

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

.REMA

IDENTIFICATION

PRODUCT CODE: AC-8524D-MC
PRODUCT NAME: CZDLCD DL11-C,D,E OFLNE TST
PRODUCT DATE: MAY 1980
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: E. CROWLEY/B. BURGESS

COPYRIGHT (C) 1975, 1980 DIGITAL EQUIPMENT CORPORATION

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

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

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

TABLE OF CONTENTS

56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112

1.0 PROGRAM PURPOSE (ABSTRACT)
2.0 SYSTEM REQUIREMENTS
3.0 RELATED DOCUMENTS AND STANDARDS
4.0 DIAGNOSTIC HIERARCHY PREREQUISITES
5.0 LOADING AND STARTING PROCEDURE
6.0 SPECIAL ENVIRONMENTS
7.0 PROGRAM OPTIONS
8.0 EXECUTION TIMES
9.0 ERROR INFORMATION
9.1 ERROR REPORTING
9.2 ERROR HALTS
10.0 PERFORMANCE AND PROGRESS REPORTS
11.0 DEVICE INFORMATION TABLES
12.0 SUBROUTINE SUMMARIES
THRU
12.20
13.0 MISCELLANEOUS
14.0 USER SELECTION PROGRAMS
14.1 PROGRAM #2 DESCRIPTION
14.2 PROGRAM #3 DESCRIPTION
14.3 PROGRAM #4 DESCRIPTION
14.4 PROGRAM #5 DESCRIPTION
15.0 PROGRAM FUNCTIONAL FLOW CHARTS
16.0 PROGRAM LISTING
17.0 ECO HISTORY

1.0 PROGRAM PURPOSE (ABSTRACT)

THIS PROGRAM HAS THE ABILITY TO TEST THE DL11 (ASYNCHRONOUS MODEM INTERFACE), OFF LINE. MODELS ABLE TO BE TESTED ARE C, D, AND E ONLY. THE USE OF A MODEM IS NOT REQUIRED FOR TESTING; HOWEVER, A SPECIAL CABLE CONNECTOR BC05C AND A SPECIAL MODEM TEST CONNECTOR H315A IS REQUIRED. THIS PROGRAM IS CAPABLE OF THE FOLLOWING:

A. VERIFICATION OF MAINTENANCE BIT

113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168

- B. VERIFICATION THAT TRANSMITTER CAN CAUSE AN INTERRUPT
- C. VERIFICATION THAT RECEIVER 'DONE' CAN CAUSE AN INTERRUPT
- D. CHECKS THAT 'REQ TO SEND' ASSERTS 'RING'
- E. CHECKS THAT 'SEC XMIT' ASSERTS 'SEC REC' AND 'DATA SET INT'
- F. CHECKS THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'
- G. VERIFIES THAT 'DATA SET I.E.' CAN CAUSE A RECVR INTR
- H. CHECKS THE 'BREAK' FEATURE
- I. PERFORMS NULL-DEL-NUL PATTERN
- J. PERFORMS BINARY UP COUNT PATTERN
- K. PERFORMS BINARY DOWN COUNT PATTERN
- L. RUNS A WORSE CASE PATTERN

INCLUDED IN THE PROGRAM ARE SPECIAL USER ROUTINES - PRG #2, PRG #3, PRG #4, AND PRG #5 (WHICH WILL BE DESCRIBED FURTHER INTO THIS DOCUMENT).

NOTE WELL TWO(2) POINTS:

1. THIS PROGRAM IS CAPABLE OF TESTING SIXTEEN(16) DL11'S AND ASSUMES CONTIGUOUS ADDRESSING FROM 1ST DEVICE TO LAST.
 - A. IF MULTIPLE DEVICES ARE NOT BEING TESTED, THUS NOT REQUIRING A PASS THRU THE PROGRAM ONCE PER DEVICE, THEN THE PROGRAM WILL DEFAULT TO TESTING THE 1ST POSSIBLE DL11-E DEVICE I.E., RCSR ADDRESS = 775610, AND TEST THIS DEVICE ONLY.
 - B. IF MULTIPLE DEVICE TESTING IS NOT BEING CONDUCTED, AND THE DEVICE EXISTING IS NOT THE DEFAULT DL11-E, THEN THE USER ON STARTING THE PROGRAM WILL HAVE TO SET SW<0>=1 TO ENTER THE QUESTION & ANSWER MODE.
2. THIS PROGRAM HAS PROVISION FOR CHARACTER LENGTH I.E., IT ASSUMES DATA IS 8 BITS, BUT ALSO HAS THE ABILITY TO HANDLE 5, 6, OR 7 BITS OF DATA AS WELL.

2.0 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

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

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)

169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224

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
MAINDEC-11-DZQAC-C3
- E. 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.

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

- 210 - SELECTS PROGRAM #2
- 220 - SELECTS PROGRAM #3
- 230 - SELECTS PROGRAM #4
- 240 - SELECTS PROGRAM #5

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

6.0 SPECIAL ENVIRONMENTS

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

6.2 FOR USE WITH PROCESSOR THAT DOES NOT HAVE A KEYBOARD.

IF A HARDWARE SWITCH REGISTER DOES NOT EXIST, THE PROGRAM WILL USE THE CONTENTS OF LOCATION 176 AS THE VALUE OF THE SWITCHES. THE PROGRAM WILL PRINT OUT THE PRESENT CONTENTS OF THE SOFTWARE SWITCH REGISTER WHEN THE PROGRAM IS STARTED. IT WILL THEN ASK FOR THE NEW CONTENTS TO BE INPUT TO THE SOFTWARE SWITCH REGISTER. TYPE CARRIAGE RETURN TO FINISH INPUT.

7.0 PROGRAM OPTIONS

<u>SWITCH</u>	<u>USE</u>
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 SW<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:
DLVCT:

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

8.0 EXECUTION TIMES

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
328
329
330
331
332
333
334
335
336

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

337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392

TO ALLOW THE INFORMATION TRANSFER.

B. IN THE POWER FAIL ROUTINE IF THE POWER UP SEQUENCE WAS

STARTED BEFORE THE POWER DOWN SEQUENCE HAD A CHANCE TO COMPLETE ITSLEF.

C. IN THE END OF PASS ROUTINE IF MULTIPLE DEVICE TESTING IS BEING CONDUCTED BUT NO DEVICES ARE SHOWN AS ACTIVE.

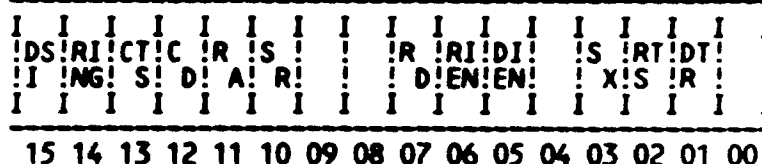
D. IN THE CASE OF SW<08> BEING SET.

10.0 PERFORMANCE AND PROGRESS REPORTS

NOT APPLICABLE.

11.0 DEVICE INFORMATION TABLES

A. THE FOLLOWING IS A PICTURE VIEW OF A DL11-E RECEIVER CONTROL STATUS REGISTER, WHICH WILL SHOW BIT ASSIGNMENTS AND DEFINITIONS, TO PROVIDE A HANDY REFERENCE:



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

- BIT15 DATA SET INTERRUPT
 - 1. INTERRUPT SEQUENCE INITIATED WHEN BIT05 SET.
 - 2. SETS WHENEVER BITS 10, 11, 12 OR 14 CHANGE STATE
 - 3. CLEARED BY INIT OR READING RCSR

- BIT14 RING
 - 1. WHEN SET, INDICATES A CONTROL SIGNAL BEING RECEIVED FROM DATASET.

- BIT13 CLEAR TO SEND
 - 1. WHEN SET INDICATES ON CONDITION: WHEN CLEAR INDICATES OFF CONDITION.
 - 2. DEPENDENT ON STATE OF 'CTS' SIGNAL FROM DATASET

- BIT12 CARRIER DETECT
 - 1. SETS WHEN DATA CARRIER

393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448

BIT11 RECEIVER ACTIVE

- RECEIVED
- 2. WHEN CLEAR INDICATES END OF CURRENT TRANSMISSION OR AN ERROR CONDITION.

- 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 INITIATE AN INTERRUPT PROVIDING BIT06 IS ALSO SET
- 2. CLEARED WHEN RDBR IS ADDRESSED OR BIT00 IS SET.
- 3. ALSO, CLEARED BY INIT

BIT06 RECEIVER INTERRUPT ENABLE

- 1. WHEN SET, ALLOWS INTERRUPT PROVIDING BIT07 IS SET.
- 2. CLEARED BY INIT
- 3. ***READ/WRITE BIT***

BIT05 DATASET INTERRUPT ENABLE

- 1. WHEN SET, ALLOWS INTERRUPT PROVIDING BIT15 IS SET.
- 2. CLEARED BY INIT
- 3. ***READ/WRITE BIT***

BIT03 SECONDARY TRANSMIT OR SUPERVISORY TRANSMITTED DATA

- 1. PROVIDES TRANSMIT CAPABILITY, WHEN SET, FOR REVERSE CHANNEL OF REMOTE STATION. SETS WHEN BIT10 IS SET.
- 2. CLEARED BY INIT
- 3. ***READ/WRITE BIT***

BIT02 REQUEST TO SEND

- 1. JUMPER TIES THIS BIT TO REQ TO SEND IN DATASET.
- 2. REQUIRED FOR TRANSMISSION
- 3. CLEARED BY INIT
- 4. ***READ/WRITE BIT***

BIT01 DATA TERMINAL READY

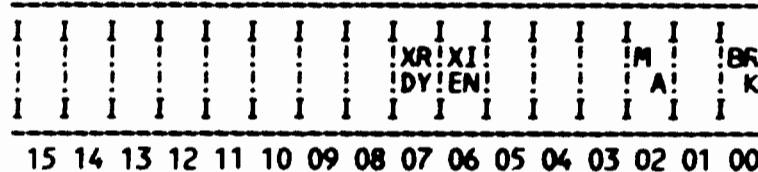
- 1. WHEN SET, PERMITS CONNECTION TO CHANNEL.
- 2. WHEN CLEAR, DISCONNECTS INTERFACE FROM CHANNEL.

449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504

- 3. MUST BE CLEARED BY PROGRAM
- 4. ***READ/WRITE BIT***

SPECIAL NOTES ON RCSR REGISTER

- 1. ADDRESSES SHOULD FALL IN THE RANGE OF 175610 TO 176170
 - 2. BIT01 (DATA TERMINAL READY) STATE IS NOT DEFINED AFTER POWER-UP.
 - 3. ON DL11-C OR -D OPTIONS BITS 15, 14, 13, 12, 10, 5, 3, 2, AND 1 ARE NOT USED.
 - 4. ON DL11-C AND -D OPTIONS BIT<00> IS 'RDR ENB' . ON A DL11-E OPTION THIS BIT IS UNUSED.
- B. THE FOLLOWING IS A PICTURE VIEW OF A DL11-E TRANSMITTER CONTROL STATUS REGISTER, WHICH WILL SHOW BIT ASSIGNMENTS AND DEFINITIONS, TO PROVIDE A HANDY REFERENCE:



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

- BIT07 TRANSMITTER READY
 - 1. SET WHEN XDOR CAN ACCEPT ANOTHER CHARACTER. WILL INITIATE AN INTERRUPT IF BIT06 ALSO SET.
 - 2. ALSO SET BY INIT
 - 3. CLEARED BY LOADING XDOR
- BIT06 TRANSMITTER INTERRUPT ENABLE
 - 1. WHEN SET, ALLOWS INTERRUPT PROVIDING BIT07 IS SET.
 - 2. CLEARED BY INIT
 - 3. ***READ/WRITE BIT***
- BIT02 MAINTENANCE
 - 1. WHEN SET, DISABLES SERIAL LINE INPUT TO RECEIVER & CONNECTS XMIT OUTPUT TO RECEIVER INPUT WHICH DISCONNECTS EXTERNAL DEVICE INPUT. THIS FORCES RECEIVER TO RUN AT XMITTER SPEED.
 - 2. CLEARED BY INIT
 - 3. ***READ/WRITE BIT***

505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560

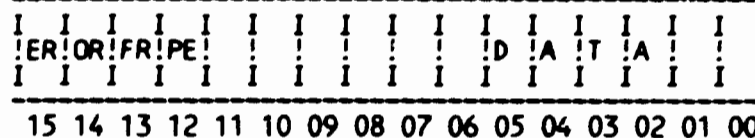
BIT00 BREAK

1. WHEN SET, TRANSMITS A CONTINUOUS SPACE TO EXTERNAL DEVICE
2. CLEARED BY INIT
3. ***READ/WRITE BIT***

!! NOTE !!

DL11-C AND -D OPTIONS ARE THE SAME.

C. THE FOLLOWING IS A PICTURE VIEW OF THE DL11-E RECEIVER AND TRANSMITTER DATA BUFFER REGISTERS, TO PROVIDE A HANDY REFERENCE.



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BITS 07-00 DATA

1. CHARACTER TO BE TRANSFERRED TO EXTERNAL DEVICE.
2. IF CHARACTER LESS THAN 8 BITS IT MUST BE LOADED RIGHT JUSTIFIED.
3. ***WRITE ONLY BITS***

BIT 15 ERROR

1. ***READ ONLY BIT***
2. CLEARED BY ERROR REMOVAL

BIT 14 OVERRUN

1. SAME AS BIT 15
2. RCVR DONE NOT CLEARED

BIT 13 FRAMING

1. SAME AS BIT 15
2. NO VALID STOP BIT

BIT 12 PARITY

1. SAME AS BIT 15
2. PARITY OTHER THAN EXPECTED

NOTE: BITS<15:12> ONLY APPEAR IN THE RCVR DATA BUFFER DL11-C AND -D OPTIONS ARE THE SAME.

12.0 SUBROUTINE SUMMARIES

561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616

12.1 DLADDR

THIS ROUTINE SETS UP THE FOLLOWING:

- RCSR - RECEIVER STATUS REGISTER
- RBUF - RECEIVER BUFFER REGISTER
- XCSR - TRANSMITTER STATUS REGISTER
- XBUF - TRANSMITTER BUFFER REGISTER

THE SETUP IS DONE, INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS TESTED, AND THEREAFTER, AT THE END OF A PROGRAM PASS TO ALLOW CYCLING THRU ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED).

12.2 \$EOP

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQACC3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS RESPONSIBLE FOR THE FOLLOWING:

- A. INCREMENTING THE PASS NUMBER (\$PASS)
- B. TYPING 'END PASS # XXX' (WHERE 'XXX' IS A DECIMAL VALUE)

NOTE

IF MULTIPLE DEVICE TESTING IS BEING CONDUCTED, THEN \$PASS IS ONLY INCREMENTED AFTER TESTING OF ALL DEVICES HAS TRANSPIRED (MULTIPLE TESTING). THEREFORE, E.G., IF 10 DEVICES HAVE BEEN TESTED THEN 'END PASS #1' WOULD BE TYPED OUT. 'END PASS #2' WOULD BE TYPED OUT AFTER THE 10 DEVICES HAVE ONCE AGAIN BEEN TESTED BY THE PROGRAM, ETC.

- C. GOES TO A MONITOR, IF THERE IS ONE
- D. IF THERE IS NO MONITOR TRANSFERS CONTROL BACK TO BEGINNING OF THE PROGRAM.

12.3 \$\$SCOPE

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE.

THIS ROUTINE IS ENTERED BEFORE AND AFTER EVERY SUBTEST TO ASCERTAIN THE FOLLOWING CONDITIONS:

- A. LOOP ON TEST JUST EXECUTED?
THIS CONDITION IS ENABLED WHEN SW<14> IS SET TO A '1'.
- B. LOOP ON TEST IF AN ERROR HAS OCCURRED DURING THE TEST?
THIS CONDITION IS ENABLED WHEN SW<09> IS SET TO A '1'.

617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672

- C. LOOP ON TEST SPECIFIED BY TEST NO. APPEARING IN SW<7:0>?
THIS CONDITION IS ENABLED WHEN SW<08> IS SET TO A '1'.
- D. INHIBIT SUBTEST ITERATIONS?
THIS CONDITION IS ENABLED WHEN SW<11> IS SET TO A '1'.

12.4 \$ERROR

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE.

THIS ROUTINE HANDLES THE FOLLOWING REACTIONS TO AN ERROR WHEN AN ERROR IS ENCOUNTERED:

- A. 'HALT' ON ERROR?
THIS CONDITION IS ENABLED WHEN SW<15> IS SET TO A '1'.
- B. RING 'BELL' ON ERROR?
THIS CONDITION IS ENABLED WHEN SW<10> IS SET TO A '1'.
- C. LOOP ON ERROR
THIS CONDITION IS ENABLED WHEN SW<09> IS SET TO A '1'.
- D. INHIBIT ERROR TYPEOUTS
THIS CONDITION IS ENABLED WHEN SW<13> IS SET TO A '1'.

NOTE

ON ENCOUNTERING AN ERROR WHILE EXECUTING THE PROGRAM THIS ROUTINE WILL TRANSFER CONTROL TO '\$ERRTYP' ROUTINE SHOWN BELOW (PRESUMES 'HALT' ON ERROR SW<15> NOT SET).

12.5 \$ERRTYP

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE.

THIS ROUTINE HANDLES THE INFORMATION FOR ERROR MESSAGE TYPEOUTS AS FOLLOWS:

THIS ROUTINE USES THE 'ITEM CONTROL BYTE' (\$ITEMB) TO DETERMINE WHICH ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' (\$ERRTB) THE ADDRESSES OF WHERE THE INFORMATION, FOR PRINTOUT, IS STORED; AND CAUSES THE APPROPRIATE INFORMATION CONCERNING THE ERROR TO BE PRINTED OUT.

673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728

- NOTE: 1. THE VARIABLE '\$ITEMB' IS SUPPLIED BY .SCMTAG, A 'SYSMAC' UTILITY PACKAGE ROUTINE.
2. THE 1ST ADDRESS '\$ERRTB' FOR LOCATION OF 'ERROR TABLE' INFORMATION IS ALSO SUPPLIED BY .SCMTAG.
3. IF THE '\$ITEMB' VALUE IS ZERO(0), THEN ONLY A PROGRAM COUNTER (PC) IS PRINTED OUT. IT HAS NO LABEL, IT IS A PURE NUMBER.

12.6 \$TYPOC

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS USED FOR ALL OCTAL TYPEOUTS (16 BIT VALUES) THROUGHOUT THE PROGRAM.

12.7 \$TYPDS

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS USED TO TYPE A DECIMAL VALUE AT THE END OF A PASS OF THE PROGRAM OF THE FORM 'END PASS # XXX' WHERE 'XXX' IS THE DECIMAL VALUE.

12.8 \$RDCHR, \$RDLIN, \$RDOCT

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THEIR USES ARE AS FOLLOWS:

- A. \$RDCHR - HANDLES A SINGLE CHARACTER COMING IN FROM THE TTY. THE CHARACTER IS PLACED ON TOP OF THE STACK FOR FUTURE USE.
- B. \$RDLIN - HANDLES A STRING OF CHARACTERS COMING IN FROM THE TTY. THE ADDRESS OF THE 1ST CHARACTER IS PLACED ON TOP OF THE STACK FOR FUTURE USE.
- C. \$RDOCT - HANDLES AN OCTAL NUMBER COMING IN FROM THE \$RDOCT 104420 TTY DECIMAL # INPUT TTY. LOW ORDER BITS ARE STORED ON TOP OF THE STACK; HIGH ORDER BITS ARE STORED IN LOCATION \$HIOCT. \$HIOCT IS SUPPLIED BY .SCMTAG, A 'SYSMAC' PACKAGE UTILITY ROUTINE.

12.9 \$TYPE

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS USED TO

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
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784

TYPE ASCII MESSAGES (WHICH MUST TERMINATE WITH A 0 BYTE) AS WELL AS ALL OTHER FORMS OF TYPED INFORMATION. THE ROUTINE IS ALSO RESPONSIBLE FOR INSERTING A NUMBER OF FILL CHARACTERS AFTER A LINE FEED.

- NOTE:
1. \$NULL CONTAINS THE CHARACTER TO BE USED AS FILL.
 2. \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQ'D.
 3. \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 4. THE ABOVE THREE(3) VARIABLES ARE SUPPLIED BY .SCMTAG, A 'SYSMAC' PACKAGE UTILITY ROUTINE.

12.10 STRAP, STRPAD

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THE 'STRAP' ROUTINE WILL STRIP OFF THE LOWER BYTE OF A TRAP INSTRUCTION AND USE IT TO INDEX THRU THE TRAP TABLE (STRPAD) FOR THE STARTING ADDRESS OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL THEN TRANSFER PROGRAM CONTROL TO THAT ROUTINE.

THE FOLLOWING TABLE DEFINES ALL ROUTINES IN THE PROGRAM CALLED BY A 'TRAP' INSTRUCTION BY SHOWING THEIR 'TRAP' EQUIVALENCES -

STYPE	104400	TTY TYPEOUT ROUTINE
STYPOC	104402	TYPE OCTAL # (WITH LEADING ZEROS)
STYPOS	104404	TYPE OCTAL # (NO LEADING ZEROS)
STYPOH	104406	TYPE OCTAL # (PER LAST CHARACTER METHOD)
STYPDS	104410	TYPE DECIMAL # (WITH SIGN)
SRDCHR	104412	TTY CHARACTER INPUT
SRDLIN	104414	TTY STRING INPUT
SRDOCT	104416	TTY OCTAL # INPUT

12.11 SPWRDN, SPWRUP

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THESE ROUTINES HANDLE THE 'POWER DOWN AND UP' SEQUENCE. THE PROGRAM MAY BE POWER FAILED WHEN RUNNING; HOWEVER, USE CAUTION IN TURNING POWER OFF/ON WHILE THE POWER FAIL MESSAGE IS BEING TYPED - IT MAY CAUSE STACK OVERFLOW.

NOTE

WHEN POWER RETURNS THE PROGRAM WILL

AUTOMATICALLY START ITSELF OVER AT THE
BEGINNING.

785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840

12.12 XINT, RINT

XINT - THIS IS THE TRANSMITTER INTERRUPT SERVICE ROUTINE
FOR 256(10) BYTE BLOCK TRANSFERS.

RINT - THIS IS THE RECEIVER INTERRUPT SERVICE ROUTINE FOR
256(10) BYTE BLOCK TRANSFERS.

12.13 DELAY, STALL, DATCHK, TIMERX, TIMETX

THESE ROUTINES ARE ALL USED BY PROGRAMS 2, 3, 4 AND 5.
PROGRAMS 2 THROUGH 5 ARE THE 'SPECIAL' USER INTERACTION
ROUTINES WHICH WILL BE DEFINED LATER IN THIS DOCUMENT. THE
ABOVE ROUTINE USES ARE AS FOLLOWS:

A. DELAY - THIS ROUTINE IS USED BY ALL THE UTILITY PROGRAMS
TO WAIT A NO. OF MILLISECONDS BETWEEN CHARACTER
TRANSFERS AS SPECIFIED BY THE USER.

B. STALL - THIS ROUTINE IS USED BY PROGRAM #4 AND WILL
ALLOW A RANDOM NO. OF MILLISECONDS TO TRANSPIRE
BEFORE A TRANSMISSION OF A CHARACTER. THIS
ROUTINE IS ACTIVATED BASED ON USER RESPONSE.

C. DATCHK - THIS ROUTINE IS USED BY PROGRAM #4 AND WILL
CHECK FOR CORRECT EXPECTED AND RECEIVED DATA
AFTER CHARACTER TRANSMISSION AS WELL AS ANY
ERROR BIT CONDITIONS.

D. TIMERX + TIMETX -
THESE TWO(2) ROUTINES ARE USED BY PROGRAM #4 TO
VERIFY THE 'DONE' BIT AFTER BOTH TRANSMITTER AND
RECEIVER OPERATIONS.

12.14 SUERR1, SUER2

THESE TWO(2) ROUTINES ARE USED THROUGHOUT THE PROGRAM TO SET
UP THE ERROR INFORMATION FOR 'ERROR REPORTING' BEFORE THE
'ERROR REPORT' CALL IS MADE. 'ERROR REPORT' CALLS APPEAR
THROUGHOUT THE PROGRAM IN THE FORM 'ERROR + XX' WHERE 'XX'
INDICATES THE PARTICULAR ERROR TABLE (ERRTB:) ENTRY USED BY
THE ERROR SERVICE ROUTINE.

12.15 PRIME

841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896

THIS ROUTINE IS USED TO SET UP THE DATA BUFFERS AN THE DEVICE UNDER TEST FOR EACH 256(10) BYTE BLOCK TRANSFER.

12.16 CLDLBF

THIS ROUTINE IS USED IN CONJUNCTION WITH ROUTINE 'PRIME' TO CLEAR INPUT AND OUTPUT BUFFERS BEFORE DATA TRANSFERS.

12.17 LDOUT1, LDOUT2, LDOUT3, LDOUT4

THE ROUTINES ARE ALL USED FOR SET UP AND LOADING PURPOSES AS FOLLOWS:

- A. LDOUT1 - IS CALLED TO SET UP THE 'NULL-DEL-NUL' PATTERN
- B. LDOUT2 - IS CALLED TO LOAD AN ASCENDING BINARY COUNT PATTERN
- C. LDOUT3 - IS CALLED TO LOAD A DESCENDING BINARY COUNT PATTERN
- D. LDOUT4 - IS CALLED TO LOAD A COMPLEMENTING WORSE CASE PATTERN

12.18 CHKDAT

THIS ROUTINE IS USED TO CHECK FOR DATA COMPARE ERRORS IN 256(10) BYTE BLOCK TRANSFERS.

12.19 BUSERR, RSVERR

THESE TWO(2) ROUTINES ARE USED TO SERVICE 'UNEXPECTED' BUS ERROR AND RESERVED INSTRUCTION TRAPS, RESPECTIVELY.

12.20 TRPCOM

THIS ROUTINE IS USED TO SET UP AND REPORT THE INFORMATION CONCERNING 'UNEXPECTED' BUS ERROR AND RESERVED INSTRUCTION TRAPS. THIS ROUTINE IS USED IN CONJUNCTION WITH ROUTINES 'BUSERR' AND 'RSVERR' DESCRIBED ABOVE.

13.0 MISCELLANEOUS

- A. THE STACK POINTER IS INITIALLY SET TO 1100.

897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952

B. THE PARITY BIT IS NOT COVERED.

14.0 USER SELECTION PROGRAMS

14.1 PROGRAM #2 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

- A. SELECTION OF 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
- THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

- 1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE
- 2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND
- 3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G., A=101)?

THE USER SHOULD RESPOND BY TYPING AN OCTAL ASCII VALUE, FOR THE CHARACTER DESIRED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982

C. WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G., 10=8(10))?

THE USER SHOULD RESPOND BY TYPING AN OCTAL VALUE FOR THE DESIRED NO. OF MSEC. DELAY AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

E.G. - IF USER DESIRED 16 MSEC. DELAY BETWEEN EACH CHARACTER TRANSFER HE SHOULD TYPE '20'.

D. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING THE CHARACTER SPECIFIED, WITH THE DESIRED MSEC. DELAY BETWEEN EACH CHARACTER TRANSMISSION.

14.2 PROGRAM #3 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

- A. SELECTION OF TRANSMITTER DATA BUFFER
- B. SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER IN MAINTENANCE MODE.
- 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

983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE
2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND
3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G., A=101)?

THE USER SHOULD RESPOND BY TYPING AN OCTAL ASCII VALUE, FOR THE CHARACTER DESIRED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

C. WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G., 10=8(10))?

THE USER SHOULD RESPOND BY TYPING AN OCTAL VALUE FOR THE DESIRED NO. OF MSEC. DELAY AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

E.G. - IF USER DESIRED 16 MSEC. DELAY BETWEEN EACH CHARACTER TRANSFER HE SHOULD TYPE '20'.

D. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING THE CHARACTER SPECIFIED, WITH THE DESIRED MSEC. DELAY

BETWEEN EACH CHARACTER TRANSMISSION.

14.3 PROGRAM #4 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094

- A. SELECTION OF A TRANSMITTER DATA BUFFER
- B. SELECTION OF A SINGLE CHARACTER TO BE SENT, RECEIVED AND CHECKED WITH MAINTENANCE BIT SET.

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

- A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE
2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND
3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT.

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

- B. IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO?

THE USER SHOULD RESPOND AS ASKED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

- C. WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G. A=101)?

THE USER SHOULD RESPOND BY TYPING AN OCTAL ASCII VALUE, FOR THE CHARACTER DESIRED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

- D. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING THE CHARACTER SPECIFIED, WITH A RANDOM MSEC. DELAY BETWEEN EACH CHARACTER TRANSMISSION. BETWEEN EACH TRANSMISSION, 'RCVR' & 'XMITTER' DONE BITS WILL BE

VERIFIED, AS WELL AS CHECKS FOR CORRECT DATA AND ANY ERROR BIT CONDITIONS.

NOTE

1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150

IF USER RESPONSE TO ITEM B. (DIRECTLY ABOVE) WAS A '0' OR A PLAIN 'CARRIAGE RETURN' THEN THERE IS NO DELAY BETWEEN CHARACTER TRANSMISSIONS.

14.4 PROGRAM #5 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW USER PARAMETERS FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE.

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE
2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND
3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT.

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO?

THE USER SHOULD RESPOND AS ASKED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

C. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING BINARY CHARACTERS, WITH A RANDOM MSEC. DELAY BETWEEN EACH CHARACTER TRANSMISSION. BETWEEN EACH TRANSMISSION, 'RCVR' & 'XMITTER' DONE BITS WILL BE VERIFIED, AS WELL AS CHECKS FOR CORRECT DATA AND ANY ERROR BIT CONDITIONS.

NOTE

IF USER RESPONSE TO ITEM B. (DIRECTLY ABOVE) WAS A '0' OR A PLAIN 'CARRIAGE RETURN' THEN THERE IS NO DELAY BETWEEN

CHARACTER TRANSMISSIONS.

1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
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
1201
1202
1203
1204
1205
1206

15.0 PROGRAM FUNCTIONAL FLOW CHARTS

16.0 PROGRAM LISTING

17.0 ECO TABLE

- CHGC1 - .SETUP MACRO EXPANDED TO INCLUDE 'SOFTSWR'.
- CHGC2 - GETSWR MACRO ADDED AFTER SETUP.
- CHGC3 - JSR PC,STKINT FOLLOWS GETSWR.
- CHGC4 - .SREAD ARGUMENTS EXPANDED TO READ '\$SREAD ,X,8.,200'
- CHGC5 - MODIFIED PROGRAM START SO THAT ALL STARTS RUN THROUGH A COMMON SOFTSWR ROUTINE AND VECTOR INIT.
- CHGC6 - ADDED DELAY TO TEST 10.

- CHGD1 - ADDED CLEAR PROC STATUS WORD UPON ENTRY INTO TEST TO CLEAR UP INTERRUPT TEST FAILURES BEING CAUSED BY THE CPU PRIORITY BEING SET TOO HIGH.
- CHGD2 - ADDED .=42 WITH A ZERO VALUE TO PREVENT THAT LOCATION FROM INADVERTANTLY CAUSING THE PROGRAM TO RETURN TO SOME FALSE LOCATION RATHER THAN ITS INTENDED PURPOSE OF RETURNING PROGRAM CONTROL TO THE MONITOR.

