

DL11-C,-D,-E

DL11-C,D,E OFFLINE TST
CZDLCC0

AH-8525C-MC

COPYRIGHT 75-80
FICHE 1 OF 1

JAN 1980

digital
MADE IN USA



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

.REMA

IDENTIFICATION

PRODUCT CODE: AC-8524C-MC
PRODUCT NAME: CZDLCCO DL11-C,D,E OFLNE TST
PRODUCT DATE: JULY 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: E. CROWLEY/B. BURGESS

COPYRIGHT (C) 1975, 1979 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

| | |
|-----|--|
| 57 | |
| 58 | |
| 59 | |
| 60 | 1.0 PROGRAM PURPOSE (ABSTRACT) |
| 61 | |
| 62 | 2.0 SYSTEM REQUIREMENTS |
| 63 | |
| 64 | 3.0 RELATED DOCUMENTS AND STANDARDS |
| 65 | |
| 66 | 4.0 DIAGNOSTIC HIERARCHY PREREQUISITES |
| 67 | |
| 68 | 5.0 LOADING AND STARTING PROCEDURE |
| 69 | |
| 70 | 6.0 SPECIAL ENVIRONMENTS |
| 71 | |
| 72 | 7.0 PROGRAM OPTIONS |
| 73 | |
| 74 | 8.0 EXECUTION TIMES |
| 75 | |
| 76 | 9.0 ERROR INFORMATION |
| 77 | 9.1 ERROR REPORTING |
| 78 | 9.2 ERROR HALTS |
| 79 | |
| 80 | 10.0 PERFORMANCE AND PROGRESS REPORTS |
| 81 | |
| 82 | 11.0 DEVICE INFORMATION TABLES |
| 83 | |
| 84 | 12.0 SUBROUTINE SUMMARIES |
| 85 | THRU |
| 86 | 12.20 |
| 87 | |
| 88 | 13.0 MISCELLANEOUS |
| 89 | |
| 90 | 14.0 USER SELECTION PROGRAMS |
| 91 | 14.1 PROGRAM #2 DESCRIPTION |
| 92 | 14.2 PROGRAM #3 DESCRIPTION |
| 93 | 14.3 PROGRAM #4 DESCRIPTION |
| 94 | 14.4 PROGRAM #5 DESCRIPTION |
| 95 | |
| 96 | 15.0 PROGRAM FUNCTIONAL FLOW CHARTS |
| 97 | |
| 98 | 16.0 PROGRAM LISTING |
| 99 | |
| 100 | 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:

101
102
103
104
105
106
107
108
109
110
111
112

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

- A. VERIFICATION OF MAINTENANCE BIT
- 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

PAGE 4

BC05C SPECIAL CABLE CONNECTOR
H315A SPECIAL MODEM TEST CONNECTOR

B. SOFTWARE REQUIREMENTS

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

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

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

3.0 RELATED DOCUMENTS AND STANDARDS

- A. PROGRAMMING PRACTICES - DOCUMENT NO. 175-003-009-00
- B. PDP11/40 PROCESSOR HANDBOOK
- C. DL11 ASYNCHRONOUS LINE INTERFACE MANUAL
DOCUMENT NO. DEC-11-HDLAA
- D. PDP-11 MAINDEC SYSMAC UTILITY PACKAGE
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:

PAGE 5

210 - SELECTS PROGRAM #2
220 - SELECTS PROGRAM #3

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

230 - SELECTS PROGRAM #4
240 - SELECTS PROGRAM #5

6.0 SPECIAL ENVIRONMENTS

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

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

| SWITCH | USE |
|------------|---|
| ----- | --- |
| 15-1 OR UP | HALT ON ERROR |
| 14-1 OR UP | LOOP ON TEST |
| 13-1 OR UP | INHIBIT ERROR TYPEOUTS |
| 12-1 OR UP | /C OR /D MODEL BEING TESTED |
| 11-1 OR UP | INHIBIT ITERATIONS |
| 10-1 OR UP | BELL ON ERROR |
| 9-1 OR UP | LOOP ON ERROR |
| 8-1 OR UP | LOOP ON TEST IN 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:
DLVECT:

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

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

8.0 EXECUTION TIMES

EXECUTION TIME IS DEPENDENT ON TYPE OF MEMORY AND

PAGE 6

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(?) 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

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

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 TO ALLOW THE INFORMATION TRANSFER.

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

PAGE 7

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

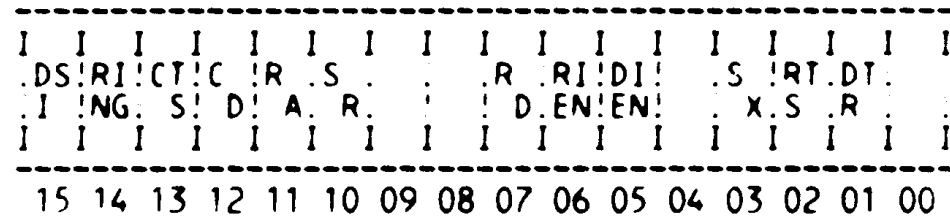
C. IN THE END OF PASS ROUTINE IF MULTIPLE DEVICE TESTING IS BEING CONDUCTED BUT NO DEVICES ARE SHOWN AS ACTIVE.

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:

- BIT 15 DATA SET INTERRUPT
 - 1. INTERRUPT SEQUENCE INITIATED WHEN BIT 05 SET.
 - 2. SETS WHENEVER BITS 10, 11, 12 OR 14 CHANGE STATE
 - 3. CLEARED BY INIT OR READING RCSR

- BIT 14 RING
 - 1. WHEN SET, INDICATES A CONTROL SIGNAL BEING RECEIVED FROM DATASET.

- BIT 13 CLEAR TO SEND
 - 1. WHEN SET INDICATES ON

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

BIT12 CARRIER DETECT

- CONDITION; WHEN CLEAR INDICATES OFF CONDITION.
2. DEPENDENT ON STATE OF 'CTS' SIGNAL FROM DATASET
1. SETS WHEN DATA CARRIER RECEIVED
 2. WHEN CLEAR INDICATES END OF CURRENT TRANSMISSION OR AN ERROR CONDITION.

PAGE 8

BIT11 RECEIVER ACTIVE

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

BIT10 SECONDARY RECEIVE OR SUPERVISORY RECEIVED DATA

1. PROVIDES RECEIVE CAPABILITY, WHEN SET, FOR REVERSE CHANNEL OF REMOTE STATION. SETS WHEN BIT03 IS SET.
2. CLEARED BY INIT

BIT07 RECEIVER DONE

1. SETS WHEN CHARACTER HAS BEEN RECEIVED. WILL 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

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

BIT01 DATA TERMINAL
READY

- 3. CLEARED BY INIT
- 4. ***READ/WRITE BIT***
- 1. WHEN SET, PERMITS CONNECTION TO CHANNEL.
- 2. WHEN CLEAR, DISCONNECTS INTERFACE FROM CHANNEL.
- 3. MUST BE CLEARED BY PROGRAM
- 4. ***READ/WRITE BIT***

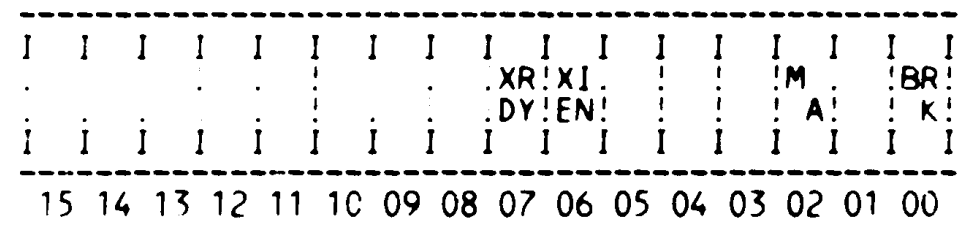
SPECIAL NOTES ON RCSR REGISTER

- 1. ADDRESSES SHOULD FALL IN THE RANGE OF 175610 TO PAGE 9

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 XDBR CAN ACCEPT ANOTHER CHARACTER. WILL INITIATE AN INTERRUPT IF BIT06 ALSO SET.
 - 2. ALSO SET BY INIT
 - 3. CLEARED BY LOADING XDBR
- 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 &

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

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***

BIT00 BREAK

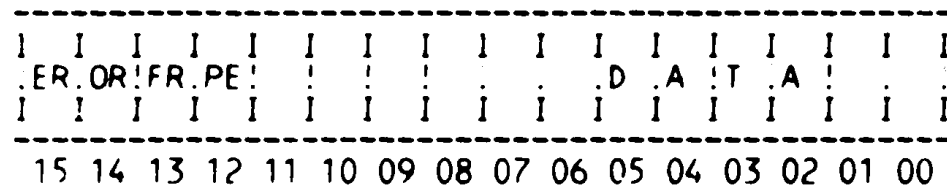
- 1. WHEN SET, TRANSMITS A
CONTINUOUS SPACE TO
EXTERNAL DEVICE
- 2. CLEARED BY INIT
- 3. ***READ/WRITE BIT***

NOTE

PAGE 10

DL11-C AND -D OPTIONS ARE THE SAME.

(. 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

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

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

12.0 SUBROUTINE SUMMARIES

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

PAGE 11

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 \$FOP

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.

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

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'.
- C. LOOP ON TEST SPECIFIED BY TEST NO. APPEARING IN SWR<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

PAGE 12

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.

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

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.

- 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.

PAGE 13

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

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

\$RDDEC 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 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.

PAGE 14

- 4. THE ABOVE THREE(3) VARIABLES ARE SUPPLIED BY .SCMTAG, A 'SYSMAC' PACKAGE UTILITY ROUTINE.

12.10 \$TRAP, \$TRPAD

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THE '\$TRAP' ROUTINE WILL STRIP OFF THE LOWER BYTE OF A TRAP INSTRUCTION AND USE IT TO INDEX THRU THE TRAP TABLE (\$TRPAD) 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 -

| | | |
|---------|--------|--|
| \$TYPE | 104400 | TTY TYPEOUT ROUTINE |
| \$TYPOC | 104402 | TYPE OCTAL # (WITH LEADING ZEROS) |
| \$TYPOS | 104404 | TYPE OCTAL # (NO LEADING ZEROS) |
| \$TYPON | 104406 | TYPE OCTAL # (PER LAST CHARACTER METHOD) |
| \$TYPDS | 104410 | TYPE DECIMAL # (WITH SIGN) |
| \$RDCHR | 104412 | TTY CHARACTER INPUT |
| \$RDLIN | 104414 | TTY STRING INPUT |
| \$RDOCT | 104416 | TTY OCTAL # INPUT |

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

12.11 \$PWRDN, \$PWRUP

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.

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.

PAGE 15

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.

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

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

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

PAGE 16

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

- A. LDOUT1 - IS CALLED TO SET UP THE 'NULL-DEL-MULL' 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.

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

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.
- 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

PAGE 17

- 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

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
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

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.

14.2 PROGRAM #3 DESCRIPTION

PAGE 18

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

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

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
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055

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 INDEF 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

PAGE 19

BETWEEN EACH CHARACTER TRANSMISSION.

14.3 PROGRAM #4 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING.

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
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111

- 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

PAGE 20

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.

4.4 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 TIM OUT' WHEN REFERENCED, AND
3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUF 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

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
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167

ABOVE) WAS A '0' OR A PLAIN 'CARRIAGE
RETURN' THEN THERE IS NO DELAY BETWEEN
CHARACTER TRANSMISSIONS.

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
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223

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,\$TKINT FOLLOWS GETSWR.
CHGC4 - .\$READ ARGUMENTS EXPANDED TO READ '\$READ ,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.

