167202 2$:  TSTB    2$TKS          ;; WAIT FOR A CHARACTER
2587 011742 100375          BPL     2$              ;; LOOP UNTIL ITS THERE
2588 011744 117746 167176          MOVE   2$TKB, -(SP)    ;; GET CHARACTER
2589 011750 042716 177600          BIC    #1C177, (SP)    ;; MAKE IT 7-BIT ASCII
2590 011754 022627 000021          CMP    (SP)+,#21       ;; IS IT A CONTROL-Q?
2591 011760 001366          BNE    2$              ;; IF NOT DISCARD IT
2592 011762 000750          BR     1$              ;; YES, RESUME
2593 011764 026627 000004 000140 3$:  CMP    4(SP),#140      ;; IS IT UPPER CASE?
2594 011772 002407          BLT    4$              ;; BRANCH IF YES
2595 011774 026627 000004 000175  CMP    4(SP),#175      ;; IS IT A SPECIAL CHAR?
2596 012002 003003          BGT    4$              ;; BRANCH IF YES
2597 012004 042766 000040 000004  BIC    #40,4(SP)       ;; MAKE IT UPPER CASE
2598 012012 000002          4$:    RTI              ;; GO BACK TO USER
2599
2600          ;*****
2601          *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2602          *CALL:
2603          *      RDLIN              ;; INPUT A STRING FROM THE TTY
2604          *      RETURN HERE        ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2605          *                          ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2606 012014 010346          $RDLIN: MOV      R3, -(SP)      ;; SAVE R3
2607 012016 005046          CLR    -(SP)          ;; CLEAR THE RUBOUT KEY
2608 012020 012703 012250 1$:  MOV    #$TTYIN,R3      ;; GET ADDRESS
2609 012024 022703 012260 2$:  CMP    #$TTYIN+8.,R3    ;; BUFFER FULL?
    
```

```

2610 012030 101456          BLOS      4$          ;; BR IF YES
2611 012032 104406          RDCHR                    ;; GO READ ONE CHARACTER FROM THE TTY
2612 012034          13          MOVVB    (SP)+,(R3)    ;; GET CHARACTER
2613 012036          713        000177    10$:    CMPB    #177,(R3)    ;; IS IT A RUBOUT
2614 012042          1022          BNE     5$          ;; BR IF NO
2615 012044          005716          TST     (SP)          ;; IS THIS THE FIRST RUBOUT?
2616 012046          001007          SNE     6$          ;; BR IF NO
2617 012050          112767    000134    000170    MOVVB    #' \,9$      ;; TYPE A BACK SLASH
2618 012056          104401    012246          TYPE    ,9$
2619 012062          012716    177777          MOV     #-1,(SP)      ;; SET THE RUBOUT KEY
2620 012066          005303          DEC     R3            ;; BACKUP BY ONE
2621 012070          020327    012250    6$:     CMP     R3,#$TTYIN    ;; STACK EMPTY?
2622 012074          103434          BLO     4$          ;; BR IF YES
2623 012076          111367    000144          MOVVB    (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
2624 012102          104401    012246          TYPE    ,9$          ;; GO TYPE
2625 012106          000746          BR     2$          ;; GO READ ANOTHER CHAR.
2626 012110          005716          TST     (SP)          ;; RUBOUT KEY SET?
2627 012112          001406          BEQ     7$          ;; BR IF NO
2628 012114          112767    000134    000124    MOVVB    #' \,9$      ;; TYPE A BACK SLASH
2629 012122          104401    012246          TYPE    ,9$
2630 012126          005016          CLR     (SP)          ;; CLEAR THE RUBOUT KEY
2631 012130          122713    000025    7$:     CMPB    #25,(R3)      ;; IS CHARACTER A CTRL U?
2632 012134          001003          BNE     8$          ;; BR IF NO
2633 012136          104401    012260          TYPE    ,SCNTLU      ;; TYPE A CONTROL "U"
2634 012142          000726          BR     1$          ;; GO START OVER
2635 012144          122713    000022    8$:     CMPB    #22,(R3)      ;; IS CHARACTER A "TR"?
2636 012150          001011          BNE     3$          ;; BRANCH IF NO
2637 012152          105013          CLRB   (R3)          ;; CLEAR THE CHARACTER
2638 012154          104401    001253          TYPE    ,SCRLF        ;; TYPE A "CR" & "LF"
2639 012160          104401    012250          TYPE    , $TTYIN      ;; TYPE THE INPUT STRING
2640 012164          000717          BR     2$          ;; GO PICKUP ANOTHER CHARACTER
2641 012166          104401    001252    4$:     TYPE    , $QUES      ;; TYPE A '?'
2642 012172          000712          BR     1$          ;; CLEAR THE BUFFER AND LOOP
2643 012174          111367    000046    3$:     MOVVB    (R3),9$      ;; ECHO THE CHARACTER
2644 012200          104401    012246          TYPE    ,9$
2645 012204          122723    000015          CMPB    #15,(R3)+    ;; CHECK FOR RETURN
2646 012210          001305          BNE     2$          ;; LOOP IF NOT RETURN
2647 012212          105063    177777          CLRB   -1(R3)        ;; CLEAR RETURN (THE 15)
2648 012216          104401    001254          TYPE    , $LF         ;; TYPE A LINE FEED
2649 012222          005726          TST     (SP)+        ;; CLEAN RUBOUT KEY FROM THE STACK
2650 012224          012603          MOV     (SP)+,R3     ;; RESTORE R3
2651 012226          011646          MOV     (SP),-(SP)   ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2652 012230          016666    000004    000002    MOV     4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
2653 012236          012766    012250    000004    MOV     #$TTYIN,4(SP)
2654 012244          000002          RTI                    ;; RETURN
2655 012246          000          9$:     .BYTE   0            ;; STORAGE FOR ASCII CHAR. TO TYPE
2656 012247          000          .BYTE   0            ;; TERMINATOR
2657 012250          000010          $TTYIN: .BLKB   8.      ;; RESERVE 8 BYTES FOR TTY INPUT
2658 012260          052536    005015    000          $CNTLU: .ASCIZ  /TU/<15><12>  ;; CONTROL "U"
2659 012265          136        006507    000012    $CNTLG: .ASCIZ  /TG/<15><12>  ;; CONTROL "G"
2660 012272          005015    053523    020122    $MSWR:  .ASCIZ  <15><12>/SWR = /
2661 012300          020075          000
2662 012303          040        047040    053505    $MNEW:  .ASCIZ  / NEW = /
2663 012310          036440    000040
2664
2665          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

```

2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
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

012314 011646
012316 016666 000004 000002
012324 010046
012326 010146
012330 010246
012332 104407
012334 012600
012336 010067 000100
012342 005001
012344 005002
012346 112046
012350 001420
012352 122716 000060
012356 003026
012360 122716 000067
012364 002423
012366 006301
012370 006102
012372 006301
012374 006102
012376 006301
012400 006102
012402 042716 177770
012406 062601
012410 000756
012412 005726
012414 010166 000012
012420 010267 000026
012424 012602
012426 012601
012430 012600
012432 000002
012434 005726
012436 105010
012440 104401
012442 000000
012444 104401 001252
012450 000730
012452 000000

```
*****  
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND  
*CHANGE IT TO BINARY.  
*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL  
*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED  
*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST  
*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.  
*CALL:  
*      RDOCT                ;; READ AN OCTAL NUMBER  
*      RETURN HERE          ;; LOW ORDER BITS ARE ON TOP OF THE STACK  
*                          ;; HIGH ORDER BITS ARE IN $HIOCT  
  
$RDOCT: MOV      (SP), -(SP)      ;; PROVIDE SPACE FOR THE  
        MOV      4(SP), 2(SP)    ;; INPUT NUMBER  
        MOV      R0, -(SP)       ;; PUSH R0 ON STACK  
        MOV      R1, -(SP)       ;; PUSH R1 ON STACK  
        MOV      R2, -(SP)       ;; PUSH R2 ON STACK  
1$:     RDLIN                    ;; READ AN ASCII LINE  
        MOV      (SP)+, R0        ;; GET ADDRESS OF 1ST CHARACTER  
        MOV      R0, 5$          ;; AND SAVE IT  
        CLR      R1              ;; CLEAR DATA WORD  
        CLR      R2  
2$:     MOV      (R0)+, -(SP)     ;; PICKUP THIS CHARACTER  
        BEQ      3$              ;; IF ZERO GET OUT  
        CMPB    #'0, (SP)        ;; MAKE SURE THIS CHARACTER  
        BGT      4$              ;; IS AN OCTAL DIGIT  
        CMPB    #'7, (SP)        ;;  
        BLT      4$              ;;  
        ASL     R1                ;; *2  
        ROL     R2  
        ASL     R1                ;; *4  
        ROL     R2  
        ASL     R1                ;; *8  
        ROL     R2  
        BIC     #'C7, (SP)       ;; STRIP THE ASCII JUNK  
        ADD     (SP)+, R1        ;; ADD IN THIS DIGIT  
        BR      2$              ;; LOOP  
3$:     TST     (SP)+            ;; CLEAN TERMINATOR FROM STACK  
        MOV     R1, 12(SP)       ;; SAVE THE RESULT  
        MOV     R2, $HIOCT  
        MOV     (SP)+, R2        ;; POP STACK INTO R2  
        MOV     (SP)+, R1        ;; POP STACK INTO R1  
        MOV     (SP)+, R0        ;; POP STACK INTO R0  
        RTI                    ;; RETURN  
4$:     TST     (SP)+            ;; CLEAN PARTIAL FROM STACK  
        CLRB   (R0)             ;; SET A TERMINATOR  
        TYPE                    ;; TYPE UP THRU THE BAD CHAR.  
5$:     .WORD   0  
        TYPE   $QUES            ;; "?" "CR" & "LF"  
        BR     1$              ;; TRY AGAIN  
$HIOCT: .WORD   0              ;; HIGH ORDER BITS GO HERE  
  