.ENDR @

167400

.NLIST CND,MD,MC
.LIST ME,SEQ,BIN
\$SWR=167400
.ENABLE ABS

000001

.TITLE CZDLCD0 DL11-C,D,E OFLNE TST
:*COPYRIGHT (C) 1980
:*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-C3), JAN 19, 1977.
:*
\$TN=1

.SBTTL OPERATIONAL SWITCH SETTINGS
:*

```

1207      :*      SWITCH      USE
1208      :*      -----
1209      :*      15      HALT ON ERROR
1210      :*      14      LOOP ON TEST
1211      :*      13      INHIBIT ERROR TYPEOUTS
1212      :*      12      /C OR /D MODEL
1213      :*      11      INHIBIT ITERATIONS
1214      :*      10      BELL ON ERROR
1215      :*      9      LOOP ON ERROR
1216      :*      8      LOOP ON TEST IN SWR<7:0>
1217
1218      :*      0      CREATION OF DEVICE/S TABLE
1219      :*      OR CHANGE CHARACTER LENGTH
1220
1221      . =174
1222      000174 000000      DISPREG: .WORD 0      ;SOFTWARE DISPLAY REGISTER
1223      000176 000000      SWREG: .WORD      ;SOFTWARE SWITCH REGISTER
1224      000200 012767 001734 001242      MOV #PRG1,STAD      ;ADDRESS OF USER PROGRAM NO. 1
1225      000206 000417      BR STCONT
1226      000210 012767 006424 001232      MOV #PRG2,STAD      ;ADDRESS OF USER PROGRAM NO.2
1227      000216 000413      BR STCONT
1228      000220 012767 006632 001222      MOV #PRG3,STAD      ;ADDRESS OF USER PROGRAM NO. 3
1229      000226 000407      BR STCONT
1230      000230 012767 007050 001212      MOV #PRG4,STAD      ;ADDRESS OF USER PROGRAM NO. 4
1231      000236 000403      BR STCONT
1232      000240 012767 007410 001202      MOV #PRG5,STAD      ;ADDRESS OF USER PROGRAM NO. 5
1233      000246
1234      000246 005067 177524      CHGD1: STCONT: CLR PS      ;CLEAR PSW
1235      000252 000137 001452      JMP @MBEGIN      ;JUMP TO COMMON START
1236
1237      000256 000042      CHGD2: . =42      ;STORAGE LOC FOR MONITOR ADDRESS
1238      000042 000000      .WORD 0      ;CLEAR LOC FOR START UP
1239      000052 000052      . =52      ;INFORMATION LOCATION FOR ACT11
1240      000052 000000      .WORD 0      ;NO POWER FAIL REQUIRED <BIT15=0>
1241      ;IS NOT MEMORY SIZE DEPENDENT <BIT14=0>
1242      ;IS SUITABLE FOR AUTOMATIC OPERATION <BIT13=0>
1243
1244      .SBTTL BASIC DEFINITIONS
1245
1246      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1247      001100      STACK= 1100
1248      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
1249      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
1250
1251      ;*MISCELLANEOUS DEFINITIONS
1252      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
1253      000012      LF= 12      ;;CODE FOR LINE FEED
1254      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
1255      000200      CRLF= 200      ;;CODE FOR CARRIAGE RETURN--LINE FEED
1256      177776      PS= 177776      ;;PROCESSOR STATUS WORD
1257      .EQUIV PS,PSW
1258      177774      STKLMT= 177774      ;;STACK LIMIT REGISTER
1259      177772      PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
1260      177570      DSWR= 177570      ;;HARDWARE SWITCH REGISTER
1261      177570      DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
1262

```

```
1263          000000      : *GENERAL PURPOSE REGISTER DEFINITIONS
1264          000001      R0=      X0          :: GENERAL REGISTER
1265          000002      R1=      X1          :: GENERAL REGISTER
1266          000003      R2=      X2          :: GENERAL REGISTER
1267          000004      R3=      X3          :: GENERAL REGISTER
1268          000005      R4=      X4          :: GENERAL REGISTER
1269          000006      R5=      X5          :: GENERAL REGISTER
1270          000007      R6=      X6          :: GENERAL REGISTER
1271          000008      R7=      X7          :: GENERAL REGISTER
1272          000009      SP=      X6          :: STACK POINTER
1273          000010      PC=      X7          :: PROGRAM COUNTER
1274
1275          000000      : *PRIORITY LEVEL DEFINITIONS
1276          000040      PR0=      0          :: PRIORITY LEVEL 0
1277          000100      PR1=      40         :: PRIORITY LEVEL 1
1278          000200      PR2=      100        :: PRIORITY LEVEL 2
1279          000400      PR3=      140        :: PRIORITY LEVEL 3
1280          000800      PR4=      200        :: PRIORITY LEVEL 4
1281          001600      PR5=      240        :: PRIORITY LEVEL 5
1282          003200      PR6=      300        :: PRIORITY LEVEL 6
1283          006400      PR7=      340        :: PRIORITY LEVEL 7
1284
1285          100000      : *'SWITCH REGISTER' SWITCH DEFINITIONS
1286          040000      SW15=     100000
1287          020000      SW14=     40000
1288          010000      SW13=     20000
1289          004000      SW12=     10000
1290          002000      SW11=     4000
1291          001000      SW10=     2000
1292          000400      SW09=     1000
1293          000200      SW08=     400
1294          000100      SW07=     200
1295          000040      SW06=     100
1296          000020      SW05=     40
1297          000010      SW04=     20
1298          000004      SW03=     10
1299          000002      SW02=     4
1300          000001      SW01=     2
1301          000000      SW00=     1
1302          .EQUIV      SW09,SW9
1303          .EQUIV      SW08,SW8
1304          .EQUIV      SW07,SW7
1305          .EQUIV      SW06,SW6
1306          .EQUIV      SW05,SW5
1307          .EQUIV      SW04,SW4
1308          .EQUIV      SW03,SW3
1309          .EQUIV      SW02,SW2
1310          .EQUIV      SW01,SW1
1311          .EQUIV      SW00,SW0
1312
1313          100000      : *DATA BIT DEFINITIONS (BIT00 TO BIT15)
1314          040000      BIT15=    100000
1315          020000      BIT14=    40000
1316          010000      BIT13=    20000
1317          004000      BIT12=    10000
1318          002000      BIT11=    4000
```


1319	002000	BIT10=	2000
1320	001000	BIT09=	1000
1321	000400	BIT08=	400
1322	000200	BIT07=	200
1323	000100	BIT06=	100
1324	000040	BIT05=	40
1325	000020	BIT04=	20
1326	000010	BIT03=	10
1327	000004	BIT02=	4
1328	000002	BIT01=	2
1329	000001	BIT00=	1
1330		.EQUIV	BIT09,BIT9
1331		.EQUIV	BIT08,BIT8
1332		.EQUIV	BIT07,BIT7
1333		.EQUIV	BIT06,BIT6
1334		.EQUIV	BIT05,BIT5
1335		.EQUIV	BIT04,BIT4
1336		.EQUIV	BIT03,BIT3
1337		.EQUIV	BIT02,BIT2
1338		.EQUIV	BIT01,BIT1
1339		.EQUIV	BIT00,BIT0

1340		.*BASIC 'CPU' TRAP VECTOR ADDRESSES		
1341		ERRVEC=	4	::TIME OUT AND OTHER ERRORS
1342	000004	RESVEC=	10	::RESERVED AND ILLEGAL INSTRUCTIONS
1343	000010	TBITVEC=	14	::'T' BIT
1344	000014	TRTVEC=	14	::TRACE TRAP
1345	000014	BPTVEC=	14	::BREAKPOINT TRAP (BPT)
1346	000014	IOTVEC=	20	::INPUT/OUTPUT TRAP (IOT) **SCOPE**
1347	000020	PWRVEC=	24	::POWER FAIL
1348	000024	EMTVEC=	30	::EMULATOR TRAP (EMT) **ERROR**
1349	000030	TRAPVEC=	34	::'TRAP' TRAP
1350	000034	TKVEC=	60	::TTY KEYBOARD VECTOR
1351	000060	TPVEC=	64	::TTY PRINTER VECTOR
1352	000064	PIRQVEC=	240	::PROGRAM INTERRUPT REQUEST VECTOR
1353	000240			
1354				

```
1355 .SBTTL COMMON TAGS
1356
1357 ;*****
1358 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
1359 ;*USED IN THE PROGRAM.
1360
1361         001100      .S=1100
1362 001100      $CMTAG:  .WORD 0          ;; START OF COMMON TAGS
1363 001100 000000      $PASS:  .WORD 0        ;; CONTAINS PASS COUNT
1364 001102      000    $TSTNM: .BYTE 0        ;; CONTAINS THE TEST NUMBER
1365 001103      000    $ERFLG: .BYTE 0        ;; CONTAINS ERROR FLAG
1366 001104 000000      $ICNT:  .WORD 0        ;; CONTAINS SUBTEST ITERATION COUNT
1367 001106 000000      $LPADR: .WORD 0        ;; CONTAINS SCOPE LOOP ADDRESS
1368 001110 000000      $LPERR: .WORD 0        ;; CONTAINS SCOPE RETURN FOR ERRORS
1369 001112 000000      $ERTTL: .WORD 0        ;; CONTAINS TOTAL ERRORS DETECTED
1370 001114      000    $ITEMB: .BYTE 0        ;; CONTAINS ITEM CONTROL BYTE
1371 001115      001    $ERMAX: .BYTE 1        ;; CONTAINS MAX. ERRORS PER TEST
1372 001116 000000      $ERRPC: .WORD 0        ;; CONTAINS PC OF LAST ERROR INSTRUCTION
1373 001120 000000      $GDADR: .WORD 0        ;; CONTAINS ADDRESS OF 'GOOD' DATA
1374 001122 000000      $BDADR: .WORD 0        ;; CONTAINS ADDRESS OF 'BAD' DATA
1375 001124 000000      $GDDAT: .WORD 0        ;; CONTAINS 'GOOD' DATA
1376 001126 000000      $BDDAT: .WORD 0        ;; CONTAINS 'BAD' DATA
1377 001130 000000      .WORD 0          ;; RESERVED—NOT TO BE USED
1378 001132 000000      .WORD 0
1379 001134      000    $AUTOB: .BYTE 0        ;; AUTOMATIC MODE INDICATOR
1380 001135      000    $INTAG: .BYTE 0        ;; INTERRUPT MODE INDICATOR
1381 001136 000000      .WORD 0
1382 001140 177570      $SWR:  .WORD DSWR    ;; ADDRESS OF SWITCH REGISTER
1383 001142 177570      DISPLAY: .WORD DDISP  ;; ADDRESS OF DISPLAY REGISTER
1384 001144 177560      $TKS:  177560    ;; TTY KBD STATUS
1385 001146 177562      $TKB:  177562    ;; TTY KBD BUFFER
1386 001150 177564      $TPS:  177564    ;; TTY PRINTER STATUS REG. ADDRESS
1387 001152 177566      $TPB:  177566    ;; TTY PRINTER BUFFER REG. ADDRESS
1388 001154      000    $NULL:  .BYTE 0        ;; CONTAINS NULL CHARACTER FOR FILLS
1389 001155      002    $FILLS: .BYTE 2        ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
1390 001156      012    $FILLC: .BYTE 12       ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
1391 001157      000    $TPFLG: .BYTE 0        ;; 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
1392 001160 000000      $REGAD: .WORD 0        ;; CONTAINS THE ADDRESS FROM
1393                                ;; WHICH ($REG0) WAS OBTAINED
1394 001162 000000      $REG0:  .WORD 0        ;; CONTAINS (($REGAD)+0)
1395 001164 000000      $REG1:  .WORD 0        ;; CONTAINS (($REGAD)+2)
1396 001166 000000      $REG2:  .WORD 0        ;; CONTAINS (($REGAD)+4)
1397 001170 000000      $REG3:  .WORD 0        ;; CONTAINS (($REGAD)+6)
1398 001172 000000      $REG4:  .WORD 0        ;; CONTAINS (($REGAD)+10)
1399 001174 000000      $REG5:  .WORD 0        ;; CONTAINS (($REGAD)+12)
1400 001176 000000      $REG6:  .WORD 0        ;; CONTAINS (($REGAD)+14)
1401 001200 000000      $REG7:  .WORD 0        ;; CONTAINS (($REGAD)+16)
1402 001202 000000      $TMP0:  .WORD 0        ;; USER DEFINED
1403 001204 000000      $TMP1:  .WORD 0        ;; USER DEFINED
1404 001206 000000      $TMP2:  .WORD 0        ;; USER DEFINED
1405 001210 000000      $TMP3:  .WORD 0        ;; USER DEFINED
1406 001212 000000      $TMP4:  .WORD 0        ;; USER DEFINED
1407 001214 000000      $TMP5:  .WORD 0        ;; USER DEFINED
1408 001216 000000      $TMP6:  .WORD 0        ;; USER DEFINED
1409 001220 000000      $TMP7:  .WORD 0        ;; USER DEFINED
1410 001222 000000      $TMP10: .WORD 0        ;; USER DEFINED
```

1411	001224	000000	\$TMP11: .WORD	0	::USER DEFINED
1412	001226	000000	\$TMP12: .WORD	0	::USER DEFINED
1413	001230	000000	\$TMP13: .WORD	0	::USER DEFINED
1414	001232	000000	\$TMP14: .WORD	0	::USER DEFINED
1415	001234	000000	\$TMP15: .WORD	0	::USER DEFINED
1416	001236	000000	\$TMP16: .WORD	0	::USER DEFINED
1417	001240	000000	\$TMP17: .WORD	0	::USER DEFINED
1418	001242	000000	\$TIMES	0	::MAX. NUMBER OF ITERATIONS
1419	001244	000000	\$ESCAPE:	0	::ESCAPE ON ERROR ADDRESS
1420	001246	177607	\$BELL: .ASCIZ	<207><377><377>	::CODE FOR BELL
1421	001252	077	\$QUES: .ASCII	/?/	::QUESTION MARK
1422	001253	015	\$CRLF: .ASCII	<15>	::CARRIAGE RETURN
1423	001254	000012	\$LF: .ASCIZ	<12>	::LINE FEED
1424			:*****		
1425			:THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO		
1426			:'MORETAGS' AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .SCMTAG		
1427					
1428					
1429	001256	000000	TABFLG: .WORD	0	:AN INDICATOR TO SHOW THAT THE
1430					:INFORMATION FOR MULTIPLE DEVICE
1431					:TESTING HAS ALREADY TRANSPIRED
1432					:& 'MAINDEC' NAME HAS BEEN PRINTED
1433	001260	000000	DLBASE: .WORD	0	:STORAGE & WORKING LOCATION FOR A DEVICE
1434					:RECEIVER STATUS REGISTER ADDRESS
1435	001262	000000	KEEPAD: .WORD	0	:STORAGE LOCATION FOR THE 1ST
1436					:DEVICE RCSR FROM WHICH
1437					:'BASEADD' IS RESTORED AT THE
1438					:END OF A COMPLETE PROGRAM PASS.
1439	001264	000000	BASEADD: .WORD	0	:STORAGE LOCATION WHICH HOLDS
1440					:THE RCSR ADDRESS OF THE 'NEXT'
1441					:DEVICE DURING MULTIPLE TESTING
1442	001266	000000	KEEPIV: .WORD	0	:STORAGE LOCATION FOR THE 1ST
1443					:DEVICE RECEIVER VECTOR FROM
1444					:WHICH 'BASEIV' IS RESTORED AT THE
1445					:END OF A COMPLETE PROGRAM PASS
1446	001270	000000	BASEIV: .WORD	0	:STORAGE LOCATION WHICH HOLDS
1447					:THE VECTOR ADDRESS OF THE 'NEXT'
1448					:DEVICE DURING MULTIPLE TESTING
1449	001272	000000	MULTD: .WORD	0	:FLAG TO INDICATE TO 'END OF PASS'
1450					:ROUTINE THAT MULTIPLE DEVICE
1451					:TESTING IS BEING CONDUCTED
1452					:0=NO, 1=YES
1453	001274	000000	ACTREG: .WORD	0	:THIS IS THE DEVICE ACTIVE REGISTER
1454					:A BIT IS SET (STARTING AT
1455					:BIT0)FOR EACH CONTIGUOUS DEVICE
1456					:(A MAX. OF 16) THAT IS TO UNDERGO
1457					:TESTING. THIS LOCATION IS
1458					:AUTOMATICALLY FILLED BASED ON
1459					:USER RESPONSE TO PROGRAM QUESTIONS
1460	001276	000000	ROTADD: .WORD	0	:A ROTATING POINTER TO SIGNAL
1461					:THE LAST DEVICE TESTED (IF
1462					:MULTIPLE DEVICE TESTING WAS BEING
1463					:DONE) IF LESS THAN A FULL COMPLE-
1464					:MENT OF DEVICES (16)WAS SELECTED
1465	001300	000000	LASTADD: .WORD	0	:STORAGE LOCATION FOR THE
1466					:RCSR ADDRESS OF THE LAST DEVICE

1467
1468
1469 001302 000000
1470
1471 001304 000000
1472
1473
1474
1475
1476
1477 001306 177740
1478
1479
1480
1481
1482
1483
1484

DLPRI: .WORD 0

LESS1: .WORD 0

STLMSK: 177740

;END OF USER SUPPLIED TAG(S)

;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'.

```
1485 .SBTTL ERROR POINTER TABLE
1486
1487 :*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1488 :*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1489 :*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1490 :*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1491 :*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1492
1493 :* EM ::POINTS TO THE ERROR MESSAGE
1494 :* DH ::POINTS TO THE DATA HEADER
1495 :* DT ::POINTS TO THE DATA
1496 :* DF ::POINTS TO THE DATA FORMAT
1497
1498
1499 001310 $ERRTB:
1500
1501 :ERROR TABLE ITEM FOR ERROR MESSAGE 1
1502
1503 001310 015714 EM1 :'DL11 REGISTER REFERENCE CAUSED TIMEOUT''
1504 001312 015763 DH1 :'' (PC) (PS) (SP) TEST DEVADR REGADR ''
1505 001314 016042 DT1 : (R7) (PSW) (R6) (R0) (R1) (R2)
1506 001316 000000 0 :PRINT ALL OCTAL
1507
1508 :ERROR TABLE ITEM FOR ERROR MESSAGE 2
1509
1510 001320 016060 EM2 :'' DL11 REGISTER ERROR ''
1511 001322 016104 DH2 :'' (PC) (PS) (SP) TEST DEVADR REGADR WAS S/B ''
1512 001324 016202 DT2 :'' (R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4) ''
1513 001326 000000 0 :PRINT ALL OCTAL
1514
1515 :ERROR TABLE ITEM FOR ERROR MESSAGE 3
1516
1517 001330 016224 EM3 :'' DL11 DATA COMPARE ERROR ''
1518 001332 016254 DH3 :'' (PC) (PS) (SP) TEST WASADR SHBADR WAS S/B ''
1519 001334 016352 DT3 :'' (R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4) ''
1520 001336 000000 0 :PRINT ALL OCTAL
1521
1522 :ERROR TABLE ITEM FOR ERROR MESSAGE 4
1523
1524 001340 016374 EM4 :'' UNEXPECTED TRAP TO VECTOR AT LOCATION XXX ''
1525 001342 016446 DH4 :'' (PC) (PS) (SP) TEST ''
1526 001344 016504 DT4 :'' (R7) (PSW) (R6) (R0) ''
1527 001346 000000 0 :PRINT ALL OCTAL
1528
1529 :ERROR TABLE ITEM FOR ERROR MESSAGE 5
1530
1531 001350 016516 EM5 :'' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) ''
1532 001352 016573 DH5 :'' (PC) (PS) (SP) TEST DEVADR REGADR (REG) ''
1533 001354 016662 DT5 :'' (R7) (PSW) (R6) (R0) (R1) (R2) (R3) ''
1534 001356 000000 0
1535
1536 :ERROR TABLE ITEM FOR ERROR MESSAGE 6
1537
1538 001360 015714 EM1 :'DL11 REGISTER REFERENCE CAUSED TIMEOUT''
1539 001362 016702 DH6 :'' (PC) (PS) (SP) REGADR''
1540 001364 016742 DT6 :$ERRPC,$TMP0,$REG6,$REG2
```

```

1541 001366 000000          0          ;PRINT ALL OCTAL
1542
1543          ;ERROR TABLE ITEM FOR ERROR MESSAGE 7
1544
1545 001370 016516          EM5          ;' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) ''
1546 001372 016754          DH7          ;' (PC) DEVADR REGADR (REG)''
1547 001374 017014          DT7          ;$ERRPC,$REG1,$REG2,$REG3
1548 001376 000000          0          ;PRINT ALL OCTAL
1549
1550          ;ERROR TABLE ITEM FOR ERROR MESSAGE 10
1551
1552 001400 016224          EM3          ;' DL11 DATA COMPARE ERROR ''
1553 001402 017026          DH10         ;' (PC) DEVADR REGADR (REG) S/B''
1554 001404 017074          DT10         ;$ERRPC,$REG1,$REG2,$REG3,$REG4
1555 001406 000000          0          ;PRINT ALL OCTAL
1556
1557          ;*****
1558          ;DL11 DEFINITIONS
1559          ;*****
1560
1561 001410 175610          DLRCR: 175610      ;CONTAINS ADDRESS OF RCVR CSR
1562 001412 175612          DLRDBR: 175612      ;CONTAINS ADDRESS OF RCVR DBR
1563 001414 175614          DLXCSR: 175614      ;CONTAINS ADDRESS OF XMIT CSR
1564 001416 175616          DLXDBR: 175616      ;CONTAINS ADDRESS OF XMIT DBR
1565 001420 000300          DLVECT: 300        ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
1566 001422 000000          XFLG0: 0          ;FLAG FOR HARD XMIT ERRORS
1567 001424 000000          RFLG0: 0          ;FLAG FOR HARD RCVR ERRORS
1568 001426 000000          RFLG1: 0          ;FLAG FOR SOFT RCVR ERRORS
1569 001430 000000          RTRY: 0          ;COUNTS NO. OF RETRIES ON SOFT ERRORS
1570 001432 000000          OPTR: 0          ;CONTAINS POINTER TO OUTPUT BUFFER
1571 001434 000000          IPTR: 0          ;CONTAINS POINTER TO INPUT BUFFER
1572 001436 000000          LDOUT: 0         ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
1573 001440 000000          TIMR1: 0         ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
1574 001442 000000          TIMR2: 0
1575 001444 000000          TIMR3: 0         ;DELAY TIMER FOR TEST 10
1576 001446 000000          INTFLG: 0        ;SOFTWARE INTR. FLAG
1577 001450 000000          STAD: 0          ;TEMPORARY ADDRESS STORAGE LOC.
1578
1579
1580
1581
1582 001452 000240          BEGIN: NOP        ;PROGRAM WILL START HERE
1583          .SBTTL INITIALIZE THE COMMON TAGS
1584          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1585 001454 012706 001100          MOV #SCMTAG,R6    ;:FIRST LOCATION TO BE CLEARED
1586 001460 005026          CLR (R6)+         ;:CLEAR MEMORY LOCATION
1587 001462 022706 001140          CMP #SWR,R6      ;:DONE?
1588 001466 001374          BNE -6           ;:LOOP BACK IF NO
1589 001470 012706 001130          MOV #STACK,SP    ;:SETUP THE STACK POINTER
1590          ;;INITIALIZE A FEW VECTORS
1591 001474 012737 010406 000020          MOV #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
1592 001502 012737 000340 000022          MOV #340,@IOTVEC+2 ;:LEVEL 7
1593 001510 012737 010660 000030          MOV #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
1594 001516 012737 000340 000032          MOV #340,@EMTVEC+2 ;:LEVEL 7
1595 001524 012737 013630 000034          MOV #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
1596 001532 012737 000340 000036          MOV #340,@TRAPVEC+2;:LEVEL 7

```

```

1597 001540 012737 013714 000024      MOV      #SPWRDN,@PWVREC      ;;POWER FAILURE VECTOR
1598 001546 012737 000340 000026      MOV      #340,@PWVREC+2      ;;LEVEL 7
1599 001554 005067 177462                CLR      $TIMES                ;;INITIALIZE NUMBER OF ITERATIONS
1600 001560 005067 177460                CLR      $ESCAPE                ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1601 001564 112767 000001 177323      MOV      #1,$ERMAX                ;;ALLOW ONE ERROR PER TEST
1602 001572 012767 000152 177306      MOV      #.,$LPADR                ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1603 001600 012767 001600 177302      MOV      #.,$LPERR                ;;SETUP THE ERROR LOOP ADDRESS
1604                                     ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
1605                                     ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1606 001606 013746 000004                MOV      @ERRVEC-(SP)          ;;SAVE ERROR VECTOR
1607 001612 012737 001646 000004      MOV      #64,$@ERRVEC          ;;SET UP ERROR VECTOR
1608 001620 012767 177570 177312      MOV      #DSWR,$SWR                ;;SETUP FOR A HARDWARE SWICH REGISTER
1609 001626 012767 177570 177306      MOV      #DDISP,$DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
1610 001634 022777 177777 177276      CMP      #-1,$SWR                ;;TRY TO REFERENCE HARDWARE SWR
1611 001642 001012                                BNE     66$                      ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1612                                     ;;AND THE HARDWARE SWR IS NOT = -1
1613 001644 000403                                BR      65$                      ;;BRANCH IF NO TIMEOUT
1614 001646 012716 001654 64$:          MOV      #65,$(SP)                ;;SET UP FOR TRAP RETURN
1615 001652 000002                                RTI
1616 001654 012767 000176 177256 65$:          MOV      #SWREG,$SWR                ;;POINT TO SOFTWARE SWR
1617 001662 012767 000174 177252      MOV      #DISPREG,$DISPLAY        ;;RESTORE ERROR VECTOR
1618 001670 012637 000004 66$:          MOV      (SP)+,@ERRVEC
1619
1620 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1621 001674 005737 000042                TST      @#42                    ;;ARE WE RUNNING UNDER XXDP/ACT?
1622 001700 001006                                BNE     67$                      ;;BRANCH IF YES
1623 001702 026727 177232 000176      CMP      $SWR,#SWREG                ;;SOFTWARE SWITCH REG SELECTED?
1624 001710 001005                                BNE     68$                      ;;BRANCH IF NO
1625 001712 104406                                GTSWR                                ;;GET SOFT-SWR SETTINGS
1626 001714 000403                                BR      68$
1627 001716 112767 000001 177210 67$:          MOV      #1,$AUTOB                ;;SET AUTO-MODE INDICATOR
1628 001724 68$:
1629 001724 004767 007714                JSR      PC,$TKINT                ;;SET TTY INTERRUPT
1630 001730 000177 177514                JMP      @STAD                    ;;JUMP TO SELECTED TEST
1631 001734 005067 177332      PRG1:  CLR      $MULTD                ;;CLEAR MULTIPLE DEVICE
1632                                     ;;TESTING FLAG
1633 001740 005067 177312                CLR      $TABFLG                ;;CLEAR TABLE CREATION FLAG
1634 001744 012767 000010 177262      MOV      #8.,$TMP15                ;;SET CHARACTER LENGTH DESIGNATOR
1635                                     ;;FOR 8 BITS --- THIS IS THE DEFAULT
1636                                     ;;LENGTH ASSUMED BY THE PROGRAM
1637                                     ;;UNLESS THE USER CHANGES IT THRU
1638                                     ;;THE QUESTION AND ANSWER CYCLE
1639                                     ;;INITIATED BY SETTING SW<0> TO A 1
1640 001752 012767 000200 177322      MOV      #200,$DLPRI                ;;SET STANDARD PRIORITY LEVEL
1641                                     ;;FOR DEVICE
1642 001760 032777 000400 177152      BIT      #SWB,$SWR                ;;IS THE 'LOOP ON TEST' SWITCH SET?
1643 001766 001411                                BEQ     1$                        ;;BRANCH IF NOT
1644
1645                                     ;;IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
1646                                     ;;INSTRUCTION THUS BYPASSING TABLE CREATION
1647
1648                                     ;;IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
1649                                     ;;THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
1650                                     ;;WITH THE DESIRED DEVICE REGISTER VALUES:
1651
1652

```

*****.*****

```

1653          :          UNDER ;DL11 DEFINITIONS          ABOVE
1654          :          *****
1655          :
1656          :          DLRCR:    PATCH THE ADDRESS OF THE RCVR CSR
1657          :          DLRDBR:   PATCH THE ADDRESS OF THE RCVR DBR
1658          :          DLXCSR:   PATCH THE ADDRESS OF THE XMIT CSR
1659          :          DLXDBR:   PATCH THE ADDRESS OF THE XMIT DBR
1660          :          DLVECT:   PATCH THE VECTOR ADDRESS OF THIS DL11
1661          :
1662 001770 104401 017417          :          TYPE, STMES          ;PRINT OUT 'MAINDEC' NAME
1663 001774 104401 021453          :          TYPE, FAILSA        ;TYPE FAILSAFE MESSAGE
1664 002000 104000          :          ERROR +0           ;TYPE OUT THE PC VALUE
1665 002002 104401 022031          :          TYPE, PCMSG         ;FOLLOWED BY =PC
1666 002006 000000          :          HALT                ;WAIT FOR USER TO RESPOND
1667 002010 000443          :          BR ONCE             ;GO TO TEST DEVICE PATCHED IN BY USER
1668 002012
1669          :
1670          :          1$:
1671          :          :ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
1672          :          :AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
1673          :          :WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
1674          :          :TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
1675          :          :HE CAN RUN THE DEFAULT DEVICE (1ST RECEIVER
1676          :          :STATUS REGISTER ADDRESS 175610)
1677          :          :NOTE: IF THIS IS NOT SUITABLE THE USER WILL
1678          :          :HAVE TO SET SWO=1 (OR UP) IN ORDER TO
1679          :          :RECREATE THE TABLE HE DESTROYED FROM
1680          :          :ABOVE
1681          :          MOV #175610,DLBASE ;1ST POSSIBLE RECEIVER CSR
1682          :          JSR PC,DLADDR      ;FORM DL ADDRESSES FOR
1683          :          :1ST POSSIBLE DEVICE
1684          :          MOV #300,DLVECT    ;1ST POSSIBLE INTERRUPT VECTOR
1685          :          CLR TABFLG        ;CLEAR TABLE CREATION FLAG
1686          :
1687          :          RESTR: MOV #STACK,SP ;SET UP STACK POINTER
1688          :          MOV #USERR,@PERRVEC ;SET UP BUS ERROR VECTOR
1689          :          MOV #340,@PERRVEC+2
1690          :          MOV #RSVERR,@PRESVEC ;SET UP RSVD INSTR. VECTOR
1691          :          MOV #340,@PRESVEC+2
1692          :
1693          :          :THIS NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
1694          :          :WILL TAKE PLACE I.E.-
1695          :          :A) HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
1696          :          :B) IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
1697          :          :
1698          :          TSTB TABFLG        ;HAS TABLE CREATION BEEN PERFORMED?
1699          :          BNE ONCE           ;BRANCH IF YES TO SKIP 'MAINDEC
1700          :          :TITLE' MESSAGE
1701          :          TYPE ,STMES        ;OTHERWISE, PRINT OUT 'MAINDEC'
1702          :          :NAME
1703          :          COMB TABFLG        ;IF TABLE CREATION HAS NOT BEEN
1704          :          :PERFORMED, THEN SET FLAG, AND DO SO
1705          :          BIT #SWO,@SWR      ;THE PROGRAM HAS OBVIOUSLY BEEN
1706          :          :RESTARTED - DOES USER WISH TO
1707          :          :RESELECT VECTOR AND CONTROL REGISTER
1708          :          :ADDRESSES I.E. - CREATE A NEW TABLE?
1709          :          :BRANCH IF YES
1710          :          ONCE: BNE GO
1711          :          CLR @DLXCSR        ;CLEAR OUT BOTH CSR'S
1712          :          CLR @DLRCR