.ENDR @

16740C

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

000001

.TITLE CZDLCCO DL11-C,D,E OFLNE TST
:*COPYRIGHT (C) 1979
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY E. CROWLEY/B. BURGESS
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINLEC-11-DZQAC-C3), JAN 19, 1977.
:*
\$TN-1

.SBTTL OPERATIONAL SWITCH SETTINGS

| SWITCH | USE |
|--------|------------------------|
| 15 | HALT ON ERROR |
| 14 | LOOP ON TEST |
| 13 | INHIBIT ERROR TYPEOUTS |
| 12 | /C OR /D MODEL |
| 11 | INHIBIT ITERATIONS |

```

1224      :*      10      BELL ON ERROR
1225      :*      9      LOOP ON ERROR
1226      :*      8      LOOP ON TEST IN SWR<7:0>
1227
1228      :*      0      CREATION OF DEVICE/S TABLE
1229      :*
1230
1231      .174
1232 000174 000174 000000 001734 001242  DISPREG: .WORD 0      ;SOFTWARE DISPLAY REGISTER
1233 000176 000000 000000 001734 001242  SWREG: .WORD 0      ;SOFTWARE SWITCH REGISTER
1234 000200 012767 001734 001242  MOV #PRG1,STAD      ;ADDRESS OF USER PROGRAM NO. 1
1235 000206 000417 000417 001734 001242  BR STCONT
1236 000210 012767 006424 001232  MOV #PRG2,STAD      ;ADDRESS OF USER PROGRAM NO.2
1237 000216 000413 000413 001232  BR STCONT
1238 000220 012767 006632 001222  MOV #PRG3,STAD      ;ADDRESS OF USER PROGRAM NO. 3
1239 000226 000407 000407 001222  BR STCONT
1240 000230 012767 007050 001212  MOV #PRG4,STAD      ;ADDRESS OF USER PROGRAM NO. 4
1241 000236 000403 000403 001212  BR STCONT
1242 000240 012767 007410 001202  MOV #PRG5,STAD      ;ADDRESS OF USER PROGRAM NO. 5
1243 000246 000137 001452 001202  STCONT: JMP @WBEGIN ;JUMP TO COMMON START
1244
1245      .52      ;INFORMATION LOCATION FOR ACT11
1246 000052 000000 .WORD 0      ;NO POWER FAIL REQUIRED <BIT15=0>
1247      ;IS NOT MEMORY SIZE DEPENDENT <BIT14=0>
1248      ;IS SUITABLE FOR AUTOMATIC OPERATION <BIT13=0>
1249
1250      .SBTTL BASIC DEFINITIONS
1251
1252      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1'00 ***
1253      001100  STACK= 1100
1254      .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
1255      .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL
1256
1257      ;*MISCELLANEOUS DEFINITIONS
1258      000011  HT= 11 ;:CODE FOR HORIZONTAL TAB
1259      000012  LF= 12 ;:CODE FOR LINE FEED
1260      000015  CR= 15 ;:CODE FOR CARRIAGE RETURN
1261      000200  CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
1262      177776  PS 177776 ;:PROCESSOR STATUS WORD
1263      .EQUIV PS,PSW
1264      177774  STKLM= 177774 ;:STACK LIMIT REGISTER
1265      177772  PIRQ 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
1266      177570  DSWR= 177570 ;:HARDWARE SWITCH REGISTER
1267      177570  DDISP= 177570 ;:HARDWARE DISPLAY REGISTER
1268
1269      ;*GENERAL PURPOSE REGISTER DEFINITIONS
1270      000000  R0 %0 ;:GENERAL REGISTER
1271      000001  R1= %1 ;:GENERAL REGISTER
1272      000002  R2= %2 ;:GENERAL REGISTER
1273      000003  R3= %3 ;:GENERAL REGISTER
1274      000004  R4= %4 ;:GENERAL REGISTER
1275      000005  R5= %5 ;:GENERAL REGISTER
1276      000006  R6= %6 ;:GENERAL REGISTER
1277      000007  R7= %7 ;:GENERAL REGISTER
1278      000006  SP %6 ;:STACK POINTER
1279      000007  PC= %7 ;:PROGRAM COUNTER

```


| | | | |
|------|--------|---|---------------------|
| 1280 | | | |
| 1281 | | ;*PRIORITY LEVEL DEFINITIONS | |
| 1282 | 000000 | PR0= 0 | :::PRIORITY LEVEL 0 |
| 1283 | 000040 | PR1= 40 | :::PRIORITY LEVEL 1 |
| 1284 | 000100 | PR2= 100 | :::PRIORITY LEVEL 2 |
| 1285 | 000140 | PR3= 140 | :::PRIORITY LEVEL 3 |
| 1286 | 000200 | PR4= 200 | :::PRIORITY LEVEL 4 |
| 1287 | 000240 | PR5= 240 | :::PRIORITY LEVEL 5 |
| 1288 | 000300 | PR6= 300 | :::PRIORITY LEVEL 6 |
| 1289 | 000340 | PR7= 340 | :::PRIORITY LEVEL 7 |
| 1290 | | | |
| 1291 | | ;'SWITCH REGISTER' SWITCH DEFINITIONS | |
| 1292 | 100000 | SW15= 100000 | |
| 1293 | 040000 | SW14= 40000 | |
| 1294 | 020000 | SW13= 20000 | |
| 1295 | 010000 | SW12= 10000 | |
| 1296 | 004000 | SW11= 4000 | |
| 1297 | 002000 | SW10= 2000 | |
| 1298 | 001000 | SW09= 1000 | |
| 1299 | 000400 | SW08= 400 | |
| 1300 | 000200 | SW07= 200 | |
| 1301 | 000100 | SW06= 100 | |
| 1302 | 000040 | SW05= 40 | |
| 1303 | 000020 | SW04= 20 | |
| 1304 | 000010 | SW03= 10 | |
| 1305 | 000004 | SW02= 4 | |
| 1306 | 000002 | SW01= 2 | |
| 1307 | 000001 | SW00= 1 | |
| 1308 | | .EQUIV SW09,SW9 | |
| 1309 | | .EQUIV SW08,SW8 | |
| 1310 | | .EQUIV SW07,SW7 | |
| 1311 | | .EQUIV SW06,SW6 | |
| 1312 | | .EQUIV SW05,SW5 | |
| 1313 | | .EQUIV SW04,SW4 | |
| 1314 | | .EQUIV SW03,SW3 | |
| 1315 | | .EQUIV SW02,SW2 | |
| 1316 | | .EQUIV SW01,SW1 | |
| 1317 | | .EQUIV SW00,SW0 | |
| 1318 | | | |
| 1319 | | ;*DATA BIT DEFINITIONS (BIT00 TO BIT15) | |
| 1320 | 100000 | BIT15= 100000 | |
| 1321 | 040000 | BIT14= 40000 | |
| 1322 | 020000 | BIT13= 20000 | |
| 1323 | 010000 | BIT12= 10000 | |
| 1324 | 004000 | BIT11= 4000 | |
| 1325 | 002000 | BIT10= 2000 | |
| 1326 | 001000 | BIT09= 1000 | |
| 1327 | 000400 | BIT08= 400 | |
| 1328 | 000200 | BIT07= 200 | |
| 1329 | 000100 | BIT06= 100 | |
| 1330 | 000040 | BIT05= 40 | |
| 1331 | 000020 | BIT04= 20 | |
| 1332 | 000010 | BIT03= 10 | |
| 1333 | 000004 | BIT02= 4 | |
| 1334 | 000002 | BIT01= 2 | |
| 1335 | 000001 | BIT00= 1 | |

```
1336 .EQUIV BIT09,BIT9
1337 .EQUIV BIT08,BIT8
1338 .EQUIV BIT07,BIT7
1339 .EQUIV BIT06,BIT6
1340 .EQUIV BIT05,BIT5
1341 .EQUIV BIT04,BIT4
1342 .EQUIV BIT03,BIT3
1343 .EQUIV BIT02,BIT2
1344 .EQUIV BIT01,BIT1
1345 .EQUIV BIT00,BIT0
1346
1347 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1348 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
1349 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
1350 000014 TBITVEC=14 ;:'T' BIT
1351 000014 TRTVEC= 14 ;:TRACE TRAP
1352 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
1353 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
1354 000024 PWRVEC= 24 ;:POWER FAIL
1355 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
1356 000034 TRAPVEC=34 ;:'TRAP' TRAP
1357 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
1358 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
1359 000240 PIRQVEC-240 ;:PROGRAM INTERRUPT REQUEST VECTOR
1360
```



```
1417 001224 000000 $TMP11: .WORD 0 ;:USER DEFINED
1418 001226 000000 $TMP12: .WORD 0 ;:USER DEFINED
1419 001230 000000 $TMP13: .WORD 0 ;:USER DEFINED
1420 001232 000000 $TMP14: .WORD 0 ;:USER DEFINED
1421 001234 000000 $TMP15: .WORD 0 ;:USER DEFINED
1422 001236 000000 $TMP16: .WORD 0 ;:USER DEFINED
1423 001240 000000 $TMP17: .WORD 0 ;:USER DEFINED
1424 001242 000000 $TIMES: 0 ;:MAX. NUMBER OF ITERATIONS
1425 001244 000000 $ESCAPE:0 ;:ESCAPE ON ERROR ADDRESS
1426 001246 177607 000377 $BELL: .ASCIZ <207><377><377> ;:CODE FOR BELL
1427 001252 077 $QUES: .ASCII /?/ ;:QUESTION MARK
1428 001253 015 $CRLF: .ASCII <15> ;:CARRIAGE RETURN
1429 001254 00C012 $LF: .ASCIZ <12> ;:LINE FEED
1430 ;:*****
1431 ;:THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO
1432 ;:'MORETAGS' AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .SCMTAG
1433
1434
1435 001256 000000 TABFLG: .WORD 0 ;:AN INDICATOR TO SHOW THAT THE
1436 ;:INFORMATION FOR MULTIPLE DEVICE
1437 ;:TESTING HAS ALREADY TRANSPIRED
1438 ;:& 'MAINDEC' NAME HAS BEEN PRINTED
1439 001260 000000 DLBASE: .WORD 0 ;:STORAGE & WORKING LOCATION FOR A DEVICE
1440 ;:RECEIVER STATUS REGISTER ADDRESS
1441 001262 000000 KEEPAD: .WORD 0 ;:STORAGE LOCATION FOR THE 1ST
1442 ;:DEVICE RCSR FROM WHICH
1443 ;:'BASEADD' IS RESTORED AT THE
1444 ;:END OF A COMPLETE PROGRAM PASS.
1445 001264 000000 BASEADD: .WORD 0 ;:STORAGE LOCATION WHICH HOLDS
1446 ;:THE RCSR ADDRESS OF THE 'NEXT'
1447 ;:DEVICE DURING MULTIPLE TESTING
1448 001266 000000 KEEPIV: .WORD 0 ;:STORAGE LOCATION FOR THE 1ST
1449 ;:DEVICE RECEIVER VECTOR FROM
1450 ;:WHICH 'BASEIV' IS RESTORED AT THE
1451 ;:END OF A COMPLETE PROGRAM PASS
1452 001270 000000 BASEIV: .WORD 0 ;:STORAGE LOCATION WHICH HOLDS
1453 ;:THE VECTOR ADDRESS OF THE 'NEXT'
1454 ;:DEVICE DURING MULTIPLE TESTING
1455 001272 000000 MULTD: .WORD 0 ;:FLAG TO INDICATE TO 'END OF PASS'
1456 ;:ROUTINE THAT MULTIPLE DEVICE
1457 ;:TESTING IS BEING CONDUCTED
1458 ;:0=NO, 1=YES
1459 001274 000000 ACTREG: .WORD 0 ;:THIS IS THE DEVICE ACTIVE REGISTER
1460 ;:A BIT IS SET (STARTING AT
1461 ;:BIT0)FOR EACH CONTIGUOUS DEVICE
1462 ;:(A MAX. OF 16) THAT IS TO UNDERGO
1463 ;:TESTING. THIS LOCATION IS
1464 ;:AUTOMATICALLY FILLED BASED ON
1465 ;:USER RESPONSE TO PROGRAM QUESTIONS
1466 001276 000000 ROTADD: .WORD 0 ;:A ROTATING POINTER TO SIGNAL
1467 ;:THE LAST DEVICE TESTED (IF
1468 ;:MULTIPLE DEVICE TESTING WAS BEING
1469 ;:DONE) IF LESS THAN A FULL COMPLE-
1470 ;:MENT OF DEVICES (16)WAS SELECTED
1471 001300 000000 LASTADD: .WORD 0 ;:STORAGE LOCATION FOR THE
1472 ;:RCSR ADDRESS OF THE LAST DEVICE
```

1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490

001302 000000
001304 000000
001306 177740

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 INTERRUPT'G.
;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'.