.SBTTL TYPE ROUTINE  
;*****
```

```

2722      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2723      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2724      ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2725      ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2726      ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2727      ;*
2728      ;*CALL:
2729      ;*1) USING A TRAP INSTRUCTION
2730      ;*      TYPE      ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2731      ;*OR
2732      ;*      TYPE
2733      ;*      MESADR
2734      ;*
2735
2736      012454 105767 166477      $TYPE:  TSTB      $TFPLG          ;; IS THERE A TERMINAL?
2737      012460 100002              BPL          1$          ;; BR IF YES
2738      012462 000000              HALT              ;; HALT HERE IF NO TERMINAL
2739      012464 000407              BR          3$          ;; LEAVE
2740      012466 010046              1$:  MOV        RO,-(SP)          ;; SAVE RO
2741      012470 017600 000002      MOV        22(SP),RO          ;; GET ADDRESS OF ASCIZ STRING
2742      012474 112046              2$:  MOVB      (RO)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2743      012476 001005              BNE        4$          ;; BR IF IT ISN'T THE TERMINATOR
2744      012500 005726              TST        (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
2745      012502 012600              60$: MOV        (SP)+,RO          ;; RESTORE RO
2746      012504 062716 000002      3$:  ADD        #2,(SP)          ;; ADJUST RETURN PC
2747      012510 000002              RTI              ;; RETURN
2748      012512 122716 000011      4$:  CMPB      #HT,(SP)          ;; BRANCH IF <HT>
2749      012516 001430              BEQ        8$          ;; BRANCH IF NOT <CRLF>
2750      012520 122716 000200      CMPB      #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
2751      012524 001006              BNE        5$          ;; POP <CR><LF> EQUIV
2752      012526 005726              TST        (SP)+          ;; TYPE A CR AND LF
2753      012530 104401              TYPE
2754      012532 001253              $CRLF
2755      012534 105067 000130      CLRB      $CHARCNT          ;; CLEAR CHARACTER COUNT
2756      012540 000755              BR          2$          ;; GET NEXT CHARACTER
2757      012542 004767 000056      5$:  JSR        PC,$TYPEPC          ;; GO TYPE THIS CHARACTER
2758      012546 126726 166404      6$:  CMPB      $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
2759      012552 001350              BNE        2$          ;; IF NO GO GET NEXT CHAR.
2760      012554 016746 166374      MOV        $NULL,-(SP)      ;; SET # OF FILLER CHARS. NEEDED
2761      ;*
2762      012560 105366 000001      7$:  DECB      1(SP)          ;; AND THE NULL CHAR.
2763      012564 002770              BLT        6$          ;; DOES A NULL NEED TO BE TYPED?
2764      012566 004767 000032      JSR        PC,$TYPEPC          ;; BR IF NO--GO POP THE NULL OFF OF STACK
2765      012572 105367 000072      DECB      $CHARCNT          ;; GO TYPE A NULL
2766      012576 000770              BR          7$          ;; DO NOT COUNT AS A COUNT
2767      ;*
2768      ;*
2769      ;*
2770      012600 112716 000040      8$:  MOVB      #' ,(SP)          ;; REPLACE TAB WITH SPACE
2771      012604 004767 000014      9$:  JSR        PC,$TYPEPC          ;; TYPE A SPACE
2772      012610 132767 000007 000052 BITB      #7,$CHARCNT          ;; BRANCH IF NOT AT
2773      012616 001372              BNE        9$          ;; TAB STC?
2774      012620 005726              TST        (SP)+          ;; POP SPACE OFF STACK
2775      012622 000724              BR          2$          ;; GET NEXT CHARACTER
2776      012624 105777 166320      $TYPEPC: TSTB      2$TPS          ;; WAIT UNTIL PRINTER IS READY
2777      012630 100375              BPL        $TYPEPC

```

```

2778 012632 116677 000002 166312      MOVB 2(SP),D$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2779 012640 122766 000015 000002      CMPB #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
2780 012646 001003                BNE 1$               ;;BRANCH IF NO
2781 012650 105067 000014                CLRB $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
2782 012654 000406                BR $TYPEX            ;;EXIT
2783 012656 122766 000012 000002 1$:    CMPB #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
2784 012664 001402                BEQ $TYPEX           ;;BRANCH IF YES
2785 012666 105227                INCB (PC)+          ;;COUNT THE CHARACTER
2786 012670 000000                $CHARCNT:.WORD 0    ;;CHARACTER COUNT STORAGE
2787 012672 000207                $TYPEX: RTS        PC

```

.SBTTL READ A DECIMAL NUMBER FROM THE TTY

```

2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833

```

```

*****
*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
*POSITIVE 32767 TO NEGATIVE 32768.
*CALL:
*
* RDDEC                ;;READ A DECIMAL NUMBER
* RETURN HERE         ;;NUMBER IS ON TOP OF THE STACK

```

```

2804 012674 011646 000004 000002 $RDDEC: MOV (SP),-(SP) ;;PROVIDE SPACE FOR
2805 012676 016666                MOV 4(SP),2(SP)    ;;THE INPUT NUMBER
2806 012704 010046                MOV R0,-(SP)       ;;PUSH R0 ON STACK
2807 012706 010146                MOV R1,-(SP)       ;;PUSH R1 ON STACK
2808 012710 010246                MOV R2,-(SP)       ;;PUSH R2 ON STACK
2809 012712 104407 1$:          RDLIN                ;;READ AN ASCII LINE
2810 012714 012600                MOV (SP)+,R0       ;;ADDRESS OF 1ST CHAR.
2811 012716 010067 000120        MOV R0,R0          ;;SAVE IN CASE OF BAD INPUT
2812 012722 005046                CLR -(SP)          ;;CLEAR DATA WORD
2813 012724 005002                CLR R2             ;;SIGN SET POSITIVE
2814 012726 122710 000055        CMPB #'-(R0)       ;;SEE IF A MINUS SIGN WAS TYPED
2815 012732 001001                BNE 2$            ;;BR IF NO MINUS SIGN
2816 012734 112002                MOVB (R0)+,R2     ;;SAVE FOR LATER USE
2817 012736 112001 2$:          MOVB (R0)+,R1     ;;PICKUP THIS CHARACTER
2818 012740 001424                BEQ 3$            ;;GET OUT IF ZERO
2819 012742 122701 000060        CMPB #'0,R1       ;;MAKE SURE THIS CHARACTER
2820 012746 003032                BGT 5$            ;;IS A DIGIT BETWEEN 0 & 9
2821 012750 122701 000071        CMPB #'9,R1
2822 012754 002427                BLT 5$
2823 012756 032716 170000        BIT #'C7777,(SP) ;;DON'T LET NUMBER GET TO BIG
2824 012762 001024                BNE 5$            ;;BR IF NUMBER WOULD OVERFLOW
2825 012764 006316                ASL (SP)          ;;*2
2826 012766 011646                MOV (SP),-(SP)    ;;SAVE FOR LATER
2827 012770 006316                ASL (SP)          ;;*4
2828 012772 006316                ASL (SP)          ;;*8
2829 012774 062616                ADD (SP)+,(SP)    ;;*10
2830 012776 102416                BVS 5$            ;;OVERFLOW ISN'T ALLOWED
2831 013000 162701 000060        SUB #'0,R1        ;;STRIP AWAY THE ASCII JUNK
2832 013004 060116                ADD R1,(SP)       ;;ADD IN THIS DIGIT
2833 013006 102412                BVS 5$            ;;OVERFLOW ISN'T ALLOWED

```