```



```
1709 002130 005777 177256      TST      @DLRDBR      ;FLUSH RCVR 'DONE' BIT
1710 002134 005777 177252      TST      @DLRDBR
1711 002140 000167 000670      JMP      TST1        ;OTHERWISE, GO WITH EXISTING
1712                                ;TABLE OR NOT USE ANY TABLE AT
1713                                ;ALL WHICHEVER THE CASE MAY BE
1714                                ;(DEFAULT CASE IS 1ST POSSIBLE
1715                                ;DEVICE)
1716                                ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES:
1717                                ; A) TO RUN MULTIPLE DEVICES
1718                                ; B) TO CREATE A NEW TABLE TO RUN FROM, OR
1719                                ; C) TO CHANGE THE CHARACTER LENGTH
1720 002144 104401 020013      GO:      TYPE,      LENGTH      ;ASK USER FOR THE CHARACTER LENGTH
1721                                ;FOR WHICH HIS DEVICE IS SET
1722 002150 104413                                ;ACCEPT THE ANSWER TYPED BY USER
1723                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1724 002152 012600      MOV      (SP)+,R0      ;GET THE ANSWER TYPED
1725 002154 020027 000010      CMP      R0,#8.        ;IS THE NUMBER TOO HIGH?
1726 002160 101114      BHI      RETRY        ;IF YES - GO TO RETRY SITUATION
1727 002162 020027 000005      CMP      R0,#5.        ;IS THE NUMBER TOO LOW?
1728 002166 103511      BLO      RETRY        ;IF YES - GO TO RETRY SITUATION
1729 002170 010067 177040      MOV      R0,$TMP15     ;THE VALUE TYPED IS OK
1730                                ;STORE FOR FUTURE USE
1731 002174 104401 020075      TYPE,      DEFAULT     ;ASK USER IF HE WISHES TO TEST OTHER
1732                                ;THAN THE DEFAULT DEVICE
1733 002200 104412      RDOCT                                ;ACCEPT THE ANSWER TYPED BY USER
1734 002202 005726      TST      (SP)+        ;LOOK AT THE ANSWER
1735 002204 001002      BNE      1$           ;BRANCH IF REPLY WAS YES
1736 002206 000137 002770      JMP      @FLUSH       ;OTHERWISE, SKIP REST OF INTERROGATION
1737 002212 012700 000300      1$:      MOV      #300,R0      ;START RESTORATION OF TRAPCATCHER
1738 002216 012701 000302      MOV      #302,R1      ;AREA FROM LOCATIONS 300 TO 776
1739 002222 012702 000004      MOV      #4,R2        ;SO THAT WE CREATE THE MULTIPLE
1740 002226 010110      2$:      MOV      R1,(R0)      ;DEVICES TABLE WITH A CLEAN SLATE
1741 002230 005011      CLR      (R1)
1742 002232 060200      ADD      R2,R0
1743 002234 060201      ADD      R2,R1
1744 002236 022701 001000      CMP      #1000,R1
1745 002242 002771      BLT      2$
1746                                ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
1747                                ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
1748                                ;ADDRESS
1749 002244 104401 020202      FIRSTD: TYPE ,MFIRSTD      ;ASK USER FOR THE RECEIVER CONTROL
1750                                ;REGISTER ADDRESS OF HIS FIRST
1751                                ;DEVICE
1752 002250 104412      RDOCT                                ;ACCEPT THE ANSWER TYPED BY USER
1753                                ;AND STORE ON TOP OF STACK
1754                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1755 002252 012600      MOV      (SP)+,R0      ;GET THE ANSWER TYPED
1756 002254 020027 176170      CMP      R0,#176170    ;IS THE NUMBER TOO HIGH?
1757 002260 101060      BHI      RETRYO       ;IF YES-GO TO RETRY SITUATION
1758 002262 020027 175610      CMP      R0,#175610    ;IS THE NUMBER TOO LOW?
1759 002266 103455      BLO      RETRYO       ;IF YES - GO TO RETRY SITUATION
1760 002270 132700 000001      BITB     #BIT0,R0      ;NUMBER IS IN RANGE BUT IS IT
1761                                ;ON AN EVEN BOUNDARY?
1762 002274 001052      BNE      RETRYO       ;IF NO - GO TO RETRY SITUATION
1763                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1764 002276 032700 000007      BIT      #7,R0        ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
```

```
1765 ;USER RESPONSE EQUAL TO A ZERO?
1766 002302 001047 ;BRANCH IF NOT
1767 002304 010067 176750 BNE R0,DLBASE
1768 MOV R0,DLBASE ;THE 1ST ADDRESS VALUE TYPED IS OK
1769 ;STORE FOR FUTURE USE
1770 ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
1771 002310 016767 176744 176744 MOV DLBASE,KEEPADD ;GET 1ST ADDRESS VALUE
1772 002316 004767 005456 JSR PC,DLADDR ;GO FORM DL ADDRESSES FOR
1773 ;1ST DEVICE SELECTED
1774 002322 016767 176734 176734 MOV KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
1775 002330 104401 020270 VECT: TYPE ,MVECT ;ASK USER FOR A VECTOR ADDRESS
1776 002334 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1777 ;AND STORE ON TOP OF STACK
1778 002336 012600 ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1779 002340 020027 000776 MOV (SP)+,R0 ;GET THE ANSWER TYPED
1780 002344 101032 CMP R0,#776 ;IS THE NUMBER TOO HIGH?
1781 002346 020027 000300 BHI RETRY1 ;IF YES - GO TO RETRY SITUATION
1782 002352 103427 CMP R0,#300 ;IS THE NUMBER TOO LOW?
1783 002354 132700 000001 BLO RETRY1 ;IF YES - GO TO RETRY SITUATION
1784 BITB #BIT0,R0 ;NUMBER IS IN RANGE BUT IS IT
1785 002360 001024 BNE RETRY1 ;ON AN EVEN BOUNDARY?
1786 ;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
1787 002362 032700 000007 BIT #7,R0 ;IF NO - GO TO RETRY SITUATION
1788 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1789 002366 001021 BNE RETRY1 ;USER RESPONSE EQUAL TO A ZERO?
1790 002370 010067 177024 MOV R0,DLVECT ;BRANCH IF NOT
1791 ;THE VECTOR VALUE TYPED IS OK
1792 002374 016767 177020 176664 MOV DLVECT,KEEPIV ;STORE FOR FUTURE USE
1793 002402 016767 177012 176660 MOV DLVECT,BASEIV ;GET THE FIRST VECTOR VALUE
1794 002410 000414 BR HOWMANY ;SAVE FIRST VECTOR VALUE
1795 ;GO TO SEE IF USER WANTS MORE
1796 002412 104401 001252 RETRY: TYPE, $QUES ;THAN 1 DEVICE
1797 ;TYPE '?' INDICATING USER TYPED
1798 002416 000167 177522 JMP GO ;SOMETHING WRONG FOR CHARACTER LENGTH
1799 002422 104401 001252 RETRY0: TYPE , $QUES ;GO BACK TO REISSUE QUESTION
1800 ;TYPE '?' INDICATING USER TYPE
1801 002426 000167 177612 JMP FIRSTD ;SOMETHING WRONG FOR 1ST ADDRESS
1802 002432 104401 001252 RETRY1: TYPE , $QUES ;GO BACK TO REISSUE QUESTION
1803 ;TYPE '?' INDICATING USER TYPED
1804 002436 000167 177666 JMP VECT ;SOMETHING WRONG FOR VECTOR
1805 002442 104401 020346 HOWMANY: TYPE ,MULDEV ;GO BACK TO REISSUE QUESTION
1806 ;ASK USER IF HE WISHES TO RUN
1807 002446 104412 RDOCT ;MULTIPLE DEVICES
1808 ;ACCEPT THE ANSWER TYPED BY USER
1809 002450 012600 MOV (SP)+,R0 ;AND STORE ON TOP OF STACK
1810 002452 005700 TST R0 ;GET THE ANSWER TYPED
1811 002454 001003 BNE 1$ ;WAS THE ANSWER YES?
1812 002456 005067 176610 CLR MULTD ;BRANCH IF IT WAS
1813 ;OTHERWISE, INITIALIZE FLAG TO
1814 002462 000402 BR 2$ ;INDICATE NON-MULTIPLE DEVICES
1815 002464 105167 176602 1$: COMB MULTD ;SKIP NEXT INSTRUCTION
1816 ;INITIALIZE FLAG TO INDICATE
1817 002470 2$: ;RUNNING OF MULTIPLE DEVICES
1818 002470 105767 176576 TSTB MULTD ;ARE THERE MULTIPLE DEVICES ON
1819 1819 ;THE SYSTEM?
1820 002474 100406 BMI LASTD ;IF SO, GO TO ASK NEXT QUESTION
```

```
1821 002476 005067 176572 CLR ACTREG ;CLEAR DEVICE ACTIVE FLAG TO
1822 ;INDICATE NO RUNNING OF MULTIPLE
1823 ;DEVICES
1824 002502 005067 176570 CLR ROTADD ;CLEAR DEVICE ADDRESS POINTER IN
1825 ;USE WHEN RUNNING MULTIPLE DEVICES
1826 002506 000167 000160 JMP CONQUES ;SKIP ASKING NEXT QUESTION
1827 002512 LASTD:
1828 ;WE WILL NOW BEGIN TO SET UP THE DEVICE ACTIVE REGISTER FOR RUNNING
1829 ;MULTIPLE DL11 DEVICES
1830 002512 104401 020433 TYPE ,MLASTD ;ASK USER FOR THE RECEIVER
1831 ;CONTROL REGISTER ADDRESS OF
1832 ;HIS LAST DEVICE
1833 002516 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY
1834 ;USER AND STORE ON TOP OF STACK
1835 ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1836 002520 012600 1$: MOV (SP)+,R0 ;GET THE ANSWER TYPED
1837 002522 020027 176170 CMP R0,#176170 ;IS THE NUMBER TOO HIGH?
1838 002526 101132 BHI RETRY2 ;IF YES - GO TO RETRY SITUATION
1839 002530 020027 175610 CMP R0,#175610 ;IS THE NUMBER TOO LOW?
1840 002534 103527 BLO RETRY2 ;IF YES - GO TO RETRY SITUATION
1841 002536 132700 000001 BITB #BIT0,R0 ;NUMBER IS IN RANGE BUT IS IT
1842 ;ON AN EVEN BOUNDARY?
1843 002542 001124 BNE RETRY2 ;IF NOT - GO TO RETRY SITUATION
1844 ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1845 002544 032700 000007 BIT #7,R0 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1846 ;USER RESPONSE EQUAL TO A ZERO?
1847 002550 001121 BNE RETRY2 ;BRANCH IF NOT
1848 002552 010067 176522 MOV R0,LASTADD ;THE LAST ADDRESS VALUE TYPED IS OK
1849 ;STORE FOR FUTURE USE
1850 ;NOW WE BEGIN TO ACTUALLY INITIALIZE THE DEVICE ACTIVE REGISTER
1851 ;FROM WHICH THE PROGRAM WILL CYCLE UNTIL ALL DEVICES HAVE BEEN TESTED
1852 002556 012767 000001 176512 MOV #1,ROTADD ;SET UP POINTER FOR 'ACTREG'
1853 002564 005067 176504 CLR ACTREG ;CLEAR DEVICE ACTIVE REGISTER
1854 002570 056767 176502 176476 2$: BIS ROTADD,ACTREG ;MAKE 1ST DEVICE ACTIVE
1855 002576 000241 CLC ;CLEAR CARRY BIT FOR POINTER
1856 ;ROTATION
1857 002600 006167 176472 ROL ROTADD ;ARE WE PAST 16 LINE RANGE?
1858 002604 103422 BCS 3$ ;BRANCH IF YES
1859 002606 062767 000010 176450 ADD #10,BASEADD ;STEP UP BASE ADDRESS
1860 002614 026767 176460 176442 CMP LASTADD,BASEADD ;IS THIS THE LAST DEVICE?
1861 002622 101362 BHI 2$ ;BRANCH IF NOT
1862 ;NOTE: IF THIS PATH IS TAKEN IT IS ASSUMED THAT AT LEAST 2 DEVICES
1863 ;EXIST AND THAT ALL ADDRESSING IS CONTIGUOUS
1864 002624 056767 176446 176442 BIS ROTADD,ACTREG ;INDICATE NEXT DEVICE ACTIVE
1865 002632 012767 000001 176436 MOV #1,ROTADD ;RESET POINTER FOR 'ACTREG' FOR
1866 ;LATER USE IN END OF PASS ROUTINE
1867 002640 016767 176416 176416 MOV KEEPADD,BASEADD ;RESET 1ST DEVICE RECEIVER
1868 ;CONTROLLER REGISTER ADDRESS FOR
1869 ;LATER USE IN END OF PASS ROUTINE
1870 002646 000167 000020 JMP CONQUES ;GO TO CONTINUE QUESTIONING OF USER
1871 002652 3$:
1872 ;IF WE TAKE THIS PATH IT APPEARS THAT THERE ARE NOT AT LEAST
1873 ;TWO DEVICES PRESENT - IN RESPONSE TO USER TYPING 'YES' TO MULTIPLE
1874 ;DEVICES QUESTION
1875 002652 016767 176404 176404 MOV KEEPADD,BASEADD ;RESET 1ST DEVICE RECEIVER
1876 ;CONTROLLER REGISTER ADDRESS
```

```

1877 002660 104401 020531          TYPE      ,MRANGE          ;INFORM USER TO CHECK AND RETYPE
1878                                RDOCT              ;THE LAST DEVICE RCSR ADDRESS
1879 002664 104412                                ;ACCEPT THE ANSWER TYPED BY USER
1880                                JMP      1$          ;AND STORE ON TOP OF STACK
1881 002666 000167 177626          CONQUES:
1882 002672                                ;IF WE HAVE REACHED THIS PORTION WE KNOW:
1883                                ;A) THE 'RXCSR' ADDRESS OF THE 1ST DEVICE
1884                                ;B) THE 'RXCSR' ADDRESS OF THE LAST DEVICE, SAND
1885                                ;C) THE INTERRUPT VECTOR OF THE 1ST DEVICE
1886                                ;NOW LET'S FIND THE PRIORITY LEVEL
1887                                TYPE      ,PLEVEL          ;ASK USER FOR PRIORITY LEVEL
1888 002672 104401 020615          RDOCT              ;ACCEPT ANSWER TYPED BY USER AND
1889 002676 104412                                ;STORE ON TOP OF STACK
1890                                MOV      (SP)+,R0          ;GET THE ANSWER TYPED
1891 002700 012600                                CMP      R0,#7          ;IS THE NUMBER TOO HIGH?
1892 002702 020027 000007          BHI      RETRY3        ;IF YES - GO TO RETRY SITUATION
1893 002706 101046                                CMP      R0,#4          ;IS THE NUMBER TOO LOW?
1894 002710 020027 000004          BLO      RETRY3        ;IF YES GO TO RETRY SITUATION
1895 002714 103443                                MOV      R0,DLPRI       ;THE PRIORITY TYPED IN IS OK
1896 002716 010067 176360                                ;STORE FOR FUTURE USE
1897
1898                                ;THIS SECTION WILL CALCULATE THE PRIORITY LEVEL FOR THE
1899                                ;PROCESSOR BASED ON THE USER RESPONSE FOR PRIORITY LEVEL OF THE
1900                                ;DEVICE
1901                                ASL      DLPRI          ;FORM BITS <7-5> OF PSW
1902 002722 006367 176354          ASL      DLPRI          ;
1903 002726 006367 176350          ASL      DLPRI          ;
1904 002732 006367 176344          ASL      DLPRI          ;
1905 002736 006367 176340          ASL      DLPRI          ;
1906 002742 006367 176334          ASL      DLPRI          ;
1907 002746 016767 176330 176330  MOV      DLPRI,LESS1    ;START TO FORM LEVEL TO ALLOW
1908                                ;INTERRUPTS
1909 002754 162767 000001 176322  SUB      #1,LESS1        ;DROP DEVICE LEVEL PRIORITY
1910                                ;BY 1 LEVEL FOR PSW
1911 002762 042767 000037 176314  BIC      #37,LESS1      ;MAKE SURE THE T,N,Z,V & C
1912                                ;BITS FOR THE PROCESSOR ARE CLEAR
1913 002770 005077 176420          FLUSH: CLR      @DLXCSR   ;CLEAR OUT BOTH CSR'S
1914 002774 005077 176410          CLR      @DLRCSR
1915 003000 005777 176406          TST     @DLRDBR
1916 003004 005777 176402          TST     @DLRDBR
1917 003010 000167 000020          JMP      TST1
1918 003014 104401 001252          RETRY2: TYPE     ,SQUES   ;BEGIN TESTING
1919                                ;TYPE '?' INDICATING USER TYPED
1920 003020 000167 177466          JMP      LASTD          ;GO BACK TO REISSUE QUESTION
1921 003024 104401 001252          RETRY3: TYPE     ,SQUES   ;TYPE '?' INDICATING USER TYPED
1922                                ;SOMETHING WRONG FOR PRIORITY
1923 003030 000167 177636          JMP      CONQUES        ;GO BACK TO REISSUE QUESTION
1924
1925
1926                                ;*****
1927                                ;*TEST 1 TEST THAT REFERENCE TO RCSR DOES NOT CAUSE TIMEOUT
1928                                ;*****
1929 003034 000004          TST1: SCOPE
1930 003036 016746 174742          MOV      ERRVEC,-(SP)   ;SAVE THE TIMEOUT VECTOR
1931 003042 012767 003060 174734  MOV      #1$,ERRVEC     ;GO TO 1$ IF TIMEOUT
1932 003050 016702 176334          MOV      DLRCR,R2      ;REGADR = RCSR ADR

```

```
1933 003054 005712          TST      (R2)          :USE REGADR ON BUS
1934 003056 000407          BR       3$           :;<GO TO NEXT TEST IF NO TIMEOUT>
1935 003060 004767 011614    1$:     JSR      PC,SUERT1   :GO SET UP ERROR INFO
1936 003064 012767 003074 176152    MOV      #2$,SESCAPE  :RETURN TO 2$ AFTER ERROR PRINT
1937 003072 104001          ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
1938 003074 022626          2$:     CMP      (SP)+,(SP)+ :CLEAN STACK FROM TIMEOUT
1939 003076 012667 174702    3$:     MOV      (SP)+,ERRVEC :RESTORE TIMEOUT VECTOR
1940
1941
1942 :*****
1943 :*TEST 2      TEST THAT REFERENCE TO XCSR DOES NOT CAUSE TIMEOUT
1944 :*****
1944 003102 000004          TST2:   SCOPE
1945 003104 016746 174674          MOV      ERRVEC,-(SP)  :SAVE THE TIMEOUT VECTOR
1946 003110 012767 003126 174666    MOV      #1$,ERRVEC   :GO TO 1$ IF TIMEOUT
1947 003116 016702 176272          MOV      DLXCSR,R2    :REGADR = XCSR ADR
1948 003122 005712          TST      (R2)          :USE REGADR ON BUS
1949 003124 000407          BR       3$           :;<GO TO NEXT TEST IF NO TIMEOUT>
1950 003126 004767 011546    1$:     JSR      PC,SUERT1   :GO SET UP ERROR INFO
1951 003132 012767 003142 176104    MOV      #2$,SESCAPE  :RETURN TO 2$ AFTER ERROR PRINT
1952 003140 104001          ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
1953 003142 022626          2$:     CMP      (SP)+,(SP)+ :CLEAN STACK FROM TIMEOUT
1954 003144 012667 174634    3$:     MOV      (SP)+,ERRVEC :RESTORE TIMEOUT VECTOR
1955
1956 :*****
1957 :*TEST 3      TEST THAT REFERENCE TO RDBR DOES NOT CAUSE TIMEOUT
1958 :*****
1959 003150 000004          TST3:   SCOPE
1960 003152 016746 174626          MOV      ERRVEC,-(SP)  :SAVE THE TIMEOUT VECTOR
1961 003156 012767 003174 174620    MOV      #1$,ERRVEC   :GO TO 1$ IF TIMEOUT
1962 003164 016702 176222          MOV      DLRDBR,R2    :REGADR = RDBR ADR
1963 003170 005712          TST      (R2)          :USE REGADR ON BUS
1964 003172 000407          BR       3$           :;<GO TO NEXT TEST IF NO TIMEOUT>
1965 003174 004767 011500    1$:     JSR      PC,SUERT1   :GO SET UP ERROR INFO
1966 003200 012767 003210 176036    MOV      #2$,SESCAPE  :RETURN TO 2$ AFTER ERROR PRINT
1967 003206 104001          ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
1968 003210 022626          2$:     CMP      (SP)+,(SP)+ :CLEAN STACK FROM TIMEOUT
1969 003212 012667 174566    3$:     MOV      (SP)+,ERRVEC :RESTORE TIMEOUT VECTOR
1970
1971 :*****
1972 :*TEST 4      TEST THAT REFERENCE TO XDBR DOES NOT CAUSE TIMEOUT
1973 :*****
1974 003216 000004          TST4:   SCOPE
1975 003220 016746 174560          MOV      ERRVEC,-(SP)  :SAVE THE TIMEOUT VECTOR
1976 003224 012767 003242 174552    MOV      #1$,ERRVEC   :GO TO 1$ IF TIMEOUT
1977 003232 016702 176160          MOV      DLXDBR,R2    :REGADR = XDBR ADR
1978 003236 005712          TST      (R2)          :USE REGADR ON BUS
1979 003240 000407          BR       3$           :;<GO TO NEXT TEST IF NO TIMEOUT>
1980 003242 004767 011432    1$:     JSR      PC,SUERT1   :GO SET UP ERROR INFO
1981 003246 012767 003256 175770    MOV      #2$,SESCAPE  :RETURN TO 2$ AFTER ERROR PRINT
1982 003254 104001          ERROR+1 :DL REFERENCE CAUSED BUS TIMEOUT
1983 003256 022626          2$:     CMP      (SP)+,(SP)+ :CLEAN STACK FROM TIMEOUT
1984 003260 012667 174520    3$:     MOV      (SP)+,ERRVEC :RESTORE TIMEOUT VECTOR
1985
1986 :*****
1987 :*TEST 5      TEST THAT RCSR IS ALL ZEROES ON ENTRY
1988 :*****
```

1989 003264 000004
1990 003266 005004
1991 003270 016702 176114
1992 003274 020412
1993 003276 001403
1994 003300 004767 011314
1995 003304 104002
1996
1997
1998
1999 003306 000004
2000 003310 012704 000200
2001 003314 016702 176074
2002 003320 020412
2003 003322 001403
2004 003324 004767 011270
2005 003330 104002
2006
2007
2008
2009 003332 000004
2010 003334 012704 000204
2011 003340 016702 176050
2012 003344 052712 000004
2013 003350 020412
2014 003352 001403
2015 003354 004767 011240
2016 003360 104002
2017 003362 012704 000200
2018 003366 042712 000004
2019 003372 020412
2020 003374 001403
2021 003376 004767 011216
2022 003402 104002
2023
2024
2025
2026 003404 000004
2027 003406 005067 176034
2028 003412 012767 000001 176024
2029 003420 016705 175774
2030 003424 012765 003520 000004
2031 003432 016765 175644 000006
2032 003440 005005
2033 003442 012704 000200
2034 003446 016702 175742
2035 003452 052712 000100
2036 003456 005767 175764
2037 003462 001023
2038 003464 005305
2039 003466 001373
2040 003470 005367 175750
2041 003474 001770
2042 003476 012704 000300
2043 003502 004767 011112
2044 003506 012767 003516 175530

```
TST5: SCOPE
      CLR R4 ;RESULT IN RCSR S/B = 0
      MOV DLXCSR,R2 ;REGADR = RCSR ADR
      CMP R4,(R2) ;[RCSR]=000000 ??
      BEQ TST6 ;<BR IF YES>
      JSR PC,SUER2 ;GO SET UP ERROR INFO
      ERROR+2 ;RCSR NOT CLEAR ON START UP
;*****
;*TEST 6 TEST THAT 'READY' BIT IS ONLY BIT SET IN XCSR
;*****
TST6: SCOPE
      MOV #200,R4 ;RESULT IN XCSR S/B = 000200
      MOV DLXCSR,R2 ;REGADR = XCSR ADR
      CMP R4,(R2) ;[XCSR]=000200 ??
      BEQ TST7 ;<BR IF YES>
      JSR PC,SUER2 ;GO SETUP ERROR INFO
      ERROR+2 ;[XCSR] INCORRECT ON START UP
;*****
;*TEST 7 TEST THAT 'MAINT' BIT CAN BE SET AND CLEARED
;*****
TST7: SCOPE
      MOV #204,R4 ;RESULT IN XCSR S/B = 000204
      MOV DLXCSR,R2 ;REGADR = XCSR ADR
      BIS #BIT2,(R2) ;SET THE 'MAINT' BIT
      CMP R4,(R2) ;RESULT IN XCSR OK ??
      BEQ 1$ ;<BR IF YES>
      JSR PC,SUER2 ;GO SET UP ERROR INFO
      ERROR+2 ;MAINT. BIT FAILED TO SET PROPERLY
1$: MOV #200,R4 ;RESULT IN XCSR S/B = 000200
      BIC #BIT2,(R2) ;NOW CLEAR THE 'MAINT' BIT
      CMP R4,(R2) ;RESULT IN XCSR OK ??
      BEQ TST10 ;<BR IF YES>
      JSR PC,SUER2 ;GO SET UP ERROR INFO
      ERROR+2 ;MAINT BIT FAILED TO CLEAR PROPERLY
;*****
;*TEST 10 TEST THAT XMIT I.E. CAN CAUSE AN INTR
;*****
TST10: SCOPE
      CLR INTFLG ;INIT SOFTWARE INTR FLAG
      MOV #1,TIMR3 ;SET TIMER FOR DELAY
      MOV DLVECT,R5 ;GET VECTOR ADDRESS
      MOV #28,4(R5) ;GO TO 4$ ON INTR
      MOV DLPRI,6(R5) ;PRIORITY LEVEL 4
      CLR R5 ;INIT INTR. TIMER
      MOV #200,R4 ;RESULT IN XCSR S/B = 000200
      MOV DLXCSR,R2 ;REGADR = XCSR ADR
      BIS #100,(R2) ;SET INTR. ENABLE BIT 06
      TST INTFLG ;DID INTR OCCUR YET ??
      BNE 3$ ;BR IF IT DID
      DEC R5 ;COUNT THE TIMER
      BNE 1$ ;BR IF NO TIMEOUT
      DEC TIMR3 ;REDUCE ADDED DELAY
      BEQ 1$ ;BRANCH BACK ON FIRST PASS
      MOV #300,R4 ;RESULT IN XCSR S/B = 000300
      JSR PC,SUER2 ;GO SETUP ERROR INFO
      MOV #4$, $ESCAPE ;RETURN TO 4$ AFTER ERROR PRINT
```

(CHGC6)
(CHGC6)
(CHGC6)

N 3

CZDLCD0 DL11-C,D,E OF LNE TST MACY11 30A(1052) 23-JUN-80 11:11 PAGE 40
 CZDLCD.P11 23-JUN-80 11:10 T10 TEST THAT XMIT I.E. CAN CAUSE AN INTR

SEQ 0039

```

2045 003514 104002          ERROR+2          ;INTR. FAILED
2046 003516
2047 003516 000412          4$: BR TST11          ;;<GO TO NEXT TEST>
2048 003520 005167 175722 2$: COM INTFLG        ;SET THE SOFTWARE FLAG
2049 003524 042712 000100  BIC #100,(R2)      ;TURN OFF I.E. BIT
2050 003530 000002          RTI              ;RETURN CONTROL TO INTR. ROUTINE
2051 003532 020412          3$: CMP R4,(R2)     ;RESULT IN XCSR OK ??
2052 003534 001403          BEQ TST11        ;;<BR IF YES>
2053 003536 004767 011056  JSR PC,SUER2     ;GO SET UP ERROR INFO.
2054 003542 104002          ERROR+2          ;XMIT INTR. NOT SERVICED PROPERLY
2055
2056 *****
2057 *TEST 11 TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
2058 *****
2058 003544 000004          TST11: SCOPE
2059 003546 012704 000100  MOV #100,R4       ;RESULT IN RCSR S/B = 000100
2060 003552 016702 175632  MOV DLRCSR,R2    ;REGADR = RCSR ADR
2061 003556 052712 000100  BIS #BIT6,(R2)   ;SET I.E. BIT
2062 003562 020412          CMP R4,(R2)     ;DID IT SET PROPERLY ??
2063 003564 001403          BEQ 1$          ;;<BR IF YES>
2064 003566 004767 011026  JSR PC,SUER2     ;GO SET UP ERROR INFO.
2065 003572 104002          ERROR+2          ;RCVR I.E. BIT FAILED TO SET PROPERLY
2066 003574 005004          1$: CLR R4         ;RESULT IN RCSR S/B = 000000
2067 003576 042712 000100  BIC #BIT6,(R2)   ;CLEAR THE I.E. BIT
2068 003602 020412          CMP R4,(R2)     ;DID IT CLEAR PROPERLY ??
2069 003604 001403          BEQ TST12        ;;<BR IF YES>
2070 003606 004767 011006  JSR PC,SUER2     ;GO SET UP ERROR INFO
2071 003612 104002          ERROR+2          ;RCVR I.E. BIT FAILED TO CLEAR PROPERLY
2072 *****
2073 *TEST 12 TEST THAT RCVR 'DONE' CAN GENERATE AN INTR.
2074 *****
2075 003614 000004          TST12: SCOPE
2076 003616 016705 175576  MOV DLVECT,R5    ;GET THE VECTOR ADDRESS
2077 003622 012725 004000  MOV #3$,(R5)+    ;GO TO 3$ ON RCVR INTR.
2078 003626 016715 175450  MOV DLPRJ,(R5)   ;AT LEVEL 4
2079 003632 005067 175610  CLR INTFLG       ;INIT THE SOFTWARE FLAG
2080 003636 005005          CLR R5          ;INIT INTR. TIMER
2081 003640 105067 175340  CLR $TMP1        ;INIT WHERE DATA WILL BE STORED
2082 003644 016702 175540  MOV DLRCSR,R2    ;REGADR = RCSR ADR
2083 003650 005012          CLR (R2)        ;INIT THE RCSR TO 000000
2084 003652 052712 000100  BIS #BIT6,(R2)   ;ENABLE RCVR INTERRUPTS
2085 003656 052762 000004 000004  BIS #BIT2,4(R2)  ;NOW TURN ON MAINT MODE
2086 003664 112767 000252 175312  MOVB #252,$TMP1 ;GET DATA PATTERN AND
2087 003672 004767 011614  JSR PC,UPMASK    ;GO MASK OFF BITS AS A FUNCTION OF
2088                                     ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2089 003676 116700 175330          MOVB $TMP14,R0  ;SAVE DATA PATTERN FOR FURTHER USE
2090 003702 116762 175324 000006  MOVB $TMP14,6(R2);LOAD XMIT BUFFER REG.
2091 003710 005767 175532          1$: TST INTFLG     ;DID RCVR INTR. YET ??
2092 003714 001044          BNE 4$          ;BR IF IT DID
2093 003716 005305          DEC R5          ;COUNT THE TIMER
2094 003720 001373          BNE 1$          ;BR IF NO TIMEOUT
2095 003722 013767 177776 175252  MOV @MP$W,$TMP0  ;SAVE ERROR PSW
2096 003730 042762 000004 000004  BIC #BIT2,4(R2)  ;DISABLE MAINT MODE
2097 003736 042712 000100          BIC #100,(R2)   ;DISABLE RCVR INTR.
2098 003742 010667 175230          MOV SP,$REG6    ;SAVE THE ERROR SP
2099 003746 010201          MOV R2,R1       ;DEVADR = RCSR ADR
2100 003750 011203          MOV (R2),R3     ;GET THE WAS DATA

```

```

2101 003752 012704 000200      MOV      #200,R4      ;[RCSR] S/B = 000200
2102 003756 004767 010664      JSR      PC,SUERR1   ;GO SET UP ERROR INFO.
2103 003762 012767 003772 175254  MOV      #2$,SESCAPE ;RETURN TO 2$ AFTER ERROR ALWAYS
2104 003770 104002                ERROR+2 ;RCVR INTERRUPT FAILED
2105 003772 005762 000002      2$: TST      2(R2)     ;REFERENCE RCVR DATA BUFFER
2106                                ;TO CLEAR RCSR IN CASE RCVR
2107                                ;INTERRUPTS COULD NOT BE ENABLED
2108                                ;<GO TO NEXT TEST>
2109 004000 042762 000004 000004 3$: BR      TST13      ;DISABLE THE MAINT MODE
2110 004006 116267 000002 175170  MOVB    2(R2),STMP1  ;GET THE RECEIVED DATA
2111 004014 042712 000100      BIC      #BIT6,(R2)  ;TURN OFF RCVR INTR. ENAB
2112 004020 005167 175422      COM     INTFLG      ;SET THE SOFTWARE FLAG
2113 004024 000002                RTI                    ;RETURN TO MAINLINE
2114 004026 005004                4$: CLR      R4      ;[RCSR] S/B=0
2115 004030 005712                TST      (R2)       ;IS IT ALL ZEROES ??
2116 004032 001403                BEQ     5$          ;<BR IF YES>
2117 004034 004767 010560      JSR      PC,SUER2   ;GO SET UP ERROR INFO
2118 004040 104002                ERROR+2 ;RCVR INTR NOT SERVICED PROPERLY
2119 004042 016701 175344      5$: MOV      DLRDBR,R1 ;SAVE WAS ADDRESS
2120 004046 016702 175344      MOV      DLXDBR,R2  ;SAVE THE S/B ADDRESS
2121 004052 004767 011434      JSR      PC,UPMASK  ;GET THE WAS DATA AND
2122                                ;GO MASK OFF BITS AS A FUNCTION OF
2123                                ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2124 004056 116703 175150      MOVB    STMP14,R3   ;SET UP FOR ERROR CHECKING
2125 004062 110004                MOVB    R0,R4      ;GET THE S/B DATA
2126 004064 020403                CMP     R4,R3      ;WAS = S/B ??
2127 004066 001403                BEQ     TST13      ;<BR IF YES>
2128 004070 004767 010552      JSR      PC,SUERR1  ;GO SET UP THE ERROR INFO
2129 004074 104003                ERROR+3 ;DATA COMPARE ERROR
2130                                ;*****
2131                                ;*TEST 13 TEST THAT 'REQ TO SEND' ASSERTS 'RING'
2132                                ;*****
2133 TST13: SCOPE
2134 004076 000004                BIT     #SW12,BSWR  ;ARE WE TESTING /C OR /D MODEL?
2135 004100 032777 010000 175032  BNE     TST14      ;<BRANCH IF YES>
2136 004106 001047                MOV     #140004,R4 ;RESULT IN RCSR S/B = 140004
2137 004110 012704 140004      MOV     DLRCSR,R2  ;REGADR = RCSR ADR
2138 004114 016702 175270      CLR     (R2)       ;INIT THE RCSR TO 000000
2139 004120 005012                BIS     #BIT2,(R2) ;SET 'REQ TO SEND'
2140 004126 032777 100000 175254  BIT     #BIT15,DLRCSR ;DID 'RING' SET 'DATA SET INT' ?
2141 004134 001003                BNE     1$          ;<BR IF YES>
2142 004136 004767 010456      JSR     PC,SUER2   ;GO SET UP ERROR INFO.
2143 004142 104002                ERROR+2 ;'RING' TRANSITION FAILED TO SET 'DATA SET INT'
2144                                ;NOTE: 'BIT #BIT15,(R2)' RESETS BIT15
2145 004144 012704 040004 1$: MOV     #40004,R4 ;RESULT IN RCSR S/B = 40004
2146 004150 020412                CMP     R4,(R2)    ;BOTH 'RING' AND 'REQ TO SEND' ASSERTED ?
2147 004152 001403                BEQ     2$          ;<BR IF YES>
2148 004154 004767 010440      JSR     PC,SUER2   ;GO SET UP ERROR INFO.
2149 004160 104002                ERROR+2 ;'RING' OR 'REQ TO SEND' FAILED TO SET
2150 004162 005004                2$: CLR     R4      ;RESULT IN RCSR S/B = 000000
2151 004164 042712 000004      BIC     #BIT2,(R2) ;CLEAR 'REQ TO SEND'
2152 004170 032777 100000 175212  BIT     #BIT15,DLRCSR ;DID 'DATA SET INT' GET SET ??
2153 004176 001403                BEQ     3$          ;<BR IF NOT>
2154 004200 004767 010414      JSR     PC,SUER2   ;GO SET UP ERROR INFO.
2155 004204 104002                ERROR+2 ;CLEARING 'RING' SET 'DATA SET INT'
2156 004206 020412                3$: CMP     R4,(R2) ;RCSR CONTAIN ALL ZEROES ??