1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505 001310
1506
1507
1508
1509 001310 015714
1510 001312 015763
1511 001314 016042
1512 001316 000000
1513
1514
1515
1516 001320 016060
1517 001322 016104
1518 001324 016202
1519 001326 000000
1520
1521
1522
1523 001330 016224
1524 001332 016254
1525 001334 016352
1526 001336 000000
1527
1528
1529
1530 001340 016374
1531 001342 016446
1532 001344 016504
1533 001346 000000
1534
1535
1536
1537 001350 016516
1538 001352 016573
1539 001354 016662
1540 001356 000000
1541
1542
1543
1544 001360 015714
1545 001362 016702
1546 001364 016742

```
.SBTTL ERROR POINTER TABLE

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

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

$ERRTB:

;ERROR TABLE ITEM FOR ERROR MESSAGE 1
      EM1      ;;DL11 REGISTER REFERENCE CAUSED TIMEOUT''
      DH1      ;;(PC) (PS) (SP) TEST DEVADR REGADR ''
      DT1      ;;(R7) (PSW) (R6) (R0) (R1) (R2)
      0        ;;PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 2
      EM2      ;;DL11 REGISTER ERROR ''
      DH2      ;;(PC) (PS) (SP) TEST DEVADR REGADR WAS S/B ''
      DT2      ;;(R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4) ''
      0        ;;PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 3
      EM3      ;;DL11 DATA COMPARE ERROR ''
      DH3      ;;(PC) (PS) (SP) TEST WASADR SHBADR WAS S/B ''
      DT3      ;;(R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4) ''
      0        ;;PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 4
      EM4      ;;UNEXPECTED TRAP TO VECTOR AT LOCATION XXX ''
      DH4      ;;(PC) (PS) (SP) TEST ''
      DT4      ;;(R7) (PSW) (R6) (R0) ''
      0        ;;PRINT ALL OCTAL

;ERROR TABLE ITEM FOR ERROR MESSAGE 5
      EM5      ;;DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) ''
      DH5      ;;(PC) (PS) (SP) TEST DEVADR REGADR (REG) ''
      DT5      ;;(R7) (PSW) (R6) (R0) (R1) (R2) (R3) ''
      0

;ERROR TABLE ITEM FOR ERROR MESSAGE 6
      EM6      ;;DL11 REGISTER REFERENCE CAUSED TIMEOUT''
      DH6      ;;(PC) (PS) (SP) REGADR''
      DT6      ;;$ERRPC,$TMP0,$REG6,$REG2
```

```

1547 001366 000000          0          ;PRINT ALL OCTAL
1548
1549          ;ERROR TABLE ITEM FOR ERROR MESSAGE 7
1550
1551 001370 016516          EMS          ;' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) '
1552 001372 016754          DH7          ;' (PC) DEVADR REGADR (REG) '
1553 001374 017014          DT7          ;$ERRPC,$REG1,$REG2,$REG3
1554 001376 000000          0          ;PRINT ALL OCTAL
1555
1556          ;ERROR TABLE ITEM FOR ERROR MESSAGE 10
1557
1558 001400 016224          EM3          ;' DL11 DATA COMPARE ERROR '
1559 001402 017026          DH10         ;' (PC) DEVADR REGADR (REG) S/B '
1560 001404 017074          DT10         ;$ERRPC,$REG1,$REG2,$REG3,$REG4
1561 001406 000000          0          ;PRINT ALL OCTAL

```

```

;:*****
;:          ;D_11 DEFINITIONS
;:*****

```

```

1567 001410 175610          DLRCR: 175610      ;CONTAINS ADDRESS OF RCVR CSR
1568 001412 175612          DLRDBR: 175612      ;CONTAINS ADDRESS OF RCVR DBR
1569 001414 175614          DLXCSR: 175614      ;CONTAINS ADDRESS OF XMIT CSR
1570 001416 175616          DLXDBR: 175616      ;CONTAINS ADDRESS OF XMIT DBR
1571 001420 000300          DLVECT: 300        ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
1572 001422 000000          XFLGO: 0          ;FLAG FOR HARD XMIT ERRORS
1573 001424 000000          RFLGO: 0          ;FLAG FOR HARD RCVR ERRORS
1574 001426 000000          RFLG1: 0          ;FLAG FOR SOFT RCVR ERRORS
1575 001430 000000          RTRY: 0           ;COUNTS NO. OF RETRIES ON SOFT ERRORS
1576 001432 000000          OPTR: 0           ;CONTAINS POINTER TO OUTPUT BUFFER
1577 001434 000000          IPTR: 0           ;CONTAINS POINTER TO INPUT BUFFER
1578 001436 000000          LDOUT: 0          ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
1579 001440 000000          TIMR1: 0          ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
1580 001442 000000          TIMR2: 0
1581 001444 000000          TIMR3: 0
1582 001446 000000          INTFLG: 0         ;DELAY TIMER FOR TEST 10
1583 001450 000000          STAD: 0           ;SOFTWARE INTR. FLAG
                                ;TEMPORARY ADDRESS STORAGE LOC.

```

```

1588 001452 000240          BEGIN: NOP          ;PROGRAM WILL START HERE
1589          .SBTTL INITIALIZE THE COMMON TAGS
1590          ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
1591 001454 012706 001100          MOV      #$CMTAG,R6      ;:FIRST LOCATION TO BE CLEARED
1592 001460 005026          CLR      (R6)+          ;:CLEAR MEMORY LOCATION
1593 001462 022706 001140          CMP      #SWR,R6 ;:DONE?
1594 001466 001374          BNE     .-6             ;:LOOP BACK IF NO
1595 001470 012706 001100          MOV      #STACK,SP      ;:SETUP THE STACK POINTER
1596          ;:INITIALIZE A FEW VECTORS
1597 001474 012737 010406 000020          MOV      #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
1598 001502 012737 000340 000022          MOV      #340,@IOTVEC+2 ;:LEVEL 7
1599 001510 012737 010660 000030          MOV      #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
1600 001516 012737 000340 000032          MOV      #340,@EMTVEC+2 ;:LEVEL 7
1601 001524 012737 013630 000034          MOV      #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
1602 001532 012737 000340 000036          MOV      #340,@TRAPVEC+2;LEVEL 7

```

```

1603 001540 012737 013714 000024      MOV    #SPWRDN,@PWRVEC  ;;POWER FAILURE VECTOR
1604 001546 012737 000340 000026      MOV    #340,@PWRVEC+2  ;;LEVEL 7
1605 001554 005067 177462                CLR    $TIMES           ;;INITIALIZE NUMBER OF ITERATIONS
1606 001560 005067 177460                CLR    $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1607 001564 112767 000001 177323      MOV    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1608 001572 012767 001572 177306      MOV    #,$SLPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1609 001600 012767 001600 177302      MOV    #,$SLPERR       ;;SETUP THE ERROR LOOP ADDRESS
1610                                     ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1611                                     ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
1612 001606 013746 000004                MOV    @ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
1613 001612 012737 001646 000004      MOV    #64$,@ERRVEC   ;;SET UP ERROR VECTOR
1614 001620 012767 177570 177312      MOV    #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1615 001626 012767 177570 177306      MOV    #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1616 001634 022777 177777 177276      CMP    #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1617 001642 001012                BNE    66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1618                                     ;;AND THE HARDWARE SWR IS NOT '-1'
1619 001644 000403                BR     65$             ;;BRANCH IF NO TIMEOUT
1620 001646 012716 001654 64$:          MOV    #65$,(SP)      ;;SET UP FOR TRAP RETURN
1621 001652 000002                RTI
1622 001654 012767 000176 177256 65$:          MOV    #SWREG,SWR     ;;POINT TO SOFTWARE SWR
1623 001662 012767 000174 177252      MOV    #DISPREG,DISPLAY
1624 001670 012637 000004 66$:          MOV    (SP)+,@ERRVEC  ;;RESTORE ERROR VECTOR
1625
1626                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1627 001674 005737 000042                TST    @#42           ;;ARE WE RUNNING UNDER XXDP/ACT?
1628 001700 001006                BNE    67$             ;;BRANCH IF YES
1629 001702 026727 177232 000176      CMP    SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
1630 001710 001005                BNE    68$             ;;BRANCH IF NO
1631 001712 104406                GTSWR                 ;;GET SOFT-SWR SETTINGS
1632 001714 000403                BR     68$
1633 001716 112767 000001 177210 67$:          MOV    #1,$AUTOB     ;;SET AUTO-MODE INDICATOR
1634 001724 68$:
1635 001724 004767 007714                JSR    PC,$TKINT      ;;SET TTY INTERRUPT
1636 001730 000177 177514                JMP    @STAD          ;;JUMP TO SELECTED TEST
1637 001734 005067 177332  PRG1:      CLR    MULTD         ;;CLEAR MULTIPLE DEVICE
1638                                     ;;TESTING FLAG
1639 001740 005067 177312                CLR    TABFLG        ;;CLEAR TABLE CREATION FLAG
1640 001744 012767 000010 177262      MOV    #8,$TMP15     ;;SET CHARACTER LENGTH DESIGNATOR
1641                                     ;;FOR 8 BITS --- THIS IS THE DEFAULT
1642                                     ;;LENGTH ASSUMED BY THE PROGRAM
1643                                     ;;UNLESS THE USER CHANGES IT THRU
1644                                     ;;THE QUESTION AND ANSWER CYCLE
1645                                     ;;INITIATED BY SETTING SW<0> TO A '
1646 001752 012767 000200 177322      MOV    #200,DLPRI    ;;SET STANDARD PRIORITY LEVEL
1647                                     ;;FOR DEVICE
1648 001760 032777 000400 177152      BIT    #SW8,@SWR     ;;IS THE 'LOOP ON TEST' SWITCH SET?
1649 001766 001411                BEQ    1$             ;;BRANCH IF NOT
1650
1651                                     ;;IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
1652                                     ;;INSTRUCTION THUS BYPASSING TABLE CREATION
1653
1654                                     ;;IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
1655                                     ;;THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
1656                                     ;;WITH THE DESIRED DEVICE REGISTER VALUES:
1657
1658                                     .....

```



```

1659          :          UNDER      ;DL11 DEFINITIONS          ABOVE
1660          :          *****
1661          :
1662          :          DLRCSE      PATCH THE ADDRESS OF THE RCVR CSR
1663          :          DLRDBR:    PATCH THE ADDRESS OF THE RCVR DBR
1664          :          DLXCSR:    PATCH THE ADDRESS OF THE XMIT CSR
1665          :          DLXDBR:    PATCH THE ADDRESS OF THE XMIT DBR
1666          :          DLVECT:    PATCH THE VECTOR ADDRESS OF THIS DL11
1667          :
1668 001770 104401 017417          :          TYPE,      STMES      ;PRINT OUT 'MAINDEC' NAME
1669 001774 104401 021453          :          TYPE,      FAILSA     ;TYPE FAILSAFE MESSAGE
1670 002000 104000          :          ERROR      +0         ;TYPE OUT THE PC VALUE
1671 002002 104401 022031          :          TYPE,      PCMSG      ;FOLLOWED BY PC
1672 002006 000000          :          HALT       ;WAIT FOR USER TO RESPOND
1673 002010 000443          :          BR         ONCE       ;GO TO TEST DEVICE PATCHED IN BY USER
1674 002012          :
1675          :          1$:
1676          :          :ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
1677          :          :AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
1678          :          :WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
1679          :          :TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
1680          :          :HE CAN RUN THE DEFAULT DEVICE (1ST RECEIVER
1681          :          :STATUS REGISTER ADDRESS 175610)
1682          :          :NOTE: IF THIS IS NOT SUITABLE THE USER WILL
1683          :          :HAVE TO SET SW0=1 (OR UP) IN ORDER TO
1684          :          :RECREATE THE TABLE HE DESTROYED FROM
1685          :          :ABOVE
1685 002012 012767 175610 177240          :          MOV        #175610,DLBASE ;1ST POSSIBLE RECEIVER CSR
1686 002020 004767 005754          :          JSR        PC,DLADDR     ;FORM DL ADDRESSES FOR
1687          :          :1ST POSSIBLE DEVICE
1688 002024 012767 000300 177366          :          MOV        #300,DLVECT   ;1ST POSSIBLE INTERRUPT VECTOR
1689 002032 005067 177220          :          CLR        TABFLG       ;CLEAR TABLE CREATION FLAG
1690          :
1691 002036 012706 001100          :          RESTR: MOV     #STACK,SP   ;SET UP STACK POINTER
1692 002042 012737 015602 000004          :          MOV     #BUSERR,@ERRVEC   ;SET UP BUS ERROR VECTOR
1693 002050 012737 000340 000006          :          MOV     #340,@ERRVEC+2
1694 002056 012737 015626 000010          :          MOV     #RSVERR,@RESVEC   ;SET UP RSVD INSTR. VECTOR
1695 002064 012737 000340 000012          :          MOV     #340,@RESVEC+2
1696          :
1697          :          :THIS NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
1698          :          :WILL TAKE PLACE I.E.-
1699          :          :A) HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
1700          :          :B) IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
1701 002072 105767 177160          :          TSTB     TABFLG         ;HAS TABLE CREATION BEEN PERFORMED?
1702 002076 001010          :          BNE     ONCE           ;BRANCH IF YES TO SKIP 'MAINDEC
1703          :          :TITLE' MESSAGE
1704 002100 104401 017417          :          TYPE     ,STMES        ;OTHERWISE, PRINT OUT 'MAINDEC'
1705          :          :NAME
1706 002104 105167 177146          :          COMB     TABFLG        ;IF TABLE CREATION HAS NOT BEEN
1707          :          :PERFORMED, THEN SET FLAG, AND DO SO
1708 002110 032777 000001 177022          :          BIT     #SW0,@SWR      ;THE PROGRAM HAS OBVIOUSLY BEEN
1709          :          :RESTARTED - DOES USER WISH TO
1710          :          :RESELECT VECTOR AND CONTROL REGISTER
1711          :          :ADDRESSES I.E. - CREATE A NEW TABLE?
1712 002116 001012          :          BNE     GO             ;BRANCH IF YES
1713 002120 005077 177270          :          CLR     @DLXCSR        ;CLEAR OUT BOTH CSR'S
1714 002124 005077 177260          :          CLR     @DLRCSR