```

2834 013010 000752          BR      2$          ;; LOOP
2835 013012 005702          3$:   TST      R2          ;; CHECK IF NUMBER IS NEG
2836 013014 001401          BEQ      4$          ;; BR IF NO
2837 013016 005416          NEG      (SP)        ;; YES--NEGATE THE NUMBER
2838 013020 012666 000012  4$:   MOV      (SP)+,12(SP) ;; SAVE THE RESULT
2839 013024 012602          MOV      (SP)+,R2    ;; POP STACK INTO R2
2840 013026 012601          MOV      (SP)+,R1    ;; POP STACK INTO R1
2841 013030 012600          MOV      (SP)+,R0    ;; POP STACK INTO R0
2842 013032 000002          RTI          ;; RETURN
2843
2844 013034 005726          5$:   TST      (SP)+    ;; CLEAN PARTIAL NUMBER FROM STACK
2845 013036 105010          CLRB     (R0)        ;; SET A TERMINATOR
2846 013040 104401          TYPE    0           ;; TYPE THE INPUT UP TO BAD CHAR.
2847 013042 000000          6$:   .WORD    0           ;; POINTER GOES HERE
2848 013044 104401 001252  TYPE    $QUES        ;; "?" "CR" &"LF"
2849 013050 000720          BR      1$          ;; TRY AGAIN
2850
2851          .SBTTL TRAP DECODER
2852
2853          ;;*****
2854          ;;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2855          ;;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2856          ;;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2857          ;;GO TO THAT ROUTINE.
2858
2859 013052 010046          $TRAP: MOV      RO,-(SP)    ;; SAVE RO
2860 013054 016600 000002  MOV      2(SP),RO    ;; GET TRAP ADDRESS
2861 013060 005740          TST      -(RO)      ;; BACKUP BY 2
2862 013062 111000          MOVB    (RO),RO    ;; GET RIGHT BYTE OF TRAP
2863 013064 006300          ASL     RO          ;; POSITION FOR INDEXING
2864 013066 016000 013106  MOV      $TRPAD(RO),RO ;; INDEX TO TABLE
2865 013072 000200          RTS      RO          ;; GO TO ROUTINE
2866
2867
2868          ;;THIS IS USE TO HANDLE THE "GETPR!" MACRO
2869
2870 013074 011646          $TRAP2: MOV      (SP),-(SP) ;; MOVE THE PC DOWN
2871 013076 016666 000004 000002 MOV      4(SP),2(SP) ;; MOVE THE PSW DOWN
2872 013104 000002          RTI          ;; RESTORE THE PSW
2873
2874          .SBTTL TRAP TABLE
2875
2876          ;;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2877          ;;BY THE "TRAP" INSTRUCTION.
2878
2879          ;          ROUTINE
2880          ;          -----
2881 013106 013074          $TRPAD: .WORD    $TRAP2          TRAP+1(104401) TTY TYPEOUT ROUTINE
2882 013110 012454          $TYPE    ;;CALL=TYPE          TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2883 013112 011246          $TYPOC   ;;CALL=TYPOC         TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2884 013114 011222          $TYPOS   ;;CALL=TYPOS         TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2885 013116 011262          $TYPON   ;;CALL=TYPON         TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
2886 013120 011450          $TYPDS   ;;CALL=TYPDS
2887
2888
2889 013122 011674          $RDCHR  ;;CALL=RDCHR          TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
  
```

2890	013124	012014			\$RDLIN	::CALL=RDLIN	TRAP+7(104407)	TTY TYPEIN STRING ROUTINE
2891	013126	012314			\$RDOCT	::CALL=RDOCT	TRAP+10(104410)	READ AN OCTAL NUMBER FROM TTY
2892	013130	012674			\$RDDEC	::CALL=RDDEC	TRAP+11(104411)	READ A DECIMAL NUMBER FROM TTY

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
$PWRDN: MOV    $SILLUP, @#PWRVEC    ;; SET FOR FAST UP
        MOV    #340, @#PWRVEC+2    ;; PRIO:7
        MOV    RD, -(SP)           ;; PUSH RD 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 UP

```

```

*****
:POWER UP ROUTINE
$PWRUP: MOV    $SILLUP, @#PWRVEC    ;; SET FOR FAST DOWN
        MOV    $SAVR6, SP         ;; GET SP
        CLR    $SAVR6            ;; WAIT LOOP FOR THE TTY
1$:     INC    $SAVR6            ;; WAIT FOR THE INC
        BNE   1$                 ;; OF WORD
        MOV   (SP)+, @SWR        ;; POP STACK INTO @SWR
        MOV   (SP)+, R5         ;; POP STACK INTO R5
        MOV   (SP)+, R4         ;; POP STACK INTO R4
        MOV   (SP)+, R3         ;; POP STACK INTO R3
        MOV   (SP)+, R2         ;; POP STACK INTO R2
        MOV   (SP)+, R1         ;; POP STACK INTO R1
        MOV   (SP)+, RD         ;; POP STACK INTO RD
        MOV    #PWRDN, @#PWRVEC    ;; SET UP THE POWER DOWN VECTOR
        MOV    #340, @#PWRVEC+2    ;; PRIO:7
        TYPE   $POWER            ;; REPORT THE POWER FAILURE
        SPWRMG: .WORD $POWER      ;; POWER FAIL MESSAGE POINTER
        MOV   (PC)+, (SP)        ;; RESTART AT RESTR
        SPWRAD: .WORD RESTR      ;; RESTART ADDRESS
        RTI
$ILLUP: HALT                    ;; THE POWER UP SEQUENCE WAS STARTED
        BR     .-2                ;; BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0                        ;; PUT THE SP HERE
$POWER: .ASCIZ <15><12>"POWER"
        .EVEN

```

```

*****
:TRANSMIT INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
*****

```

2944	013314	105777	166074		XINT:	TSTB	@DLXCSR	;"READY" SET ??
2945	013320	100416				BMI	1\$;;BR IF YES


```

2946 013322 013767 177776 165652      MOV      Q#PSW,$TMP0      ;SAVE THE ERROR PSW
2947 013330 010667 165642      MOV      SP,$REG6        ;SAVE THE ERROR STACK POINTER
2948 013334 005167 166062      COM      XFLGO           ;SET XMIT SOFTWARE ERROR FLAG
2949 013340 042777 000100 166042      BIC      #100,$DLRCSR    ;TURN OFF THE INTERRUPT ENABLES
2950 013346 042777 000100 166040      BIC      #100,$DLXCSR
2951 013354 000411                BR      2$              ;GO TO EXIT
2952 013356 022767 021722 166046 1$:    CMP      #DLBUFI,OPTR    ;XMITTED 256. BYTES YET ??
2953 013364 001405                BEQ      2$              ;BR IF YES
2954 013366 117777 166040 166022      MOVB     $OPTR,$DLXDBR   ;OUTPUT A BYTE
2955 013374 005267 166032                INC      OPTR            ;UPDATE BUFFER POINTER
2956 013400 000002                2$:    RTI                 ;RETURN TO MAINLINE TEST
2957
2958                                     ;*****
2959                                     ;RECEIVER INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
2960                                     ;*****
2961
2962 013402 105777 166002      RINT:    TSTB     $DLRCSR ;"DONE" SET ??
2963 013406 100410                BMI     1$              ;BR IF YES
2964 013410 013767 177776 165564      MOV      Q#PSW,$TMP0      ;SAVE THE ERROR PSW
2965 013416 010667 165554      MOV      SP,$REG6        ;SAVR THE ERROR STACK POINTER
2966 013422 005167 165776      COM      RFLGO           ;SET HARD RCVR ERROR FLAG
2967 013426 000415                BR      2$              ;GO EXIT
2968 013430 005777 165756      1$:    TST      $DLRDBR    ;ANY SOFT ERRORS ??
2969 013434 100021                BPL     3$              ;BR IF NOT
2970 013436 013767 177776 165536      MOV      Q#PSW,$TMP0      ;SAVE THE ERROR PSW
2971 013444 010667 165526      MOV      SP,$REG6        ;SAVE THE ERROR STACK POINTER
2972 013450 017767 165736 165526      MOV      $DLRDBR,$TMP1    ;SAVE THE ERROR REGISTER IN TMP1
2973 013456 005167 165744      COM      RFLG1           ;SET THE SOFT ERROR FLAG
2974 013462 042777 000100 165724 2$:    BIC      #100,$DLXCSR    ;TURN OFF THE INTR. ENABLES
2975 013470 042777 000100 165712      BIC      #100,$DLRCSR
2976 013476 000411                BR      4$              ;GO TO EXIT
2977 013500 022767 022322 165726 3$:    CMP      #BUFEND,IPTR    ;RECEIVED 256. BYTES YET ??
2978 013506 001405                BEQ      4$              ;BR IF YES
2979 013510 117777 165676 165716      MOVB     $DLRDBR,$IPTR    ;INPUT A BYTE FROM THE DL11
2980 013516 005267 165712      INC      IPTR            ;UPDATE BUFFER POINTER
2981 013522 000002                4$:    RTI                 ;RETURN TO MAINLINE TEST
2982
2983                                     ;THE FOLLOWING ROUTINE IS USED BY THE USER UTILITY PROGRAMS TO WAIT
2984                                     ;A SPECIFIED NO. OF MILLISECONDS BETWEEN CHARACTER TRANSFERS
2985
2986 013524 017667 000000 000034 DELAY:  MOV      Q(R6),DELCNT ;GET THE NO. OF MSEC. DELAY COUNT
2987                                     ;TYPED IN BY USER
2988 013532 062716 000002                ADD     #2,(R6)         ;SET UP THIS ROUTINE'S EXIT ADDRESS
2989 013536 005767 000024                TST     DELCNT          ;IS THE DELAY COUNT ZERO?
2990 013542 001410                BEQ     3$              ;BRANCH IF YES
2991 013544 012746 000226      1$:    MOV      #226,-(SP)    ;PUSH A 1 MSEC. COUNT TO STACK
2992 013550 005316                2$:    DEC     (SP)          ;DECREMENT THE 1 MSEC. COUNT BY 1
2993 013552 001376                BNE     2$              ;BRANCH IF 1 MSEC. NOT EATEN
2994                                     ;AWAY YET
2995 013554 005726                TST     (SP)+           ;RESET STACK AFTER 1 MSEC. TIME UP
2996 013556 005367 000004                DEC     DELCNT          ;DECREMENT THE TOTAL NO. OF
2997                                     ;MSECS. COUNT
2998 013562 001370                BNE     1$              ;BRANCH IF WE HAVE MORE MSECS.
2999                                     ;TO WAIT
3000 013564 000207                3$:    RTS      PC         ;GO BACK TO REISSUE A CHARACTER
3001 013566 000000      DELCNT: .WORD 0        ;THE NO. OF MSECS. NEEDED TO