```



```
2157 004210 001406          BEQ     TST14          ;;<BR IF YES>
2158 004212 004767 010402    JSR     PC,SUER2      ;GO SET UP ERROR INFO.
2159 004216 016767 000002 174754  MOV     .+6,$REG7     ;SAVE THE ERROR PC
2160 004224 104002          ERROR+2 ;CLEARING 'REQ TO SEND' FAILED TO CLEAR 'RING'
2161          ;*****
2162          ;*TEST 14 TEST THAT 'SEC XMIT' ASSERTS 'SEC REC' AND 'DATA SET INT'
2163          ;*****
2164 004226 000004          TST14: SCOPE
2165 004230 032777 010000 174702  BIT     #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
2166 004236 001046          BNE     TST15         ;;<BRANCH IF YES>
2167 004240 016702 175144    MOV     DLRCR,R2      ;REGADR = RCSR ADR
2168 004244 005012          CLR     (R2)         ;INIT RCSR TO 000000
2169 004246 012704 102010    MOV     #102010,R4    ;CONTENTS OF RCSR S/B = 102010
2170 004252 052712 000010    BIS     #BIT3,(R2)    ;SET 'SEC XMIT' BIT
2171 004256 032777 100000 175124  BIT     #BIT15,@DLRCR ;DID 'DATA SET INT' SET ??
2172 004264 001003          BNE     1$           ;;<BR IF YES>
2173 004266 004767 010326    JSR     PC,SUER2      ;GO SET UP ERROR INFO
2174 004272 104002          ERROR+2 ;'DATA SET INT' FAILED TO SET-NOTE THAT
2175          ;'BIT #BIT15,(R2)' RESETS BIT15
2176 004274 012704 002010    1$:    MOV     #2010,R4   ;RESULT IN RCSR S/B = 2010
2177 004300 020412          CMP     R4,(R2)      ;ARE 'SEC XMIT' AND 'SEC REC' BOTH SET ?
2178 004302 001403          BEQ     2$           ;;<BR IF YES>
2179 004304 004767 010310    JSR     PC,SUER2      ;GO SET UP ERROR INFO
2180 004310 104002          ERROR+? ;'SEC XMIT' OR 'SEC REC' FAILED TO SET
2181          ;OR 'DATA SET INT' FAILED TO BE CLEARED
2182          ;WHEN REFERENCING RCSR
2183 004312 012704 100000    2$:    MOV     #BIT15,R4  ;RESULT IN RCSR S/B = 100000
2184 004316 042712 000010    BIC     #BIT3,(R2)   ;CLEAR 'SEC XMIT' BIT
2185 004322 032777 100000 175060  BIT     #BIT15,@DLRCR ;DID CLEARING IT SET 'DATA SET INT'??
2186 004330 001003          BNE     3$           ;;<BR IF YES>
2187 004332 004767 010262    JSR     PC,SUER2      ;GO SET UP ERROR INFO.
2188 004336 104002          ERROR+2 ;CLEARING 'SEC XMIT' FAILED TO SET 'DATA
2189          ;SET INT. (NOTE THAT REFERENCING RCSR
2190          ;CLEARS 'DATA SET INT'
2191 004340 005004          3$:    CLR     R4         ;RESULT IN RCSR S/B = 000000
2192 004342 020412          CMP     R4,(R2)      ;'SEC XMIT' AND 'SEC REC' CLEAR ?
2193 004344 001403          BEQ     TST15         ;;<BR IF YES>
2194 004346 004767 010246    JSR     PC,SUER2      ;GO SETUP ERROR INFO
2195 004352 104002          ERROR+2 ;'SEC XMIT' OR 'SEC REC' FAILED TO CLEAR
2196          ;OR REFERENCING RCSR FAILED TO CLEAR 'DATA SET INT'
2197          ;*****
2198          ;*TEST 15 TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'
2199          ;*****
2200 004354 000004          TST15: SCOPE
2201 004356 032777 010000 174554  BIT     #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
2202 004364 001046          BNE     TST16         ;;<BRANCH IF YES>
2203 004366 016702 175016    MOV     DLRCR,R2      ;REGADR = RCSR ADR
2204 004372 005012          CLR     (R2)         ;INIT RCSR TO 000000
```

```

2205 004374 012704 130002      MOV      #130002,R4      ;RESULT IN RCSR S/B = 130002
2206 004400 052712 000002      BIS      #BIT1 (R2)     ;SET 'DTR' BIT
2207 004404 032777 100000 174776 BIT      #BIT15,@DLRCSR ;DID 'DATA SET INT' SET ??
2208 004412 001003              BNE     1$             ;; <BR IF YES>
2209 004414 004767 010200      JSR     PC,SUER2       ;GO SET UP ERROR INFO.
2210 004420 104002      ERROR+2 ;'DATA SET INT' FAILED TO SET -
2211 ;NOTE: THE REFERENCE TO RCSR ABOVE WILL
2212 ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.
2213 004422 012704 030002      1$: MOV      #30002,R4      ;RESULT IN RCSR S/B = 30002
2214 004426 020412      CMP     R4,(R2)        ;'DTR','CLR TO SEND', AND 'CAR DET' ALLSET
2215 004430 001403      BEQ     2$             ;; <BR IF ALL SET>
2216 004432 004767 010162      JSR     PC,SUER2       ;GO SET UP ERROR INFO
2217 004436 104002      ERROR+2 ;'DTR','CLR TO SEND' OR 'CAR DET' FAILED
2218 ;TO SET OR 'DATA SET INT' FAILED TO CLEAR
2219
2220 004440 012704 100000      2$: MOV      #BIT15,R4      ;RESULT IN RCSR S/B = 100000
2221
2222 004444 042712 000002      BIC     #BIT1 (R2)     ;NOW CLEAR 'DTR'
2223 004450 032777 100000 174732 BIT      #BIT15,@DLRCSR ;DID 'DATA SET INT' SET ??
2224 004456 001003      BNE     3$             ;; <BR IF YES>
2225 004460 004767 010134      JSR     PC,SUER2       ;GO SETUP ERROR INFO
2226
2227 004464 104002      ERROR+2 ;'DATA SET INT' FAILED TO SET WHEN 'DTR'
2228 ;WENT TO A ZERO.
2229 004466 005004      3$: CLR     R4           ;RESULT IN RCSR S/B = 000000
2230 004470 020412      CMP     R4,(R2)        ;DID ALL BITS CLEAR??
2231 004472 001403      BEQ     TST16          ;; <BR IF YES>
2232 004474 004767 010120      JSR     PC,SUER2       ;GO SET UP ERROR INFO
2233 004500 104002      ERROR+2 ;'DTR','CLR TO SEND' OR 'CAR DET' FAILED
2234 ;TO CLEAR PROPERLY

```

*TEST 16 TEST THAT 'DATA SET INT ENAB' CAN SET AND CLEAR

```

2235
2236
2237 TST16: SCOPE
2238 004502 000004
2239 004504 032777 010000 174426 BIT      #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2240 004512 001023      BNE     TST17          ;; <BRANCH IF YES>
2241 004514 016702      MOV     DLRCSR,R2      ;REGADR = RCSR ADR
2242 004520 012704 000040      MOV     #40,R4         ;RESULT IN RCSR S/B = 000040
2243 004524 052712 000040      BIS     #BIT5 (R2)     ;SET THE 'DATA SET I.E.' BIT
2244 004530 020412      CMP     R4,(R2)        ;DID IT SET OK ??
2245 004532 001403      BEQ     1$             ;; <BR IF YES>
2246 004534 004767 010060      JSR     PC,SUER2       ;GO SET UP ERROR INFO
2247 004540 104002      ERROR+2 ;'DAT SET I. E.' FAILED TO SET
2248 004542 005004      1$: CLR     R4           ;MAKE S/B DATA = 000000
2249 004544 042712 000040      BIC     #BIT5 (R2)     ;NOW CLEAR THE 'DATA SET I.E.' BIT
2250 004550 020412      CMP     R4,(R2)        ;DID IT CLEAR OK ??
2251 004552 001403      BEQ     TST17          ;; <BR IF YES>
2252 004554 004767 010040      JSR     PC,SUER2       ;GO SET UP ERROR INFO.
2253 004560 104002      ERROR+2 ;'DATA SET I.E.' FAILED TO CLEAR

```

*TEST 17 TEST THE 'DATA SET I.E.' CAN CAUSE A RCVR INTR

```

2254
2255
2256 TST17: SCOPE
2257 004562 000004
2258 004564 032777 010000 174346 BIT      #SW12,@SWR      ;ARE WE TESTING A /C OR /D MODEL?
2259 004572 001054      BNE     TST20          ;; <BRANCH IF YES>
2260 004574 016705 174620      MOV     DLVECT,R5     ;GET THE VECTOR ADDR

```

```
2261 004600 012725 004666      MOV      #3$, (R5)+      ;GO TO 3$ ON RCVR INTR.
2262 004604 016715 174472      MOV      DLPRI, (R5)    ;AT LEVEL 4
2263 004610 005005              CLR      R5             ;INIT INTR. TIMER
2264 004612 005067 174630      CLR      INTFLG        ;INIT SOFTWARE FLAG
2265 004616 005004              CLR      R4             ;RESULT IN RCSR S/B = 0 AFTER INTR.
2266 004620 016702 174564      MOV      DLRCR, R2      ;REGADR = RCSR ADR
2267 004624 052712 000040      BIS      #BIT5, (R2)    ;SET THE 'DATA SET I.E.' BIT
2268 004630 052712 000002      BIS      #BIT1, (R2)    ;NOW SET 'DTR' TO GEN INTR.
2269 004634 005767 174606      1$:     TST      INTFLG    ;DID INTR OCCUR YET ??
2270 004640 001016              BNE     4$             ;BR IF YES
2271 004642 005305              DEC     R5             ;COUNT THE TIMER
2272 004644 001373              BNE     1$             ;BR IF NO TIMEOUT
2273 004646 004767 007746      JSR     PC, SUER2      ;GO SET UP ERROR INFO
2274 004652 005012              CLR     (R2)           ;TURN IT ALL OFF
2275 004654 012767 004664 174362  MOV     #2$, $ESCAPE    ;COME BACK TO 2$ IN ALL CASES
2276 004662 104002              ERROR+2 ;'DATA SET' INTR FAILED TO OCCUR
2277 004664              2$:
2278 004664 000417              BR     TST20           ;; <GO TO NEXT TEST>
2279 004666 005012              3$:     CLR     (R2)         ;ZERO THE RCSR
2280 004670 005167 174552      COM     INTFLG        ;SET THE SOFTWARE FLAG
2281 004674 000002              RTI                    ;RETURN TO SENDER
2282 004676 032712 100000      4$:     BIT     #BIT15, (R2) ;DID 'DATA SET INT' GET SET BY INTR. SERVICE ??
2283 004702 001003              BNE     5$             ;; <BR IF YES>
2284 004704 004767 007710      JSR     PC, SUER2      ;GO SET UP ERROR INFO
2285 004710 104002              ERROR+2 ;DATA SET INTR. NOT SERVICED PROPERLY
2286 004712 020412              5$:     CMP     R4, (R2)    ;ALL BITS IN RCSR CLEAR ??
2287 004714 001403              BEQ     TST20         ;; <BR IF YES>
2288 004716 004767 007676      JSR     PC, SUER2      ;GO SET UP ERROR INFO
2289 004722 104002              ERROR+2 ;INTR. SERVICE FAILED TO CLEAR RCSR
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
```

*TEST 20 TEST THAT THE 'BREAK' BIT CAN BE SET AND CLEARED

TST20: SCOPE
BIT #SW12, @SWR ;ARE WE TESTING /C OR /D MODEL?
BNE TST21 ;; <BRANCH IF YES>
MOV #201, R4 ;RESULT S/B = 201 IN XCSR
MOV DLXCSR, R2 ;SET UP REGADR
BIS #BIT0, (R2) ;SET THE 'BREAK' BIT
CMP R4, (R2) ;DID IT SET PROPERLY ??
BEQ 1\$;; <BR IF YES>
JSR PC, SUER2 ;GO SET UP ERROR INFO.
ERROR+2 ;'BREAK' BIT FAILED TO SET PROPERLY
1\$: MOV #200, R4 ;RESULT S/B = 200 IN XCSR
BIC #BIT0, (R2) ;CLEAR THE 'BREAK' BIT
CMP R4, (R2) ;DID IT CLEAR PROPERLY ??
BEQ TST21 ;; <BR IF YES>
JSR PC, SUER2 ;GO SET UP ERROR INFO
ERROR+2 ;'BREAK' FAILED TO CLEAR PROPERLY

*TEST 21 TEST THAT A 'RESET' CLEARS THE 'BREAK' BIT

TST21: SCOPE
MOV #200, R4 ;RESULT S/B = 200
MOV DLXCSR, R2 ;SET UP REGADR

2317 005020 052712 000001
2318 005024 000005
2319 005026 020412
2320 005030 001403
2321 005032 004767 007562
2322 005036 104002
2323

BIS #BIT0,(R2) ;SET THE 'BREAK' BIT
RESET ;CLEAR IT WITH A 'RESET'
CMP R4,(R2) ;DID IT CLEAR ??
BEQ TST22 ;:<BR IF YES>
JSR PC,SUER2 ;GO SET UP ERROR INFO.
ERROR+2 ;RESET INSTR. FAILED TO CLEAR 'BREAK'

```
2324
2325
2326
2327 005040 000004
2328 005042 012767 000001 174172
2329 005050 004767 007700
2330 005054 005067 174350
2331 005060 012767 015130 174350 1$:
2332 005066 004767 007710
2333 005072 005767 174324 2$:
2334 005076 001040
2335 005100 005767 174320
2336 005104 001053
2337 005106 005767 174314
2338 005112 001065
2339 005114 022767 023040 174312
2340 005122 001003
2341 005124 004767 010144
2342 005130 000500
2343 005132 005367 174302 3$:
2344 005136 001355
2345 005140 005367 174276
2346 005144 001352
2347 005146 042777 000100 174234
2348 005154 042777 000104 174232
2349 005162 104401 017110
2350 005166 012767 005176 174050
2351 005174 104000
2352 005176 4$:
2353 005176 000455
2354 005200 016701 174204 5$:
2355 005204 016702 174204
2356 005210 011203
2357 005212 012704 000204
2358 005216 004767 007424
2359 005222 012767 005232 174014
2360 005230 104002
2361 005232 6$:
2362 005232 000437
2363 005234 016701 174150 7$:
2364 005240 010102
2365 005242 011203
2366 005244 012704 000200
2367 005250 004767 007372
2368 005254 012767 005264 173762
2369 005262 104002
2370 005264 8$:
2371 005264 000422
2372 005266 016701 174116 9$:
2373 005272 016702 174114
2374 005276 016703 173702
2375 005302 004767 007340
2376 005306 012767 005316 173730
2377 005314 104005
2378 005316 005267 174106 10$:
2379 005322 022767 000003 174100
```

*TEST 22 TEST TO TURN AROUND NULL-DEL-NULP PATTERN

TST22: SCOPE
MOV #1,\$TIMES ;:DO 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

```
2380 005330 001253 BNE 1$ ;BR IF NOT
2381
2382
2383 *****
2384 *TEST 23 TEST TO TURN AROUND BINARY UP COUNT PATTERN
2385 *****
2386 TST23: SCOPE
2387 005332 000004 MOV #1,$TIMES ;DO 1 ITERATION
2388 005334 012767 000001 173700 JSR PC,SUVEC ;GO SET UP VECTORS
2389 005342 004767 007406 CLR RTRY ;INITIALIZE RETRY FLAG
2390 005346 005067 174056 1$: MOV #LDOUT2,LDOUT ;SET POINTER TO LOAD ROUTINE
2391 005352 012767 015152 JSR PC,PRIME ;GO SET UP BUFFERS AND DEVICE
2392 005360 004767 007416 2$: TST XFLGO ;ANY HARD XMIT ERRORS ??
2393 005364 005767 174032 BNE 5$ ;BR IF YES
2394 005370 001040 TST RFLGO ;ANY HARD RECEIVER ERROR ??
2395 005372 005767 174026 BNE 7$ ;BR IF YES
2396 005376 001053 TST RFLG1 ;ANY SOFT RECEIVER ERRORS ??
2397 005400 005767 174022 BNE 9$ ;BR IF YES
2398 005404 001065 CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??
2399 005406 022767 023040 174020 BNE 3$ ;BR IF NOT
2400 005414 001003 JSR PC,CHKDAT ;GO CHECK THE DATA BUFFERS
2401 005416 004767 007652 BR TST24 ;<GO TO NEXT TEST>
2402 005422 000500 174010 3$: DEC TIMR1 ;DEC TIMEOUT COUNTER 1
2403 005424 005367 174010 BNE 2$ ;BR IF NO TIMEOUT
2404 005430 001355 DEC TIMR2 ;DEC TIMEOUT COUNTER 2
2405 005432 005367 174004 BNE 2$ ;BR IF NO TIMEOUT
2406 005436 001352 BIC #100,@DLRCSR ;TURN OFF THE INTRS.
2407 005440 042777 000100 173742 BIC #104,@DLXCSR
2408 005446 042777 000104 173740 TYPE ,XMSG2 ;GO TYPE TIMEOUT MESSAGE
2409 005454 104401 017167 MOV #4$,$ESCAPE ;GO TO 4$ AFTER ERROR PRINT
2410 005460 012767 005470 173556 ERROR ;PRINT ERROR PC
2411 005470 4$:
2412 005472 000455 BR TST24 ;<GO TO NEXT TEST>
2413 005474 016701 173712 5$: MOV DLRCSR,R1 ;PUT DEVADR IN R1
2414 005476 016702 173712 MOV DLXCSR,R2 ;PUT REGADR IN R2
2415 005502 011203 MOV (R2),R3 ;GET THE WAS DATA
2416 005504 012704 000204 MOV #204,R4 ;PUT S/B DATA IN R4
2417 005510 004767 007132 JSR PC,SUERR1 ;GO SET UP ERROR INFO
2418 005514 012767 005524 173522 MOV #6$,$ESCAPE ;GO TO 6$ AFTER PRINTING ERROR
2419 005522 104002 ERROR+2 ;TRANSMITTER FALSE INTERRUPT
2420 005524 6$:
2421 005526 000437 BR TST24 ;<GO TO NEXT TEST>
2422 005528 016701 173656 7$: MOV DLRCSR,R1 ;SAVE THE DEVADR
2423 005532 010102 MOV R1,R2 ;SAVE THE REGADR
2424 005534 011203 MOV (R2),R3 ;GET THE WAS DATA
2425 005536 012704 000200 MOV #200,R4 ;RESULT S/B = 200
2426 005542 004767 007100 JSR PC,SUERR1 ;GO SET UP ERROR INFO
2427 005546 012767 005556 173470 MOV #8$,$ESCAPE ;GO TO 8$ AFTER ERROR PRINT
2428 005554 104002 ERROR+2 ;RECEIVER FALSE INTERRUPT
2429 005556 8$:
2430 005558 000422 BR TST24 ;<GO TO NEXT TEST>
2431 005560 016701 173624 9$: MOV DLRCSR,R1 ;SAVE THE DEVADR
2432 005564 016702 173622 MOV DLRDBR,R2 ;SAVE REGADR
2433 005570 016703 173410 MOV $TMP1,R3 ;GET CONTENTS OF ERROR RDBR
2434 005574 004767 007046 JSR PC,SUERR1 ;GO SETUP ERROR INFO
2435 005600 012767 005610 173436 MOV #10$,$ESCAPE ;GO TO 10$ AFTER ERROR PRINT
```

```

2436 005606 104005          ERROR+5          ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
2437 005610 005267 173614    10$: INC RTRY          ;COUNT ONE TRY
2438 005614 022767 000003 173606 CMP #3,RTRY      ;TRIED THREE TIMES
2439 005622 001253          BNE 1$          ;BR IF NOT
2440          ;:*****
2441          ;*TEST 24 TEST TO TURN AROUND BINARY DOWN COUNT PATTERN
2442          ;:*****
2443 005624 000004          TST24: SCOPE
2444 005626 012767 000001 173406 MOV #1,$TIMES    ;;DO 1 ITERATION
2445 005634 004767 007114     JSR PC,SUVEC    ;GO SET UP VECTORS
2446 005640 005067 173564     CLR RTRY        ;INITIALIZE RETRY FLAG
2447 005644 012767 015172 173564 1$: MOV #LDOUT3,LDOUT ;SET POINTER TO LOAD ROUTINE
2448 005652 004767 007124     JSR PC,PRIME    ;GO SET UP BUFFERS AND DEVICE
2449 005656 005767 173540     2$: TST XFLGO     ;ANY HARD XMIT ERRORS ??
2450 005662 001040          BNE 5$          ;BR IF YES
2451 005664 005767 173534     TST RFLGO      ;ANY HARD RECEIVER ERROR ??
2452 005670 001053          BNE 7$          ;BR IF YES
2453 005672 005767 173530     TST RFLG1      ;ANY SOFT RECEIVER ERRORS ??
2454 005676 001065          BNE 9$          ;BR IF YES
2455 005700 022767 023040 173526 CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??
2456 005706 001003          BNE 3$          ;BR IF NOT
2457 005710 004767 007360     JSR PC,CHKDAT   ;GO CHECK THE DATA BUFFERS
2458 005714 000500          BR TST25        ;;<GO TO NEXT TEST>
2459 005716 005367 173516     3$: DEC TIMR1     ;DEC TIMEOUT COUNTER 1
2460 005722 001355          BNE 2$          ;BR IF NO TIMEOUT
2461 005724 005367 173512     DEC TIMR2      ;DEC TIMEOUT COUNTER 2
2462 005730 001352          BNE 2$          ;BR IF NO TIMEOUT
2463 005732 042777 000100 173450 BIC #100,@DLRCSR ;TURN OFF THE INTRS.
2464 005740 042777 000104 173446 BIC #104,@DLXCSR
2465 005746 104401 017250     TYPE #,MSG3     ;GO TYPE TIMEOUT MESSAGE
2466 005752 012767 005762 173264 MOV #4$, $ESCAPE ;GO TO 4$ AFTER ERROR PRINT
2467 005760 104000          ERROR          ;PRINT ERROR PC
2468 005762          4$:
2469 005762 000455          BR TST25        ;;<GO TO NEXT TEST>
2470 005764 016701 173420     5$: MOV DLRCSR,R1   ;PUT DEVADR IN R1
2471 005770 016702 173420     MOV DLXCSR,R2   ;PUT REGADR IN R2
2472 005774 011203          MOV (R2),R3     ;GET THE WAS DATA
2473 005776 012704 000204     MOV #204,R4     ;PUT S/B DATA IN R4
2474 006002 004767 006640     JSR PC,SUERR1   ;GO SET UP ERROR INFO
2475 006006 012767 006016 173230 MOV #6$, $ESCAPE ;GO TO 6$ AFTER PRINTING ERROR
2476 006014 104002          ERROR+2        ;TRANSMITTER FALSE INTERRUPT
2477 006016          6$:
2478 006016 000437          BR TST25        ;;<GO TO NEXT TEST>
2479 006020 016701 173364     7$: MOV DLRCSR,R1   ;SAVE THE DEVADR
2480 006024 010102          MOV R1,R2       ;SAVE THE REGADR
2481 006026 011203          MOV (R2),R3     ;GET THE WAS DATA
2482 006030 012704 000200     MOV #200,R4     ;RESULT S/B = 200
2483 006034 004767 006606     JSR PC,SUERR1   ;GO SET UP ERROR INFO
2484 006040 012767 006050 173176 MOV #8$, $ESCAPE ;GO TO 8$ AFTER ERROR PRINT
2485 006046 104002          ERROR+2        ;RECEIVER FALSE INTERRUPT
2486 006050          8$:
2487 006050 000422          BR TST25        ;;<GO TO NEXT TEST>
2488 006052 016701 173332     9$: MOV DLRCSR,R1   ;SAVE THE DEVADR
2489 006056 016702 173330     MOV DLRDBR,R2   ;SAVE REGADR
2490 006062 016703 173116     MOV $TMP1,R3    ;GET CONTENTS OF ERROR RDBR
2491 006066 004767 006554     JSR PC,SUERR1   ;GO SETUP ERROR INFO

```

```
2492 006072 012767 006102 173144      MOV      #10$, $ESCAPE      ;GO TO 10$ AFTER ERROR PRINT
2493 006100 104005                ERROR+5      ;REPORT SOFT ERROR (PARITY,FRAMING OR OVERRUN
2494 006102 005267 173322      10$:      INC      RTRY              ;COUNT ONE TRY
2495 006106 022767 000003 173314      CMP      #3,RTRY           ;TRIED THREE TIMES
2496 006114 001253                BNE                ;BR IF NOT
2497                                     ;*****
2498                                     ;*TEST 25      TEST TO TURN AROUND WORST CASE PATTERN
2499                                     ;*****
2500      006116 000004                TST25:      SCOPE
2501 006120 012767 000001 173114      MOV      #1,$TIMES         ;;DO 1 ITERATION
2502 006126 004767 006622                JSR      PC,SUVEC          ;GO SET UP VECTORS
2503 006132 005067 173272                CLR      RTRY              ;INITIALIZE RETRY FLAG
2504 006136 012767 015226 173272      1$:      MOV      #LDOUT4,LDOUT     ;SET POINTER TO LOAD ROUTINE
2505 006144 004767 006632                JSR      PC,PRIME         ;GO SET UP BUFFERS AND DEVICE
2506 006150 005767 173246      2$:      TST      XFLGO            ;ANY HARD XMIT ERRORS ??
2507 006154 001042                BNE                ;BR IF YES
2508 006156 005767 173242                TST      RFLGO            ;ANY HARD RECEIVER ERROR ??
2509 006162 001056                BNE                ;BR IF YES
2510 006164 005767 173236                TST      RFLG1            ;ANY SOFT RECEIVER ERRORS ??
2511 006170 001071                BNE                ;BR IF YES
2512 006172 022767 023040 173234      CMP      #256,$IBUFEND,IPTR ;RECEIVED 256. BYTES ??
2513 006200 001004                BNE                ;BR IF NOT
2514 006202 004767 007066                JSR      PC,CHKDAT        ;GO CHECK THE DATA BUFFERS
2515 006206 000167 001642                JMP      $EOP              ;GO TO NEXT TEST
2516 006212 005367 173222      3$:      DEC      TIMR1            ;DEC TIMEOUT COUNTER 1
2517 006216 001354                BNE                ;BR IF NO TIMEOUT
2518 006220 005367 173216                DEC      TIMR2            ;DEC TIMEOUT COUNTER 2
2519 006224 001351                BNE                ;BR IF NO TIMEOUT
2520 006226 042777 000100 173154      BIC      #100,$DLRCSR      ;TURN OFF THE INTRS.
2521 006234 042777 000104 173152      BIC      #104,$DLXCSR
2522 006242 104401 017333                TYPE      ,XMSG4          ;GO TYPE TIMEOUT MESSAGE
2523 006246 012767 006256 172770      MOV      #4$, $ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
2524 006254 104000                ERROR      ;PRINT ERROR PC
2525 006256 000167 001572      4$:      JMP      $EOP              ;GO TO NEXT TEST
2526 006262 016701 173122      5$:      MOV      DLRCSR,R1        ;PUT DEVADR IN R1
2527 006266 016702 173122                MOV      DLXCSR,R2        ;PUT REGADR IN R2
2528 006272 011203                MOV      (R2),R3          ;GET THE WAS DATA
2529 006274 012704 000204                MOV      #204,R4          ;PUT S/B DATA IN R4
2530 006300 004767 006342                JSR      PC,SUERR1        ;GO SET UP ERROR INFO
2531 006304 012767 006314 172732      MOV      #6$, $ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
2532 006312 104002                ERROR+2      ;TRANSMITTER FALSE INTERRUPT
2533 006314 000167 001534      6$:      JMP      $EOP              ;GO TO NEXT TEST
2534 006320 016701 173064      7$:      MOV      DLRCSR,R1        ;SAVE THE DEVADR
2535 006324 010102                MOV      R1,R2            ;SAVE THE REGADR
2536 006326 011203                MOV      (R2),R3          ;GET THE WAS DATA
2537 006330 012704 000200                MOV      #200,R4          ;RESULT S/B = 200
2538 006334 004767 006306                JSR      PC,SUERR1        ;GO SET UP ERROR INFO
2539 006340 012767 006350 172676      MOV      #8$, $ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
2540 006346 104002                ERROR+2      ;RECEIVER FALSE INTERRUPT
2541 006350 000167 001500      8$:      JMP      $EOP              ;GO TO NEXT TEST
2542 006354 016701 173030      9$:      MOV      DLRCSR,R1        ;SAVE THE DEVADR
2543 006360 016702 173026                MOV      DLRDBR,R2        ;SAVE REGADR
2544 006364 016703 172614                MOV      $TMP1,R3         ;GET CONTENTS OF ERROR RDBR
2545 006370 004767 006252                JSR      PC,SUERR1        ;GO SETUP ERROR INFO
2546 006374 012767 006404 172642      MOV      #10$, $ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
2547 006402 104005                ERROR+5      ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
```



```

2548 006404 005267 173020 10$: INC RTRY ;COUNT ONE TRY
2549 006410 022767 000003 173012 CMP #3,RTRY ;TRIED THREE TIMES
2550 006416 001247 BNE 1$ ;BR IF NOT
2551 006420 000167 001430 JMP SEOP ;GO TO END OF PASS ROUTINE
2552
2553 ;THIS IS PROGRAM #2
2554 ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
2555 ; A) SELECTION OF A TRANSMITTER DATA BUFFER
2556 ; B) SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
2557 ; C) SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
2558 ; BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
2559 ; D) A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
2560 ;
2561
2562 006424 000240 PRG2: NOP ;INDICATE THAT USER SELECTED
2563 006426 104401 017460 TYPE ,PROG2M ;PROGRAM #2
2564
2565 006432 104401 021102 PRG2A: TYPE ,LINTAD ;ASK USER FOR THE TRANSMITTER
2566 ;DATA BUFFER ADDRESS OF THE DEVICE
2567 ;HE WISHES TO TEST
2568 006436 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
2569 ;AND STORE ON TOP OF STACK
2570 ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
2571 006440 012602 MOV (SP)+,R2 ;GET THE ANSWER TYPED
2572 006442 020227 176176 CMP R2,#176176 ;IS THE NUMBER TOO HIGH?
2573 006446 101065 BHI RED01 ;IF YES - GO TO RETRY SITUATION
2574 006450 020227 175616 CMP R2,#175616 ;IS THE NUMBER TOO LOW?
2575 006454 103462 BLO RED01 ;IF YES - GO TO RETRY SITUATION
2576 006456 132702 000001 BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
2577 ;ON AN EVEN BOUNDARY?
2578 006462 001057 BNE RED01 ;IF NOT GO TO RETRY SITUATION
2579 ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
2580 006464 010203 MOV R2,R3 ;GET THE USER RESPONSE
2581 006466 142703 000370 BICB #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
2582 ;LEAST SIGNIFICANT DIGIT
2583 006472 122703 000006 CMPB #6,R3 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2584 ;USER RESPONSE EQUAL TO A SIX?
2585 006476 001051 BNE RED01 ;BRANCH IF NOT
2586 006500 010267 172476 MOV R2,$TMP0 ;THE TRANSMITTER ADDRESS
2587 ;TYPED IS OK - STORE FOR
2588 ;FUTURE USE
2589 ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2590 006504 016746 171274 MOV ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
2591 006510 012767 006522 171266 MOV #2$,ERRVEC ;SET UP TIMEOUT SERVICE ADDRESS
2592 006516 005712 TST (R2) ;IF PRESENT WE WILL EXECUTE THE
2593 ;NEXT INSTRUCTION - IF NOT
2594 ;WE GO TO 2$:
2595 006520 000412 BR 4$ ;BRANCH IF PRESENT
2596 006522 004767 006200 2$: JSR PC,SUERT2 ;GO SET UP FOR ERROR INFORMATION
2597 006526 012767 006536 172510 MOV #3$, $ESCAPE ;POINT OF RETURN AFTER ERROR REPORT
2598 006534 104006 ERROR +6 ;XDBR REFERENCE CAUSED TIMEOUT
2599 006536 022626 3$: CMP (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
2600 006540 012667 171240 MOV (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
2601 006544 000426 BR RED01 ;GO TO RETRY SITUATION
2602 006546 012667 171232 4$: MOV (SP)+,ERRVEC ;DEVICE REGISTER IS PRESENT!
2603 ;RESTORE TIMEOUT VECTOR