```

```
1715 002130 005777 177256      TST   @DLRDBR      ;FLUSH RCVR 'DONE' BIT
1716 002134 005777 177252      TST   @DLRDBR
1717 002140 000167 000670      JMP   TST1         ;OTHERWISE, GO WITH EXISTING
1718                                ;TABLE OR NOT USE ANY TABLE AT
1719                                ;ALL WHICHEVER THE CASE MAY BE
1720                                ;(DEFAULT CASE IS 1ST POSSIBLE
1721                                ;DEVICE)
1722                                ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES
1723                                ;A) TO RUN MULTIPLE DEVICES
1724                                ;B) TO CREATE A NEW TABLE TO RUN FROM, OR
1725                                ;C) TO CHANGE THE CHARACTER LENGTH
1726 002144 104401 020013      GO:   TYPE, LENGTH ;ASK USER FOR THE CHARACTER LENGTH
1727                                ;FOR WHICH HIS DEVICE IS SET
1728 002150 104413                                ;ACCEPT THE ANSWER TYPED BY USER
1729                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1730                                MOV   (SP)+,R0      ;GET THE ANSWER TYPED
1731 002152 012600                                ;IS THE NUMBER TOO HIGH?
1732 002154 020027 000010      CMP   R0,#8        ;IF YES - GO TO RETRY SITUATION
1733 002160 101114                                ;IS THE NUMBER TOO LOW?
1734 002162 020027 000005      CMP   R0,#5        ;IF YES - GO TO RETRY SITUATION
1735 002166 103511                                ;THE VALUE TYPED IS OK
1736 002170 010067 177040      BLO  RETRY         ;STORE FOR FUTURE USE
1737 002174 104401 020075      MOV   R0,$TMP15    ;ASK USER IF HE WISHES TO TEST OTHER
1738                                ;THAN THE DEFAULT DEVICE
1739                                RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1740 002200 104412                                ;LOOK AT THE ANSWER
1741 002202 005726                                BNE  1$            ;BRANCH IF REPLY WAS YES
1742 002204 001002                                ;OTHERWISE, SKIP REST OF INTERROGATION
1743 002206 000137 002770      JMP   @#FLUSH      ;START RESTORATION OF TRAPCATCHER
1744 002212 012700 000300      1$:  MOV   #300,R0    ;AREA FROM LOCATIONS 300 TO 776
1745 002216 012701 000302      MOV   #302,R1      ;SO THAT WE CREATE THE MULTIPLE
1746 002222 012702 000004      MOV   #4,R2        ;DEVICES TABLE WITH A CLEAN SLATE
1747 002226 010110      2$:  MOV   R1,(R0)    ;
1748 002230 005011                                ;
1749 002232 060200                                ;
1750 002234 060201                                ;
1751 002236 022701 001000      ADD  R2,R1         ;
1752 002242 002771                                ;
1753                                ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
1754                                ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
1755                                ;ADDRESS
1756 002244 104401 020202      FIRSTD: TYPE ,MFIRSTD ;ASK USER FOR THE RECEIVER CONTROL
1757                                ;REGISTER ADDRESS OF HIS FIRST
1758                                ;DEVICE
1759                                RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1760                                ;AND STORE ON TOP OF STACK
1761                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1762 002252 012600                                MOV   (SP)+,R0      ;GET THE ANSWER TYPED
1763 002254 020027 176170      CMP   R0,#176170  ;IS THE NUMBER TOO HIGH?
1764 002260 101060                                ;IF YES-GO TO RETRY SITUATION
1765 002262 020027 175610      CMP   R0,#175610  ;IS THE NUMBER TOO LOW?
1766 002266 103455                                ;IF YES - GO TO RETRY SITUATION
1767 002270 132700 000001      BITB #BIT0,R0     ;NUMBER IS IN RANGE BUT IS IT
1768                                ;ON AN EVEN BOUNDARY?
1769 002274 001052                                BNE  RETRY0        ;IF NO - GO TO RETRY SITUATION
1770 002276 032700 000007      ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1771                                ;BIT #7,R0 ;WAS THE LEAST SIGNIFICANT DIGIT
```

```

1771                                     ;USER RESPONSE EQUAL TO A ZERO?
1772 002302 001047                       BNE   RETRY0   ;BRANCH IF NOT
1773 002304 010067 176750                 MOV   RO,DLBASE ;THE 1ST ADDRESS VALUE TYPED IS OK
1774                                     ;STORE FOR FUTURE USE
1775                                     ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
1776 002310 016767 176744 176744         MOV   DLBASE,KEEPADD ;GET 1ST ADDRESS VALUE
1777 002316 004767 005456                 JSR   PC,DLADDR   ;GO FORM DL ADDRESSES FOR
1778                                     ;1ST DEVICE SELECTED
1779 002322 016767 176734 176734         MOV   KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
1780 002330 104401 020270                 VECT:  TYPE   ,MVECT ;ASK USER FOR A VECTOR ADDRESS
1781 002334 104412                         RDOCT ;ACCEPT THE ANSWER TYPED BY USER
1782                                     ;AND STORE ON TOP OF STACK
1783                                     ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1784 002336 012600                         MOV   (SP)+,RO    ;GET THE ANSWER TYPED
1785 002340 020027 000776                 CMP   RO,#776    ;IS THE NUMBER TOO HIGH?
1786 002344 101032                         BHI   RETRY1     ;IF YES - GO TO RETRY SITUATION
1787 002346 020027 000300                 CMP   RO,#300    ;IS THE NUMBER TOO LOW?
1788 002352 103427                         BLO   RETRY1     ;IF YES - GO TO RETRY SITUATION
1789 002354 132700 000001                 BITB  #BIT0,RO   ;NUMBER IS IN RANGE BUT IS IT
1790                                     ;ON AN EVEN BOUNDARY?
1791 002360 001024                         BNE   RETRY1     ;IF NO - GO TO RETRY SITUATION
1792                                     ;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
1793 002362 032700 000007                 BIT   #7,RO      ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1794                                     ;USER RESPONSE EQUAL TO A ZERO?
1795 002366 001021                         BNE   RETRY1     ;BRANCH IF NOT
1796 002370 010067 177024                 MOV   RO,DLVECT  ;THE VECTOR VALUE TYPED IS OK
1797                                     ;STORE FOR FUTURE USE
1798 002374 016767 177020 176664         MOV   DLVECT,KEEPIV ;GET THE FIRST VECTOR VALUE
1799 002402 016767 177012 176660         MOV   DLVECT,BASEIV ;SAVE FIRST VECTOR VALUE
1800 002410 000414                         BR    HOWMANY    ;GO TO SEE IF USER WANTS MORE
1801                                     ;THAN 1 DEVICE
1802 002412 104401 001252                 RETRY: TYPE,    $QUES ;TYPE '?' INDICATING USER TYPED
1803                                     ;SOMETHING WRONG FOR CHARACTER LENGTH
1804 002416 000167 177522                 JMP   GO         ;GO BACK TO REISSUE QUESTION
1805 002422 104401 001252                 RETRY0: TYPE   , $QUES ;TYPE '?' INDICATING USER TYPE
1806                                     ;SOMETHING WRONG FOR 1ST ADDRESS
1807 002426 000167 177612                 JMP   FIRSTD    ;GO BACK TO REISSUE QUESTION
1808 002432 104401 001252                 RETRY1: TYPE   , $QUES ;TYPE '?' INDICATING USER TYPED
1809                                     ;SOMETHING WRONG FOR VECTOR
1810 002436 000167 177666                 JMP   VECT      ;GO BACK TO REISSUE QUESTION
1811 002442 104401 020346                 HOWMANY:TYPE   ,MULDEV ;ASK USER IF HE WISHES TO RUN
1812                                     ;MULTIPLE DEVICES
1813                                     ;ACCEPT THE ANSWER TYPED BY USER
1814                                     ;AND STORE ON TOP OF STACK
1815 002450 012600                         MOV   (SP)+,RO  ;GET THE ANSWER TYPED
1816 002452 005700                         TST   RO        ;WAS THE ANSWER YES?
1817 002454 001003                         BNE   1$        ;BRANCH IF IT WAS
1818 002456 005067 176610                 CLR   MULTD     ;OTHERWISE, INITIALIZE FLAG TO
1819                                     ;INDICATE NON-MULTIPLE DEVICES
1820 002462 000402                         BR    2$        ;SKIP NEXT INSTRUCTION
1821 002464 105167 176602                 1$:  COMB   MULTD ;INITIALIZE FLAG TO INDICATE
1822                                     ;RUNNING OF MULTIPLE DEVICES
1823 002470                                     2$:
1824 002470 105767 176576                 TSTB  MULTD     ;ARE THERE MULTIPLE DEVICES ON
1825                                     ;THE SYSTEM?
1826 002474 100406                         BMI   LASTD    ;IF SO, GO TO ASK NEXT QUESTION