```



```

00000000 014036 013767 177776 165136 SUER2: MOV      2#PSW,$TMPD      ;SAVE THE [PSW]
00000000 014044 016701 165340      MOV      DLRCR,R1      ;PUT DEVADR IN R1
00000000 014050 011203      MOV      (R2),R3      ;PUT WAS INFO IN R3
00000000 014056 010657 165120      MOV      SP,$REG6     ;SAVE THE [SP]
00000000 014062 062767 000002 165112      ADD      #2,$REG6     ;CORRECT FOR CALLING JSR
00000000 014068 116700 165012      SUERR1: MOVB     $TSTNM,R0 ;PUT TEST NO. IN R0
00000000 014074 010067 165066      MOV      R0,$REG0     ;SAVE [R0] THRU [R4]
00000000 014080 010167 165064      MOV      R1,$REG1
00000000 014086 010267 165062      MOV      R2,$REG2
00000000 014092 010367 165060      MOV      R3,$REG3
00000000 014098 010467 165056      MOV      R4,$REG4
00000000 014104      RTS      PC           ;RETURN TO CALLING TEST
00000000 014110      ;
00000000 014116 013767 177776 165056 SUERT1: MOV      2#PSW,$TMPD      ;SAVE THE [PSW]
00000000 014122 116700 164752      MOVB     $TSTNM,R0     ;PUT TEST NO. IN R0
00000000 014128 016701 165254      MOV      DLRCR,R1     ;PUT DEVADR IN R1
00000000 014134 010067 165022      MOV      R0,$REG0     ;SAVE [R0]
00000000 014140 010167 165020      MOV      R1,$REG1     ;SAVE [R1]
00000000 014146 013767 177776 165032 SUERT2: MOV      2#PSW,$TMP1     ;SAVE THE [PSW]
00000000 014152 010667 165020      MOV      SP,$REG6     ;SAVE THE [SP]
00000000 014158 062767 000002 165012      ADD      #2,$REG6     ;CORRECT FOR CALLING JSR
00000000 014164 010267 164776      MOV      R2,$REG2     ;SAVE [R2]
00000000 014170      RTS      PC           ;RETURN
00000000 014176      ;
00000000 014182      ;SUBROUTINE TO SETUP VECTORS FOR 256. BYTE BLOCK TRANSFER TESTS
00000000 014188      ;
00000000 014194 016705 165222      SUVEC:  MOV      DLVECT,R5     ;GET FIRST VECTOR ADDRESS
00000000 014200 012725 013402      MOV      #RINT,(R5)+    ;SET UP RCVR VECTOR
00000000 014206 016725 165074      MOV      DLPRI,(R5)+
00000000 014212 012725 013314      MOV      #XINT,(R5)+   ;SET UP XMIT VECTOR
00000000 014218 016715 165064      MOV      DLPRI,(R5)
00000000 014224      RTS      PC           ;RETURN TO CALLER
00000000 014230      ;
00000000 014236      ;SUBROUTINE TO PRIME DATA BUFFERS AND DEVICE FOR 256. BYTE TRANSFER
00000000 014242      ;
00000000 014248 005077 165170      PRIME:  CLR      2DLXCSR     ;CLEAR XMIT AND RCVR CSR'S
00000000 014254 005077 165160      CLR      2DLRCR
00000000 014260 005067 165166      CLR      XFLGO          ;INITIALIZE ERROR FLAGS
00000000 014266 005067 165164      CLR      RFLGO
00000000 014272 005067 165162      CLR      RFLG1
00000000 014278 012767 021322 165160      MOV      #DLBUFO,OPTR   ;SET UP OUTPUT POINTER
00000000 014284 012767 021722 165154      MOV      #DLBUFI,IPTR   ;SET UP INPUT POINTER
00000000 014290 004767 000044      JSR      PC,CLDLBF      ;GO CLEAR THE BUFFERS
00000000 014296 004777 165146      JSR      PC,2ALDOUT     ;GO SET UP THE PATTERN
00000000 014302 005067 165144      CLR      TIMR1          ;INIT TIMEOUT COUNTERS
00000000 014308 012767 000036 165140      MOV      #30,TIMR2
00000000 014314 005777 165104      TST      2DLRDBR        ;FLUSH "DONE" BIT IN RCVR CSR
00000000 014320 005777 165100      TST      2DLRDBR
00000000 014326 052777 000100 165070      BIS      #100,2DLRCR    ;ENABLE RCVR INTR.
00000000 014332 052777 000104 165066      BIS      #104,2DLXCSR  ;ENABLE XMIT INTR. AND MAINT MODE
00000000 014338      RTS      PC
  
```

;THIS ROUTINE IS CALLED TO CLEAR THE INPUT AND OUTPUT BUFFERS

```

0138
0139 014330 012705 021322      CLDLBF: MOV      #DLBUF0,R5      ;R5 POINTS TO BEGINNING OF BUFFER AREA
0140 014334 005025          1$: CLR        (R5)+          ;CLEAR A WORD
0141 014336 022705 022322      CMP        #BUFEND,R5      ;DONE ALL WORDS ??
0142 014342 001374          BNE        1$              ;BR IF NOT
0143 014344 000207          RTS        PC              ;RETURN TO CALLER

;THIS ROUTINE IS CALLED TO SET UP THE NULL-DEL-NULL PATTERN

0144 014346 012705 021322      LDOUT1: MOV      #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
0145 014352 105025          1$: CLRB       (R5)+          ;MOVE A NULL CHAR
0146 014354 112725 000377      MOVB      #377,(R5)+        ;MOV A DEL CHAR
0147 014356 022705 021722      CMP        #DLBUF1,R5      ;ALL DONE ??
0148 014364 001372          BNE        1$              ;BR IF NOT
0149 014366 000207          RTS        PC              ;RETURN TO CALLER

;THIS ROUTINE IS USED TO LOAD AN ASCENDING BINARY COUNT PATTERN

0150 014370 005005          LDOUT2: CLR        R5          ;START WITH 000
0151 014372 110565 021322      1$: MOVB     R5,DLBUF0(R5)    ;LOAD ONE BYTE
0152 014376 005205          INC        R5              ;INCREMENT BYTE
0153 014400 022705 000400      CMP        #400,R5          ;DONE 000 THRU 377 ??
0154 014404 001372          BNE        1$              ;BR IF NOT
0155 014406 000207          RTS        PC              ;RETURN TO CALLER

;THIS ROUTINE IS USED TO LOAD A DESCENDING BINARY COUNT PATTERN

0156 014410 112767 000377 164602      LDOUT3: MOVB    #377,$TMP7    ;START WITH A 377 BYTE
0157 014416 012705 021322      MOV      #DLBUF0,R5          ;R5 POINTS TO OUTPUT BUFFER
0158 014422 116725 164572      1$: MOVB     $TMP7,(R5)+      ;LOAD ONE BYTE
0159 014426 022705 021722      CMP      #DLBUF1,R5          ;ALL DONE ??
0160 014432 001403          BEQ        2$              ;BR IF YES
0161 014434 105367 164560      DECB     $TMP7              ;GENERATE NEXT BYTE
0162 014440 000770          BR        1$              ;GO MOVE IT
0163 014442 000207      2$: RTS        PC              ;RETURN TO CALLER

;THIS ROUTINE LOADS A COMPLEMENTING WORST CASE PATTERN

0164 014444 012705 021322      LDOUT4: MOV      #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
0165 014450 005067 164544      CLR      $TMP7              ;INIT. BYTE GENERATOR
0166 014454 116725 164540      1$: MOVB     $TMP7,(R5)+      ;MOVE A BYTE
0167 014460 105167 164534      COMB     $TMP7              ;COMPLEMENT IT
0168 014464 116725 164530      MOVB     $TMP7,(R5)+        ;NOW LOAD THE 1'S COMPLEMENT
0169 014470 105267 164525      INCB     $TMP7+1            ;INCREMENT THE BYTE
0170 014474 116767 164521 164516      MOVB     $TMP7+1,$TMP7      ;SET UP TO LOAD NEXT TWO
0171 014502 022705 021722      CMP      #DLBUF1,R5          ;ALL DONE ??
0172 014506 001362          BNE        1$              ;BR IF NOT
0173 014510 000207          RTS        PC              ;RETURN TO CALLER

;THIS ROUTINE CHECKS FOR DATA COMPARE ERRORS IN 256. BYTE BLOCK TRANSFERS

0174
0175
0176
0177 014444 012705 021322
0178 014450 005067 164544
0179 014454 116725 164540
0180 014460 105167 164534
0181 014464 116725 164530
0182 014470 105267 164525
0183 014474 116767 164521 164516
0184 014502 022705 021722
0185 014506 001362
0186 014510 000207
0187
0188
0189
0190 014512 042777 000104 164674      CHKDAT: BIC      #104,$DLXCSR    ;DISABLE BOTH XMIT AND RCVR INTR. ENAB.
0191 014520 042777 000100 164662      BIC      #100,$DLRCSR
0192 014526 012702 021322      MOV      #DLBUF0,R2          ;R2 POINTS TO S/B DATA IN OUTPUT BUFFER
0193 014532 004767 000070      JSR      PC,MASKING          ;GO TO MASK OFF BITS AS A FUNCTION OF
    
```

```

3194
3195 014536 012701 021722
3196 014542 122221
3197 014544 001004
3198 014546 022701 022322
3199 014552 001373
3200 014554 000207
3201 014556 013767 177776 164416
3202 014564 010667 164406
3203 014570 114204
3204 014572 042704 177400
3205 014576 114103
3206 014600 042703 177400
3207 014604 004767 177254
3208 014610 012767 014620 164426
3209 014616 104003
3210 014620 005202
3211 014622 005201
3212 014624 000750
3213
3214
3215
3216
3217
3218
3219
3220 014626 005005
3221
3222 014630 022767 000010 164376
3223 014636 001427
3224 014640 062705 000002
3225
3226 014644 022767 000007 164362
3227 014652 001410
3228 014654 062705 000002
3229
3230 014660 022767 000006 164346
3231 014666 001402
3232 014670 062705 000002
3233
3234 014674 016505 014720
3235 014700 005105
3236 014702 140522
3237 014704 022702 021722
3238
3239 014710 001374
3240 014712 012702 021322
3241 014716 000207
3242
3243 014720 000377
3244 014722 000177
3245 014724 000077
3246 014726 000037
3247
3248
3249