```

```
2604 :WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
2605 :DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN
2606 :SUCCESSIVE CHARACTER TRANSFERS
2607 006552 104401 021163 PRG2B: TYPE ,SELCAR :ASK USER FOR THE CHARACTER HE
2608 :WISHES TO TRANSFER
2609 006556 104412 RDOCT :ACCEPT THE ANSWER TYPED BY
2610 :USER AND STORE ON TOP OF STACK
2611 006560 012667 172420 MOV (SP)+,$TMP1 :GET THE ANSWER TYPED
2612 :NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
2613 :OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
2614 006564 104401 021271 TYPE ,SELDLY :ASK THE USER FOR THE DELAY
2615 :IN MSEC (OCTAL NO.) BETWEEN
2616 :CHARACTER TRANSFERS
2617 006570 104412 RDOCT :ACCEPT THE ANSWER TYPED BY
2618 :USER AND STORE ON TOP OF STACK
2619 006572 012667 172410 MOV (SP)+,$TMP2 :GET THE ANSWER TYPED
2620 006576 116767 172404 000012 1$: MOVSB $TMP2,$2$ :SET THE DELAY COUNT ARGUMENT
2621 :FOR TIMER ROUTINE
2622 006604 116777 172374 172370 MOVSB $TMP1,@$TMP0 :LOAD THE TRANSMITTER DATA
2623 :BUFFER WITH THE CHARACTER
2624 006612 004767 005470 JSR PC,DELAY :GO OFF TO WAIT THE SPECIFIED
2625
2626
2627
2628
2629
2630 :NO. OF MSEC. BEFORE ISSUING
2631 :ANOTHER CHARACTER
2632 006616 000000 2$: .WORD 0 :THIS IS WHERE THE DELAY COUNT RESIDES
2633 006620 000766 BR 1$ :GO BACK TO ISSUE ANOTHER CHARACTER
2634 006622 104401 001252 REDO1: TYPE ,SQUES :TYPE A QUESTION MARK(?)
2635 006626 000167 177600 JMP PRG2A :REITERATE THE XDBR QUESTION TO USER
2636 :THIS IS PROGRAM #3
2637 :THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
2638 : A) SELECTION OF A TRANSMITTER DATA BUFFER
2639 : B) SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
2640 : IN MAINTENANCE MODE
2641 : C) SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
2642 : BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
2643 : D) A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
2644 :
2645
2646 006632 000240 PRG3: NOP
2647 006634 104401 017524 TYPE ,PROG3M :INDICATE THAT USER SELECTED
2648 :PROGRAM #3
2649 006640 104401 021102 PRG3A: TYPE ,LINTAD :ASK USER FOR THE TRANSMITTER DATA
2650 :BUFFER ADDRESS OF THE DEVICE
2651 :HE WISHES TO TEST
2652 006644 104412 RDOCT :ACCEPT THE ANSWER TYPED BY
2653 :USER AND STORE ON TOP OF STACK
2654 :CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2655 006646 012602 MOV (SP)+,R2 :GET THE ANSWER TYPED
2656 006650 020227 176176 CMP R2,#176176 :IS THE NUMBER TOO HIGH?
2657 006654 101071 BHI REDO2 :IF YES - GO TO RETRY SITUATION
2658 006656 020227 175616 CMP R2,#175616 :IS THE NUMBER TOO LOW?
2659 006662 103466 BLO REDO2 :IF YES - GO TO RETRY SITUATION
```

```

2660 006664 132702 000001          BITB  #BIT0,R2          ;NUMBER IS IN RANGE BUT IS IT
2661                                     ;ON AN EVEN BOUNDARY?
2662 006670 001063          BNE   REDO2            ;IF NOT - GO TO RETRY SITUATION
2663                                     ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR DBR ADDRESS
2664 006672 010203          MOV   R2,R3           ;GET THE USER RESPONSE
2665 006674 142703 000370          BICB  #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
2666                                     ;LEAST SIGNIFICANT DIGIT
2667 006700 122703 000006          CMPB  #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2668                                     ;USER RESPONSE EQUAL TO A TWO?
2669 006704 001055          BNE   REDO2            ;BRANCH IF NOT
2670 006706 010267 172270          MOV   R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED IS
2671                                     ;OK - STORE FOR FUTURE USE
2672                                     ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2673 006712 016746 171066          MOV   ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
2674 006716 012767 006730 171060          MOV   #2$,ERRVEC     ;SET UP TIMEOUT SERVICE ADDRESS
2675 006724 005712          TST   (R2)            ;IF PRESENT WE WILL EXECUTE THE
2676                                     ;NEXT INSTRUCTION - IF NOT WE
2677                                     ;GO TO 2$:
2678 006726 000412          BR    4$              ;BRANCH IF PRESENT
2679 006730 004767 005772 2$:      JSR   PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
2680 006734 012767 006744 172302          MOV   #3$, $ESCAPE   ;POINT OF RETURN AFTER ERROR REPORT
2681 006742 104006          ERROR +6             ;XDBR REFERENCE CAUSED TIMEOUT
2682 006744 022626 3$:      CMP   (SP)+,(SP)+    ;CLEAN STACK FROM TIMEOUT
2683 006746 012667 171032          MOV   (SP)+,ERRVEC   ;RESTORE TIMEOUT VECTOR
2684 006752 000432          BR    REDO2           ;GO TO RETRY SITUATION
2685 006754 012667 171024 4$:      MOV   (SP)+,ERRVEC   ;DEVICE REGISTER IS PRESENT!
2686                                     ;RESTORE TIMEOUT VECTOR
2687                                     ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
2688                                     ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN SUCCESSIVE
2689                                     ;CHARACTER TRANSFERS
2690 006760 104401 021163          PRG3B: TYPE  ,SELCAR  ;ASK USER FOR THE CHARACTER
2691                                     ;HE WISHES TO TRANSFER
2692 006764 104412          RDOCT                ;ACCEPT THE ANSWER TYPED BY USER
2693                                     ;AND STORE ON TOP OF STACK
2694 006766 012667 172212          MOV   (SP)+,$TMP1    ;GET THE ANSWER TYPED
2695                                     ;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
2696                                     ;OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. B=102
2697 006772 104401 021271          TYPE  ,SELDLY        ;ASK THE USER FOR THE DELAY
2698                                     ;IN MSEC (OCTAL NO.) BETWEEN
2699                                     ;CHARACTER TRANSFERS
2700 006776 104412          RDOCT                ;ACCEPT THE ANSWER TYPED BY
2701                                     ;USER AND STORE ON TOP OF STACK
2702 007000 012667 172202          MOV   (SP)+,$TMP2    ;GET THE ANSWER TYPED
2703 007004 162702 000002          SUB   #2,R2          ;GET THE CORRESPONDING XCSR
2704                                     ;ADDRESS FOR TRANSMITTER UNDER-
2705                                     ;GOING TEST
2706 007010 052712 000004 1$:      BIS   #BIT2,(R2)     ;SET MAINTENANCE BIT IN XCSR
2707 007014 116767 172166 000012          MOVB  $TMP2,2$       ;SET THE DELAY COUNT ARGUMENT
2708                                     ;FOR TIMER ROUTINE
2709 007022 116777 172156 172152          MOVB  $TMP1,2$TMP0   ;LOAD THE TRANSMITTER DATA BUFFER
2710                                     ;WITH THE CHARACTER
2711 007030 004767 005252          JSR   PC,DELAY        ;GO OFF TO WAIT THE SPECIFIED
2712                                     ;NO. OF MSEC. BEFORE ISSUING
2713                                     ;ANOTHER CHARACTER
2714 007034 000000 2$:      .WORD 0              ;THIS IS WHERE THE DELAY COUNT RESIDES
2715 007036 000764          BR    1$              ;GO BACK TO ISSUE ANOTHER CHARACTER

```

2716 007040 104401 001252
 2717 007044 000167 177570
 2718
 2719
 2720
 2721
 2722
 2723
 2724
 2725
 2726
 2727 007050 000240
 2728 007052 104401 017570
 2729
 2730 007056 104401 021102
 2731
 2732
 2733 007062 104412
 2734
 2735
 2736 007064 012602
 2737 007066 020227 176176
 2738 007072 101136
 2739 007074 020227 175616
 2740 007100 103533
 2741 007102 132702 000001
 2742
 2743 007106 001130
 2744
 2745 007110 010203
 2746 007112 142703 000370
 2747
 2748 007116 122703 000006
 2749
 2750 007122 001122
 2751 007124 010267 172052
 2752
 2753
 2754 007130 016746 170650
 2755 007134 012767 007146 170642
 2756 007142 005712
 2757
 2758
 2759 007144 000412
 2760 007146 004767 005554
 2761 007152 012767 007162 172064
 2762 007160 104006
 2763 007162 022626
 2764 007164 012667 170614
 2765 007170 000477
 2766 007172 012667 170606
 2767
 2768 007176 104401 021364
 2769
 2770
 2771

```

REDO2: TYPE ,SQUES ;TYPE A QUESTION MARK(?)
      JMP PRG3A ;REITERATE THE XDBR QUESTION TO
                ;USER

;THIS IS PROGRAM #4
;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
;
;A) SELECTION OF A TRANSMITTER DATA BUFFER
;B) SELECTION OF A SINGLE CHARACTER TO BE SENT, RECEIVED
;    AND CHECKED WITH MAINTENANCE BIT SET
;
PRG4:  NOP ;INDICATE THAT USER SELECTED
      TYPE ,PROG4M ;PROGRAM #4
PRG4A: 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 REDO3 ;IF YES - GO TO RETRY SITUATION
      CMP R2,#175616 ;IS THE NUMBER TOO LOW?
      BLO REDO3 ;IF YES - GO TO RETRY SITUATION
      BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
                ;ON AN EVEN BOUNDARY?
      BNE REDO3 ;IF NO - GO TO RETRY SITUATION
;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR 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 REDO3 ;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$: JSR PC,SUERT2 ;GO SET UP FOR ERROR INFORMATION
    MOV #3$, $ESCAPE ;POINT OF RETURN AFTER ERROR REPORT
    ERROR +6 ;XDBR REFERENCE CAUSED TIMEOUT
3$: CMP (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
    MOV (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
    BR REDO3 ;GO TO RETRY SITUATION
4$: MOV (SP)+,ERRVEC ;DEVICE REGISTER IS PRESENT!
    ;RESTORE TIMEOUT VECTOR
    TYPE ,RSTALL ;ASK THE USER IF HE DESIRES SOME
                ;RANDOM NO. OF MSEC. WAIT TIME
                ;BEFORE CHECKING FOR XCSR DONE
                ;FLAG
  
```

```

2772 007202 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
2773 ;AND STORE ON TOP OF STACK
2774 007204 012667 171776 MOV (SP)+,$TMP2 ;GET THE ANSWER TYPED
2775 ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED
2776 007210 104401 021163 PRG4B: TYPE ,SELCAR ;ASK USER FOR THE CHARACTER HE
2777 ;WISHES TO TRANSFER
2778 007214 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
2779 ;AND STORE ON TOP OF STACK
2780 007216 012667 171762 MOV (SP)+,$TMP1 ;GET THE ANSWER TYPED
2781 ;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE OCTAL
2782 ; ASCII EQUIVALENT OF THE CHARACTER E.G. C=103
2783 ;
2784 007222 104401 020013 PRG4C: TYPE, LENGTH ;ASK USER FOR THE CHARACTER LENGTH
2785 ;FOR WHICH HIS DEVICE IS SET
2786 007226 104413 RDDEC ;ACCEPT THE ANSWER TYPED BY USER
2787 ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2788 007230 012600 MOV (SP)+,$RO ;GET THE ANSWER TYPED
2789 007232 020027 000010 CMP RO,#8. ;IS THE NUMBER TOO HIGH?
2790 007236 101060 BHI RED03A ;IF YES - GO TO RETRY SITUATION
2791 007240 020027 000005 CMP RO,#5. ;IS THE NUMBER TOO LOW?
2792 007244 103455 BLO RED03A ;IF YES - GO TO RETRY SITUATION
2793 007246 010067 171762 MOV RO,$TMP15 ;THE VALUE TYPED IS OK
2794 ;STORE FOR FUTURE USE
2795 007252 016767 171724 171730 MOV $TMP0,$TMP3 ;GET THE XDBR ADDRESS
2796 007260 162767 000002 171722 SUB #2,$TMP3 ;FORM THE XCSR ADDRESS
2797 007266 005767 171714 1$: TST $TMP2 ;DO WE RANDOM STALL?
2798 007272 001402 BEQ 2$ ;BRANCH IF IT WASN'T DESIRED
2799 007274 004767 005052 JSR PC,STALL ;GO STALL RANDOM VALUE OF MSEC.
2800 007300 004767 005156 2$: JSR PC,TIMETX ;GO WAIT FOR TRANSMITTER DONE
2801 ;BIT TO SET
2802 007304 104401 017700 TYPE ,XDB ;TYPE TRANSMITTER DONE BIT MESSAGE
2803 007310 104000 ERROR +0 ;XCSR DONE BIT NEVER SET
2804 007312 052777 000004 171670 BIS #BIT2,$TMP3 ;SET THE MAINTENANCE BIT IN THE
2805 ;TRANSMITTER CONTROL STATUS REGISTER
2806 007320 016777 171660 171654 MOV $TMP1,$TMP0 ;LOAD TRANSMITTER DATA BUFFER
2807 ;WITH SELECTED CHARACTER
2808 007326 004767 005112 JSR PC,TIMERX ;GO WAIT FOR RECEIVER DONE BIT
2809 ;TO SET
2810 007332 104401 017747 TYPE ,RDB ;TYPE RECEIVER DONE BIT MESSAGE
2811 007336 104000 ERROR +0 ;RCSR DONE BIT NEVER SET
2812 007340 016767 171644 171644 MOV $TMP3,$TMP4 ;GET THE TRANSMITTER CONTROL
2813 ;STATUS REGISTER ADDRESS
2814 007346 162767 000002 171636 SUB #2,$TMP4 ;FORM THE RECEIVER DATA BUFFER
2815 ;ADDRESS
2816 007354 017767 171632 171632 MOV $TMP4,$TMP5 ;STORE THE CHARACTER FROM THE
2817 ;RECEIVER BUFFER - REST OF CONTENTS
2818 007362 004767 005132 JSR PC,DATCHK ;GO TO COMPARE EXPECTED & RECEIVED
2819 ;DATA
2820 007366 000737 BR 1$ ;GO BACK TO ISSUE ANOTHER CHARACTER
2821 007370 104401 001252 RED03: TYPE ,SQUES ;TYPE A QUESTION MARK(?)
2822 007374 000167 177456 JMP PRG4A ;REITERATE THE XDBR QUESTION TO USER
2823 007400 104401 001252 RED03A: TYPE, SQUES ;TYPE '?' INDICATING USER TYPED
2824 ;SOMETHING WRONG FOR CHARACTER LENGTH
2825 007404 000167 177612 JMP PRG4C ;GO BACK TO REISSUE QUESTION
2826
2827 ;THIS IS PROGRAM #5

```

C 5

```

2828 ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
2829 ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
2830 ;
2831 ;
2832 007410 000240 PRG5: NOP
2833 007412 104401 017634 TYPE ,PROG5M ;INDICATE THAT USER SELECTED
2834 ;PROGRAM #5
2835 007416 104401 021102 PRG5A: TYPE ,LINTAD ;ASK USER FOR THE TRANSMITTER DATA
2836 ;BUFFER ADDRESS OF THE DEVICE
2837 ;HE WISHES TO TEST
2838 007422 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
2839 ;AND STORE ON TOP OF STACK
2840 ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
2841 007424 012602 MOV (SP)+,R2 ;GET THE ANSWER TYPED
2842 007426 020227 176176 CMP R2,#176176 ;IS THE NUMBER TOO HIGH?
2843 007432 101152 BHI REDO4 ;IF YES - GO TO RETRY SITUATION
2844 007434 020227 175616 CMP R2,#175616 ;IS THE NUMBER TOO LOW?
2845 007440 103547 BLO REDO4 ;IF YES - GO TO RETRY SITUATION
2846 007442 132702 000001 BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
2847 ;ON AN EVEN BOUNDARY?
2848 007446 001144 BNE REDO4 ;IF NOT - GO TO RETRY SITUATION
2849 ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
2850 007450 010203 MOV R2,R3 ;GET THE USER RESPONSE
2851 007452 142703 000370 BICB #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
2852 ;LEAST SIGNIFICANT DIGIT
2853 007456 122703 000006 CMPB #6,R3 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2854 ;USER RESPONSE EQUAL TO A SIX?
2855 007462 001136 BNE REDO4 ;BRANCH IF NOT
2856 007464 010267 171512 MOV R2,$TMP0 ;THE TRANSMITTER ADDRESS TYPED
2857 ;IS OK - STORE FOR FUTURE USE
2858 ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2859 007470 016746 170310 MOV ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
2860 007474 012767 007506 170302 MOV #2$,ERRVEC ;SET UP TIMEOUT SERVICE ADDRESS
2861 007502 005712 TST (R2) ;IF PRESENT WE WILL EXECUTE THE
2862 ;NEXT INSTRUCTION - IF NOT WE
2863 ;GO TO 2$:
2864 007504 000412 BR 4$ ;BRANCH IF PRESENT
2865 007506 004767 005214 JSR PC,SUERT2 ;GO SETUP FOR ERROR INFORMATION
2866 007512 012767 007522 171524 MOV #3$,SESCAPE ;POINT OF RETURN AFTER ERROR REPORT
2867 007520 104006 ERROR +6 ;XCSR REFERENCE CAUSED TIMEOUT
2868 007522 022626 3$: CMP (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
2869 007524 012667 170254 MOV (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
2870 007530 000513 BR REDO4 ;GO TO RETRY SITUATION
2871 007532 012667 170246 4$: MOV (SP)+,ERRVEC ;DEVICE REGISTER IS PRESENT!
2872 ;ASK THE USER IF HE DESIRES SOME
2873 ;RANDOM NO. OF MSEC. WAIT TIME
2874 ;BEFORE CHECKING XCSR DONE FLAG
2875 007536 104401 020013 PRG5C: TYPE, LENGTH ;ASK USER FOR THE CHARACTER LENGTH
2876 ;FOR WHICH HIS DEVICE IS SET
2877 007542 104413 RDDEC ;ACCEPT THE ANSWER TYPED BY USER
2878 ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2879 007544 012600 MOV (SP)+,R0 ;GET THE ANSWER TYPED
2880 007546 020027 000010 CMP R0,#8. ;IS THE NUMBER TOO HIGH?
2881 007552 101106 BHI REDO4A ;IF YES - GO TO RETRY SITUATION
2882 007554 020027 000005 CMP R0,#5. ;IS THE NUMBER TOO LOW?
2883 007560 103503 BLO REDO4A ;IF YES - GO TO RETRY SITUATION

```

```

2884 007562 010067 171446      MOV      R0,$TMP15      ;THE VALUE TYPED IS OK
2885                                ;STORE FOR FUTURE USE
2886 007566 104401 021364      TYPE     ,RSTALL       ;RANDOM NO. OF MSEC. WAIT TIME
2887 007572 104412                RDOCT                ;ACCEPT THE ANSWER TYPED BY USER
2888                                ;AND STORE ON TOP OF STACK
2889 007574 012667 171406      MOV      (SP)+,$TMP2    ;GET THE ANSWER TYPED
2890                                ;WE ARE NOW READY TO INITIALIZE THE BINARY COUNT AND GET
2891                                ;THE BINARY CHARACTER
2892                                ;
2893 007600 012767 177777 171416      MOV      #-1,$TMP11     ;SET LEAD IN VARIABLE TO -1
2894 007606 016767 171412 171412 PRG5B: MOV      $TMP11,$TMP12 ;STORE PREVIOUS BINARY CHARACTER
2895 007614 005267 171406      INC      $TMP12         ;FLIP BINARY CHARACTER AGAIN
2896 007620 042767 177400 171400      BIC      #177400,$TMP12 ;MASK TO 8 BITS
2897 007626 016767 171374 171370      MOV      $TMP12,$TMP11 ;STORE BINARY CHARACTER
2898 007634 016767 171366 171342      MOV      $TMP12,$TMP1  ;STORE BINARY CHARACTER
2899                                ;FOR FUTURE USE
2900 007642 016767 171334 171340      MOV      $TMP0,$TMP3   ;GET THE XDBR ADDRESS
2901 007650 162767 000002 171332      SUB      #2,$TMP3      ;FORM THE XCSR ADDRESS
2902 007656 005767 171324      1$: TST      $TMP2      ;DO WE RANDOM STALL?
2903 007662 001402                BEQ      2$            ;BRANCH IF IT WASN'T DESIRED
2904 007664 004767 004462      JSR      PC,STALL      ;GO STALL RANDOM VALUE OF MSEC.
2905 007670 004767 004566      2$: JSR      PC,TIMETX   ;GO WAIT FOR TRANSMITTER DONE
2906                                ;BIT TO SET
2907 007674 104401 017700      TYPE     ,XDB          ;TYPE TRANSMITTER DONE BIT MESSAGE
2908 007700 104000                ERROR    +0           ;XCSR DONE BIT NEVER SET
2909 007702 052777 000004 171300      BIS      #BIT2,@$TMP3  ;SET THE MAINTENANCE BIT IN THE
2910                                ;TRANSMITTER CONTROL STATUS REGISTER
2911 007710 016777 171270 171264      MOV      $TMP1,@$TMP0  ;LOAD TRANSMITTER DATA BUFFER
2912                                ;WITH SELECTED CHARACTER
2913 007716 004767 004522      JSR      PC,TIMERX     ;GO WAIT FOR RECEIVER DONE BIT TO SET
2914 007722 104401 017747      TYPE     ,RDB          ;TYPE RECEIVER DONE BIT MESSAGE
2915 007726 104000                ERROR    +0           ;RCSR DONE BIT NEVER SET
2916 007730 016767 171254 171254      MOV      $TMP3,$TMP4   ;GET THE TRANSMITTER CONTROL
2917                                ;STATUS REGISTER ADDRESS
2918 007736 162767 000002 171246      SUB      #2,$TMP4      ;FORM THE RECEIVER DATA BUFFER
2919                                ;ADDRESS
2920 007744 017767 171242 171242      MOV      @$TMP4,$TMP5  ;STORE THE CHARACTER FROM THE
2921                                ;RECEIVER BUFFER + REST OF CONTENTS
2922 007752 004767 004542      JSR      PC,DATCHK     ;GO TO COMPARE EXPECTED & RECEIVED
2923                                ;DATA
2924 007756 000713                BR       PRG5B         ;GO BACK TO ISSUE ANOTHER BINARY
2925                                ;CHARACTER
2926 007760 104401 001252      RED04: TYPE     ,QUES   ;TYPE A QUESTION MARK(?)
2927 007764 000167 177426      JMP      PRG5A         ;GO BACK TO REITERATE XDBR QUESTION
2928 007770 104401 001252      RED04A: TYPE     ,QUES  ;TYPE '?' INDICATING USER TYPED
2929                                ;SOMETHING WRONG FOR CHARACTER LENGTH
2930 007774 000167 177536      JMP      PRG5C         ;GO BACK TO REISSUE QUESTION
2931
2932                                ;THIS ROUTINE WILL SET UP:
2933                                ; RCSR - RECEIVER STATUS REGISTER
2934                                ; RBUF - RECEIVER BUFFER REGISTER
2935                                ; XCSR - TRANSMITTER STATUS REGISTER
2936                                ; XBUF - TRANSMITTER BUFFER REGISTER
2937                                ;INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS
2938                                ;TESTED, AND THEREAFTER, AT THE END OF A PROGRAM TO CYCLE THRU
2939                                ;ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED)

```

```

2940 010000 016767 171254 171402 DLADDR: MOV DLBASE,DLRCSR :STORE RECEIVER STATUS REGISTER
2941 :OF CURRENT DEVICE
2942 010006 062767 000002 171244 ADD #2,DLBASE :FORM RECEIVER BUFFER REGISTER
2943 :OF CURRENT DEVICE
2944 010014 016767 171240 171370 MOV DLBASE,DLRDBR :STORE RECEIVER BUFFER REGISTER
2945 :OF CURRENT DEVICE
2946 010022 062767 000002 171230 ADD #2,DLBASE :FORM TRANSMITTER STATUS REGISTER
2947 :OF OF CURRENT DEVICE
2948 010030 016767 171224 171356 MOV DLBASE,DLXCSR :STORE TRANSMITTER STATUS REGISTER
2949 :OF CURRENT DEVICE
2950 010036 062767 000002 171214 ADD #2,DLBASE :FOR TRANSMITTER BUFFER REGISTER
2951 :OF CURRENT DEVICE
2952 010044 016767 171210 171344 MOV DLBASE,DLXDBR :STORE TRANSMITTER BUFFER REGISTER
2953 :OF CURRENT DEVICE
2954 010052 000207 RTS PC :RETURN
2955
2956 .SBTTL END OF PASS ROUTINE
2957
2958 ::*****
2959 :*INCREMENT THE PASS NUMBER ($PASS)
2960 :*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
2961 :*IF THERES A MONITOR GO TO IT
2962 :*IF THERE ISN'T JUMP TO RESTRT
2963
2964 010054 $EOP:
2965 :THIS NEXT SECTION UP TO THE NEXT LINE OF ASTERISKS WAS
2966 :SUPPLIED BY THE MACRO 'EOPBEG'. THE MACRO NAME APPEARS IN
2967 :THE SOURCE PROGRAM AS ONE OF THE ARGUMENTS TO THE .SEOP
2968 :SYSMAC UTILITY ROUTINE CALL.
2969 010054 000004 SCOPE
2970 010056 105767 171210 TSTB MULTD :ARE WE RUNNING MULTIPLE DEVICES?
2971 010062 001501 BEQ 3$ :BRANCH IF NOT FOR NORMAL
2972 :'END PASS # XX' TYPEOUT
2973 010064 005767 171204 TST ACTREG :ARE ANY DEVICES ACTIVE?
2974 010070 001011 BNE 1$ :BRANCH IF YES
2975 010072 104401 020670 TYPE ,FOULUP :INDICATE SOMETHING WRONG!
2976 :MULTIPLE DEVICES ARE BEING
2977 :RUN SUPPOSEDLY, BUT NONE ARE
2978 :SHOWN ACTIVE
2979 010076 000000 HALT :WAIT FOR A USER RESPONSE
2980 010100 005067 171166 CLR MULTD :CLEAR MULTIPLE DEVICE FLAG
2981 010104 005067 171146 CLR TABFLG :CLEAR TABLE CREATION FLAG
2982 010110 000167 171722 JMP RESTRT :GET READY TO START ALL OVER
2983 :AGAIN - ALL DEVICES WERE
2984 :Deselected somehow!
2985 010114 062767 000010 171142 1$: ADD #10,BASEADD :FORM A NEW BASE
2986 :ADDRESS FOR START OF NEXT BLOCK
2987 :OF REGISTERS FOR NEXT DEVICE
2988 010122 062767 000010 171140 ADD #10,BASEIV :FORM A NEW BASE ADDRESS FOR
2989 :START OF NEXT BLOCK OF VECTORS
2990 :FOR NEXT DEVICE
2991 010130 000241 CLC :CLEAR LAST DEVICE INDICATOR
2992 010132 006167 171140 ROL ROTADD :UPDATE NEXT POSSIBLE DEVICE ACTIVE
2993 :POINTER
2994 010136 103431 BCS 2$ :BRANCH IF THIS WAS THE
2995 :LAST DEVICE TO BE TESTED ON
  
```



```

2996                                     :THIS PASS
2997 010140 036767 171132 171126      BIT   ROTADD,ACTREG  :IS THIS DEVICE TRULY ACTIVE
2998                                     : (AS PER USER)
2999 010146 001762                      BEQ   1$           :BRANCH IF NOT TO SEE IF NEXT
3000                                     :ONE POSSIBLE IS
3001 010150 016767 171110 171102      MOV   BASEADD,DLBASE :FORM THE RECEIVER STATUS REGISTER
3002                                     :ADDRESS OF NEXT DEVICE
3003 010156 016767 171106 171234      MOV   BASEIV,DLVECT  :GET NEXT DEVICE RCVR VECTOR
3004 010164 000240                      NOP
3005 010166 004767 177606              JSR   PC,DLADDR     :GO FORM DL ADDRESSES FOR NEXT
3006                                     :DEVICE SELECTED
3007 010172 005067 170704              CLR   $TSTNM        :INITIALIZE TEST NO. FOR A PROGRAM
3008                                     :PASS OVER THE NEXT DEVICE ACTIVE
3009 010176 005077 171212              CLR   @DLXCSR       :CLEAR OUT BOTH CSR'S
3010 010202 005077 171202              CLR   @DLRCSR
3011 010206 005777 171200              TST   @DLRDBR
3012 010212 005777 171174              TST   @DLRDBR
3013 010216 000167 172612              JMP   TST1          :START TESTING THIS DEVICE
3014 010222
3015 2$:
3016 :IF WE TAKE THIS PATH WE HAVE MADE A CYCLE THRU THE PROGRAM ONCE
3017 :PER DEVICE I.E. - WE HAVE MADE A COMPLETE PASS
3018 :NOW WE NEED TO RESTORE EVERYTHING FOR THE NEXT COMPLETE PASS
3019 010222 012767 000001 171046      MOV   #1,ROTADD     :SET UP ROTATING POINTER FOR NEXT
3020                                     :MULTIPLE PASS
3021 010230 016767 171026 171026      MOV   KEEPADD,BASEADD :RESTORE BASE ADDRESS
3022 010236 016767 171024 171024      MOV   KEEPIV,BASEIV  :RESTORE BASE INTERRUPT VECTOR
3023 010244 016767 171014 171006      MOV   BASEADD,DLBASE :RESTORE 1ST DEVICE BASE ADDRESS
3024 010252 016767 171012 171140      MOV   BASEIV,DLVECT  :RESTORE 1ST DEVICE VECTOR ADDRESS
3025 010260 000240                      NOP
3026 010262 004767 177512              JSR   PC,DLADDR     :FORM ADDRESSES FOR 1ST DEVICE
3027 3$:
3028 :*****
3029 010266 005067 170610              CLR   $TSTNM        ;; ZERO THE TEST NUMBER
3030 010272 005067 170744              CLR   $TIMES        ;; ZERO THE NUMBER OF ITERATIONS
3031 010276 005267 170576              INC   $PASS         ;; INCREMENT THE PASS NUMBER
3032 010302 042767 100000 170570      BIC   #100000,$PASS :DON'T ALLOW A NEG. NUMBER
3033 010310 005327                      DEC   (PC)+         ;; LOOP?
3034 010312 000001                      SEOPCT: .WORD 1
3035 010314 003022                      BGT   $DOAGN        ;; YES
3036 010316 012737                      MOV   (PC)+,@(PC)+  ;; RESTORE COUNTER
3037 010320 000001                      SENDCT: .WORD 1
3038 010322 010312                      SEOPCT
3039 010324 104401 010371              TYPE  $SENDMG       ;; TYPE 'END PASS #'
3040 010330 016746 170544              MOV   $PASS,-(SP)  ;; SAVE $PASS FOR TYPEOUT
3041 010334 104405                      TYPDS              ;; GO TYPE--DECIMAL ASCII WITH SIGN
3042 010336 104401 010366              TYPE  $SENULL       ;; TYPE A NULL CHARACTER
3043 010342 013700 000042              $GET42: MOV   @#42,R0  ;; GET MONITOR ADDRESS
3044 010346 001405                      BEQ   $DOAGN        ;; BRANCH IF NO MONITOR
3045 010350 000005                      RESET              ;; CLEAR THE WORLD
3046 010352 004710                      SENDAD: JSR   PC,(R0) ;; GO TO MONITOR
3047 010354 000240                      NOP                ;; SAVE ROOM
3048 010356 000240                      NOP                ;; FOR
3049 010360 000240                      NOP                ;; ACT11
3050 010362 000137                      $DOAGN: JMP   @(PC)+   ;; RETURN
3051 010364 002036                      $RTNAD: .WORD  RESTRT

```

3052	010366	377	377	000
3053	010371	015	042412	042116
3054	010376	050040	051501	020123
3055	010404	000043		
3056				
3057				
3058				
3059				
3060				
3061				
3062				
3063				
3064				
3065				
3066				
3067				
3068				
3069				
3070	010406			
3071	010406	104407		
3072	010410	032777	040000	170522
3073	010416	001111		
3074				
3075	010420	000416		
3076				
3077	010422	013746	000004	
3078	010426	012737	010446	000004
3079	010434	005737	177060	
3080	010440	012637	000004	
3081	010444	000463		
3082	010446	022626		
3083	010450	012637	000004	
3084	010454	000423		
3085	010456			
3086	010456	032777	000400	170454
3087	010464	001404		
3088	010466	127767	170446	170406
3089	010474	001462		
3090	010476	105767	170401	
3091	010502	001421		
3092	010504	126767	170405	170371
3093	010512	101015		
3094	010514	032777	001000	170416
3095	010522	001404		
3096	010524	016767	170360	170354
3097	010532	000443		
3098	010534	105067	170343	
3099	010540	005067	170476	
3100	010544	000415		
3101	010546	032777	004000	170364
3102	010554	001011		
3103	010556	005767	170316	
3104	010562	001406		
3105	010564	005267	170314	
3106	010570	026767	170446	170306
3107	010576	002021		

```

$ENULL: .BYTE -1,-1,0          ;;NULL CHARACTER STRING
$ENDMG: .ASCIZ <15><12>/END PASS #/

.SBTTL SCOPE HANDLER ROUTINE

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

$SCOPE:
1$:      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
        BIT          #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
        BNE          $OVER          ;;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR:  BR          6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
        ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
        MOV          @ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
        MOV          #5$,@ERRVEC          ;;SET FOR TIMEOUT
        TST          @#177060          ;;TIME OUT ON XOR?
        MOV          (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
        BR          $$VLAD          ;;GO TO THE NEXT TEST
5$:      CMP          (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
        MOV          (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
        BR          7$          ;;LOOP ON THE PRESENT TEST
6$:      *****END OF CODE FOR THE XOR TESTER*****
        BIT          #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
        BEQ          2$          ;;BR IF NO
        CMPB         @SWR,$STNM          ;;ON THE RIGHT TEST? SWR<7:0>
        BEQ          $OVER          ;;BR IF YES
2$:      TSTB         $ERFLG          ;;HAS AN ERROR OCCURRED?
        BEQ          3$          ;;BR IF NO
        CMPB         $ERMAX,$ERFLG          ;;MAX. ERRORS FOR THIS TEST OCCURRED?
        BHI          3$          ;;BR IF NO
        BIT          #BIT09,@SWR          ;;LOOP ON ERROR?
        BEQ          4$          ;;BR IF NO
7$:      MOV          $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
        BR          $OVER
4$:      CLRB         $ERFLG          ;;ZERO THE ERROR FLAG
        CLR          $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
        BR          1$          ;;ESCAPE TO THE NEXT TEST
3$:      BIT          #BIT11,@SWR          ;;INHIBIT ITERATIONS?
        BNE          1$          ;;BR IF YES
        TST          $PASS          ;;IF FIRST PASS OF PROGRAM
        BEQ          1$          ;;INHIBIT ITERATIONS
        INC          $ICNT          ;;INCREMENT ITERATION COUNT
        CMP          $TIMES,$ICNT          ;;CHECK THE NUMBER OF ITERATIONS MADE
        BGE          $OVER          ;;BR IF MORE ITERATION REQUIRED