```



```

1883 002660 104401 020531          TYPE      ,MRANGE          ;INFORM USER TO CHECK AND RETYPE
1884                                RDOCT                ;THE LAST DEVICE RCSR ADDRESS
1885 002664 104412                                ;ACCEPT THE ANSWER TYPED BY USER
1886                                JMP      1$              ;AND STORE ON TOP OF STACK
1887 002666 000167 177626
1888 002672
1889                                CONQUES:
1890                                ;IF WE HAVE REACHED THIS PORTION WE KNOW:
1891                                ;A) THE 'RXCSR' ADDRESS OF THE 1ST DEVICE
1892                                ;B) THE 'RXCSR' ADDRESS OF THE LAST DEVICE, SAND
1893                                ;C) THE INTERRUPT VECTOR OF THE 1ST DEVICE
1894                                ;NOW LET'S FIND THE PRIORITY LEVEL
1894 002672 104401 020615          TYPE      ,PLEVEL          ;ASK USER FOR PRIORITY LEVEL
1895 002676 104412                                RDOCT                ;ACCEPT ANSWER TYPED BY USER AND
1896                                ;STORE ON TOP OF STACK
1897 002700 012600                                MOV      (SP)+,R0      ;GET THE ANSWER TYPED
1898 002702 020027 000007          CMP      R0,#7         ;IS THE NUMBER TOO HIGH?
1899 002706 101046                                BHI     RETRY3         ;IF YES - GO TO RETRY SITUATION
1900 002710 020027 000004          CMP      R0,#4         ;IS THE NUMBER TOO LOW?
1901 002714 103443                                BLO     RETRY3         ;IF YES GO TO RETRY SITUATION
1902 002716 010067 176360          MOV      R0,DLPRI     ;THE PRIORITY TYPED IN IS OK
1903                                ;STORE FOR FUTURE USE
1904
1905                                ;THIS SECTION WILL CALCULATE THE PRIORITY LEVEL FOR THE
1906                                ;PROCESSOR BASED ON THE USER RESPONSE FOR PRIORITY LEVEL OF THE
1907                                ;DEVICE
1908 002722 006367 176354          ASL     DLPRI         ;FORM BITS <7-5> OF PSW
1909 002726 006367 176350          ASL     DLPRI
1910 002732 006367 176344          ASL     DLPRI
1911 002736 006367 176340          ASL     DLPRI
1912 002742 006367 176334          ASL     DLPRI
1913 002746 016767 176330 176330  MOV      DLPRI,LESS1  ;START TO FORM LEVEL TO ALLOW
1914                                ;INTERRUPTS
1915 002754 162767 000001 176322  SUB      #1,LESS1     ;DROP DEVICE LEVEL PRIORITY
1916                                ;BY 1 LEVEL FOR PSW
1917 002762 042767 000037 176314  BIC     #37,LESS1    ;MAKE SURE THE T,N,Z,V & C
1918                                ;BITS FOR THE PROCESSOR ARE CLEAR
1919 002770 005077 176420          FLUSH: CLR     @DLXCSR  ;CLEAR OUT BOTH CSR'S
1920 002774 005077 176410          CLR     @DLRCSR
1921 003000 005777 176406          TST     @DLRDBR
1922 003004 005777 176402          TST     @DLRDBR      ;FLUSH RCVR 'DONE' BIT
1923 003010 000167 000020          JMP     TST1
1924 003014 104401 001252          RETRY2: TYPE     ,SQUES ;BEGIN TESTING
1925                                ;TYPE '?' INDICATING USER TYPED
1926 003020 000167 177466          JMP     LASTD        ;SOMETHING WRONG FOR LAST ADDRESS
1927 003024 104401 001252          RETRY3: TYPE     ,SQUE  ;GO BACK TO REISSUE QUESTION
1928                                ;TYPE '?' INDICATING USER TYPED
1929 003030 000167 177636          JMP     CONQUES      ;SOMETHING WRONG FOR PRIORITY
1930                                ;GO BACK TO REISSUE QUESTION
1931
1932                                ;*****
1933                                ;*TEST 1 TEST THAT REFERENCE TO RCSR DOES NOT CAUSE TIMEOUT*
1934                                ;*****
1935 003034 000004          TST1: SCOPE
1936 003036 016746 174742          MOV     ERRVEC,-(SP)  ;SAVE THE TIMEOUT VECTOR
1937 003042 012767 003060 174734  MOV     #1$,ERRVEC   ;GO TO 1$ IF TIMEOUT
1938 003050 016702 176334          MOV     DLRCR,R2     ;REGADR - RCSR ADR

```

```

1939 003054 005712          TST      (R2)          ;USE REGADR ON BUS
1940 003056 000407          BR       3$            ;;<GO TO NEXT TEST IF NO TIMEOUT>
1941 003060 004767 011614    1$:     JSR      PC,SUERT1 ;GO SET UP ERROR INFO
1942 003064 012767 003074 176152    MOV     #2$, $ESCAPE ;RETURN TO 2$ AFTER ERROR PRINT
1943 003072 104001          ERROR+1 ;DL REFERENCE CAUSED BUS TIMEOUT
1944 003074 022626          2$:     CMP      (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
1945 003076 012667 174702    3$:     MOV     (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
1946
1947
1948
1949
1950 003102 000004          TST2:   SCOPE
1951 003104 016746 174674    MOV     ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
1952 003110 012767 003126 174666    MOV     #1$,ERRVEC ;GO TO 1$ IF TIMEOUT
1953 003116 016702 176272    MOV     DLXCSR,R2 ;REGADR - XCSR ADR
1954 003122 005712          TST      (R2)          ;USE REGADR ON BUS
1955 003124 000407          BR       3$            ;;<GO TO NEXT TEST IF NO TIMEOUT>
1956 003126 004767 011546    1$:     JSR      PC,SUERT1 ;GO SET UP ERROR INFO
1957 003132 012767 003142 176104    MOV     #2$, $ESCAPE ;RETURN TO 2$ AFTER ERROR PRINT
1958 003140 104001          ERROR+1 ;DL REFERENCE CAUSED BUS TIMEOUT
1959 003142 022626          2$:     CMP      (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
1960 003144 012667 174634    3$:     MOV     (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
1961
1962
1963
1964
1965 003150 000004          TST3:   SCOPE
1966 003152 016746 174626    MOV     ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
1967 003156 012767 003174 174620    MOV     #1$,ERRVEC ;GO TO 1$ IF TIMEOUT
1968 003164 016702 176222    MOV     DLRDBR,R2 ;REGADR = RDBR ADR
1969 003170 005712          TST      (R2)          ;USE REGADR ON BUS
1970 003172 000407          BR       3$            ;;<GO TO NEXT TEST IF NO TIMEOUT>
1971 003174 004767 011500    1$:     JSR      PC,SUERT1 ;GO SET UP ERROR INFO
1972 003200 012767 003210 176036    MOV     #2$, $ESCAPE ;RETURN TO 2$ AFTER ERROR PRINT
1973 003206 104001          ERROR+1 ;DL REFERENCE CAUSED BUS TIMEOUT
1974 003210 022626          2$:     CMP      (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
1975 003212 012667 174566    3$:     MOV     (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
1976
1977
1978
1979
1980 003216 000004          TST4:   SCOPE
1981 003220 016746 174560    MOV     ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
1982 003224 012767 003242 174552    MOV     #1$,ERRVEC ;GO TO 1$ IF TIMEOUT
1983 003232 016702 176160    MOV     DLXDDBR,R2 ;REGADR = XDDBR ADR
1984 003236 005712          TST      (R2)          ;USE REGADR ON BUS
1985 003240 000407          BR       3$            ;;<GO TO NEXT TEST IF NO TIMEOUT>
1986 003242 004767 011432    1$:     JSR      PC,SUERT1 ;GO SET UP ERROR INFO
1987 003246 012767 003256 175770    MOV     #2$, $ESCAPE ;RETURN TO 2$ AFTER ERROR PRINT
1988 003254 104001          ERROR+1 ;DL REFERENCE CAUSED BUS TIMEOUT
1989 003256 022626          2$:     CMP      (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
1990 003260 012667 174520    3$:     MOV     (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
1991
1992
1993
1994

```

```

*****
*TEST 5      TEST THAT RCSR IS ALL ZEROES ON ENTRY
*****

```

1995 003264 000004
 1996 003266 005004
 1997 003270 016702 176114
 1998 003274 020412
 1999 003276 001403
 2000 003300 004767 011314
 2001 003304 104002
 2002
 2003
 2004
 2005 003306 000004
 2006 003310 012704 000200
 2007 003314 016702 176074
 2008 003320 020412
 2009 003322 001403
 2010 003324 004767 011270
 2011 003330 104002
 2012
 2013
 2014
 2015 003332 000004
 2016 003334 012704 000204
 2017 003340 016702 176050
 2018 003344 052712 000004
 2019 003350 020412
 2020 003352 001403
 2021 003354 004767 011240
 2022 003360 104002
 2023 003362 012704 000200
 2024 003366 042712 000004
 2025 003372 020412
 2026 003374 001403
 2027 003376 004767 011216
 2028 003402 104002
 2029
 2030
 2031
 2032 003404 000004
 2033 003406 005067 176034
 2034 003412 012767 000001 176024
 2035 003420 016705 175774
 2036 003424 012765 003520 000004
 2037 003432 016765 175644 000006
 2038 003440 005005
 2039 003442 012704 000200
 2040 003446 016702 175742
 2041 003452 052712 000100
 2042 003456 005767 175764
 2043 003462 001023
 2044 003464 005305
 2045 003466 001373
 2046 003470 005367 175750
 2047 003474 001770
 2048 003476 012704 000300
 2049 003502 004767 011112
 2050 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$:    MOV      #200,R4          ;RESULT IN XCSR S/B  000200
        BIC      #BIT2,(R2)       ;NOW CLEAR THE 'MAINT' BIT
        CMP      R4,(R2)         ;RESULT IN XCSR OK ??
        BEQ      TST10           ;;<BR IF YES>
        JSR      PC,SUER2        ;GO SET UP ERROR INFO
        ERROR+2                   ;MAINT BIT FAILED TO CLEAR PROPERLY
;*****
;*TEST 10     TEST THAT XMIT I.E. CAN CAUSE AN INTR
;*****
TST10: SCOPE
        CLR      INTFLG          ;INIT SOFTWARE INTR FLAG
        MOV      #1,TIMR3        ;SET TIMER FOR DELAY
        MOV      DLVECT,R5       ;GET VECTOR ADDRESS
        MOV      #2$,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
1$:    TST      INTFLG           ;DID INTR OCCUR YET ??
        BNE     3$              ;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)

```

2051 003514 104002          ERROR+2          ;INTR. FAILED
2052 003516                4$:
2053 003516 000412          BR      TST11          ;;<GO TO NEXT TEST>
2054 003520 005167 175722  2$:      COM      INTFLG          ;SET THE SOFTWARE FLAG
2055 003524 042712 000100  BIC      #100,(R2)     ;TURN OFF I.E. BIT
2056 003530 000002          RTI                    ;RETURN CONTROL TO INTR. ROUTINE
2057 003532 020412          3$:      CMP      R4,(R2)      ;RESULT IN XCSR OK ??
2058 003534 001403          BEQ      TST11         ;;<BR IF YES>
2059 003536 004767 011056  JSR      PC,SUER2     ;GO SET UP ERROR INFO.
2060 003542 104002          ERROR+2          ;XMIT INTR. NOT SERVICED PROPERLY
2061
2062 *****
2063 *TEST 11      TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
2064 *****
TST11:  SCOPE
2064 003544 000004          MOV      #100,R4      ;RESULT IN RCSR S/B = 000100
2065 003546 012704 000100  MOV      DLRCR,R2     ;REGADR = RCSR ADR
2066 003552 016702 175632  BIS      #BIT6,(R2)   ;SET I.E. BIT
2067 003556 052712 000100  CMP      R4,(R2)     ;DID IT SET PROPERLY ??
2068 003562 020412          BEQ      1$          ;;<BR IF YES>
2069 003564 001403          JSR      PC,SUER2     ;GO SET UP ERROR INFO.
2070 003566 004767 011026  ERROR+2          ;RCVR I.E. BIT FAILED TO SET PROPERLY
2071 003572 104002          1$:      CLR      R4          ;RESULT IN RCSR S/B = 000000
2072 003574 005004          BIC      #BIT6,(R2)   ;CLEAR THE I.E. BIT
2073 003576 042712 000100  CMP      R4,(R2)     ;DID IT CLEAR PROPERLY ??
2074 003602 020412          BEQ      TST12         ;;<BR IF YES>
2075 003604 001403          JSR      PC,SUER2     ;GO SET UP ERROR INFO
2076 003606 004767 011006  ERROR+2          ;RCVR I.E. BIT FAILED TO CLEAR PROPERLY
2077 003612 104002
2078 *****
2079 *TEST 12      TEST THAT RCVR 'DONE' CAN GENERATE AN INTR.
2080 *****
TST12:  SCOPE
2081 003614 000004          MOV      DLVECT,R5    ;GET THE VECTOR ADDRESS
2082 003616 016705 175576  MOV      #3$(R5)+     ;GO TO 3$ ON RCVR INTR.
2083 003622 012725 004000  MOV      DLPRI,(R5)   ;AT LEVEL 4
2084 003626 016715 175450  CLR      INTFLG       ;INIT THE SOFTWARE FLAG
2085 003632 005067 175610  CLR      R5           ;INIT INTR. TIMER
2086 003636 005005          CLR      $TMP1        ;INIT WHERE DATA WILL BE STORED
2087 003640 105067 175340  MOV      DLRCR,R2     ;REGADR = RCSR ADR
2088 003644 016702 175540  CLR      (R2)         ;INIT THE RCSR TO 000000
2089 003650 005012          BIS      #BIT6,(R2)   ;ENABLE RCVR INTERRUPTS
2090 003652 052712 000100  BIS      #BIT2,4(R2)  ;NOW TURN ON MAINT MODE
2091 003656 052762 000004 000004  MOV      #252,$TMP1   ;GET DATA PATTERN AND
2092 003664 112767 000252 175312  JSR      PC,UPMASK    ;GO MASK OFF BITS AS A FUNCTION OF
2093 003672 004767 011614          ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2094
2095 003676 116700 175330          MOV      $TMP14,R0    ;SAVE DATA PATTERN FOR FURTHER USE
2096 003702 116762 175324 000006  MOV      $TMP14,6(R2) ;LOAD XMIT BUFFER REG.
2097 003710 005767 175532          1$:      TST      INTFLG       ;DID RCVR INTR. YET ??
2098 003714 001044          BNE      4$          ;BR IF IT DID
2099 003716 005305          DEC      R5           ;COUNT THE TIMER
2100 003720 001373          BNE      1$          ;BR IF NO TIMEOUT
2101 003722 013767 177776 175252  MOV      @PSW,$TMP0   ;SAVE ERROR PSW
2102 003730 042762 000004 000004  BIC      #BIT2,4(R2)  ;DISABLE MAINT MODE
2103 003736 042712 000100          BIC      #100,(R2)   ;DISABLE RCVR INTR.
2104 003742 010667 175230          MOV      SP,$REG6    ;SAVE THE ERROR SP
2105 003746 010201          MOV      R2,R1       ;DEVADR = RCSR ADR
2106 003750 011203          MOV      (R2),R3     ;GET THE WAS DATA