```

```

MOV #DLBUF1,R1
1$: CMPB (R2)+,(R1)+
BNE 3$
2$: CMP #BUFEND,R1
BNE 1$
RTS PC
3$: MOV @PSW,$TMP0
MOV SP,$REG6
MOVB -(R2),R4
BIC #177400,R4
MOVB -(R1),R3
BIC #177400,R3
JSR PC,SUERR1
MOV #4,$$ESCAPE
4$: INC R2
INC R1
BR 2$

```

```

; CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
; R1 POINTS TO WAS DATA IN RCVR. BUFFER
; DID S/B = WAS ??
; BR IF NOT
; CHECK'D ALL BYTES ??
; BR IF NOT
; RETURN TO CALLER
; SAVE THE [PSW]
; SAVE THE [SP]
; GET THE S/B DATA
; CLEAR JUNK FROM HI BYTE
; GET THE WAS DATA
; CLEAR JUNK FROM HI BYTE
; GO SET UP ERROR INFO.
; RETURN TO 4$ AFTER ERROR PRINT
; DATA COMPARE ERROR
; REPOSITION BUFFER POINTERS
; GO CHECK NEXT BYTE

```

```

; THIS ROUTINE IS USED BY THE PATTERN TESTS
; IT WILL MASK OFF THE CHARACTER SENT OUT BY THE XMITTER
; BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS TRANSMITTED
; IS DONE. THE MASKING IS DONE AS A FUNCTION OF CHARACTER LENGTH WHICH
; CAN BE EITHER 5, 6, 7, OR 8 BITS .

```

```

MASKING: CLR R5
; INITIALIZE TABLE OFFSET
; FOR PICKING UP MASK WORD
; IS THE CHARACTER LENGTH 8 BITS?
; BRANCH IF IT IS
; SET UP FOR NEXT MASK WORD
; IT COULD BE THIS ONE
; IS THE CHARACTER LENGTH 7 BITS?
; BRANCH IF IT IS
; SET UP FOR NEXT MASK WORD
; IT COULD BE THIS ONE
; IS THE CHARACTER LENGTH 6 BITS?
; BRANCH IF IT IS
; SET UP FOR NEXT MASK WORD
; IT MUST BE THIS ONE!!!!
; PICK UP THE MASK WORD
; FORM THE BITS THAT ARE TO BE MASKED
; MASK A BYTE
; ARE WE AT THE END OF THE XMITTER
; OUTPUT BUFFER
; BRANCH IF NO TO MASK NEXT BYTE
; RESTORE R2 BEFORE RETURNING
; RETURN TO MAINLINE CODE

```

```

1$: MOV CHARL(R5),R5
COM R5
2$: BICB R5,(R2)+
CMP #DLBUF1,R2
BNE 2$
MOV #DLBUF0,R2
3$: RTS PC

```

```

; TABLE OF MASK WORDS
CHARL: .WORD 377
        .WORD 177
        .WORD 77
        .WORD 37
; 8. BITS IN LENGTH
; 7. BITS IN LENGTH
; 6. BITS IN LENGTH
; 5. BITS IN LENGTH

```

```

; THIS ROUTINE IS USED BY PROGRAMS #4 & :
; IT WILL MASK OFF THE CHARACTER SENT OUT BY THE TRANSMITTER

```

```

3250                                     ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS
3251                                     ;TRANSMITTED IS DONE. THE MASKING IS DONE AS A FUNCTION OF CHARACTER
3252                                     ;LENGTH WHICH CAN BE EITHER 5, 6, 7, OR 8 BITS.
3253
3254 014730 016767 164250 164274 UPMASK: MOV     $TMP1,$TMP14   ;PICK UP THE CHARACTER THAT WAS
3255                                     ;SENT OUT FROM THE XMITTER
3256 014736 005005                 CLR     R5           ;INITIALIZE TABLE OFFSET
3257                                     ;FOR PICKING UP MASK WORD
3258 014740 022767 000010 164266     CMP     #8.,$TMP15   ;IS THE CHARACTER LENGTH 8 BITS?
3259 014746 001423                 BEQ     2$,          ;BRANCH IF IT IS
3260 014750 062705 000002                 ADD     #2,R5       ;SET UP FOR NEXT MASK WORD
3261                                     ;IT COULD BE THIS ONE
3262 014754 022767 000007 164252     CMP     #7.,$TMP15   ;IS THE CHARACTER LENGTH 7 BITS?
3263 014762 001410                 BEQ     1$,          ;BRANCH IF IT IS
3264 014764 062705 000002                 ADD     #2,R5       ;SET UP FOR NEXT MASK WORD
3265                                     ;IT COULD BE THIS ONE
3266 014770 022767 000006 164236     CMP     #6.,$TMP15   ;IS THE CHARACTER LENGTH 6 BITS?
3267 014776 001402                 BEQ     1$,          ;BRANCH IF IT IS
3268 015000 062705 000002                 ADD     #2,R5       ;SET UP FOR NEXT MASK WORD
3269                                     ;IT MUST BE THIS ONE!!!!
3270 015004 016505 014720     1$:   MOV     CHARL(R5),R5   ;PICK UP THE MASK WORD
3271 015010 005105                 COM     R5           ;FORM THE BITS THAT ARE TO BE MASKED
3272 015012 140567 164214     BICB   R5,$TMP14     ;MASK THE LOW BYTE
3273 015016 000207                 RTS     PC           ;RETURN TO MAINLINE CODE
3274
3275                                     ;ROUTINE TO SERVICE BUS ERROR TRAPS
3276
3277 015020 112767 000060 000632 BUSERR: MOVB   #60,EM4+46   ;SET UP ERROR MESSAGE
3278 015026 112767 000060 000625     MOVB   #60,EM4+47
3279 015034 112767 000064 000620     MOVB   #64,EM4+50
3280 015042 000412                 BR     TRPCOM       ;GO SET UP AND REPORT BUS ERROR
3281
3282                                     ;ROUTINE TO SERVICE RSVD INSTRUCTION TRAPS
3283
3284 015044 112767 000060 000606 RSVERR: MOVB   #60,EM4+46   ;SET UP ERROR MESSAGE
3285 015052 112767 000061 000601     MOVB   #61,EM4+47
3286 015060 112767 000060 000574     MOVB   #60,EM4+50
3287 015066 000400                 BR     TRPCOM       ;GO SET UP AND REPORT RSVD INSTR. ERROR
3288
3289                                     ;ROUTINE TO SET UP AND REPORT BUS ERROR AND RSVD INSTR ERRORS
3290
3291 015070 010667 164102 TRPCOM: MOV     SP,$REG6   ;SAVE THE TRAP SP
3292 015074 116700 164002     MOVB   $STNM,R0       ;PUT TEST NO. IN R0
3293 015100 010067 164056     MOV     R0,$REG0      ;SAVE TEST #
3294 015104 016667 000002 164070     MOV     2(SP),$TMP0    ;SAVE THE ERROR PSW
3295 015112 012767 015126 164124     MOV     #1,$$ESCAPE   ;GO TO 1$ AFTER ERROR PRINT
3296 015120 011667 164054     MOV     (SP),$REG7    ;SAVE THE ERROR PC
3297 015124 104004                 ERROR+4              ;REPORTED TRAP ERROR
3298 015126 000137 001772     1$:   JMP     @#RESTRT    ;ATTEMPT TO RESTART THE PROGRAM
3299                                     ;AND TRY AGAIN
3300
3301
3302                                     ;*****
3303                                     ;ERROR MESSAGE INFORMATION
3304                                     ;*****
3305