```

```
3108 010600 012767 000001 170276 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
3109 010606 016767 000044 170426 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
3110 010614 105267 170262 $$VLAD: INCB $STNM ;;COUNT TEST NUMBERS
3111 010620 011667 170262 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
3112 010624 011667 170260 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
3113 010630 005067 170410 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3114 010634 112767 000001 170253 MOV #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3115 010642 016777 170234 170272 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
3116 010650 016716 170232 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
3117 010654 000002 RTI ;;FIXES PS
3118 010656 000100 $MXCNT: 100 ;;MAX. NUMBER OF ITERATIONS
3119
3120 .SBTTL ERROR HANDLER ROUTINE
3121
3122 ;;*****
3123 ;;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
3124 ;;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
3125 ;;*AND GO TO $ERRTYP ON ERROR
3126 ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3127 ;;*SW15=1 HALT ON ERROR
3128 ;;*SW13=1 INHIBIT ERROR TYPEOUTS
3129 ;;*SW10=1 BELL ON ERROR
3130 ;;*SW09=1 LOOP ON ERROR
3131 ;;*CALL
3132 ;;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
3133
3134 $ERROR:
3135 010660 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3136 010662 105267 170215 7$: INCB $ERFLG ;;SET THE ERROR FLAG
3137 010666 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
3138 010670 016777 170206 170244 MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
3139 010676 032777 002000 170234 BIT #BIT10,@SWR ;;BELL ON ERROR?
3140 010704 001402 BEQ 1$ ;;NO - SKIP
3141 010706 104401 001246 TYPE ,SBELL ;;RING BELL
3142 010712 005267 170174 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
3143 010716 011667 170174 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
3144 010722 162767 000002 170166 SUB #2,$ERRPC
3145 010730 117767 170162 170156 MOV @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
3146 010736 032777 020000 170174 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
3147 010744 001004 BNE 20$ ;;SKIP TYPEOUTS
3148 010746 004767 000046 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
3149 010752 104401 001253 TYPE ,$CRLF
3150 010756 20$:
3151 010756 005777 170156 2$: TST @SWR ;;HALT ON ERROR
3152 010762 100002 BPL 3$ ;;SKIP IF CONTINUE
3153 010764 000000 HALT ;;HALT ON ERROR!
3154 010766 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3155 010770 032777 001000 170142 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
3156 010776 001402 BEQ 4$ ;;BR IF NO
3157 011000 016716 170104 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
3158 011004 005767 170234 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
3159 011010 001402 BEQ 5$ ;;BR IF NONE
3160 011012 016716 170226 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
3161 011016 5$:
3162 011016 000002 RTI ;;RETURN
3163
```

3164
3165
3166
3167
3168
3169
3170
3171 011020
3172 011020 104401 001253
3173 011024 010046
3174 011026 005000
3175 011030 153700 001114
3176 011034 001004
3177
3178 011036 016746 170054
3179
3180 011042 104402
3181 011044 000426
3182 011046 005300
3183 011050 006300
3184 011052 006300
3185 011054 006300
3186 011056 062700 001310
3187 011062 012067 000004
3188 011066 001404
3189 011070 104401
3190 011072 000000
3191 011074 104401 001253
3192 011100 012067 000004
3193 011104 001404
3194 011106 104401
3195 011110 000000
3196 011112 104401 001253
3197 011116 011000
3198 011120 001004
3199 011122 012600
3200 011124 104401 001253
3201 011130 000207
3202 011132
3203 011132 013046
3204 011134 104402
3205 011136 005710
3206 011140 001770
3207 011142 104401 011150
3208 011146 000771
3209 011150 020040 000
3210 011154

.SBTTL ERROR MESSAGE TIMEOUT ROUTINE

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

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

.SBTTL 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.
*$TYPOS--ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)    ::NUMBER TO BE TYPED
```

3211
3212
3213
3214
3215
3216
3217
3218
3219

```
3220      *      TYPOS      ::CALL FOR TYPEOUT
3221      *      .BYTE      N      ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3222      *      .BYTE      M      ::M=1 OR 0
3223      *      *      *      ;1=TYPE LEADING ZEROS
3224      *      *      *      ;0=SUPPRESS LEADING ZEROS
3225      *
3226      *$TYPON-----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3227      *$TYPOS OR $TYPOC
3228      *CALL:
3229      *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
3230      *      TYPON      ::CALL FOR TYPEOUT
3231      *
3232      *$TYPOC-----ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3233      *CALL:
3234      *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
3235      *      TYPOC      ::CALL FOR TYPEOUT
3236
3237      011154 017646 000007 000211 $TYPOS: MOV      @ (SP),-(SP)      ::PICKUP THE MODE
3238      011160 116667 000001 000211 MOVB     1(SP), $OFILL      ::LOAD ZERO FILL SWITCH
3239      011166 112667 000207 000211 MOVB     (SP)+, $OMODE+1    ::NUMBER OF DIGITS TO TYPE
3240      011172 062716 000002 000211 ADD      #2, (SP)      ::ADJUST RETURN ADDRESS
3241      011176 000406 000000 000211 BR       $TYPON
3242      011200 112767 000001 000171 $TYPOC: MOVB     #1, $OFILL      ::SET THE ZERO FILL SWITCH
3243      011206 112767 000006 000165 MOVB     #6, $OMODE+1    ::SET FOR SIX(6) DIGITS
3244      011214 112767 000005 000154 $TYPON: MOVB     #5, $OCNT      ::SET THE ITERATION COUNT
3245      011222 010346 000000 000154 MOV      R3,-(SP)      ::SAVE R3
3246      011224 010446 000000 000154 MOV      R4,-(SP)      ::SAVE R4
3247      011226 010546 000000 000154 MOV      R5,-(SP)      ::SAVE R5
3248      011230 116704 000145 000154 MOVB     $OMODE+1, R4    ::GET THE NUMBER OF DIGITS TO TYPE
3249      011234 005404 000000 000154 NEG      R4
3250      011236 062704 000006 000154 ADD      #6, R4      ::SUBTRACT IT FOR MAX. ALLOWED
3251      011242 110467 000132 000154 MOVB     R4, $OMODE      ::SAVE IT FOR USE
3252      011246 116704 000125 000154 MOVB     $OFILL, R4     ::GET THE ZERO FILL SWITCH
3253      011252 016605 000012 000154 MOV      12(SP), R5    ::PICKUP THE INPUT NUMBER
3254      011256 005003 000000 000154 CLR      R3      ::CLEAR THE OUTPUT WORD
3255      011260 006105 000000 000154 1$: ROL     R5      ::ROTATE MSB INTO 'C'
3256      011262 000404 000000 000154 BR       3$      ::GO DO MSB
3257      011264 006105 000000 000154 2$: ROL     R5      ::FORM THIS DIGIT
3258      011266 006105 000000 000154 ROL     R5
3259      011270 006105 000000 000154 ROL     R5
3260      011272 010503 000000 000154 MOV      R5, R3
3261      011274 006103 000000 000154 3$: ROL     R3      ::GET LSB OF THIS DIGIT
3262      011276 105367 000076 000154 DECB     $OMODE      ::TYPE THIS DIGIT?
3263      011302 100016 000000 000154 BPL     7$      ::BR IF NO
3264      011304 042703 177770 000154 BIC     #177770, R3   ::GET RID OF JUNK
3265      011310 001002 000000 000154 BNE     4$      ::TEST FOR 0
3266      011312 005704 000000 000154 TST     R4      ::SUPPRESS THIS 0?
3267      011314 001403 000000 000154 BEQ     5$      ::BR IF YES
3268      011316 005204 000000 000154 4$: INC     R4      ::DON'T SUPPRESS ANYMORE 0'S
3269      011320 052703 000060 000154 BIS     #'0, R3     ::MAKE THIS DIGIT ASCII
3270      011324 052703 000040 000154 5$: BIS     #' , R3     ::MAKE ASCII IF NOT ALREADY
3271      011330 110367 000040 000154 MOVB     R3, 8$      ::SAVE FOR TYPING
3272      011334 104401 011374 000154 TYPE     8$      ::GO TYPE THIS DIGIT
3273      011340 105367 000032 000154 7$: DECB     $OCNT      ::COUNT BY 1
3274      011344 003347 000000 000154 BGT     2$      ::BR IF MORE TO DO
3275      011346 002402 000000 000154 BLT     6$      ::BR IF DONE
```

```

3276 011350 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
3277 011352 000744          BR       2$          ;;GO DO THE LAST DIGIT
3278 011354 012605          6$:     MOV      (SP)+,R5      ;;RESTORE R5
3279 011356 012604          MOV      (SP)+,R4      ;;RESTORE R4
3280 011360 012603          MOV      (SP)+,R3      ;;RESTORE R3
3281 011362 016666 000002 000004  MOV      2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
3282 011370 012616          MOV      (SP)+,(SP)
3283 011372 000002          RTI          ;;RETURN
3284 011374 000          8$:     .BYTE   0          ;;STORAGE FOR ASCII DIGIT
3285 011375 000          .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
3286 011376 000          $OCNT:  .BYTE   0          ;;OCTAL DIGIT COUNTER
3287 011377 000          $OFILL: .BYTE   0          ;;ZERO FILL SWITCH
3288 011400 000000          $OMODE: .WORD   0          ;;NUMBER OF DIGITS TO TYPE
3289
3290          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3291
3292          ;*****
3293          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3294          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3295          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3296          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3297          ;*REPLACED WITH SPACES.
3298          ;*CALL:
3299          ;*   MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
3300          ;*   TYPDS          ;;GO TO THE ROUTINE
3301
3302          $TYPDS:
3303          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
3304          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
3305          MOV      R2,-(SP)          ;;PUSH R2 ON STACK
3306          MOV      R3,-(SP)          ;;PUSH R3 ON STACK
3307          MOV      R5,-(SP)          ;;PUSH R5 ON STACK
3308          MOV      #20200,-(SP)     ;;SET BLANK SWITCH AND SIGN
3309          MOV      20(SP),R5        ;;GET THE INPUT NUMBER
3310          BPL     1$                ;;BR IF INPUT IS POS.
3311          NEG     R5                ;;MAKE THE BINARY NUMBER POS.
3312          MOVB   #'-,1(SP)         ;;MAKE THE ASCII NUMBER NEG.
3313          CLR     R0                ;;ZERO THE CONSTANTS INDEX
3314          MOV     #SDELK,R3        ;;SETUP THE OUTPUT POINTER
3315          MOVB   #' ,(R5)+         ;;SET THE FIRST CHARACTER TO A BLANK
3316          2$:   CLR     R2          ;;CLEAR THE BCD NUMBER
3317          MOV     $DTBL(R0),R1     ;;GET THE CONSTANT
3318          3$:   SUB     R1,R5       ;;FORM THIS BCD DIGIT
3319          BLT     4$                ;;BR IF DONE
3320          INC     R2                ;;INCREASE THE BCD DIGIT BY 1
3321          BR     3$
3322          4$:   ADD     R1,R5       ;;ADD BACK THE CONSTANT
3323          TST     R2                ;;CHECK IF BCD DIGIT=0
3324          BNE     5$                ;;FALL THROUGH IF 0
3325          TSTB   (SP)              ;;STILL DOING LEADING 0'S?
3326          BMI     7$                ;;BR IF YES
3327          5$:   ASLB   (SP)         ;;MSD?
3328          BCC     6$                ;;BR IF NO
3329          MOVB   1(SP),-1(R3)      ;;YES--SET THE SIGN
3330          6$:   BIS     #'0,R2      ;;MAKE THE BCD DIGIT ASCII
3331          7$:   BIS     #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT

```

```

3332 011522 110223      MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3333 011524 005720      TST    (R0)+        ;;JUST INCREMENTING
3334 011526 020027 000010  CMP    R0,#10       ;;CHECK THE TABLE INDEX
3335 011532 002746      BLT    2$           ;;GO DO THE NEXT DIGIT
3336 011534 003002      BGT    8$           ;;GO TO EXIT
3337 011536 010502      MOV    R5,R2        ;;GET THE LSD
3338 011540 000764      BR     6$           ;;GO CHANGE TO ASCII
3339 011542 105726      8$:   TSTB   (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
3340 011544 100003      BPL    9$           ;;BR IF NO
3341 011546 116663 177777 177776  MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
3342 011554 105013      9$:   CLRB   (R3)      ;;SET THE TERMINATOR
3343 011556 012605      MOV    (SP)+,R5     ;;POP STACK INTO R5
3344 011560 012603      MOV    (SP)+,R3     ;;POP STACK INTO R3
3345 011562 012602      MOV    (SP)+,R2     ;;POP STACK INTO R2
3346 011564 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
3347 011566 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
3348 011570 104401 011616      TYPE   $DBLK        ;;NOW TYPE THE NUMBER
3349 011574 016666 000002 000004  MOV    2(SP),4(SP)  ;;ADJUST THE STACK
3350 011602 012616      MOV    (SP)+,(SP)
3351 011604 000002      RTI                    ;;RETURN TO USER
3352 011606 023420      $DTBL: 10000.
3353 011610 001750      1000.
3354 011612 000144      100.
3355 011614 000012      10.
3356 011616 000004      $DBLK: .BLKW 4
3357
3358      .SBTTL TTY INPUT ROUTINE
3359
3360      ;;*****
3361      .ENABL LSB
3362 011626 000000      $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
3363 011630 000000      $TKQIN: .WORD 0     ;;INPUT POINTER
3364 011632 000000      $TKQOUT: .WORD 0    ;;OUTPUT POINTER
3365 011634 000010      $TKQSRT: .BLKB 8.   ;;TTY KEYBOARD QUEUE
3366      $TKQEND=.
3367
3368      ;*TK INITIALIZE ROUTINE
3369      ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
3370      ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
3371      ;
3372      ;*CALL:
3373      ;*   JSR   PC,$TKINT
3374      ;*   RETURN
3375      ;
3376 011644 005067 177756      $TKINT: CLR    $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
3377 011650 012767 011634 177752  MOV    #$TKQSRT,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
3378 011656 016767 177746 177746  MOV    $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
3379 011664 012737 011714 000060  MOV    #$TKSRV,@TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
3380 011672 012737 000200 000062  MOV    #200,@TKVEC+2  ;;'BR' LEVEL 4
3381 011700 005777 167242      TST    @STKB          ;;CLEAR DONE FLAG
3382 011704 012777 000100 167232  MOV    #100,@STKS     ;;ENABLE TTY KEYBOARD INTERRUPT
3383 011712 000207      RTS    PC              ;;RETURN TO CALLER
3384
3385      ;*TK SERVICE ROUTINE
3386      ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
3387      ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING

```

```

3388
3389
3390
3391
3392 011714 117746 167226
3393 011720 042716 177600
3394 011724 021627 000003
3395 011730 001007
3396 011732 104401 013030
3397 011736 004767 177702
3398 011742 005726
3399 011744 000167 166230
3400 011750 021627 000007
3401 011754 001004
3402 011756 022767 000176 167154
3403 011764 007500
3404
3405 011766
3406 011766 022757 000010 177632
3407 011774 001004
3408 011776 104401 001246
3409 012002 005726
3410 012004 000451
3411 012006 021627 000023
3412 012012 001021
3413 012014 005077 167124
3414 012020 005726
3415 012022 105777 167116
3416 012026 100375
3417 012030 117746 167112
3418 012034 042716 177600
3419 012040 022627 000021
3420 012044 001366
3421 012046 012777 000100 167070
3422 012054 000002
3423 012056 005267 177544
3424 012062 021627 000140
3425 012066 002405
3426 012070 021627 000175
3427 012074 003002
3428 012076 042716 000040
3429 012102 112677 177522
3430 012106 005267 177516
3431 012112 026727 177512 011644
3432 012120 001003
3433 012122 012767 011634 177500
3434 012130 000002
3435
3436
3437
3438
3439
3440
3441 012132 022767 000176 167000
3442 012140 001124
3443 012142 105777 166776

```

```

;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (200)
$TKSRV: MOVB @STKB,-(SP) ;;PICKUP THE CHARACTER
BIC #^C177,(SP) ;;STRIP THE JUNK
CMP (SP),#3 ;;IS IT A CONTROL C?
BNE 1$ ;;BRANCH IF NO
TYPE ,%CNTLC ;;TYPE A CONTROL-C (^C)
JSR PC,$TKINT ;;INIT THE KEYBOARD
TST (SP)+ ;;CLEAN UP STACK
JMP 200 ;;CONTROL C RESTART
1$: CMP (SP),#7 ;;IS IT A CONTROL G?
BNE 2$ ;;BRANCH IF NO
CMP #SWREG,SWR ;;IS SOFT-SWR SELECTED?
BEQ 6$ ;;GO TO SWR CHANGE
2$:
CMP #8,,$TKCNT ;;IS THE QUEUE FULL?
BNE 3$ ;;BRANCH IF NO
TYPE ,%BELL ;;RING THE TTY BELL
TST (SP)+ ;;CLEAN CHARACTER OFF OF STACK
BR 5$ ;;EXIT
3$: CMP (SP),#23 ;;IS IT A CONTROL-S?
BNE 32$ ;;BRANCH IF NO
CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
TST (SP)+ ;;CLEAN CHAR OFF STACK
31$: TSTB @STKS ;;WAIT FOR A CHAR
BPL 31$ ;;LOOP UNTIL ITS THERE
MOVB @STKB,-(SP) ;;GET THE CHARACTER
BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
BNE 31$ ;;BRANCH IF NO
MOV #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
RTI ;;RETURN
32$: INC $TKCNT ;;COUNT THIS CHARACTER
CMP (SP),#140 ;;IS IT UPPER CASE?
BLT 4$ ;;BRANCH IF YES
CMP (SP),#175 ;;IS IT A SPECIAL CHAR?
BGT 4$ ;;BRANCH IF YES
BIC #40,(SP) ;;MAKE IT UPPER CASE
4$: MOVB (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
INC $TKQIN ;;UPDATE THE POINTER
CMP $TKQIN,@STKQEND ;;GO OFF THE END?
BNE 5$ ;;BRANCH IF NO
MOV #STKQSR,$TKQIN ;;RESET THE POINTER
5$: RTI ;;RETURN
;*****
;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
;CALL WHEN OPERATING IN TTY INTERRUPT MODE.
$CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED
BNE 15$ ;;EXIT IF NOT
TSTB @STKS ;;IS A CHAR WAITING?

```



```
3444 012146 100121          BPL      15$          ::IF NOT, EXIT
3445 012150 117746 166772    MOVB     @STKB,-(SP)  ::YES
3446 012154 042716 177600    BIC      #^C177,(SP) ::MAKE IT 7-BIT ASCII
3447 012160 021627 000007    CMP      (SP),#7     ::IS IT A CONTROL-G?
3448 012164 001300          BNE      2$          ::IF NOT, PUT IT IN THE TTY QUEUE
3449                                ::AND EXIT
3450
3451 ::*****
3452 ::*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
3453 ::*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
3454 ::*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
3455 012166 126727 166742 000001 6$:  CMPB     $AUTOB,#1   ::ARE WE RUNNING IN AUTO-MODE?
3456 012174 001674          BEQ      2$          ::BRANCH IF YES
3457 012176 005726          TST      (SP)+       ::CLEAR CONTROL-G OFF STACK
3458 012200 004767 177440    JSR      PC,$TKINT   ::FLUSH THE TTY INPUT QUEUE
3459 012204 005077 166734    CLR      @STKS       ::DISABLE TTY KEYBOARD INTERRUPTS
3460 012210 112767 000001 166717    MOVB     #1,$INTAG   ::SET INTERRUPT MODE INDICATOR
3461
3462 012216 104401 013042          TYPE     ,SCNTLG     ::ECHO THE CONTROL-G (^G)
3463 012222 104401 013047    $GTSWR: TYPE     ,SMSWR     ::TYPE CURRENT CONTENTS
3464 012226 016746 165744    MOV      SWREG,-(SP) ::SAVE SWREG FOR TYPEOUT
3465 012232 104402          TYPOC          ::GO TYPE—OCTAL ASCII(ALL DIGITS)
3466 012234 104401 013060    TYPE     ,SNEW      ::PROMPT FOR NEW SWR
3467 012240 005046          CLR      -(SP)      ::CLEAR COUNTER
3468 012242 005046          CLR      -(SP)      ::THE NEW SWR
3469 012244 105777 166674    7$:  TSTB     @STKS     ::CHAR THERE?
3470 012250 100375          BPL      7$          ::IF NOT TRY AGAIN
3471
3472 012252 117746 166670    MOVB     @STKB,-(SP) ::PICK UP CHAR
3473 012256 042716 177600    BIC      #^C177,(SP) ::MAKE IT 7-BIT ASCII
3474
3475 012262 021627 000003    CMP      (SP),#3     ::IS IT A CONTROL-C?
3476 012266 001015          BNE      9$          ::BRANCH IF NOT
3477 012270 104401 013030    TYPE     ,SCNTLC    ::YES, ECHO CONTROL-C (^C)
3478 012274 062706 000006    ADD      #6,SP       ::CLEAN UP STACK
3479 012300 126727 166631 000001    CMPB     $INTAG,#1   ::REENABLE TTY KEYBOARD INTERRUPTS?
3480 012306 001003          BNE      8$          ::BRANCH IF NO
3481 012310 012777 000100 166626    MOV      #100,@STKS  ::ALLOW TTY KEYBOARD INTERRUPTS
3482 012316 000167 165656    8$:  JMP      200        ::CONTROL-C RESTART
3483
3484
3485 012322 021627 000025    9$:  CMP      (SP),#25   ::IS IT A CONTROL-U?
3486 012326 001005          BNE      10$         ::BRANCH IF NOT
3487 012330 104401 013035    TYPE     ,SCNTLU    ::YES, ECHO CONTROL-U (^U)
3488 012334 062706 000006    20$: ADD      #6,SP       ::IGNORE PREVIOUS INPUT
3489 012340 000737          BR      19$         ::LET'S TRY IT AGAIN
3490
3491
3492 012342 021627 000015    10$: CMP      (SP),#15   ::IS IT A <CR>?
3493 012346 001022          BNE      16$         ::BRANCH IF NO
3494 012350 005766 000004    TST      4(SP)       ::YES, IS IT THE FIRST CHAR?
3495 012354 001403          BEQ      11$         ::BRANCH IF YES
3496 012356 016677 000002 166554    MOV      2(SP),@SWR  ::SAVE NEW SWR
3497 012364 062706 000006    11$: ADD      #6,SP       ::CLEAR UP STACK
3498 012370 104401 001253    14$: TYPE     ,$CRLF   ::ECHO <CR> AND <LF>
3499 012374 126727 166535 000001    CMPB     $INTAG,#1   ::RE-ENABLE TTY KBD INTERRUPTS?
```

```

3500 012402 001003      BNE      15$      ;;BRANCH IF NOT
3501 012404 012777 000100 166532      MOV      #100,@$TKS  ;;RE-ENABLE TTY KBD INTFRRUPTS
3502 012412 000002      RTI          ;;RETURN
3503 012414 004767 000762      JSR      PC,$TYPEC  ;;ECHO CHAR
3504 012420 021627 000060      CMP      (SP),#60   ;;CHAR < 0?
3505 012424 002420      BLT      18$      ;;BRANCH IF YES
3506 012426 021627 000067      CMP      (SP),#67   ;;CHAR > ??
3507 012432 003015      BGT      18$      ;;BRANCH IF YES
3508 012434 042726 000060      BIC      #60,(SP)+  ;;STRIP-OFF ASCII
3509 012440 005766 000002      TST      2(SP)     ;;IS THIS THE FIRST CHAR
3510 012444 001403      BEQ      17$      ;;BRANCH IF YES
3511 012446 006316      ASL      (SP)     ;;NO, SHIFT PRESENT
3512 012450 006316      ASL      (SP)     ;; CHAR OVER TO MAKE
3513 012452 006316      ASL      (SP)     ;; ROOM FOR NEW ONE.
3514 012454 005266 000002 17$:      INC      2(SP)     ;;KEEP COUNT OF CHAR
3515 012460 056616 177776      BIS      -2(SP), (SP) ;;SET IN NEW CHAR
3516 012464 000667      BR       7$       ;;GET THE NEXT ONE
3517 012466 104401 001252 18$:      TYPE    $QUES     ;;TYPE ?<CR><LF>
3518 012472 000720      BR       20$     ;;SIMULATE CONTROL-U
3519      .DSABL  LSB
3520
3521
3522      ;*****
3523      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3524      ;*CALL:
3525      ;*      RDCHR          ;;GET A CHARACTER FROM THE QUEUE
3526      ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
3527      ;*                  ;;WITH PARITY BIT STRIPPED OFF
3528      ;*
3529      ;
3530 012474 011646      SRDCHR: MOV      (SP),-(SP)  ;;PUSH DOWN THE PC AND
3531 012476 016666 000004 000002      MOV      4(SP),2(SP)  ;;THE PS
3532 012504 005066 000004      CLR      4(SP)        ;;GET READY FOR A CHARACTER
3533 012510 005046      CLR      -(SP)       ;;PUT NEW PS ON STACK
3534 012512 012746 012520      MOV      #64$,-(SP)  ;;PUT NEW PC ON STACK
3535 012516 000002      RTI          ;;POP NEW PC AND PS
3536 012520
3537 012520 005767 177102 64$:      TST      $TKCNT     ;;WAIT ON A CHARACTER
3538 012524 001775      BEQ      1$        1$:
3539 012526 005367 177074      DEC      $TKCNT     ;;DECREMENT THE COUNTER
3540 012532 117766 177074 000004      MOVB    @$TKQOUT,4(SP) ;;GET ONE CHARACTER
3541 012540 005267 177066      INC      $TKQOUT    ;;UPDATE THE POINTER
3542 012544 026727 177062 011644      CMP      $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
3543 012552 001003      BNE      2$        2$:
3544 012554 012767 011634 177050      MOV      #$TKQSR,$TKQOUT ;;RESET THE POINTER
3545 012562 000002      RTI          ;;RETURN
3546      ;*****
3547      ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3548      ;*CALL:
3549      ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
3550      ;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3551      ;*                  ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3552      ;*
3553 012564 010346      SRDLIN: MOV      R3,-(SP)  ;;SAVE R3
3554 012566 005046      CLR      -(SP)     ;;CLEAR THE RUBOUT KEY
3555 012570 012703 013020 1$:      MOV      #$TTYIN,R3  ;;GET ADDRESS
    
```

```

3556 012574 022703 013030      2$:  CMP      #STTYIN+8.,R3  ;;BUFFER FULL?
3557 012600 101456                BLOS     4$              ;;BR IF YES
3558 012602 104410                RDCHR                    ;;GO READ ONE CHARACTER FROM THE TTY
3559 012604 112613                MOVBS   (SP)+,(R3)      ;;GET CHARACTER
3560 012606 122713 000177      10$:  MOVBS   #177,(R3)      ;;IS IT A RUBOUT
3561 012612 001022                BNE     5$              ;;BR IF NO
3562 012614 005716                TST     (SP)           ;;IS THIS THE FIRST RUBOUT?
3563 012616 001007                BNE     6$              ;;BR IF NO
3564 012620 112767 000134 000170  MOVBS   #'\",9$         ;;TYPE A BACK SLASH
3565 012626 104401 013016                TYPE    ,9$
3566 012632 012716 177777                MOV     #-1,(SP)       ;;SET THE RUBOUT KEY
3567 012636 005303                6$:  DEC     R3            ;;BACKUP BY ONE
3568 012640 020327 013020                CMP     R3,#STTYIN    ;;STACK EMPTY?
3569 012644 103434                BLO     4$              ;;BR IF YES
3570 012646 111367 000144                MOVBS   (R3),9$       ;;SETUP TO TYPEOUT THE DELETED CHAR.
3571 012652 104401 013016                TYPE    ,9$
3572 012656 000746                BR      2$              ;;GO TYPE
3573 012660 005716                5$:  TST     (SP)           ;;GO READ ANOTHER CHAR.
3574 012662 001406                BEQ     7$              ;;RUBOUT KEY SET?
3575 012664 112767 000134 000124  MOVBS   #'\",9$         ;;BR IF NO
3576 012672 104401 013016                TYPE    ,9$           ;;TYPE A BACK SLASH
3577 012676 005016                CLR     (SP)           ;;CLEAR THE RUBOUT KEY
3578 012700 122713 000025                7$:  CMPS   #25,(R3)       ;;IS CHARACTER A CTRL U?
3579 012704 001003                BNE     8$              ;;BR IF NO
3580 012706 104401 013035                TYPE    ,SCNTLU       ;;TYPE A CONTROL 'U'
3581 012712 000726                BR      1$              ;;GO START OVER
3582 012714 122713 000022                8$:  CMPS   #22,(R3)       ;;IS CHARACTER A '^R'?
3583 012720 001011                BNE     3$              ;;BRANCH IF NO
3584 012722 105013                CLRB   (R3)           ;;CLEAR THE CHARACTER
3585 012724 104401 001253                TYPE    ,SCRLF        ;;TYPE A 'CR' & 'LF'
3586 012730 104401 013020                TYPE    ,STTYIN       ;;TYPE THE INPUT STRING
3587 012734 000717                BR      2$              ;;TYPE THE INPUT STRING
3588 012736 104401 001252                4$:  TYPE    ,SQUES       ;;GO PICKUP ANOTHER CHACTER
3589 012742 000712                BR      1$              ;;TYPE A '?'
3590 012744 111367 000046                3$:  MOVBS   (R3),9$       ;;CLEAR THE BUFFER AND LOOP
3591 012750 104401 013016                TYPE    ,9$           ;;ECHO THE CHARACTER
3592 012754 122723 000015                CMPS   #15,(R3)+      ;;CHECK FOR RETURN
3593 012760 001305                BNE     2$              ;;LOOP IF NOT RETURN
3594 012762 105063 177777                CLRB   -1(R3)         ;;CLEAR RETURN (THE 15)
3595 012766 104401 001254                TYPE    ,SLF          ;;TYPE A LINE FEED
3596 012772 005726                TST     (SP)+         ;;CLEAN RUBOUT KEY FROM THE STACK
3597 012774 012603                MOV     (SP)+,R3      ;;RESTORE R3
3598 012776 011646                MOV     (SP)-,(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3599 013000 016666 000004 000002  MOV     4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
3600 013006 012766 013020 000004  MOV     #STTYIN,4(SP)
3601 013014 000002                RTI                    ;;RETURN
3602 013016 000                9$:  .BYTE  0              ;;STORAGE FOR ASCII CHAR. TO TYPE
3603 013017 000                .BYTE  0              ;;TERMINATOR
3604 013020 000010                $TTYIN: .BLKB 8        ;;RESERVE 8 BYTES FOR TTY INPUT
3605 013030 041536 005015 000  $CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'c'
3606 013035 0136 006525 000012  $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
3607 013042 043536 005015 000  $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
3608 013047 015 051412 051127  $MSWR:  .ASCIZ <15><12>/SWR = /
3609 013054 036440 000040
3610 013060 020040 042516 020127  $MNEW:  .ASCIZ / NEW = /
3611 013066 020075 000

```

```

3612          013072          .EVEN
3613
3614          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
3615
3616          ;;*****
3617          ;;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
3618          ;;*CHANGE IT TO BINARY.
3619          ;;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
3620          ;;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
3621          ;;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
3622          ;;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
3623          ;;*CALL:
3624          ;;*          RDOCT          ;;:READ AN OCTAL NUMBER
3625          ;;*          RETURN HERE  ;;:LOW ORDER BITS ARE ON TOP OF THE STACK
3626          ;;*          ;;:HIGH ORDER BITS ARE IN $HI OCT
3627
3628 013072 011646          $RDOCT: MOV      (SP),-(SP)          ;;:PROVIDE SPACE FOR THE
3629 013074 016666 000004 000002  MOV      4(SP),2(SP)          ;;:INPUT NUMBER
3630 013102 010046          MOV      R0,-(SP)          ;;:PUSH R0 ON STACK
3631 013104 010146          MOV      R1,-(SP)          ;;:PUSH R1 ON STACK
3632 013106 010246          MOV      R2,-(SP)          ;;:PUSH R2 ON STACK
3633 013110 104411          1$:  RDLIN          ;;:READ AN ASCII LINE
3634 013112 012600          MOV      (SP)+,R0          ;;:GET ADDRESS OF 1ST CHARACTER
3635 013114 010067 000100  MOV      R0,$$          ;;:AND SAVE IT
3636 013120 005001          CLR      R1          ;;:CLEAR DATA WORD
3637 013122 005002          CLR      R2
3638 013124 112046          2$:  MOVB     (R0)+,-(SP)          ;;:PICKUP THIS CHARACTER
3639 013126 001420          BEQ      3$          ;;:IF ZERO GET OUT
3640 013130 122716 000060  CMPEB   #'0,(SP)          ;;:MAKE SURE THIS CHARACTER
3641 013134 003026          BGT      4$          ;;:IS AN OCTAL DIGIT
3642 013136 122716 000067  CMPEB   #'7,(SP)
3643 013142 002423          BLT      4$
3644 013144 006301          ASL      R1          ;;:*2
3645 013146 006102          ROL      R2
3646 013150 006301          ASL      R1          ;;:*4
3647 013152 006102          ROL      R2
3648 013154 006301          ASL      R1          ;;:*8
3649 013156 006102          ROL      R2
3650 013160 042716 177770  BIC      #'C7,(SP)          ;;:STRIP THE ASCII JUNK
3651 013164 062601          ADD      (SP)+,R1          ;;:ADD IN THIS DIGIT
3652 013166 00C756          BR       2$          ;;:LOOP
3653 013170 005726          3$:  TST      (SP)+          ;;:CLEAN TERMINATOR FROM STACK
3654 013172 010166 000012  MOV      R1,12(SP)          ;;:SAVE THE RESULT
3655 013176 010267 000026  MOV      R2,$HI OCT
3656 013202 012602          MOV      (SP)+,R2          ;;:POP STACK INTO R2
3657 013204 012601          MOV      (SP)+,R1          ;;:POP STACK INTO R1
3658 013206 012600          MOV      (SP)+,R0          ;;:POP STACK INTO R0
3659 013210 000002          RTI          ;;:RETURN
3660 013212 005726          4$:  TST      (SP)+          ;;:CLEAN PARTIAL FROM STACK
3661 013214 105010          CLR     (R0)          ;;:SET A TERMINATOR
3662 013216 104401          TYPE          ;;:TYPE UP THRU THE BAD CHAR.
3663 013220 000000          5$:  .WORD    0
3664 013222 104401 001252  TYPE    ,SQUES          ;;:'?' 'CR' & 'LF'
3665 013226 000730          BR      1$          ;;:TRY AGAIN
3666 013230 000000          $HI OCT: .WORD    0          ;;:HIGH ORDER BITS GO HERE
3667

```