```



```

2107 003752 012704 000200      MOV      #200,R4      ;[RCSR] S/B - 000200
2108 003756 004767 010664      JSR      PC,SUERR1   ;GO SET UP ERROR INFO.
2109 003762 012767 003772 175254  MOV      #2$,S-SCAPE ;RETURN TO 2$ AFTER ERROR ALWAYS
2110 003770 104002                ERROR+2 ;RCVR INTERRUPT FAILED
2111 003772 005762 000002      2$:     TST      2(R2)    ;REFERENCE RCVR DATA BUFFER
2112                ;TO CLEAR RCSR IN CASE RCVR
2113                ;INTERRUPTS COULD NOT BE ENABLED
2114 003776 000437                BR      TST13        ;;<GO TO NEXT TEST>
2115 004000 042762 000004 000004 3$:     BIC      #BIT2,4(R2) ;DISABLE THE MAINT MODE
2116 004006 116267 000002 175170  MOVB    2(R2), $TMP1 ;GET THE RECEIVED DATA
2117 004014 042712 000100      BIC      #BIT6,(R2)  ;TURN OFF RCVR INTR. ENAB
2118 004020 005167 175422      COM     INTFLG      ;SET THE SOFTWARE FLAG
2119 004024 000002                RTI                    ;RETURN TO MAINLINE
2120 004026 005004                4$:     CLR      R4      ;[RCSR] S/B=0
2121 004030 005712                TST      (R2)        ;IS IT ALL ZEROES ??
2122 004032 001403                BEQ     5$          ;;<BR IF YES>
2123 004034 004767 010560      JSR      PC,SUER2   ;GO SET UP ERROR INFO
2124 004040 104002                ERROR+2 ;RCVR INTR NOT SERVICED PROPERLY
2125 004042 016701 175344      5$:     MOV      DLRDBR,R1 ;SAVE WAS ADDRESS
2126 004046 016702 175344      MOV      DLXDBR,R2  ;SAVE THE S/B ADDRESS
2127 004052 004767 011434      JSR      PC,UPMASK  ;GET THE WAS DATA AND
2128                ;GO MASK OFF BITS AS A FUNCTION OF
2129                ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2130 004056 116703 175150      MOVB    $TMP14,R3   ;SET UP FOR ERROR CHECKING
2131 004062 110004                MOVB    R0,R4       ;GET THE S/B DATA
2132 004064 020403                CMP     R4,R3       ;WAS = S/B ??
2133 004066 001403                BEQ     TST13       ;;<BR IF YES>
2134 004070 004767 010552      JSR      PC,SUERR1  ;GO SET UP THE ERROR INFO
2135 004074 104003                ERROR+3 ;DATA COMPARE ERROR
2136
2137 *****
2138 *TEST 13 TEST THAT 'REQ TO SEND' ASSERTS 'RING'
2139 *****
2139 004076 000004      TST13: SCOPE
2140 004100 032777 010000 175032  BIT      #SW12,@SWR ;ARE WE TESTING /C OR /D MODEL?
2141 004106 001047                BNE     TST14       ;;<BRANCH IF YES>
2142 004110 012704 140004      MOV      #140004,R4 ;RESULT IN RCSR S/B = 140004
2143 004114 016702 175270      MOV      DLRCSR,R2  ;REGADR = RCSR ADR
2144 004120 005012                CLR     (R2)        ;INIT THE RCSR TO 000000
2145 004122 052712 000004      BIS      #BIT2,(R2) ;SET 'REQ TO SEND'
2146 004126 032777 100000 175254  BIT      #BIT15,@DLRCSR ;DID 'RING' SET 'DATA SET INT' ?
2147 004134 001003                BNE     1$          ;;<BR IF YES>
2148 004136 004767 010456      JSR      PC,SUER2   ;GO SET UP ERROR INFO.
2149 004142 104002                ERROR+2 ;'RING' TRANSITION FAILED TO SET 'DATA SET INT'
2150                ;NOTE: 'BIT #BIT15,(R2)' RESETS BIT15
2151 004144 012704 040004      1$:     MOV      #40004,R4 ;RESULT IN RCSR S/B = 40004
2152 004150 020412                CMP     R4,(R2)     ;BOTH 'RING' AND 'REQ TO SEND' ASSERTED
2153 004152 001403                BEQ     2$          ;;<BR IF YES>
2154 004154 004767 010440      JSR      PC,SUER2   ;GO SET UP ERROR INFO.
2155 004160 104002                ERROR+2 ;'RING' OR 'REQ TO SEND' FAILED TO SET
2156 004162 005004                2$:     CLR      R4      ;RESULT IN RCSR S/B = 000000
2157 004164 042712 000004      BIC      #BIT2,(R2) ;CLEAR 'REQ TO SEND'
2158 004170 032777 100000 175212  BIT      #BIT15,@DLRCSR ;DID 'DATA SET INT' GET SET ??
2159 004176 001403                BEQ     3$          ;;<BR IF NOT>
2160 004200 004767 010414      JSR      PC,SUER2   ;GO SET UP ERROR INFO
2161 004204 104002                ERROR+2 ;CLEARING 'RING' SET 'DATA SET INT'
2162 004206 020412                3$:     CMP     R4,(R2)  ;RCSR CONTAIN ALL ZEROES ??

```

```

2163 004210 001406          BEQ     TST14          ;;<BR IF YES>
2164 004212 004767 010402    JSR     PC,SUER2      ;GO SET UP ERROR INFO:
2165 004216 016767 000002 174754  MOV     .+6,$REG7     ;SAVE THE ERROR PC
2166 004224 104002          ERROR+2 ;CLEARING 'REQ TO SEND' FAILED TO CLEAR 'RING'
2167                                     ;*****
2168                                     ;*TEST 14      TEST THAT 'SEC XMIT' ASSERTS 'SEC REC' AND 'DATA SET INT'
2169                                     ;*****
2170 004226 000004          TST14: SCOPE
2171 004230 032777 010000 174702  BIT     #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
2172 004236 001046          BNE     TST15        ;;<BRANCH IF YES>
2173 004240 016702 175144    MOV     DLRCR,R2     ;REGADR = RCSR ADR
2174 004244 005012          CLR     (R2)         ;INIT RCSR TO 000000
2175 004246 012704 102010    MOV     #102010,R4   ;CONTENTS OF RCSR S/B = 102010
2176 004252 052712 000010    BIS     #BIT3,(R2)   ;SET 'SEC XMIT' BIT
2177 004256 032777 100000 175124  BIT     #BIT15,@DLRCR ;DID 'DATA SET INT' SET ??
2178 004264 001003          BNE     1$          ;;<BR IF YES>
2179 004266 004767 010326    JSR     PC,SUER2     ;GO SET UP ERROR INFO
2180 004272 104002          ERROR+2 ;'DATA SET INT' FAILED TO SET-NOTE THAT
2181                                     ;'BIT #BIT15,(R2)' RESETS BIT15
2182 004274 012704 002010    1$:    MOV     #2010,R4  ;RESULT IN RCSR S/B = 2010
2183 004300 020412          CMP     R4,(R2)     ;ARE 'SEC XMIT' AND 'SEC REC' BOTH SET ?
2184 004302 001403          BEQ     2$          ;;<BR IF YES>
2185 004304 004767 010310    JSR     PC,SUER2     ;GO SET UP ERROR INFO
2186 004310 104002          ERROR+2 ;'SEC XMIT' OR 'SEC REC' FAILED TO SET
2187                                     ;OR 'DATA SET INT' FAILED TO BE CLEARED
2188                                     ;WHEN REFERENCING RCSR
2189 004312 012704 100000    2$:    MOV     #BIT15,R4 ;RESULT IN RCSR S/B = 100000
2190 004316 042712 000010    BIC     #BIT3,(R2)  ;CLEAR 'SEC XMIT' BIT
2191 004322 032777 100000 175060  BIT     #BIT15,@DLRCR ;DID CLEARING IT SET 'DATA SET INT'??
2192 004330 001003          BNE     3$          ;;<BR IF YES>
2193 004332 004767 010262    JSR     PC,SUER2     ;GO SET UP ERROR INFO.
2194 004336 104002          ERROR+2 ;CLEARING 'SEC XMIT' FAILED TO SET 'DATA
2195                                     ;SET INT. (NOTE THAT REFERENCING RCSR
2196                                     ;CLEARS 'DATA SET INT'
2197 004340 005004    3$:    CLR     R4          ;RESULT IN RCSR S/B = 000000
2198 004342 020412          CMP     R4,(R2)     ;'SEC XMIT' AND 'SEC REC' CLEAR ?
2199 004344 001403          BEQ     TST15        ;;<BR IF YES>
2200 004346 004767 010246    JSR     PC,SUER2     ;GO SETUP ERROR INFO
2201 004352 104002          ERROR+2 ;'SEC XMIT' OR 'SEC REC' FAILED TO CLEAR
2202                                     ;OR REFERENCING RCSR FAILED TO CLEAR 'DATA SET INT'
2203                                     ;*****
2204                                     ;*TEST 15      TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'
2205                                     ;*****
2206 004354 000004          TST15: SCOPE
2207 004356 032777 010000 174554  BIT     #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
2208 004364 001046          BNE     TST16        ;;<BRANCH IF YES>
2209 004366 016702 175016    MOV     DLRCR,R2     ;REGADR = RCSR ADR
2210 004372 005012          CLR     (R2)         ;INIT RCSR TO 000000
2211 004374 012704 130002    MOV     #130002,R4   ;RESULT IN RCSR S/B = 130002
2212 004400 052712 000002    BIS     #BIT1,(R2)  ;SET 'DTR' BIT
2213 004404 032777 100000 174776  BIT     #BIT15,@DLRCR ;DID 'DATA SET INT' SET ??
2214 004412 001003          BNE     1$          ;;<BR IF YES>
2215 004414 004767 010200    JSR     PC,SUER2     ;GO SET UP ERROR INFO.
2216 004420 104002          ERROR+2 ;'DATA SET INT' FAILED TO SET -
2217                                     ;NOTE: THE REFERENCE TO RCSR ABOVE WILL
2218                                     ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.