```

```

3306 ;INFORMATION FOR ERROR MESSAGE 1
3307
3308 015132 046104 030461 051040 EM1: .ASCIZ 'DL11 REGISTER REFERENCE CAUSED TIMEOUT'
3309 015140 043505 051511 042524
3310 015146 020122 042522 042506
3311 015154 042522 041516 020105
3312 015162 040503 051525 042105
3313 015170 052040 046511 047505
3314 015176 052125 000
3315 015201 040 050050 024503 DH1: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR'
3316 015206 020040 020040 050050
3317 015214 024523 020040 020040
3318 015222 051450 024520 020040
3319 015230 020040 042524 052123
3320 015236 020040 042040 053105
3321 015244 042101 020122 051040
3322 015252 043505 042101 000122
3323 .EVEN
3324 015260 001116 001202 001176 DT1: .WORD $ERRPC,$TMPD,$REG6,$REG0,$REG1,$REG2,0
3325 015266 001162 001164 001166
3326 015274 000000
3327
3328 ;INFORMATION FOR ERROR MESSAGE 2
3329
3330 015276 046104 030461 051040 EM2: .ASCIZ 'DL11 REGISTER ERROR'
3331 015304 043505 051511 042524
3332 015312 020122 051105 047522
3333 015320 000122
3334 015322 024040 041520 020051 DH2: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR WAS S/B'
3335 015330 020040 024040 051520
3336 015336 020051 020040 024040
3337 015344 050123 020051 020040
3338 015352 052040 051505 020124
3339 015360 020040 042504 040526
3340 015366 051104 020040 042522
3341 015374 040507 051104 020040
3342 015402 053440 051501 020040
3343 015410 020040 051440 041057
3344 015416 000
3345 .EVEN
3346 015420 001116 001202 001176 DT2: .WORD $ERRPC,$TMPD,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
3347 015426 001162 001164 001166
3348 015434 001170 001172 000000
3349
3350 ;INFORMATION FOR MESSAGE 3
3351
3352 015442 046104 030461 042040 EM3: .ASCIZ 'DL11 DATA COMPARE ERROR'
3353 015450 052101 020101 047503
3354 015456 050115 051101 020105
3355 015464 051105 047522 000122
3356 015472 024040 041520 020051 DH3: .ASCIZ ' (PC) (PS) (SP) TEST WASADR SHSADR WAS S/B'
3357 015500 020040 024040 051520
3358 015506 020051 020040 024040
3359 015514 050123 020051 020040
3360 015522 052040 051505 020124
3361 015530 020040 040527 040523

```


3362	015536	051104	020040	044123	
3363	015544	040502	051104	020040	
3364	015552	020040	040527	020123	
3365	015560	020040	020040	027523	
3366	015566	000102			
3367					.EVEN
3368	015570	001116	001202	001176	DT3: .WORD \$ERRPC,\$TMPD,\$REG6,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0
3369	015576	001162	001164	001166	
3370	015604	001170	001172	000000	
3371					
3372					; INFORMATION FOR MESSAGE 4
3373					
3374	015612	047125	054105	042520	EM4: .ASCIZ 'UNEXPECTED TRAP TO VECTOR AT LOCATION '
3375	015620	052103	042105	052040	
3376	015626	040522	020120	047524	
3377	015634	053040	041505	047524	
3378	015642	020122	052101	046040	
3379	015650	041517	052101	047511	
3380	015656	020116	020040	000040	
3381	015664	024040	041520	020051	DH4: .ASCIZ ' (PC) (PS) (SP) TEST'
3382	015672	020040	024040	051520	
3383	015700	020051	020040	024040	
3384	015706	050123	020051	020040	
3385	015714	052040	051505	000124	
3386					.EVEN
3387	015722	001200	001202	001176	DT4: .WORD \$REG7,\$TMPD,\$REG6,\$REG0,0
3388	015730	001162	000000		
3389					
3390					; ERROR INFORMATION FOR ERROR MESSAGE 5
3391					
3392	015734	046104	030461	051440	EM5: .ASCIZ 'DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)'
3393	015742	043117	020124	051105	
3394	015750	047522	020122	050050	
3395	015756	051101	052111	026131	
3396	015764	051106	046501	047111	
3397	015772	026107	047440	020122	
3398	016000	053117	051105	052522	
3399	016006	024516	000		
3400	016011	040	050050	024503	DH5: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR (REG)'
3401	016016	020040	020040	050050	
3402	016024	024523	020040	020040	
3403	016032	051450	024520	020040	
3404	016040	020040	042524	052123	
3405	016046	020040	042040	053105	
3406	016054	042101	020122	051040	
3407	016062	043505	042101	020122	
3408	016070	020040	051050	043505	
3409	016076	000051			
3410					.EVEN
3411	016100	001116	001202	001176	DT5: .WORD \$ERRPC,\$TMPD,\$REG6,\$REG0,\$REG1,\$REG2,\$REG3,0
3412	016106	001162	001164	001166	
3413	016114	001170	000000		
3414					
3415					; INFORMATION FOR ERROR MESSAGE 6
3416					
3417	016120	024040	041520	020051	DH6: .ASCIZ ' (PC) (PS) (SP) REGADR'

3418	016126	020040	024040	051520	
3419	016134	020051	020040	024040	
3420	016142	050123	020051	020040	
3421	016150	042522	040507	051104	
3422	016156	000			
3423		016160			
3424	016160	001116	001204	001176	.EVEN DT6: .WORD \$ERRPC,\$TMP1,\$REG6,\$REG2,0
3425	016166	001166	000000		
3426					
3427					
3428					; INFORMATION FOR ERROR MESSAGE 7
3429	016172	024040	041520	020051	DH7: .ASCIZ '(PC) DEVADR: REGADR (REG)'
3430	016200	020040	042504	040526	
3431	016206	051104	020040	042522	
3432	016214	040507	051104	020040	
3433	016222	024040	042522	024507	
3434	016230	000			
3435		016232			.EVEN
3436	016232	001116	001164	001166	DT7: .WORD \$ERRPC,\$REG1,\$REG2,\$REG3,0
3437	016240	001170	000000		
3438					
3439					
3440					; INFORMATION FOR ERROR MESSAGE 10
3441	016244	024040	041520	020051	DH10: .ASCIZ '(PC) DEVADR REGADR (REG) S/B'
3442	016252	020040	042504	040526	
3443	016260	051104	020040	042522	
3444	016266	040507	051104	020040	
3445	016274	024040	042522	024507	
3446	016302	020040	020040	027523	
3447	016310	000102			
3448					.EVEN
3449	016312	001116	001164	001166	DT10: .WORD \$ERRPC,\$REG1,\$REG2,\$REG3,\$REG4,0
3450	016320	001170	001172	000000	
3451					; MISCELLANEOUS MESSAGES
3452					
3453	016326	052516	046114	042055	XMSG1: .ASCIZ 'NULL-DEL-NULL SEQUENCE TIMEOUT AT FOLLOWING PC'
3454	016334	046105	047055	046125	
3455	016342	020114	042523	052521	
3456	016350	047105	042503	052040	
3457	016356	046511	047505	052125	
3458	016364	040440	020124	047506	
3459	016372	046114	053517	047111	
3460	016400	020107	041520	000	
3461	016405	102	047111	051101	XMSG2: .ASCIZ 'BINARY UP COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
3462	016412	020131	050125	041440	
3463	016420	052517	052116	051440	
3464	016426	050505	042525	041516	
3465	016434	020105	044524	042515	
3466	016442	052517	020124	052101	
3467	016450	043040	046117	047514	
3468	016456	044527	043516	050040	
3469	016464	000103			
3470	016466	044502	040516	054522	XMSG3: .ASCIZ 'BINARY DOWN COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
3471	016474	042040	053517	020116	
3472	016502	047503	047125	020124	
3473	016510	042523	052521	047105	