```

3668 .SBTTL TYPE ROUTINE
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685 013232 105767 165721
3686 013236 100002
3687 013240 000000
3688 013242 000407
3689 013244 010046
3690 013246 017600 000002
3691 013252 112046
3692 013254 001005
3693 013256 005726
3694 013260 012600
3695 013262 062716 000002
3696 013266 000002
3697 013270 122716 000011
3698 013274 001430
3699 013276 122716 000200
3700 013302 001006
3701 013304 005726
3702 013306 104401
3703 013310 001253
3704 013312 105067 000130
3705 013316 000755
3706 013320 004767 000056
3707 013324 126726 165626
3708 013330 001350
3709 013332 016746 165616
3710
3711 013336 105366 000001
3712 013342 002770
3713 013344 004767 000032
3714 013350 105367 000072
3715 013354 000770
3716
3717
3718
3719 013356 112716 000040
3720 013362 004767 000014
3721 013366 132767 000007 000052
3722 013374 001372
3723 013376 005726

*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLC CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
BPL 1$ ;;BR IF YES
HALT ;;HALT HERE IF NO TERMINAL
BR 3$ ;;LEAVE
1$: MOV RO,-(SP) ;;SAVE RO
MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,RO ;;RESTORE RO
3$: ADD #2,(SP) ;;ADJUST RETURN PC
RTI ;;RETURN
4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
BR 2$ ;;GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;;IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,$TYPEC ;;GO TYPE A NULL
DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
BR 7$ ;;LOOP

;HORIZONTAL TAB PROCESSOR
8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
9$: JSR PC,$TYPEC ;;TYPE A SPACE
BITB #7,$CHARCNT ;;BRANCH IF NOT AT
BNE 9$ ;;TAB STOP
TST (SP)+ ;;POP SPACE OFF STACK

```

```

3724 013400 000724          BR      2$          ;;GET NEXT CHARACTER
3725 013402 105777 165542  $TYPEC: TSTB  @STPS          ;;WAIT UNTIL PRINTER IS READY
3726 013406 100375          BPL      $TYPEC
3727 013410 116677 000002 165534  MOVB    2(SP),@STPB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3728 013416 122766 000015 000002  CMPB    #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
3729 013424 001003          BNE     1$          ;;BRANCH IF NO
3730 013426 105067 000014          CLRB    $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
3731 013432 000406          BR      $TYPEX          ;;EXIT
3732 013434 122766 000012 000002 1$:  CMPB    #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
3733 013442 001402          BEQ    $TYPEX          ;;BRANCH IF YES
3734 013444 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
3735 013446 000000          $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
3736 013450 000207          $TYPEX: RTS      PC
3737
3738
3739          .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
3740
3741          ;*****
3742          ;THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
3743          ;CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY !LEGAL CHARACTERS
3744          ;ARE READ A '?' FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
3745          ;THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
3746          ;USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
3747          ;POSITIVE 32767 TO NEGATIVE 32768.
3748          ;CALL:
3749          ;*      RDDEC          ;;READ A DECIMAL NUMBER
3750          ;*      RETURN HERE  ;;NUMBER IS ON TOP OF THE STACK
3751          ;
3752
3753 013452 011646          $RDDEC: MOV    (SP),-(SP)          ;;PROVIDE SPACE FOR
3754 013454 016665 000004 000002  MOV    4(SP),2(SP)          ;;THE INPUT NUMBER
3755 013462 010046          MOV    R0,-(SP)          ;;PUSH R0 ON STACK
3756 013464 010146          MOV    R1,-(SP)          ;;PUSH R1 ON STACK
3757 013466 010246          MOV    R2,-(SP)          ;;PUSH R2 ON STACK
3758 013470 104411          1$:  RDLIN          ;;READ AN ASCII LINE
3759 013472 012600          MOV    (SP)+,R0          ;;ADDRESS OF 1ST CHAR.
3760 013474 010067 000120          MOV    R0,6$          ;;SAVE INCASE OF BAD INPUT
3761 013500 005046          CLR    -(SP)          ;;CLEAR DATA WORD
3762 013502 005002          CLR    R2          ;;SIGN SET POSITIVE
3763 013504 122710 000055          CMPB   #'-,(R0)          ;;SEE IF A MINUS SIGN WAS TYPED
3764 013510 001001          BNE     2$          ;;BR IF NO MINUS SIGN
3765 013512 112002          MOVB   (R0)+,R2          ;;SAVE FOR LATER USE
3766 013514 112001          2$:  MOVB   (R0)+,R1          ;;PICKUP THIS CHARACTER
3767 013516 001424          BEQ    3$          ;;GET OUT IF ZERO
3768 013520 122701 000060          CMPB   #'0,R1          ;;MAKE SURE THIS CHARACTER
3769 013524 003032          BGT    5$          ;;IS A DIGIT BETWEEN 0 & 9
3770 013526 122701 000071          CMPB   #'9,R1
3771 013532 002427          BLT    5$
3772 013534 032716 170000          BIT    #'C7777,(SP)          ;;DON'T LET NUMBER GET TO BIG
3773 013540 001024          BNE     5$          ;;BR IF NUMBER WOULD OVERFLOW
3774 013542 006316          ASL    (SP)          ;;*2
3775 013544 011646          MOV    (SP),-(SP)          ;;SAVE FOR LATER
3776 013546 006316          ASL    (SP)          ;;*4
3777 013550 006316          ASL    (SP)          ;;*8
3778 013552 062616          ADD    (SP)+,(SP)          ;;*10
3779 013554 102416          BVS    5$          ;;OVERFLOW ISN'T ALLOWED

```

3780 013556 162701 000060
3781 013562 060116
3782 013564 102412
3783 013566 000752
3784 013570 005702
3785 013572 001401
3786 013574 005416
3787 013576 012666 000012
3788 013602 012602
3789 013604 012601
3790 013606 012600
3791 013610 000002

```

SUB #'0,R1          ;;STRIP AWAY THE ASCII JUNK
ADD R1,(SP)         ;;ADD IN THIS DIGIT
BVS 5$              ;;OVERFLOW ISN'T ALLOWED
BR 2$               ;;LOOP
3$: TST R2          ;;CHECK IF NUMBER IS NEG
BEQ 4$              ;;BR IF NO
NEG (SP)            ;;YES--NEGATE THE NUMBER
4$: MOV (SP)+,12(SP) ;;SAVE THE RESULT
MOV (SP)+,R2        ;;POP STACK INTO R2
MOV (SP)+,R1        ;;POP STACK INTO R1
MOV (SP)+,R0        ;;POP STACK INTO R0
RTI                 ;;RETURN

5$: TST (SP)+       ;;CLEAN PARTIAL NUMBER FROM STACK
CLRB (R0)           ;;SET A TERMINATOR
TYPE                ;;TYPE THE INPUT UP TO BAD CHAR.
6$: .WORD 0          ;;POINTER GOES HERE
TYPE $QUES          ;;'?' 'CR' & 'LF'
BR 1$               ;;TRY AGAIN
    
```

3792
3793 013612 005726
3794 013614 105010
3795 013616 104401
3796 013620 000000
3797 013622 104401 001252
3798 013626 000720
3799

3800
3801
3802
3803
3804
3805
3806
3807

.SBTTL TRAP DECODER

```

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

3808 013630 010046
3809 013632 016600 000002
3810 013636 005740
3811 013640 111000
3812 013642 006300
3813 013644 016000 013664
3814 013650 000200
3815
3816
3817

```

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

3818
3819 013652 011646
3820 013654 016666 000004 000002
3821 013662 000002
3822
3823

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO

```

STRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
MOV 4(SP),2(SP)       ;;MOVE THE PSW DOWN
RTI                   ;;RESTORE THE PSW
    
```

3824
3825
3826
3827
3828
3829

.SBTTL TRAP TABLE

```

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

3830 013664 013652
3831 013666 013232
3832 013670 011200
3833 013672 011154
3834 013674 011214
3835 013676 011402

```

: ROUTINE
:
STRPAD: .WORD STRAP2
STYPER ;;CALL=TYPER TRAP+1(104401) TTY TYPEOUT ROUTINE
STYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
STYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
STYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
STYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
    
```

```
3836  
3837 015700 012222          SGTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING  
3838  
3839 013702 012132          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR  
3840 013704 012474          SRDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE  
3841 013706 012564          SRDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE  
3842 013710 013072          SRDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY  
3843 013712 013452          SRDDEC  ;;CALL=RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY  
3844
```

.SBTTL POWER DOWN AND UP ROUTINES

```
3845  
3846  
3847  
3848  
3849 013714 012737 014060 000024  
3850 013722 012737 000340 000026  
3851 013730 010046  
3852 013732 010146  
3853 013734 010246  
3854 013736 010346  
3855 013740 010446  
3856 013742 010546  
3857 013744 017746 165170  
3858 013750 010667 000110  
3859 013754 012737 013766 000024  
3860 013762 000000  
3861 013764 000776  
3862  
3863  
3864  
3865 013766 012737 014060 000024  
3866 013774 016706 000064  
3867 014000 005067 000060  
3868 014004 005267 000054  
3869 014010 001375  
3870 014012 012677 165122  
3871 014016 012605  
3872 014020 012604  
3873 014022 012603  
3874 014024 012602  
3875 014026 012601  
3876 014030 012600  
3877 014032 012737 013714 000024  
3878 014040 012737 000340 000026  
3879 014046 104401  
3880 014050 014066  
3881 014052 012716  
3882 014054 002036  
3883 014056 000002  
3884 014060 000000  
3885 014062 000776  
3886 014064 000000  
3887 014066 005015 047520 042527  
3888 014074 000122  
3889  
3890  
3891
```

```
:::*****
```

:POWER DOWN ROUTINE

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

```
:::*****
```

:POWER UP ROUTINE

```
SPWRUP: 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)+,R0 ;;POP STACK INTO R0  
MOV #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR  
MOV #340,@PWRVEC+2 ;;PRIO:7  
TYPE ;;REPORT THE POWER FAILURE  
SPWRMG: .WORD SPOWER ;;POWER FAIL MESSAGE POINTER  
MOV (PC)+,(SP) ;;RESTART AT RESTRT  
SPWRAD: .WORD RESTRT ;;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  
SPOWER: .ASCIZ <15><12>'POWER'  
  
.EVEN
```

```
:::*****
```



```
3892 ;TRANSMIT INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
3893 ;:*****
3894
3895 014076 105777 165312 XINT: TSTB @DLXCSR ;'READY' SET ??
3896 014102 100416 BMI 1$ ;BR IF YES
3897 014104 013767 177776 165070 MOV @MPSW,$TMP0 ;SAVE THE ERROR PSW
3898 014112 010667 165060 MOV SP,$REG6 ;SAVE THE ERROR STACK POINTER
3899 014116 005167 165300 COM XFLGO ;SET XMIT SOFTWARE ERROR FLAG
3900 014122 042777 000100 165260 BIC #100,@DLRCSR ;TURN OFF THE INTERRUPT ENABLES
3901 014130 042777 000100 165256 BIC #100,@DLXCSR
3902 014136 000411 BR 2$ ;GO TO EXIT
3903 014140 022767 022440 165264 1$: CMP #DLBUF1,OPTR ;XMITTED 256. BYTES YET ??
3904 014146 001405 BEQ 2$ ;BR IF YES
3905 014150 117777 165256 165240 MOVB @OPTR,@DLXDBR ;OUTPUT A BYTE
3906 014156 005267 165250 INC OPTR ;UPDATE BUFFER POINTER
3907 014162 000002 2$: RTI ;RETURN TO MAINLINE TEST
3908
3909 ;:*****
3910 ;RECEIVER INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
3911 ;:*****
3912
```

```
3913 014164 105777 165220 RINT: TSTB @DLRCSR ;'DONE' SET ??
3914 014170 100410 BMI 1$ ;BR IF YES
3915 014172 013767 177776 165002 MOV @MPSW,$TMP0 ;SAVE THE ERROR PSW
3916 014200 010667 164772 MOV SP,$REG6 ;SAVR THE ERROR STACK POINTER
3917 014204 005167 165214 COM RFLGO ;SET HARD RCVR ERROR FLAG
3918 014210 000415 BR 2$ ;GO EXIT
3919 014212 005777 165174 1$: TST @DLRDBR ;ANY SOFT ERRORS ??
3920 014216 100021 BPL 3$ ;BR IF NOT
3921 014220 013767 177776 164754 MOV @MPSW,$TMP0 ;SAVE THE ERROR PSW
3922 014226 010667 164744 MOV SP,$REG6 ;SAVE THE ERROR STACK POINTER
3923 014232 017767 165154 164744 MOV @DLRDBR,$TMP1 ;SAVE THE ERROR REGISTER IN TMP1
3924 014240 005167 165162 COM RFLG1 ;SET THE SOFT ERROR FLAG
3925 014244 042777 000100 165142 2$: BIC #100,@DLXCSR ;TURN OFF THE INTR. ENABLES
3926 014252 042777 000100 165130 BIC #100,@DLRCSR
3927 014260 000411 BR 4$ ;GO TO EXIT
3928 014262 022767 023040 165144 3$: CMP #BUFEND,IPTR ;RECEIVED 256. BYTES YET ??
3929 014270 001405 BEQ 4$ ;BR IF YES
3930 014272 117777 165114 165134 MOVB @DLRDBR,@IPTR ;INPUT A BYTE FROM THE DL11
3931 014300 005267 165130 INC IPTR ;UPDATE BUFFER POINTER
3932 014304 000002 4$: RTI ;RETURN TO MAINLINE TEST
3933
3934 ;THE FOLLOWING ROUTINE IS USED BY THE USER UTILITY PROGRAMS TO WAIT
3935 ;A SPECIFIED NO. OF MILLISECONDS BETWEEN CHARACTER TRANSFERS
3936
3937 014306 017667 000000 000034 DELAY: MOV @R6),DELCNT ;GET THE NO. OF MSEC. DELAY COUNT
3938 ;TYPED IN BY USER
3939 014314 062716 000002 ADD #2,(R6) ;SET UP THIS ROUTINE'S EXIT ADDRESS
3940 014320 005767 000024 TST DELCNT ;IS THE DELAY COUNT ZERO?
3941 014324 001410 BEQ 3$ ;BRANCH IF YES
3942 014326 012746 000226 1$: MOV #226,-(SP) ;PUSH A 1 MSEC. COUNT TO STACK
3943 014332 005316 2$: DEC (SP) ;DECREMENT THE 1 MSEC. COUNT BY 1
3944 014334 001376 BNE 2$ ;BRANCH IF 1 MSEC. NOT EATEN
3945 ;AWAY YET
3946 014336 005726 TST (SP)+ ;RESET STACK AFTER 1 MSEC. TIME UP
3947 014340 005367 000004 DEC DELCNT ;DECREMENT THE TOTAL NO. OF
```

```

3948                                     ;MSECS. COUNT
3949 014344 001370                       BNE 1$                               ;BRANCH IF WE HAVE MORE MSECS.
3950                                     ;TO WAIT
3951 014346 000207                       3$: RTS PC                               ;GO BACK TO REISSUE A CHARACTER
3952 014350 000000                       DELCNT: .WORD 0                          ;THE NO. OF MSECS. NEEDED TO
3953                                     ;TRANSPIRE RESIDES HERE
3954                                     ;THE FOLLOWING ROUTINE IS USED BY USER PROGRAM #4 AND WILL ALLOW
3955                                     ;A RANDOM NUMBER OF MILLISECONDS BEFORE TRANSMISSION OF CHARACTER
3956                                     ;
3957 014352 016700 000062                 STALL: MOV NUMONE,RO                      ;GET THE LOW LIMIT
3958 014356 006100                       ROL RO                                   ;MULTIPLY BY 4
3959 014360 006100                       ROL RO                                   ;
3960 014362 066700 000054                 ADD NUMTWO,RO                            ;ADD IN THE HIGH LIMIT
3961 014366 010067 000046                 MOV RO,NUMONE                            ;STORE THIS AS NEW LOW LIMIT
3962 014372 006100                       ROL RO                                   ;MULTIPLY NEW LOW LIMIT BY 4
3963 014374 006100                       ROL RO                                   ;
3964 014376 066700 000040                 ADD NUMTWO,RO                            ;ADD IN THE HIGH LIMIT
3965 014402 006100                       ROL RO                                   ;MULTIPLY BY 4 AGAIN
3966 014404 006100                       ROL RO                                   ;
3967 014406 010067 000030                 MOV RO,NUMTWO                            ;STORE THIS AS NEW HIGH LIMIT
3968 014412 016700 000022                 MOV NUMONE,RO                            ;SAVE THE RANDOMLY GENERATED NO.
3969 014416 046700 164664                 BIC STLMASK,RO                          ;STRIP ALL BUT 1ST 5 BITS SO AS
3970                                     ;NOT TO ALLOW THE STALL TO BE TOO
3971                                     ;LARGE
3972 014422 001405                       BEQ 2$                                   ;BRANCH IF RESULT WAS ZERO
3973 014424 010067 000004                 MOV RO,1$                                ;SET STALL TIME FOR DELAY ROUTINE
3974 014430 004767 177652                 JSR PC,DELAY                             ;GO OFF TO STALL
3975 014434 000000                       1$: .WORD 0                              ;THIS IS WHERE STALL TIME RESIDES
3976 014436 000207                       2$: RTS PC                               ;RETURN TO ISSUE CHARACTER
3977 014440 001233                       NUMONE: 1233                             ;LOW LIMIT FOR RANDOM NO.
3978 014442 007622                       NUMTWO: 7622                             ;HIGH LIMIT FOR RANDOM NO.
3979                                     ;THE FOLLOWING ROUTINE CHECKS THE 'DONE' BIT FOR BOTH THE RECEIVER
3980                                     ;AND TRANSMITTER. THIS ROUTINE IS USED BY PROGRAM #4
3981                                     ;
3982 014444 016767 164540 000044         TIMERX: MOV $TMP3,DUT                     ;GET THE TRANSMITTER CONTROL
3983                                     ;STATUS REGISTER ADDRESS
3984 014452 162767 000004 000036         SUB #4,DUT                               ;FORM THE RECEIVER CONTROL
3985                                     ;STATUS REGISTER ADDRESS
3986 014460 000403                       BR TCONT                                ;GO TO TIME OUT THE RECEIVERS'
3987                                     ;DONE BIT
3988 014462 016767 164522 000026         TIMETX: MOV $TMP3,DUT                     ;GET THE TRANSMITTER CONTROL
3989                                     ;STATUS REGISTER ADDRESS
3990 014470 005067 164522                   TCONT: CLR $TMP6                          ;INITIALIZE A TIME COUNT
3991 014474 005267 164516                   1$: INC $TMP6                             ;INCREMENT THE TIME COUNT
3992 014500 001405                       BEQ 2$                                   ;BRANCH IF TIME COUNTER OVERFLOWED
3993                                     ;INDICATING DONE BIT NEVER SET
3994                                     ;WITH PLENTY OF TIME ELAPSED
3995 014502 105777 000010                   TSTB @DUT                                ;SEE IF DONE BIT IS SET YET
3996 014506 100372                       BPL 1$                                   ;WAIT SOME MORE IF IT ISN'T
3997 014510 062716 000006                   ADD #6,@R6                              ;DONE BIT IS SET - SET UP EXIT
3998                                     ;RETURN TO SKIP ERROR REPORT
3999 014514 000207                       2$: RTS PC                               ;RETURN TO PROGRAM #4
4000 014516 000000                       DUT: .WORD 0                             ;THIS IS WHERE THE RCSR OR XCSR
4001                                     ;ADDRESS RESIDES
4002                                     ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5, AND WILL CHECK FOR CORRECT
4003                                     ;EXPECTED AND RECEIVED DATA, IN ADDITION TO ANY ERROR BITS

```

```
4004  
4005 014520 016767 164470 164470 DATCHK: MOV STMP5,STMP6 ;GET THE CONTENTS OF THE RECEIVER  
4006 ;BUFFER  
4007 014526 016767 164460 164432 MOV STMP4,$REG2 ;STORE THE ADDRESS OF THE RECEIVER  
4008 ;DATA BUFFER  
4009 014534 016767 164426 164422 MOV $REG2,$REG1 ;GET THE ADDRESS OF THE RECEIVER  
4010 ;DATA BUFFER  
4011 014542 162767 000002 164414 SUB #2,$REG1 ;FORM THE ADDRESS OF THE RECEIVER  
4012 ;STATUS REGISTER FROM IT  
4013 014550 016767 164440 164412 MOV STMP5,$REG3 ;STORE THE CONTENTS OF THE RECEIVER  
4014 ;DATA BUFFER  
4015 014556 032767 170000 164432 BIT #170000,STMP6 ;ARE ANY ERROR BITS SET?  
4016 014564 001013 BNE 1$ ;BRANCH IF YES  
4017 014566 004767 000720 JSR PC,UPMASK ;GO TO MASK OFF BITS AS A FUNCTION OF  
4018 ;CHARACTER LENGTH( 5, 6, 7, OR 8 BITS)  
4019 014572 026767 164416 164432 CMP STMP5,STMP14 ;WAS RECEIVED CHARACTER THE  
4020 ;SAME AS THE ONE TRANSMITTED?  
4021 014600 001406 BEQ 2$ ;BRANCH IF YES  
4022 014602 016767 164424 164362 MOV STMP14,$REG4 ;STORE WHAT THE CONTENTS OF THE  
4023 ;RECEIVER DATA BUFFER SHOULD BE  
4024 014610 104010 ERROR +10 ;DATA RECEIVED WRONG!  
4025 014612 000401 BR 2$ ;GET SET TO RETURN AFTER ERROR REPORT  
4026 014614 104007 1$: ERROR +7 ;ERROR BIT/S SET FROM TRANSMISSION  
4027 014616 000207 2$: RTS PC ;RETURN TO PROGRAM #4  
4028  
4029 ;:*****  
4030 ;SUBROUTINE TO SETUP ERROR INFORMATION FOR ERROR MESSAGES  
4031 ;:*****  
4032  
4033  
4034 014620 013767 177776 164354 SUER2: MOV @PSW,STMP0 ;SAVE THE [PSW]  
4035 014626 016701 164556 MOV DLRCR,R1 ;PUT DEVADR IN R1  
4036 014632 011203 MOV (R2),R3 ;PUT WAS INFO IN R3  
4037 014634 010667 164336 MOV SP,$REG6 ;SAVE THE [SP]  
4038 014640 062767 000002 164330 ADD #2,$REG6 ;CORRECT FOR CALLING JSR  
4039 014646 116700 164230 SUERR1: MOVBT STSTNA,R0 ;PUT TEST NO. IN R0  
4040 014652 010067 154304 MOV R0,$REG0 ;SAVE [R0] THRU [R4]  
4041 014656 010167 164302 MOV R1,$REG1  
4042 014662 010267 164300 MOV R2,$REG2  
4043 014666 010367 164276 MOV R3,$REG3  
4044 014672 010467 164274 MOV R4,$REG4  
4045 014676 000207 RTS PC ;RETURN TO CALLING TEST  
4046  
4047 014700 013767 177776 164274 SUERT1: MOV @PSW,STMP0 ;SAVE THE [PSW]  
4048 014706 116700 164170 MOVBT STSTNA,R0 ;PUT TEST NO. IN R0  
4049 014712 016701 164472 MOV DLRCR,R1 ;PUT DEVADR IN R1  
4050 014716 010067 164240 MOV R0,$REG0 ;SAVE [R0]  
4051 014722 010167 164236 MOV R1,$REG1 ;SAVE [R1]  
4052 014726 013767 177776 164250 SUERT2: MOV @PSW,STMP1 ;SAVE THE [PSW]  
4053 014734 010667 164236 MOV SP,$REG6 ;SAVE THE [SP]  
4054 014740 062767 000002 164230 ADD #2,$REG6 ;CORRECT FOR CALLING JSR  
4055 014746 010267 164214 MOV R2,$REG2 ;SAVE [R2]  
4056 014752 000207 RTS PC ;RETURN  
4057  
4058 ;SUBROUTINE TO SETUP VECTORS FOR 256. BYTE BLOCK TRANSFER TESTS  
4059
```

4060 014754 016705 164440
 4061 014760 012725 014164
 4062 014764 016725 164312
 4063 014770 012725 014076
 4064 014774 016715 164302
 4065 015000 000207
 4066
 4067
 4068
 4069 015002 005077 164406
 4070 015006 005077 164376
 4071 015012 005067 164404
 4072 015016 005067 164402
 4073 015022 005067 164400
 4074 015026 012767 022040 164376
 4075 015034 012767 022440 164372
 4076 015042 004767 000044
 4077 015046 004777 164364
 4078 015052 005067 164362
 4079 015056 012767 000036 164356
 4080 015064 005777 164322
 4081 015070 005777 164316
 4082 015074 052777 000100 164306
 4083 015102 052777 000104 164304
 4084 015110 000207
 4085
 4086
 4087
 4088
 4089
 4090
 4091 015112 012705 022040
 4092 015116 005025
 4093 015120 022705 023040
 4094 015124 001374
 4095 015126 000207
 4096
 4097
 4098
 4099 015130 012705 022040
 4100 015134 105025
 4101 015136 112725 000377
 4102 015142 022705 022440
 4103 015146 001372
 4104 015150 000207
 4105
 4106
 4107
 4108 015152 005005
 4109 015154 110565 022040
 4110 015160 005205
 4111 015162 022705 000400
 4112 015166 001372
 4113 015170 000207
 4114
 4115

```

SUEC:  MOV    DLVECT,R5      ;GET FIRST VECTOR ADDRESS
        MOV    #RINT,(R5)+   ;SET UP RCVR VECTOR
        MOV    DLPRI,(R5)+
        MOV    #XINT,(R5)+   ;SET UP XMIT VECTOR
        MOV    DLPRI,(R5)
        RTS    PC            ;RETURN TO CALLER

;SUBROUTINE TO PRIME DATA BUFFERS AND DEVICE FOR 256. BYTE TRANSFER
PRIME:  CLR    @DLXCSR        ;CLEAR XMIT AND RCVR CSR'S
        CLR    @DLRCSR
        CLR    XFLGO         ;INITIALIZE ERROR FLAGS
        CLR    RFLGO
        CLR    RFLG1
        MOV    #DLBUFO,OPTR  ;SET UP OUTPUT POINTER
        MOV    #DLBUFI,IPTR  ;SET UP INPUT POINTER
        JSR    PC,CLDLBF     ;GO CLEAR THE BUFFERS
        JSR    PC,@LDOUT     ;GO SET UP THE PATTERN
        CLR    TIMR1         ;INIT TIMEOUT COUNTERS
        MOV    #30,TIMR2
        TST    @DLRDBR       ;FLUSH 'DONE' BIT IN RCVR CSR
        TST    @DLRDBR
        BIS    #100,@DLRCSR  ;ENABLE RCVR INTR.
        BIS    #104,@DLXCSR  ;ENABLE XMIT INTR. AND MAINT MODE
        RTS    PC

;THIS ROUTINE IS CALLED TO CLEAR THE INPUT AND OUTPUT BUFFERS
CLDLBF: MOV    #DLBUFO,R5    ;R5 POINTS TO BEGINNING OF BUFFER AREA
1$:     CLR    (R5)+         ;CLEAR A WORD
        CMP    #BUFEND,R5   ;DONE ALL WORDS ??
        BNE   1$           ;BR IF NOT
        RTS    PC          ;RETURN TO CALLER

;THIS ROUTINE IS CALLED TO SET UP THE NULL-DEL-NUL PATTERN
LDOUT1: MOV    #DLBUFO,R5    ;R5 POINTS TO OUTPUT BUFFER
1$:     CLRB   (R5)+         ;MOVE A NULL CHAR
        MOVB  #377,(R5)+    ;MOV A DEL CHAR
        CMP   #DLBUFI,R5   ;ALL DONE ??
        BNE   1$           ;BR IF NOT
        RTS    PC          ;RETURN TO CALLER

;THIS ROUTINE IS USED TO LOAD AN ASCENDING BINARY COUNT PATTERN
LDOUT2: CLR    R5           ;START WITH 000
1$:     MOVB  R5,DLBUFO(R5)  ;LOAD ONE BYTE
        INC   R5           ;INCREMENT BYTE
        CMP   #400,R5      ;DONE 000 THRU 377 ??
        BNE   1$           ;BR IF NOT
        RTS    PC          ;RETURN TO CALLER

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

```

```

4116
4117 015172 112767 000377 164020 LDOUT3: MOVB #377,$TMP7 ;START WITH A 377 BYTE
4118 015200 012705 022040          MOV #DLBUFO,R5 ;R5 POINTS TO OUTPUT BUFFER
4119 015204 116725 164010          1$: MOVB $TMP7,(R5)+ ;LOAD ONE BYTE
4120 015210 022705 022440          CMP #DLBUF1,R5 ;ALL DONE ??
4121 015214 001403          BEQ 2$ ;BR IF YES
4122 015216 105367 163776          DECB $TMP7 ;GENERATE NEXT BYTE
4123 015222 000770          BR 1$ ;GO MOVE IT
4124 015224 000207          2$: RTS PC ;RETURN TO CALLER
4125
4126
4127 ;THIS ROUTINE LOADS A COMPLEMENTING WORST CASE PATTERN
4128
4129 015226 012705 022040 LDOUT4: MOV #DLBUFO,R5 ;R5 POINTS TO OUTPUT BUFFER
4130 015232 005067 163762          CLR $TMP7 ;INIT. BYTE GENERATOR
4131 015236 116725 163756          1$: MOVB $TMP7,(R5)+ ;MOVE A BYTE
4132 015242 105167 163752          COMB $TMP7 ;COMPLEMENT IT
4133 015246 116725 163746          MOVB $TMP7,(R5)+ ;NOW LOAD THE 1'S COMPLEMENT
4134 015252 105267 163743          INCB $TMP7+1 ;INCREMENT THE BYTE
4135 015256 116767 163737 163734 MOVB $TMP7+1,$TMP7 ;SET UP TO LOAD NEXT TWO
4136 015264 022705 022440          CMP #DLBUF1,R5 ;ALL DONE ??
4137 015270 001362          BNE 1$ ;BR IF NOT
4138 015272 000207          RTS PC ;RETURN TO CALLER
4139
4140 ;THIS ROUTINE CHECKS FOR DATA COMPARE ERRORS IN 256. BYTE BLOCK TRANSFERS
4141
4142 015274 042777 000104 164112 CHKDAT: BIC #104,@DLXCSR ;DISABLE BOTH XMIT AND RCVR INTR. ENAB.
4143 015302 042777 000100 164100          BIC #100,@DLRCSR
4144 015310 012702 022040          MOV #DLBUFO,R2 ;R2 POINTS TO S/B DATA IN OUTPUT BUFFER
4145 015314 004767 000070          JSR PC,MASKING ;GO TO MASK OFF BITS AS A FUNCTION OF
4146 ;CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
4147 015320 012701 022440          MOV #DLBUF1,R1 ;R1 POINTS TO WAS DATA IN RCVR. BUFFER
4148 015324 122221          1$: CMPB (R2)+,(R1)+ ;DID S/B = WAS ??
4149 015326 001004          BNE 3$ ;BR IF NOT
4150 015330 022701 023040          2$: CMP #BUFEND,R1 ;CHECKED ALL BYTES ??
4151 015334 001373          BNE 1$ ;BR IF NOT
4152 015336 000207          RTS PC ;RETURN TO CALLER