```

CZDLCCO DL11-C,D,E OFLINE IST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 43
T15 TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'

SEQ 0042

| | | | | | | | |
|------|--------|--------|--------|------|---------|-----------|--|
| 2219 | 004422 | 012704 | 030002 | 1\$: | MOV | #30002,R4 | :RESULT IN RCSR S/B = 30002 |
| 2220 | 004426 | 020412 | | | CMP | R4,(R2) | : 'DTR', 'CLR TO SEND', AND 'CAR DET' ALLSET |
| 2221 | 004430 | 001403 | | | BEQ | 2\$ | :: <BR IF ALL SET> |
| 2222 | 004432 | 004767 | 010162 | | JSR | PC,SUER2 | :GO SET UP ERROR INFO |
| 2223 | 004436 | 104002 | | | ERROR+2 | | : 'DTR', 'CLR TO SEND' OR 'CAR DET' AILED |
| 2224 | | | | | | | : TO SET OR 'DATA SET INT' FAILED TO CLEAR |

CZDLCCO DL11-C.D.E OFFLINE TST
CZDLCC.P11 21-SEP-79 09:07

MACV11 30A(1052) 21-SEP-79 09:11 PAGE 44
T15

TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'

SEQ 0043

2225 004440 012704 100000

28: MOV #BIT15,R4 ;RESULT IN RCSR S/B = 100000

CZDLCCO DL11-C,D,E OFLINE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 45
T15

TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'

SEQ 0044

| | | | |
|------|--------|--------|--------|
| 2226 | 004444 | 042712 | 000002 |
| 2227 | 004450 | 032777 | 100000 |
| 2228 | 004456 | 001003 | |
| 2229 | 004460 | 004767 | 010134 |

174732

| | | |
|-----|-----------------|----------------------------|
| BIC | #BIT1 (R2) | :NOW CLEAR 'DTR' |
| BIT | #BIT15, @DLRCSR | :DID 'DATA SET INT' SET ?? |
| BNE | ZS | ::<BR IF YES> |
| SR | D', S JER2 | :GO SETUP ERROR INFO |

CZDLCCO DL11-C,D,E OF LNE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 46
T15 TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'

SEQ 0045

```

2230 004464 104002          ERROR+2          ;'DATA SET INT' FAILED TO SET WHEN 'DTR'
2231                                ;WENT TO A ZERO.
2232 004466 005004          3$: CLR R4          ;RESULT IN RCSR S/B = 000000
2233 004470 020412          CMP R4,(R2)       ;DID ALL BITS CLEAR??
2234 004472 001403          BEQ TST16        ;;<BR IF YES>
2235 004474 004767 010120  JSR PC,SUER2     ;GO SET UP ERROR INFO
2236 004500 104002          ERROR+2          ;'DTR','CLR TO SEND' OR 'CAR DET' FAILED
2237                                ;TO CLEAR PROPERLY
2238
2239                                ;*****
2240                                ;*TEST 16 TEST THAT 'DATA SET INT ENAB' CAN SET AND CLEAR
2241                                ;*****
2241 004502 000004          TST16: SCOPE
2242 004504 032777 010000 174426 BIT #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
2243 004512 001023          BNE TST17        ;;<BRANCH IF YES>
2244 004514 016702 174670  MOV DLRCR,R2     ;REGADR = RCSR ADR
2245 004520 012704 000040  MOV #40,R4       ;RESULT IN RCSR S/B - 000040
2246 004524 052712 000040  BIS #BIT5,(R2)   ;SET THE 'DATA SET I.E.' BIT
2247 004530 020412          CMP R4,(R2)       ;DID IT SET OK ??
2248 004532 001403          BEQ 1$          ;;<BR IF YES>
2249 004534 004767 010060  JSR PC,SUER2     ;GO SET UP ERROR INFO
2250 004540 104002          ERROR+2          ;'DAT SET I. E.' FAILED TO SET
2251 004542 005004          1$: CLR R4          ;MAKE S/B DATA = 000000
2252 004544 042712 000040  BIC #BIT5,(R2)   ;NOW CLEAR THE 'DATA SET I.E.' BIT
2253 004550 020412          CMP R4,(R2)       ;DID IT CLEAR OK ??
2254 004552 001403          BEQ TST17        ;;<BR IF YES>
2255 004554 004767 010040  JSR PC,SUER2     ;GO SET UP ERROR INFO.
2256 004560 104002          ERROR+2          ;'DATA SET I.E.' FAILED TO CLEAR
2257
2258                                ;*****
2259                                ;*TEST 17 TEST THE 'DATA SET I.E.' CAN CAUSE A RCVR INTR
2260                                ;*****
2260 004562 000004          TST17: SCOPE
2261 004564 032777 010000 174346 BIT #SW12,@SWR    ;ARE WE TESTING A /C OR /D MODEL?
2262 004572 001054          BNE TST20        ;;<BRANCH IF YES>
2263 004574 016705 174620  MOV DLVECT,R5    ;GET THE VECTOR ADDR
2264 004600 012725 004666  MOV #3$,R5+      ;GO TO 3$ ON RCVR INTR.
2265 004604 016715 174472  MOV DLPRI,(R5)   ;AT LEVEL 4
2266 004610 005005          CLR R5           ;INIT INTR. TIMER
2267 004612 005067 174630  CLR INTFLG       ;INIT SOFTWARE FLAG
2268 004616 005004          CLR R4           ;RESULT IN RCSR S/B = 0 AFTER INTR.
2269 004620 016702 174564  MOV DLRCR,R2     ;REGADR = RCSR ADR
2270 004624 052712 000040  BIS #BIT5,(R2)   ;SET THE 'DATA SET I.E.' BIT
2271 004630 052712 000002  BIS #BIT1,(R2)   ;NOW SET 'DTR' TO GEN INTR.
2272 004634 005767 174606  1$: TST INTFLG    ;DID INTR OCCUR YET ??
2273 004640 001016          BNE 4$          ;BR IF YES
2274 004642 005305          DEC R5           ;COUNT THE TIMER
2275 004644 001373          BNE 1$          ;BR IF NO TIMEOUT
2276 004646 004767 007746  JSR PC,SUER2     ;GO SET UP ERROR INFO
2277 004652 005012          CLR (R2)         ;TURN IT ALL OFF
2278 004654 012767 004664 174362  MOV #2$, $ESCAPE ;COME BACK TO 2$ IN ALL CASES
2279 004662 104002          ERROR+2          ;'DATA SET' INTR FAILED TO OCCUR
2280 004664          2$:
2281 004664 000417          BR TST20         ;;<GO TO NEXT TEST>
2282 004666 005012          3$: CLR (R2)       ;ZERO THE RCSR
2283 004670 005167 174552  COM INTFLG       ;SET THE SOFTWARE FLAG
2284 004674 000002          RTI            ;RETURN TO SENDER
2285 004676 032712 100000  4$: BIT #BIT15,(R2) ;DID 'DATA SET INT' GET SET BY INTR. SER.??

```

```

2286 004702 001003          BNE      5$          ;;<BR IF YES>
2287 004704 004767 007710  JSR      PC,SUER2   ;GO SET UP ERROR INFO
2288 004710 104002          ERROR+2  ;DATA SET INTR. NOT SERVICED PROPERLY
2289 004712 020412          5$:      CMP      R4,(R2) ;ALL BITS IN RCSR CLEAR ??
2290 004714 001403          BEQ      TST20      ;;<BR IF YES>
2291 004716 004767 007676  JSR      PC,SUER2   ;GO SET UP ERROR INFO
2292 004722 104002          ERROR+2  ;INTR. SERVICE FAILED TO CLEAR RCSR
2293
2294          ;*****
2295          ;*TEST 20      TEST THAT THE 'BREAK' BIT CAN BE SET AND CLFARED
2296          ;*****
2296 004724 000004          TST20:  SCOPE
2297 004726 032777 010000 174204  BIT      #SW12,@SWR ;ARE WE TESTING /C OR /D MODEL?
2298 004734 001024          BNE      TST21      ;;<BRANCH IF YES>
2299 004736 012704 000201  MOV      #201,R4    ;RESULT S/B = 201 IN XCSR
2300 004742 016702 174446  MOV      DLXCSR,R2  ;SET UP REGADR
2301 004746 052712 000001  BIS      #BIT0,(R2) ;SET THE 'BREAK' BIT
2302 004752 020412          CMP      R4,(R2)   ;DID IT SET PROPERLY ??
2303 004754 001403          BEQ      1$        ;;<BR IF YES>
2304 004756 004767 007636  JSR      PC,SUER2   ;GO SET UP ERROR INFO.
2305 004762 104002          ERROR+2  ;'BREAK' BIT FAILED TO SET PROPERLY
2306 004764 012704 000200  1$:      MOV      #200,R4    ;RESULT S/B = 200 IN XCSR
2307 004770 042712 000001  BIC      #BIT0,(R2) ;CLEAR THE 'BREAK' BIT
2308 004774 020412          CMP      R4,(R2)   ;DID IT CLEAR PROPERLY ??
2309 004776 001403          BEQ      TST21      ;;<BR IF YES>
2310 005000 004767 007614  JSR      PC,SUER2   ;GO SET UP ERROR INFO
2311 005004 104002          ERROR+2  ;'BREAK' FAILED TO CLEAR PROPERLY
2312
2313
2314          ;*****
2315          ;*TEST 21      TEST THAT A 'RESET' CLEARS THE 'BREAK' BIT
2316          ;*****
2317 005006 000004          TST21:  SCOPE
2318 005010 012704 000200  MOV      #200,R4    ;RESULT S/B = 200
2319 005014 016702 174374  MOV      DLXCSR,R2  ;SET UP REGADR
2320 005020 052712 000001  BIS      #BIT0,(R2) ;SET THE 'BREAK' BIT
2321 005024 000005          RESET      ;CLEAR IT WITH A 'RESET'
2322 005026 020412          CMP      R4,(R2)   ;DID IT CLEAR ??
2323 005030 001403          BEQ      TST22      ;;<BR IF YES>
2324 005032 004767 007562  JSR      PC,SUER2   ;GO SET UP ERROR INFO.
2325 005036 104002          ERROR+2  ;RESET INSTR. FAILED TO CLEAR 'BREAK'
2326

```

```
2327 .....  
2328 :*TEST 22 TEST TO TURN AROUND NULL-DEL-NULL PATTERN  
2329 .....  
2330 TST22: SCOPE  
2331 005040 000004 000001 174172 MOV #1,$TIMES ;;DO 1 ITERATION  
2332 005050 004767 007700 JSR PC,SUVEC ;GO SET UP VECTORS  
2333 005054 005067 174350 CLR RTRY ;INITIALIZE RETRY FLAG  
2334 005060 012767 015130 174350 1$: MOV #LDOUT1,LDOUT ;SET POINTER TO LOAD ROUTINE  
2335 005066 004767 007710 JSR PC,PRIME ;GO SET UP BUFFERS AND DEVICE  
2336 005072 005767 174324 2$: TST XFLGO ;ANY HARD XMIT ERRORS ??  
2337 005076 001040 BNE 5$ ;BR IF YES  
2338 005100 005767 174320 TST RFLGO ;ANY HARD RECEIVER ERROR ??  
2339 005104 001053 BNE 7$ ;BR IF YES  
2340 005106 005767 174314 TST RFLG1 ;ANY SOFT RECEIVER ERRORS ??  
2341 005112 001065 BNE 9$ ;BR IF YES  
2342 005114 022767 023040 174312 CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??  
2343 005122 001003 BNE 3$ ;BR IF NOT  
2344 005124 004767 010144 JSR PC,CHKDAT ;GO CHECK THE DATA BUFFERS  
2345 005130 000500 BR TST23 ;;<GO TO NEXT TEST>  
2346 005132 005367 174302 3$: DEC TIMR1 ;DEC TIMEOUT COUNTER 1  
2347 005136 001355 BNE 2$ ;BR IF NO TIMEOUT  
2348 005140 005367 174276 DEC TIMR2 ;DEC TIMEOUT COUNTER 2  
2349 005144 001352 BNE 2$ ;BR IF NO TIMEOUT  
2350 005146 042777 000100 174234 BIC #100,@DLRCSR ;TURN OFF THE INTRS.  
2351 005154 042777 000104 174232 BIC #104,@DLXCSR  
2352 005162 104401 017110 TYPE ,XMSG1 ;GO TYPE TIMEOUT MESSAGE  
2353 005166 012767 005176 174050 MOV #4$, $ESCAPE ;GO TO 4$ AFTER ERROR PRINT  
2354 005174 104000 ERROR ;PRINT ERROR PC  
2355 005176 4$:  
2356 005176 000455 BR TST23 ;;<GO TO NEXT TEST>  
2357 005200 016701 174204 5$: MOV DLRCSR,R1 ;PUT DEVADR IN R1  
2358 005204 016702 174204 MOV DLXCSR,R2 ;PUT REGADR IN R2  
2359 005210 011203 MOV (R2),R3 ;GET THE WAS DATA  
2360 005212 012704 000204 MOV #204,R4 ;PUT S/B DATA IN R4  
2361 005216 004767 007424 JSR PC,SUERR1 ;GO SET UP ERROR INFO  
2362 005222 012767 005232 174014 MOV #6$, $ESCAPE ;GO TO 6$ AFTER PRINTING ERROR  
2363 005230 104002 ERROR+2 ;TRANSMITTER FALSE INTERRUPT  
2364 005232 6$:  
2365 005232 000437 BR TST23 ;;<GO TO NEXT TEST>  
2366 005234 016701 174150 7$: MOV DLRCSR,R1 ;SAVE THE DEVADR  
2367 005240 010102 MOV R1,R2 ;SAVE THE REGADR  
2368 005242 011203 MOV (R2),R3 ;GET THE WAS DATA  
2369 005244 012704 000200 MOV #200,R4 ;RESULT S/B = 200  
2370 005250 004767 007372 JSR PC,SUERR1 ;GO SET UP ERROR INFO  
2371 005254 012767 005264 173762 MOV #8$, $ESCAPE ;GO TO 8$ AFTER ERROR PRINT  
2372 005262 104002 ERROR+2 ;RECEIVER FALSE INTERRUPT  
2373 005264 8$:  
2374 005264 000422 BR TST23 ;;<GO TO NEXT TEST>  
2375 005266 016701 174116 9$: MOV DLRCSR,R1 ;SAVE THE DEVADR  
2376 005272 016702 174114 MOV DLRDBR,R2 ;SAVE REGADR  
2377 005276 016703 173702 MOV $TMP1,R3 ;GET CONTENTS OF ERROR RDBR  
2378 005302 004767 007340 JSR PC,SUERR1 ;GO SETUP ERROR INFO  
2379 005306 012767 005316 173730 MOV #10$, $ESCAPE ;GO TO 10$ AFTER ERROR PRINT  
2380 005314 104005 ERROR+5 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVFERRUN)  
2381 005316 005267 174106 10$: INC RTRY ;COUNT ONE TRY  
2382 005322 022767 000003 174100 CMP #3,RTRY ;TRIED THREE TIMES
```