3474	016516	042503	052040	046511	
3475	016524	047505	052125	040440	
3476	016532	020124	047506	046114	
3477	016540	053517	047111	020107	
3478	016546	041520	000		
3479	016551	127	051117	052123	XMSG4: .ASCIZ 'WORST CASE PATTERN SEQUENCE TIMEOUT AT FOLLOWING PC'
3480	016556	041440	051501	020105	
3481	016564	040520	052124	051105	
3482	016572	020116	042523	052521	
3483	016607	047105	042503	052040	
3484	016606	046511	047505	052125	
3485	016614	040440	020124	047506	
3486	016622	046114	053517	047111	
3487	016630	020107	041520	000	
3488					
3489	016635	015	046412	044501	STMES: .ASCIZ <15><12>'MAINDEC-11-DZDLC-A'<15><12>
3490	016642	042116	041505	030455	
3491	016650	026461	055104	046104	
3492	016656	026503	006501	000012	
3493					
3494	016664	005015	047531	020125	PROG2M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 2'<15><12>
3495	016672	040510	042526	051440	
3496	016700	046105	041505	042524	
3497	016706	020104	051120	043517	
3498	016714	040522	020115	047516	
3499	016722	020056	006462	000012	
3500	016730	005015	047531	020125	PROG3M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 3'<15><12>
3501	016736	040510	042526	051440	
3502	016744	046105	041505	042524	
3503	016752	020104	051120	043517	
3504	016760	040522	020115	047516	
3505	016766	020056	006463	000012	
3506	016774	005015	047531	020125	PROG4M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 4'<15><12>
3507	017002	040510	042526	051440	
3508	017010	046105	041505	042524	
3509	017016	020104	051120	043517	
3510	017024	040522	020115	047516	
3511	017032	020056	006464	000012	
3512	017040	005015	047531	020125	PROG5M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 5'<15><12>
3513	017046	040510	042526	051440	
3514	017054	046105	041505	042524	
3515	017062	020104	051120	043517	
3516	017070	040522	020115	047516	
3517	017076	020056	006465	000012	
3518	017104	005015	051124	047101	XDB: .ASCIZ <15><12>'TRANSMITTER DONE BIT NEVER SET PC= '
3519	017112	046523	052111	042524	
3520	017120	020122	047504	042516	
3521	017126	041040	052111	047040	
3522	017134	053105	051105	051440	
3523	017142	052105	020040	041520	
3524	017150	020075	000		
3525	017153	015	051012	041505	RDB: .ASCIZ <15><12>'RECEIVER DONE BIT NEVER SET PC= '
3526	017160	044505	042526	020122	
3527	017166	047504	042516	041040	
3528	017174	052111	047040	053105	
3529	017202	051105	051440	052105	

3530	017210	020040	041520	020075	
3531	017216	000			
3532					;MESSAGES SEEKING USER RESPONSE
3533					
3534	017217	015	053412	040510	LENGTH: .ASCIZ <15><12>'WHAT IS THE CHARACTER LENGTH (5,6,7 OR 9 BITS)?'
3535	017224	020124	051511	052040	
3536	017232	042510	041440	040510	
3537	017240	040522	052103	051105	
3538	017246	046040	047105	052107	
3539	017254	020110	032450	033054	
3540	017262	033454	047440	020122	
3541	017270	020070	044502	051524	
3542	017276	037451	000		
3543	017301	015	042012	020117	DEFAULT: .ASCII <15><12>'DO YOU WISH TO TEST OTHER THAN THE'
3544	017306	047531	020125	044527	
3545	017314	044123	052040	020117	
3546	017322	042524	052123	047440	
3547	017330	044124	051105	052040	
3548	017336	040510	020116	044124	
3549	017344	105			
3550	017345	015	042012	043105	.ASCIZ <15><12>'DEFAULT DEVICE (1/0 = YES/NO)?'
3551	017352	052501	052114	042040	
3552	017360	053105	041511	020105	
3553	017366	030450	030057	036440	
3554	017374	054440	051505	047057	
3555	017402	024517	000077		
3556	017406	005015	044127	052101	MFIRSTD: .ASCIZ <15><12>'WHAT IS THE 1ST RECEIVER STATUS REGISTER ADDRESS?'
3557	017414	044440	020123	044124	
3558	017422	020105	051461	020124	
3559	017430	042522	042503	053111	
3560	017436	051105	051440	040524	
3561	017444	052524	020123	042522	
3562	017452	044507	052123	051105	
3563	017460	040440	042104	042522	
3564	017466	051523	020077	000040	
3565	017474	005015	044127	052101	MVECT: .ASCIZ <15><12>'WHAT IS THE 1ST RECEIVER'S VECTOR ADDRESS?'
3566	017502	044440	020123	044124	
3567	017510	020105	051461	020124	
3568	017516	042522	042503	053111	
3569	017524	051105	020123	042526	
3570	017532	052103	051117	040440	
3571	017540	042104	042522	051523	
3572	017546	020077	000040		
3573	017552	005015	047504	054440	MULDEV: .ASCIZ <15><12>'DO YOU WANT TO TEST MULTIPLE DEVICES 1/0=YES/NO?'
3574	017560	052517	053440	047101	
3575	017566	020124	047524	052040	
3576	017574	051505	020124	052515	
3577	017602	052114	050111	042514	
3578	017610	042040	053105	041511	
3579	017616	051505	030440	030057	
3580	017624	054475	051505	047057	
3581	017632	037517	020040	000	
3582	017637	015	053412	040510	MLASTD: .ASCIZ <15><12>'WHAT IS THE STATUS REGISTER ADDRESS OF THE LAST RECEIVER?'
3583	017644	020124	051511	052040	
3584	017652	042510	051440	040524	
3585	017660	052524	020123	042522	

3586	017666	044507	052123	051105
3587	017674	040440	042104	042522
3588	017702	051523	047440	020106
3589	017710	044124	020105	040514
3590	017716	052123	051040	041505
3591	017724	044505	042526	037522
3592	017732	020040	000	
3593	017735	015	051412	046517
3594	017742	052105	044510	043516
3595	017750	053440	047522	043516
3596	017756	040455	051516	042527
3597	017764	020122	044124	020105
3598	017772	040514	052123	050440
3599	020000	042525	052123	047511
3600	020006	020116	043501	044501
3601	020014	020516	020040	000
3602	020021	015	053412	040510
3603	020026	020124	051511	054440
3604	020034	052517	020122	047111
3605	020042	042524	051122	050125
3606	020050	020124	051120	047511
3607	020056	044522	054524	046040
3608	020064	053105	046105	020077
3609	020072	000040		
3610	020074	005015	051120	043517
3611	020102	040522	020115	047504
3612	020110	044526	042503	040440
3613	020116	052103	053111	020105
3614	020124	047514	040503	044524
3615	020132	047117	051440	047510
3616	020140	051527	047040	020117
3617	020146	042504	044526	042503
3618	020154	040440	052103	053111
3619	020162	105		
3620	020163	015	051412	052105
3621	020170	051440	044527	041524
3622	020176	020110	020060	047524
3623	020204	040440	047440	042516
3624	020212	024040	024461	040440
3625	020220	042116		
3626	020222	005015	044510	020124
3627	020230	047503	052116	047111
3628	020236	042525	052040	020117
3629	020244	047507	041040	041501
3630	020252	020113	047524	042040
3631	020260	053105	041511	020105
3632	020266	042523	042514	052103
3633	020274	047511	020116	043501
3634	020302	044501	000116	
3635	020306	005015	044127	052101
3636	020314	044440	020123	044124
3637	020322	020105	051124	047101
3638	020330	046523	052111	042524
3639	020336	020122	040504	040524
3640	020344	041040	043125	042506
3641	020352	020122	042101	051104

MRANGE: .ASCIZ <15><12>'SOMETHING WRONG-ANSWER THE LAST QUESTION AGAIN! '

PLEVEL: .ASCIZ <15><12>'WHAT IS YOUR INTERRUPT PRIORITY LEVEL? '

FOULUP: .ASCII <15><12>'PROGRAM DEVICE ACTIVE LOCATION SHOWS NO DEVICE ACTIVE'

.ASCII <15><12>'SET SWITCH 0 TO A ONE (1) AND'

.ASCIZ <15><12>'HIT CONTINUE TO GO BACK TO DEVICE 'LECTION AGAIN'

LINTAD: .ASCIZ <15><12>'WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS? '