```

```

4153 015340 013767 177776 163634 3$: MOV @#PSW,$TMP0 ;SAVE THE [PSW]
4154 015346 010667 163624 MOV SP,$REG6 ;SAVE THE [SP]
4155 015352 114204 MOV#B -(R2),R4 ;GET THE S/B DATA
4156 015354 042704 177400 BIC #177400,R4 ;CLEAR JUNK FROM HI BYTE
4157 015360 114103 MOV#B -(R1),R3 ;GET THE WAS DATA
4158 015362 042703 177400 BIC #177400,R3 ;CLEAR JUNK FROM HI BYTE
4159 015366 004767 177254 JSR PC,SUERR1 ;GO SET UP ERROR INFO.
4160 015372 012767 015402 163644 MOV #4$, $ESCAPE ;RETURN TO 4$ AFTER ERROR PRINT
4161 015400 104003 ERROR+3 ;DATA COMPARE ERROR
4162 015402 005202 4$: INC R2 ;REPOSITION BUFFER POINTERS
4163 015404 005201 INC R1
4164 015406 000750 BR 2$ ;GO CHECK NEXT BYTE
4165
4166 ;THIS ROUTINE IS USED BY THE PATTERN TESTS
4167 ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE XMITTER
4168 ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS TRANSMITTED
4169 ;IS DONE. THE MASKING IS DONE AS A FUNCTION OF CHARACTER LENGTH WHICH
4170 ;CAN BE EITHER 5, 6, 7, OR 8 BITS.
4171
4172 015410 005005 MASKING: CLR R5 ;INITIALIZE TABLE OFFSET
4173 ;FOR PICKING UP MASK WORD
4174
4175 015412 022767 000010 163614 CMP #8,$TMP15 ;IS THE CHARACTER LENGTH 8 BITS?
4176 015420 001427 BEQ 3$ ;BRANCH IF IT IS
4177 015422 062705 000002 ADD #2,R5 ;SET UP FOR NEXT MASK WORD
4178 ;IT COULD BE THIS ONE
4179 015426 022767 000007 163600 CMP #7,$TMP15 ;IS THE CHARACTER LENGTH 7 BITS?
4180 015434 001410 BEQ 1$ ;BRANCH IF IT IS
4181 015436 062705 000002 ADD #2,R5 ;SET UP FOR NEXT MASK WORD
4182 ;IT COULD BE THIS ONE
4183 015442 022767 000006 163564 CMP #6,$TMP15 ;IS THE CHARACTER LENGTH 6 BITS?
4184 015450 001402 BEQ 1$ ;BRANCH IF IT IS
4185 015452 062705 000002 ADD #2,R5 ;SET UP FOR NEXT MASK WORD
4186 ;IT MUST BE THIS ONE!!!!
4187 015456 016505 015502 1$: MOV CHARL(R5),R5 ;PICK UP THE MASK WORD
4188 015462 005105 COM R5 ;FORM THE BITS THAT ARE TO BE MASKED
4189 015464 140522 2$: BICB R5,(R2)+ ;MASK A BYTE
4190 015466 022702 022440 CMP #DLBUF1,R2 ;ARE WE AT THE END OF THE XMITTER
4191 ;OUTPUT BUFFER
4192 015472 001374 BNE 2$ ;BRANCH IF NO TO MASK NEXT BYTE
4193 015474 012702 022040 MOV #DLBUF0,R2 ;RESTORE R2 BEFORE RETURNING
4194 015500 000207 3$: RTS PC ;RETURN TO MAINLINE CODE
4195 ;TABLE OF MASK WORDS
4196 015502 000377 CHARL: .WORD 377 ;8. BITS IN LENGTH
4197 015504 000177 .WORD 177 ;7. BITS IN LENGTH
4198 015506 000077 .WORD 77 ;6. BITS IN LENGTH
4199 015510 000037 .WORD 37 ;5. BITS IN LENGTH
4200
4201 ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5
4202 ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE TRANSMITTER
4203 ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS
4204 ;TRANSMITTED IS DONE. THE MASKING IS DONE AS A FUNCTION OF CHARACTER
4205 ;LENGTH WHICH CAN BE EITHER 5, 6, 7, OR 8 BITS.
4206
4207 015512 016767 163466 163512 UPMASK: MOV $TMP1,$TMP14 ;PICK UP THE CHARACTER THAT WAS
4208 ;SENT OUT FROM THE XMITTER

```

```

4209 015520 005005 CLR R5 ;INITIALIZE TABLE OFFSET
4210 ;FOR PICKING UP MASK WORD
4211 015522 022767 000010 163504 CMP #8, $TMP15 ;IS THE CHARACTER LENGTH 8 BITS?
4212 015530 001423 BEQ 2$ ;BRANCH IF IT IS
4213 015532 062705 000002 ADD #2, R5 ;SET UP FOR NEXT MASK WORD
4214 ;IT COULD BE THIS ONE
4215 015536 022767 000007 163470 CMP #7, $TMP15 ;IS THE CHARACTER LENGTH 7 BITS?
4216 015544 001410 BEQ 1$ ;BRANCH IF IT IS
4217 015546 062705 000002 ADD #2, R5 ;SET UP FOR NEXT MASK WORD
4218 ;IT COULD BE THIS ONE
4219 015552 022767 000006 163454 CMP #6, $TMP15 ;IS THE CHARACTER LENGTH 6 BITS?
4220 015560 001402 BEQ 1$ ;BRANCH IF IT IS
4221 015562 062705 000002 ADD #2, R5 ;SET UP FOR NEXT MASK WORD
4222 ;IT MUST BE THIS ONE!!!!
4223 015566 016505 015502 1$: MOV CHARL(R5), R5 ;PICK UP THE MASK WORD
4224 015572 005105 COM R5 ;FORM THE BITS THAT ARE TO BE MASKED
4225 015574 140567 163432 BICB R5, $TMP14 ;MASK THE LOW BYTE
4226 015600 000207 2$: RTS PC ;RETURN TO MAINLINE CODE
4227
4228 ;ROUTINE TO SERVICE BUS ERROR TRAPS
4229
4230 015602 112767 000060 000632 BUSERR: MOVB #60, EM4+46 ;SET UP ERROR MESSAGE
4231 015610 112767 000060 000625 MOVB #60, EM4+47
4232 015616 112767 000064 000620 MOVB #64, EM4+50
4233 015624 000412 BR TRPCOM ;GO SET UP AND REPORT BUS ERROR
4234
4235 ;ROUTINE TO SERVICE RSVD INSTRUCTION TRAPS
4236
4237 015626 112767 000060 000606 RSVERR: MOVB #60, EM4+46 ;SET UP ERROR MESSAGE
4238 015634 112767 000061 000601 MOVB #61, EM4+47
4239 015642 112767 000060 000574 MOVB #60, EM4+50
4240 015650 000400 BR TRPCOM ;GO SET UP AND REPORT RSVD INSTR. ERROR
4241
4242 ;ROUTINE TO SET UP AND REPORT BUS ERROR AND RSVD INSTR ERRORS
4243
4244 015652 010667 163320 TRPCOM: MOV SP, $REG6 ;SAVE THE TRAP SP
4245 015656 116700 163220 MOVB $STMP, R0 ;PUT TEST NO. IN R0
4246 015662 010067 163274 MOV R0, $REG0 ;SAVE TEST #
4247 015666 016667 000002 163306 MOV 2(SP), $TMP0 ;SAVE THE ERROR PSW
4248 015674 012767 015710 163342 MOV #1$, $ESCAPE ;GO TO 1$ AFTER ERROR PRINT
4249 015702 011667 163272 MOV (SP), $REG7 ;SAVE THE ERROR PC
4250 015706 104004 ERROR+4 ;REPORTED TRAP ERROR
4251 015710 000137 002036 1$: JMP @PRESTR ;ATTEMPT TO RESTART THE PROGRAM
4252 ;AND TRY AGAIN
4253
4254
4255 ;*****
4256 ;ERROR MESSAGE INFORMATION
4257 ;*****
4258
4259 ;INFORMATION FOR ERROR MESSAGE 1
4260
4261 015714 046104 030461 051040 EM1: .ASCIZ 'DL11 REGISTER REFERENCE CAUSED TIMEOUT'
4262 015722 043505 051511 042524
4263 015730 020122 042522 042506
4264 015736 042522 041516 020105

```

4265 015744 040503 051525 042105
4266 015752 052040 046511 047505
4267 015760 052125 000
4268 015763 040 050050 024503
4269 015770 020040 020040 050050
4270 015776 024523 020040 020040
4271 016004 051450 024520 020040
4272 016012 020040 042524 052123
4273 016020 020040 042040 053105
4274 016026 042101 020122 051040
4275 016034 043505 042101 000122
4276
4277 016042 001116 001202 001176
4278 016050 001162 001164 001166
4279 016056 000000
4280
4281
4282
4283 016060 046104 030461 051040
4284 016066 043505 051511 042524
4285 016074 020122 051105 047522
4286 016102 000122
4287 016104 024040 041520 020051
4288 016112 020040 024040 051520
4289 016120 020051 020040 024040
4290 016126 050123 020051 020040
4291 016134 052040 051505 020124
4292 016142 020040 042504 040526
4293 016150 051104 020040 042522
4294 016156 040507 051104 020040
4295 016164 053440 051501 020040
4296 016172 020040 051440 041057
4297 016200 000
4298
4299 016202 016202 001116 001202 001176
4300 016210 001162 001164 001166
4301 016216 001170 001172 000000
4302
4303
4304
4305 016224 046104 030461 042040
4306 016232 052101 020101 047503
4307 016240 050115 051101 020105
4308 016246 051105 047522 000122
4309 016254 024040 041520 020051
4310 016262 020040 024040 051520
4311 016270 020051 020040 024040
4312 016276 050123 020051 020040
4313 016304 052040 051505 020124
4314 016312 020040 040527 040523
4315 016320 051104 020040 044123
4316 016326 040502 051104 020040
4317 016334 020040 040527 020123
4318 016342 020040 020040 027523
4319 016350 000102
4320

DH1: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR'

.EVEN
DT1: .WORD \$ERRPC,\$TMP0,\$REG6,\$REG0,\$REG1,\$REG2,0

; INFORMATION FOR ERROR MESSAGE 2

EM2: .ASCIZ 'DL11 REGISTER ERROR'

DH2: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR WAS S/B'

.EVEN
DT2: .WORD \$ERRPC,\$TMP0,\$REG6,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0

; INFORMATION FOR MESSAGE 3

EM3: .ASCIZ 'DL11 DATA COMPARE ERROR'

DH3: .ASCIZ ' (PC) (PS) (SP) TEST WASADR SHBADR WAS S/B'

.EVEN


```
4321 016352 001116 001202 001176 DT3: .WORD $ERRPC,$TMPO,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
4322 016360 001162 001164 001166
4323 016366 001170 001172 000000
4324
4325 ;INFORMATION FOR MESSAGE 4
4326
4327 016374 047125 054105 042520 EM4: .ASCIZ 'UNEXPECTED TRAP TO VECTOR AT LOCATION '
4328 016402 052103 042105 052040
4329 016410 040522 020120 047524
4330 016416 053040 041505 047524
4331 016424 020122 052101 046040
4332 016432 041517 052101 047511
4333 016440 020116 020040 000040
4334 016446 024040 041520 020051 DH4: .ASCIZ ' (PC) (PS) (SP) TEST'
4335 016454 020040 024040 051520
4336 016462 020051 020040 024040
4337 016470 050123 020051 020040
4338 016476 052040 051505 000124
4339 .EVEN
4340 016504 001200 001202 001176 DT4: .WORD $REG7,$TMPO,$REG6,$REG0,0
4341 016512 001162 000000
4342
4343 ;ERROR INFORMATION FOR ERROR MESSAGE 5
4344
4345 016516 046104 030461 051440 EM5: .ASCIZ 'DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) '
4346 016524 043117 020124 051105
4347 016532 047522 020122 050050
4348 016540 051101 052111 026131
4349 016546 051106 046501 047111
4350 016554 026107 047440 020122
4351 016562 053117 051105 052522
4352 016570 024516 000
4353 016573 040 050050 024503 DH5: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR (REG) '
4354 016600 020040 020040 050050
4355 016606 024523 020040 020040
4356 016614 051450 024520 020040
4357 016622 020040 042524 052123
4358 016630 020040 042040 053105
4359 016636 042101 020122 051040
4360 016644 043505 042101 020122
4361 016652 020040 051050 043505
4362 016660 000051
4363 .EVEN
4364 016662 001116 001202 001176 DT5: .WORD $ERRPC,$TMPO,$REG6,$REG0,$REG1,$REG2,$REG3,0
4365 016670 001162 001164 001166
4366 016676 001170 000000
4367
4368 ;INFORMATION FOR ERROR MESSAGE 6
4369
4370 016702 024040 041520 020051 DH6: .ASCIZ ' (PC) (PS) (SP) REGADR '
4371 016710 020040 024040 051520
4372 016716 020051 020040 024040
4373 016724 050123 020051 020040
4374 016732 042522 040507 051104
4375 016740 000
4376 .EVEN 016742
```

```
4377 016742 001116 001204 001176 DT6: .WORD $ERRPC,$TMP1,$REG6,$REG2,0
4378 016750 001166 000000
4379
4380 ;INFORMATION FOR ERROR MESSAGE 7
4381
4382 016754 024040 041520 020051 DH7: .ASCIZ ' (PC) DEVADR REGADR (REG) '
4383 016762 020040 042504 040526
4384 016770 051104 020040 042522
4385 016776 040507 051104 020040
4386 017004 024040 042522 024507
4387 017012 000
4388 .EVEN
4389 017014 001116 001164 001166 DT7: .WORD $ERRPC,$REG1,$REG2,$REG3,0
4390 017022 001170 000000
4391
4392 ;INFORMATION FOR ERROR MESSAGE 10
4393
4394 017026 024040 041520 020051 DH10: .ASCIZ ' (PC) DEVADR REGADR (REG) S/B '
4395 017034 020040 042504 040526
4396 017042 051104 020040 042522
4397 017050 040507 051104 020040
4398 017056 024040 042522 024507
4399 017064 020040 020040 027523
4400 017072 000102
4401 .EVEN
4402 017074 001116 001164 001166 DT10: .WORD $ERRPC,$REG1,$REG2,$REG3,$REG4,0
4403 017102 001170 001172 000000
4404 ;MISCELLANEOUS MESSAGES
4405
4406 017110 052516 046114 042055 XMSG1: .ASCIZ 'NULL-DEL-NUL SEQUENCE TIMEOUT AT FOLLOWING PC'
4407 017116 046105 047055 046125
4408 017124 020114 042523 052521
4409 017132 047105 042503 052040
4410 017140 046511 047505 052125
4411 017146 040440 020124 047506
4412 017154 046114 053517 047111
4413 017162 020107 041520 000
4414 017167 102 047111 051101 XMSG2: .ASCIZ 'BINARY UP COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
4415 017174 020131 050125 041440
4416 017202 052517 052116 051440
4417 017210 050505 042525 041516
4418 017216 020105 044524 042515
4419 017224 052517 020124 052101
4420 017232 043040 046117 047514
4421 017240 044527 043516 050040
4422 017246 000103
4423 017250 044502 040516 054522 XMSG3: .ASCIZ 'BINARY DOWN COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
4424 017256 042040 053517 020116
4425 017264 047503 047125 020124
4426 017272 042523 052521 047105
4427 017300 042503 052040 046511
4428 017306 047505 052125 040440
4429 017314 020124 047506 046114
4430 017322 053517 047111 020107
4431 017330 041520 000
4432 017333 127 051117 052123 XMSG4: .ASCIZ 'WORST CASE PATTERN SEQUENCE TIMEOUT AT FOLLOWING PC'
```

4433 017340 041440 051501 020105
4434 017346 040520 052124 051105
4435 017354 020116 042523 052521
4436 017362 047105 042503 052040
4437 017370 046511 047505 052125
4438 017376 040440 020124 047506
4439 017404 046114 053517 047111
4440 017412 020107 041520 000
4441
4442 017417 015 041412 042132
4443 017424 041514 030104 042040
4444 017432 030514 026461 026103
4445 017440 026104 020105 043117
4446 017446 047114 020105 051524
4447 017454 006524 000012
4448
4449 017460 005015 047531 020125
4450 017466 040510 042526 051440
4451 017474 046105 041505 042524
4452 017502 020104 051120 043517
4453 017510 040522 020115 047516
4454 017516 020056 006462 000012
4455 017524 005015 047531 020125
4456 017532 040510 042526 051440
4457 017540 046105 041505 042524
4458 017546 020104 051120 043517
4459 017554 040522 020115 047516
4460 017562 020056 006463 000012
4461 017570 005015 047531 020125
4462 017576 040510 042526 051440
4463 017604 046105 041505 042524
4464 017612 020104 051120 043517
4465 017620 040522 020115 047516
4466 017626 020056 006464 000012
4467 017634 005015 047531 020125
4468 017642 040510 042526 051440
4469 017650 046105 041505 042524
4470 017656 020104 051120 043517
4471 017664 040522 020115 047516
4472 017672 020056 006465 000012
4473 017700 005015 051124 047101
4474 017706 046523 052111 042524
4475 017714 020122 047504 042516
4476 017722 041040 052111 047040
4477 017730 053105 051105 051440
4478 017736 052105 020040 041520
4479 017744 020075 000
4480 017747 015 051012 041505
4481 017754 044505 042526 020122
4482 017762 047504 042516 041040
4483 017770 052111 047040 053105
4484 017776 051105 051440 052105
4485 020004 020040 041520 020075
4486 020012 000
4487
4488

STMES: .ASCIZ <15><12>'CZDLCD0 DL11-C,D,E OFLNE TST'<15><12>

PROG2M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 2'<15><12>

PROG3M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 3'<15><12>

PROG4M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 4'<15><12>

PROG5M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 5'<15><12>

XDB: .ASCIZ <15><12>'TRANSMITTER DONE BIT NEVER SET PC= '

RDB: .ASCIZ <15><12>'RECEIVER DONE BIT NEVER SET PC= '

;MESSAGES SEEKING USER RESPONSE

4489	020013	015	053412	040510	LENGTH: .ASCIZ <15><12>'WHAT IS THE CHARACTER LENGTH (5,6,7 OR 8 BITS)?'
4490	020020	020124	051511	052040	
4491	020026	042510	041440	040510	
4492	020034	040522	052103	051105	
4493	020042	046040	047105	052107	
4494	020050	020110	032450	033054	
4495	020056	033454	047440	020122	
4496	020064	020070	044502	051524	
4497	020072	037451	000		
4498	020075	015	042012	020117	DEFAULT: .ASCII <15><12>'DO YOU WISH TO TEST OTHER THAN THE'
4499	020102	047531	020125	044527	
4500	020110	044123	052040	020117	
4501	020116	042524	052123	047440	
4502	020124	044124	051105	052040	
4503	020132	040510	020116	044124	
4504	020140	105			
4505	020141	015	042012	043105	.ASCIZ <15><12>'DEFAULT DEVICE (1/0 = YES/NO)?'
4506	020146	052501	052114	042040	
4507	020154	053105	041511	020105	
4508	020162	030450	030057	036440	
4509	020170	054440	051505	047057	
4510	020176	024517	000077		
4511	020202	005015	044127	052101	MFIRSTD: .ASCIZ <15><12>'WHAT IS THE 1ST RECEIVER STATUS REGISTER ADDRESS? '
4512	020210	044440	020123	044124	
4513	020216	020105	051461	020124	
4514	020224	042522	042503	053111	
4515	020232	051105	051440	040524	
4516	020240	052524	020123	042522	
4517	020246	044507	052123	051105	
4518	020254	040440	042104	042522	
4519	020262	051523	020077	000040	
4520	020270	005015	044127	052101	MVECT: .ASCIZ <15><12>'WHAT IS THE 1ST RECEIVER''S VECTOR ADDRESS? '
4521	020276	044440	020123	044124	
4522	020304	020105	051461	020124	
4523	020312	042522	042503	053111	
4524	020320	051105	020123	042526	
4525	020326	052103	051117	040440	
4526	020334	042104	042522	051523	
4527	020342	020077	000040		
4528	020346	005015	047504	054440	MULDEV: .ASCIZ <15><12>'DO YOU WANT TO TEST MULTIPLE DEVICES 1/0=YES/NO? '
4529	020354	052517	053440	047101	
4530	020362	020124	047524	052040	
4531	020370	051505	020124	052515	
4532	020376	052114	050111	042514	
4533	020404	042040	053105	041511	
4534	020412	051505	030440	030057	
4535	020420	054475	051505	047057	
4536	020426	037517	020040	000	
4537	020433	015	053412	040510	MLASTD: .ASCIZ <15><12>'WHAT IS THE STATUS REGISTER ADDRESS OF THE LAST RECEIVER? '
4538	020440	020124	051511	052040	
4539	020446	042510	051440	040524	
4540	020454	052524	020123	042522	
4541	020462	044507	052123	051105	
4542	020470	040440	042104	042522	
4543	020476	051523	047440	020106	
4544	020504	044124	020105	040514	

4545	020512	052123	051040	041505	
4546	020520	044505	042526	037522	
4547	020526	020040	000		
4548	020531	015	051412	046517	MRANGE: .ASCIZ <15><12>'SOMETHING WRONG-ANSWER THE LAST QUESTION AGAIN! '
4549	020536	052105	044510	043516	
4550	020544	053440	047522	043516	
4551	020552	040455	051516	042527	
4552	020560	020122	044124	020105	
4553	020566	040514	052123	050440	
4554	020574	042525	052123	047511	
4555	020602	020116	043501	044501	
4556	020610	020516	020040	000	
4557	020615	015	053412	040510	PLEVEL: .ASCIZ <15><12>'WHAT IS YOUR INTERRUPT PRIORITY LEVEL? '
4558	020622	020124	051511	054440	
4559	020630	052517	020122	047111	
4560	020636	042524	051122	050125	
4561	020644	020124	051120	047511	
4562	020652	044522	054524	046040	
4563	020660	053105	046105	020077	
4564	020666	000040			
4565	020670	005015	051120	043517	FOULUP: .ASCII <15><12>'PROGRAM DEVICE ACTIVE LOCATION SHOWS NO DEVICE ACTIVE'
4566	020676	040522	020115	042504	
4567	020704	044526	042503	040440	
4568	020712	052103	053111	020105	
4569	020720	047514	040503	044524	
4570	020726	047117	051440	047510	
4571	020734	051527	047040	020117	
4572	020742	042504	044526	042503	
4573	020750	040440	052103	053111	
4574	020756	105			
4575	020757	015	051412	052105	.ASCII <15><12>'SET SWITCH 0 TO A ONE (1) AND'
4576	020764	051440	044527	041524	
4577	020772	020110	020060	047524	
4578	021000	040440	047440	042516	
4579	021006	024040	024461	040440	
4580	021014	042116			
4581	021016	005015	044510	020124	.ASCII <15><12>'HIT CONTINUE TO GO BACK TO DEVICE SELECTION AGAIN'
4582	021024	047503	052116	047111	
4583	021032	042525	052040	020117	
4584	021040	047507	041040	041501	
4585	021046	020113	047524	042040	
4586	021054	053105	041511	020105	
4587	021062	042523	042514	052103	
4588	021070	047511	020116	043501	
4589	021076	044501	000116		
4590	021102	005015	044127	052101	LINTAD: .ASCIZ <15><12>'WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS? '
4591	021110	044440	020123	044124	
4592	021116	020105	051124	047101	
4593	021124	046523	052111	042524	
4594	021132	020122	040504	040524	
4595	021140	041040	043125	042506	
4596	021146	020122	042101	051104	
4597	021154	051505	037523	020040	
4598	021162	000			
4599	021163	015	053412	040510	SELCAR: .ASCIZ <15><12>'WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G. A=101
4600	021170	020124	051511	052040	

4601 021176 042510 041440 040510
4602 021204 040522 052103 051105
4603 021212 052040 020117 042502
4604 021220 052040 040522 051516
4605 021226 044515 052124 042105
4606 021234 024040 041517 040524
4607 021242 020114 051501 044503
4608 021250 020111 027105 027107
4609 021256 040440 030475 030460
4610 021264 037451 020040 000
4611 021271 015 053412 040510
4612 021276 020124 051511 052040
4613 021304 042510 042040 051505
4614 021312 051111 042105 046440
4615 021320 042523 027103 042040
4616 021326 046105 054501 024040
4617 021334 041517 040524 020114
4618 021342 027105 027107 030440
4619 021350 036460 024070 030061
4620 021356 024451 020077 000040
4621 021364 005015 051511 040440
4622 021372 051040 047101 047504
4623 021400 020115 040527 052111
4624 021406 052040 046511 020105
4625 021414 046450 042523 027103
4626 021422 020051 042504 044523
4627 021430 042522 020104 030440
4628 021436 030057 054475 051505
4629 021444 047057 037517 020040
4630 021452 000
4631 021453 015 054412 052517
4632 021460 044040 053101 020105
4633 021466 053523 034122 051440
4634 021474 052105 044440 042116
4635 021502 041511 052101 047111
4636 021510 020107 047514 050117
4637 021516 047440 020116 042524
4638 021524 052123
4639 021526 005015 040510 042526
4640 021534 054440 052517 046440
4641 021542 042117 043111 042511
4642 021550 020104 044124 020105
4643 021556 051120 050117 051105
4644 021564 046040 041517 052101
4645 021572 047511 051516 043040
4646 021600 051117 052040 042510
4647 021606 005015 042504 044526
4648 021614 042503 052040 040510
4649 021622 020124 047531 020125
4650 021630 040527 052116 052040
4651 021636 020117 042524 052123
4652 021644 077
4653 021645 015 044412 020106
4654 021652 047523 026440 050040
4655 021660 042522 051523 052040
4656 021666 042510 041440 047117

SELDLY: .ASCIZ <15><12>'WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G. 10=8(10))? '

RSTALL: .ASCIZ <15><12>'IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO? '

FAILSA: .ASCII <15><12>'YOU HAVE SWR8 SET INDICATING LOOP ON TEST'

.ASCII <15><12>'HAVE YOU MODIFIED THE PROPER LOCATIONS FOR THE'

.ASCII <15><12>'DEVICE THAT YOU WANT TO TEST?'

.ASCII <15><12>'IF SO - PRESS THE CONTINUE SWITCH'

```
4657 021674 044524 052516 020105
4658 021702 053523 052111 044103
4659 021710 005015 043111 047040      .ASCII <15><12>'IF NOT - MODIFY THE PROPER LOCATIONS, THEN'
4660 021716 052117 026440 046440
4661 021724 042117 043111 020131
4662 021732 044124 020105 051120
4663 021740 050117 051105 046040
4664 021746 041517 052101 047511
4665 021754 051516 020054 044124
4666 021762 047105
4667 021764 005015 042522 052123      .ASCIIZ <15><12>'RESTART THE PROGRAM AT ADDRESS 200'
4668 021772 051101 020124 044124
4669 022000 020105 051120 043517
4670 022006 040522 020115 052101
4671 022014 040440 042104 042522
4672 022022 051523 031040 030060
4673 022030      000
4674
4675 022031      040 020075 041520      PCMSG: .ASCII ' = PC'
4676 022036 000040      .ASCIIZ ' '
4677
4678      .EVEN
4679      ;512. WORDS RESERVED FOR TWO 256. BYTE INPUT/OUTPUT DATA BUFFERS
4680
4681 022040 000400      DLBUFO: .BLKB 256.      ;RSVD FOR OUTPUT BUFFER
4682      ;THIS IS THE DATA BEING SENT OUT
4683      ;BY THE TRANSMITTER
4684 022440 000400      DLBUFI: .BLKB 256.      ;RSVD FOR INPUT BUFFER
4685      ;THIS IS THE DATA THAT WAS PICKED
4686      ;UP BY THE RECEIVER (I.E. DATA
4687      ;SENT BY THE TRANSMITTER - HOPEFULLY)
4688 023040 000000      BUFEND: 0      ;TAG MARKS END OF BUFFERS
4689
4690      000001      .END
```

ACTREG	001274	1453#	1821*	1853*	1854*	1864*	2973	2957					
BASEAD	001264	1439#	1773*	1859*	1860	1867*	1875*	2985*	3001	3020*	3022		
BASEIV	001270	1446#	1793*	2988*	3003	3021*	3023						
BEGIN	001452	1235	1582#										
BIT0	= 000001	1339#	1760	1783	1841	2298	2304	2317	2576	2660	2741	2846	
BIT00	= 000001	1329#	1339										
BIT01	= 000002	1328#	1338										
BIT02	= 000004	1327#	1337										
BIT03	= 000010	1326#	1336										
BIT04	= 000020	1325#	1335										
BIT05	= 000040	1324#	1334										
BIT06	= 000100	1323#	1333										
BIT07	= 000200	1322#	1332										
BIT08	= 000400	1321#	1331	3086									
BIT09	= 001000	1320#	1330	3094	3155								
BIT1	= 000002	1338#	2206	2222	2268								
BIT10	= 002000	1319#	3139										
BIT11	= 004000	1318#	3101										
BIT12	= 010000	1317#											
BIT13	= 020000	1316#	3146										
BIT14	= 040000	1315#	3072										
BIT15	= 100000	1314#	2140	2152	2171	2183	2185	2207	2220	2223	2282		
BIT2	= 000004	1337#	2012	2018	2085	2096	2109	2139	2151	2706	2804	2909	
BIT3	= 000010	1336#	2170	2184									
BIT4	= 000020	1335#											
BIT5	= 000040	1334#	2243	2249	2267								
BIT6	= 000100	1333#	2061	2067	2084	2111							
BIT7	= 000200	1332#											
BIT8	= 000400	1331#											
BIT9	= 001000	1330#											
BPTVEC	= 000014	1346#											
BUFEND	023040	2339	2398	2455	2512	3928	4093	4150	4688#				
BUSERR	715602	1686	4230#										
CHARL	015502	4187	4196#	4223									
CHGD1	000246	1233#											
CHGD2	000256	1237#											
CHKDAT	015274	2341	2400	2457	2514	4142#							
CKSWR	= 104407	3071	3135	3154	3839#								
CLDLBF	015112	4076	4091#										
CONQUE	002672	1826	1870	1882#	1923								
CR	= 000015	1254#	3728	3738									
CRLF	= 000200	1255#	3699	3738									
DATCHK	014520	2818	2922	4005#									
DDISP	= 177570	1261#	1383	1609									
DEFAULT	020075	1731	4498#										
DELAY	014306	2624	2711	3937#	3974								
DELCNT	014350	3937*	3940	3947*	3952#								
DH1	015763	1504	4268#										
DH10	017026	1553	4394#										
DH2	016104	1511	4287#										
DH3	016254	1518	4309#										
DH4	016446	1525	4334#										
DH5	016573	1532	4353#										
DH6	016702	1539	4370#										
DH7	016754	1546	4382#										
DISPLA	001142	1383#	1609*	1617*	3115*	3138*							

LDOUT4	015226	2504	4129#																	
LENGTH	020013	1720	2784	2875	4489#															
LESS1	001304	1471#	1907*	1909*	1911*															
LF =	000012	1253#	3732	3738																
LINTAD	021102	2565	2649	2730	2835	4590#														
MASKIN	015410	4145	4172#																	
MFIRST	020202	1749	4511#																	
MLASTD	020433	1830	4537#																	
MRANGE	020531	1877	4548#																	
MULDEV	020346	1805	4528#																	
MULTD	001272	1449#	1631*	1812*	1815*	1818	2970	2980*												
MVECT	020270	1774	4520#																	
MUPONE	014440	3957	3961*	3968	3977#															
MUP TWO	014442	3960	3964	3967*	3978#															
ONCE	002120	1667	1696	1707#																
OPTR	001432	1570#	3903	3905	3906*	4074*														
PCMSG	022031	1665	4675#																	
PIRQ =	177772	1259#																		
PIRQVE =	000240	1353#																		
PLEVEL	020615	1888	4557#																	
PRG1	001734	1224	1631#																	
PRG2	006424	1226	2562#																	
PRG2A	006432	2565#	2635																	
PRG2B	006552	2607#																		
PRG3	006632	1228	2646#																	
PRG3A	006640	2649#	2717																	
PRG3B	006760	2690#																		
PRG4	007050	1230	2727#																	
PRG4A	007056	2730#	2822																	
PRG4B	007210	2776#																		
PRG4C	007222	2784#	2825																	
PRG5	007410	1232	2832#																	
PRG5A	007416	2835#	2927																	
PRG5B	007606	2894#	2924																	
PRG5C	007536	2875#	2930																	
PRIME	015002	2332	2391	2448	2505	4069#														
PROG2M	017460	2563	4449#																	
PROG3M	017524	2647	4455#																	
PROG4M	017570	2728	4461#																	
PROG5M	017634	2833	4467#																	
PRO =	000000	1276#																		
PR1 =	000040	1277#																		
PR2 =	000100	1278#																		
PR3 =	000140	1279#																		
PR4 =	000200	1280#																		
PR5 =	000240	1281#																		
PR6 =	000300	1282#																		
PR7 =	000340	1283#																		
PS =	177776	1234*	1256#	1257																
PSW =	177776	1257#	2095	3897	3915	3921	4034	4047	4052	4153										
PWRVEC =	000024	1348#	1597*	1598*	3849*	3850*	3859*	3865*	3877*	3878*										
RDB	017747	2810	2914	4480#																
RDCHR =	104410	3558	3840#																	
RDDEC =	104413	1722	2786	2877	3843#															
RDLIN =	104411	3633	3758	3841#																
RDOCT =	104412	1733	1752	1775	1807		1879	1889	2568	2609	2617	2652	2692	2700						

CZDLCD.DL11-C.D.E OF LNE TST
CZDLCD.P11 23-JUN-80 11:10

MACY11 30A(1052) 23-JUN-80 11:11 F 8 PAGE 98
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0076

3861 3885 4298# 4376# 4388# 4681# 4684#

.SETUP	1#	1189#	1581
.SWRHI	1#	1189#	1205
.SWRLO	1189#	1217#	1218
.SACT1	1#		
.SAPT8	1#		
.SAPTH	1#		
.SAPTY	1#		
.SASTA	1#		
.SCATC	1#	1189#	
.SCMTA	1#	1189#	1355
.SDB2D	1#		
.SDB2O	1#		
.SDIV	1#		
.SEOP	1#	1189#	2956
.SERRO	1#	1189#	3120
.SERRT	1#	1189#	3164
.SMILT	1#		
.SPOWE	1#	1189#	3845
.SRAND	1#		
.SRDDE	1#	1189#	3739
.SRDOC	1#	1189#	3614
.SREAD	1#	1189#	3358
.SR2AZ	1#		
.SSAVE	1#		
.SSB2D	1#		
.SSB2O	1#		
.SSCOP	1#	1189#	3056
.SSIZE	1#		
.SSUPR	1#		
.STRAP	1#	1189#	3800
.STYP8	1#		
.STYPD	1#	1189#	3290
.STYPE	1#	1189#	3668
.STYPO	1#	1189#	3212
.S4OCA	1#		
.1170	1#		

. ABS. 023042 000

ERRORS DETECTED: 0

CZDLCD.BIN,CZDLCD.LST/CRF/SOL/NL:TOC=CZDLCD.SML,CZDLCD.P11
RUN-TIME: 43 60 3 SECONDS
RUN-TIME RATIO: 164/107=1.5
CORE USED: 34K (67 PAGES)