3642	020360	051505	037523	020040	
3643	020366	000			
3644	020367	015	053412	040510	SELCAR: .ASCIZ <15><12>'WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G. A=101
3645	020374	020124	051511	052040	
3646	020402	042510	041440	040510	
3647	020410	040522	052103	051105	
3648	020416	052040	020117	042502	
3649	020424	052040	040522	051516	
3650	020432	044515	052124	042105	
3651	020440	024040	041517	040524	
3652	020446	020114	051501	044503	
3653	020454	020111	027105	027107	
3654	020462	040440	030475	030460	
3655	020470	037451	020040	000	
3656	020475	015	053412	040510	SELDLY: .ASCIZ <15><12>'WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G. 10=8(10))?
3657	020502	020124	051511	052040	
3658	020510	042510	042040	051505	
3659	020516	051111	042105	046440	
3660	020524	042523	027103	042040	
3661	020532	046105	054501	024040	
3662	020540	041517	040524	020114	
3663	020546	027105	027107	030440	
3664	020554	036460	024070	030061	
3665	020562	024451	020077	000040	
3666	020570	005015	044127	052101	LINRAD: .ASCIZ <15><12>'WHAT IS THE RECEIVER DATA BUFFER ADDRESS?
3667	020576	044440	020123	044124	
3668	020604	020105	042522	042503	
3669	020612	053111	051105	042040	
3670	020620	052101	020101	052502	
3671	020626	043106	051105	040440	
3672	020634	042104	042522	051523	
3673	020642	020077	000040		
3674	020646	005015	051511	040440	RSTALL: .ASCIZ <15><12>'IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO?
3675	020654	051040	047101	047504	
3676	020662	020115	040527	052111	
3677	020670	052040	046511	020105	
3678	020676	046450	042523	027103	
3679	020704	020051	042504	044523	
3680	020712	042522	020104	030440	
3681	020720	030057	054475	051505	
3682	020726	047057	037517	020040	
3683	020734	000			
3684	020735	015	054412	052517	FAILSA: .ASCII <15><12>'YOU HAVE SWRB SET INDICATING LOOP ON TEST'
3685	020742	044040	053101	020105	
3686	020750	053523	034122	051440	
3687	020756	052105	044440	042116	
3688	020764	041511	052101	047111	
3689	020772	020107	047514	050117	
3690	021000	047440	020116	042524	
3691	021006	052123			
3692	021010	005015	040510	042526	.ASCII <15><12>'HAVE YOU MODIFIED THE PROPER LOCATIONS FOR THE'
3693	021016	054440	052517	046440	
3694	021024	042117	043111	042511	
3695	021032	020104	044124	020105	
3696	021040	051120	050117	051105	
3697	021046	046040	041517	052101	

```

3698 021054 047511 051516 043040
3699 021062 051117 052040 042510
3700 021070 005015 042504 044526
3701 021076 042503 052040 040510
3702 021104 020124 047531 020125
3703 021112 040527 052116 052040
3704 021120 020117 042524 052123
3705 021126 077
3706 021127 015 044412 020106
3707 021134 047523 026440 050040
3708 021142 042522 051523 052040
3709 021150 042510 041440 047117
3710 021156 044524 052516 020105
3711 021164 053523 052111 044103
3712 021172 005015 043111 047040
3713 021200 052117 026440 046440
3714 021206 042117 043111 020131
3715 021214 044124 020105 051120
3716 021222 050117 051105 046040
3717 021230 041517 052101 047511
3718 021236 051516 020054 044124
3719 021244 047105
3720 021246 005015 042522 052123
3721 021254 051101 020124 044124
3722 021262 020105 051120 043517
3723 021270 040522 020115 052101
3724 021276 040440 042104 042522
3725 021304 051523 031040 030060
3726 021312 000
3727
3728 021313 040 020075 041520
3729 021320 000040
3730
3731
3732
3733
3734 021322 000400
3735
3736
3737 021722 000400
3738
3739
3740
3741 022322 000000
3742
3743 000001
  
```

```

.ASCII <15><12>'DEVICE THAT YOU WANT TO TEST?'
.ASCII <15><12>'IF SO - PRESS THE CONTINUE SWITCH'
.ASCII <15><12>'IF NOT - MODIFY THE PROPER LOCATIONS, THEN'
.ASCIIZ <15><12>'RESTART THE PROGRAM AT ADDRESS 200'
  
```

```

PCMSG: .ASCII : = PC'
        .ASCIIZ : ;
  
```

```

.EVEN
;512. WORDS RESERVED FOR TWO 256. BYTE INPUT/OUTPUT DATA BUFFERS
DLBUFO: .BLKB 256. ;RSVD FOR OUTPUT BUFFER
;THIS IS THE DATA BEING SENT OUT
;BY THE TRANSMITTER
DLBUFI: .BLKB 256. ;RSVD FOR INPUT BUFFER
;THIS IS THE DATA THAT WAS PICKED
;UP BY THE RECEIVER (I.E. DATA
;SENT BY THE TRANSMITTER - HOPEFULLY)
;TAG MARKS END OF BUFFERS
BUFEND: 0
.END
  
```


	2236	2251	2257	2259	2269	2270	2271	2272	2273	2278	2290	2292	2293	
	2296	2297	2298	2305	2306	2307	2318	2321	2324	2332	2333	2334	2335	
	2336	2344	2351	2356	2359	2367	2932							
SSWRMK=	000000	442	443	2273	2274	2294								
STIMES	001242	642#	821*	1527*	1584*	1641*	1698*	2236*	2305*	2312	2315*	2324		
STKB	001146	609#	2565	2582	2588									
STIS	001144	608#	2565	2580	2586									
STMP0	001202	626#	1299*	1787*	1823*	1869*	1908*	1954*	1998	2009*	2063*	2107	2118*	2946*
		2964*	2970*	3082*	3095*	3201*	3294*	3324	3346	3368	3387	3411		
STMP1	001204	627#	1289*	1314*	1325	1573	1630	1687	1741	1812*	1823	1893*	1908	1983*
		2009	2105*	2118	2972*	3100*	3254	3424						
STMP10	001222	634#												
STMP11	001224	635#	2100*	2101	2104*									
STMP12	001226	636#	2101*	2102*	2103*	2104	2105							
STMP13	001230	637#												
STMP14	001232	638#	3069	3071	3254*	3272*								
STMP15	001234	639#	845*	940*	1996*	2091*	3222	3226	3230	3258	3262	3266		
STMP16	001236	640#												
STMP17	001240	641#												
STMP2	001206	628#	1820*	1821	1901*	1906	1977*	2000	2096*	2109				
STMP3	001210	629#	1998*	1999*	2007*	2015	2107*	2108*	2116*	2123	3031	3037		
STMP4	001212	630#	2015*	2017*	2019	2123*	2125*	2127	3056					
STMP5	001214	631#	2019*	2127*	3054	3062	3068							
STMP6	001216	632#	3039*	3040*	3054*	3064								
STMP7	001220	633#	3165*	3167	3170*	3178*	3179	3180*	3181	3182*	3193			
STN =	000026	427#	1137	1141#	1152	1156#	1167	1171#	1182	1186#	1197			
		1211#	1214	1217	1221#	1231	1234	1238#	1255	1260	1263			
		1284#	1312	1329	1332	1336#	1337	1359	1363	1367#	1368			
		1404	1430	1434	1438#	1439	1450	1453	1457#	1458	1477			
		1494	1505	1510	1514#	1519	1523	1527#	1541	1552	1561			
		1598	1609	1618	1627	1637	1641#	1655	1666	1675	1684			
STPB	001152	611#	2778*	2789										
STPFLG	001157	615#	2736	2789										
STPS	001150	610#	2776	2789										
STRAP	013052	817	1760	1842	1927	2036	2859#							
STRAP2	013074	2870#	2881											
STRP =	000012	2874#	2883#	2884#	2885#	2886#	2887#	2889	2890#	2891#	2892#	2893#		
STRPAD	013106	2864	2881#											
STSTNM	001102	588#	2214*	2235*	2268	2294	2316*	2321	2325	2343	2367	3087	3096	3292
STTYIN	012250	2608	2609	2621	2639	2653	2657#							
STYPBN=	***** U	2887												
STYPDS	011450	2506#	2886											
STYPE	012454	2736#	2874	2882										
STYPEC	012624	2757	2764	2771	2776#	2777								
STYPEX	012672	2782	2784	2787#										
STYPOC	011246	2446#	2883											
STYPON	011262	2445	2448#	2885										
STYPOS	011222	2441#	2884											
SXTSTR	010472	2281#												
SSGET4=	000000	2251#												
SOFILL	011445	2442*	2446*	2456	2491#									
S4OCAT=	***** U	2278	2353											
	= 022324	449#	453#	463#	585#	648	810	824	825	1361	2259	2263	2324	2325
		2367	2414#	2560#	2565	2657#	2658	2664	2718	2799	2850	2910	2934	3345#
		3423#	3435#	3734#	3737#									



.SWRHI	412#	431
.SWRLO	412#	443#
.SCATC	412#	447
.SCMTA	411#	579
.SEOP	412#	2163
.SERRO	411#	2326
.SERRT	413#	2368
.SPOWE	413#	2894
.SRDDE	414#	2790
.SRDOC	414#	2665
.SREAD	414#	2562
.SSCOP	413#	2263
.STRAP	413#	2851
.STYPD	413#	2494
.STYPE	413#	2719
.STYPO	413#	2416

444

1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487	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
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

1174

1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200

1174

1157

1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200

1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200

111111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111
111111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111
111111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111	411111

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

* DZDLOA SEQ/SOL/CRF/PAGNUM-DZDLOA
CPU TIME: 60.56 6 SECONDS
CPU I/O: 188/103=1.8
CORE USED: 28K (51 PAGES)