CZDLCCO DL11-C.D.E OFLNE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 49
T22 TEST TO TURN AROUND NULL-DEL-NULL PATTERN

SEQ 0048

2383 005330 001253

BNE 18

;BR IF NOT

```

2384
2385
2386
2387 005332 000004
2388 005334 012767 000001 173700
2389 005342 004767 007406
2390 005346 005067 174056
2391 005352 012767 015152 174056 1$:
2392 005360 004767 007416
2393 005364 005767 174032 2$:
2394 005370 001040
2395 005372 005767 174026
2396 005376 001053
2397 005400 005767 174022
2398 005404 001065
2399 005406 022767 023040 174020
2400 005414 001003
2401 005416 004767 007652
2402 005422 000500
2403 005424 005367 174010 3$:
2404 005430 001355
2405 005432 005367 174004
2406 005436 001352
2407 005440 042777 000100 173742
2408 005446 042777 000104 173740
2409 005454 104401 017167
2410 005460 012767 005470 173556
2411 005466 104000
2412 005470 4$:
2413 005470 000455
2414 005472 016701 173712 5$:
2415 005476 016702 173712
2416 005502 011203
2417 005504 012704 000204
2418 005510 004767 007132
2419 005514 012767 005524 173522
2420 005522 104002
2421 005524 6$:
2422 005524 000437
2423 005526 016701 173656 7$:
2424 005532 010102
2425 005534 011203
2426 005536 012704 000200
2427 005542 004767 007100
2428 005546 012767 005556 173470
2429 005554 104002
2430 005556 8$:
2431 005556 000422
2432 005560 016701 173624 9$:
2433 005564 016702 173622
2434 005570 016703 173410
2435 005574 004767 007046
2436 005600 012767 005610 173436
2437 005606 104005
2438 005610 005267 173614 10$:
2439 005614 022767 000003 173606

```

```

:*****
:*TEST 23 TEST TO TURN AROUND BINARY UP COUNT PATTERN
:*****
TST23: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
JSR PC,SUVEC ;;GO SET UP VECTORS
CLR RTRY ;;INITIALIZE RETRY FLAG
MOV #LDOUT2,LDOUT ;;SET POINTER TO LOAD ROUTINE
JSR PC,PRIME ;;GO SET UP BUFFERS AND DEVICE
TST XFLGO ;;ANY HARD XMIT ERRORS ??
BNE 5$ ;;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 TST24 ;;<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 ,XMSG2 ;;GO TYPE TIMEOUT MESSAGE
MOV #4$, $ESCAPE ;;GO TO 4$ AFTER ERROR PRINT
ERROR ;;PRINT ERROR PC
4$:
BR TST24 ;;<GO TO NEXT TEST>
5$:
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$:
BR TST24 ;;<GO TO NEXT TEST>
7$:
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$:
BR TST24 ;;<GO TO NEXT TEST>
9$:
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

```

CZDLCCO DL11-C,D,E OFLINE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 51
T23 TEST TO TURN AROUND BINARY UP COUNT PATTERN

L 4

SEQ 0050

2440 005622 001253

BNE 1\$

;BR IF NOT

```

2441
2442
2443
2444 005624 000004
2445 005626 012767 000001 173406
2446 005634 004767 007114
2447 005640 005067 173564
2448 005644 012767 015172 173564 1$:
2449 005652 004767 007124
2450 005656 005767 173540 2$:
2451 005662 001040
2452 005664 005767 173534
2453 005670 001053
2454 005672 005767 173530
2455 005676 001065
2456 005700 022767 023040 173526
2457 005706 001003
2458 005710 004767 007360
2459 005714 000500
2460 005716 005367 173516 3$:
2461 005722 001355
2462 005724 005367 173512
2463 005730 001352
2464 005732 042777 000100 173450
2465 005740 042777 000104 173446
2466 005746 104401 017250
2467 005752 012767 005762 173264
2468 005760 104000
2469 005762 4$:
2470 005762 000455
2471 005764 016701 173420 5$:
2472 005770 016702 173420
2473 005774 011203
2474 005776 012704 000204
2475 006002 004767 006640
2476 006006 012767 006016 173230
2477 006014 104002
2478 006016 6$:
2479 006016 000437
2480 006020 016701 173364 7$:
2481 006024 010102
2482 006026 011203
2483 006030 012704 000200
2484 006034 004767 006606
2485 006040 012767 006050 173176
2486 006046 104002
2487 006050 8$:
2488 006050 000422
2489 006052 016701 173332 9$:
2490 006056 016702 173330
2491 006062 016703 173116
2492 006066 004767 006554
2493 006072 012767 006102 173144
2494 006100 104005
2495 006102 005267 173322 10$:
2496 006106 022767 000003 173314

```

```

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

```

CZDLCCO DL11-C,D,E OFLNE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 53
T24 TEST TO TURN AROUND BINARY DOWN COUNT PATTERN

N 4

SEQ 0052

2497 006114 00253

BNE 1\$

;BR IF NOT

```

2498
2499
2500
2501 006116 000004
2502 006120 012767 000001 173114
2503 006126 004767 006622
2504 006132 005067 173272
2505 006136 012767 015226 173272 1$:
2506 006144 004767 006632
2507 006150 005767 173246 2$:
2508 006154 001042
2509 006156 005767 173242
2510 006162 001056
2511 006164 005767 173236
2512 006170 001071
2513 006172 022767 023040 173234
2514 006200 001004
2515 006202 004767 007066
2516 006206 000167 001642
2517 006212 005367 173222 3$:
2518 006216 001354
2519 006220 005367 173216
2520 006224 001351
2521 006226 042777 000100 173154
2522 006234 042777 000104 173152
2523 006242 104401 017333
2524 006246 012767 006256 172770
2525 006254 104000
2526 006256 000167 001572 4$:
2527 006262 016701 173122 5$:
2528 006266 016702 173122
2529 006272 011203
2530 006274 012704 000204
2531 006300 004767 006342
2532 006304 012767 006314 172732
2533 006312 104002
2534 006314 000167 001534 6$:
2535 006320 016701 173064 7$:
2536 006324 010102
2537 006326 011203
2538 006330 012704 000200
2539 006334 004767 006306
2540 006340 012767 006350 172676
2541 006346 104002
2542 006350 000167 001500 8$:
2543 006354 016701 173030 9$:
2544 006360 016702 173026
2545 006364 016703 172614
2546 006370 004767 006252
2547 006374 012767 006404 172642
2548 006402 104005
2549 006404 005267 173020 10$:
2550 006410 022767 000003 173012
2551 006416 001247
2552 006420 000167 001430
2553

```

```

*****
*TEST 25 TEST TO TURN AROUND WORST CASE PATTERN
*****
TST25: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
JSR PC,SUVEC ;GO SET UP VECTORS
CLR RTRY ;INITIALIZE RETRY FLAG
MOV #LDOUT4,LDOUT ;SET POINTER TO LOAD ROUTINE
JSR PC,PRIME ;GO SET UP BUFFERS AND DEVICE
TST XFLGO ;ANY HARD XMIT ERRORS ??
BNE 5$ ;BR IF YES
TST RFLGO ;ANY HARD RECEIVER ERROR ??
BNE 7$ ;BR IF YES
TST RFLG1 ;ANY SOFT RECEIVER ERRORS ??
BNE 9$ ;BR IF YES
CMP #BUFEND,IPTR ;RECEIVED 256. BYTES ??
BNE 3$ ;BR IF NOT
JSR PC,CHKDAT ;GO CHECK THE DATA BUFFERS
JMP $EOP ;GO TO NEXT TEST
3$: 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 #04,@DLXCSR
TYPE ,XMSG4 ;GO TYPE TIMEOUT MESSAGE
MOV #4$,$ESCAPE ;GO TO 4$ AFTER ERROR PRINT
ERROR ;PRINT ERROR PC
4$: JMP $EOP ;GO TO NEXT TEST
5$: 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$: JMP $EOP ;GO TO NEXT TEST
7$: 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$: JMP $EOP ;GO TO NEXT TEST
9$: 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$: INC RTRY ;COUNT ONE TRY
CMP #3,RTRY ;TRIED THREE TIMES
BNE 1$ ;BR IF NO
JMP $EOP ;GO TO END OF PASS ROUTINE

```


D 5

2610 006556 104412
 2611
 2612 006560 012667 172420
 2613
 2614
 2615 006564 104401 021271
 2616
 2617
 2618 006570 104412
 2619
 2620 006572 012667 172410
 2621 006576 116767 172404 000012 1\$:
 2622
 2623 006604 116777 172374 172370
 2624
 2625 006612 004767 005470
 2626
 2627
 2628
 2629
 2630
 2631
 2632
 2633 006616 000000
 2634 006620 000766
 2635 006622 104401 001252
 2636 006626 000167 177600
 2637
 2638
 2639
 2640
 2641
 2642
 2643
 2644
 2645
 2646
 2647 006632 000240
 2648 006634 104401 017524
 2649
 2650 006640 104401 021102
 2651
 2652
 2653 006644 104412
 2654
 2655
 2656 006644 012602
 2657 006650 020227 176176
 2658 006654 101071
 2659 006656 020227 175616
 2660 006662 103466
 2661 006664 132702 000001
 2662
 2663 006670 001063
 2664
 2665 006672 010203

```

RDOCT          ;ACCEPT THE ANSWER TYPED BY
                ;USER AND STORE ON TOP OF STACK
MOV (SP)+,$TMP1 ;GET THE ANSWER TYPED
;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
;      OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
TYPE ,SELDLY   ;ASK THE USER FOR THE DELAY
                ;IN MSEC (OCTAL NO.) BETWEEN
                ;CHARACTER TRANSFERS
RDOCT          ;ACCEPT THE ANSWER TYPED BY
                ;USER AND STORE ON TOP OF STACK
MOV (SP)+,$TMP2 ;GET THE ANSWER TYPED
MOVB $TMP2,2$  ;SET THE DELAY COUNT ARGUMENT
                ;FOR TIMER ROUTINE
MOVB $TMP1,@$TMP0 ;LOAD THE TRANSMITTER DATA
                ;BUFFER WITH THE CHARACTER
JSR PC,DELAY   ;GO OFF TO WAIT THE SPECIFIED
                ;NO. OF MSEC. BEFORE ISSUING
                ;ANOTHER CHARACTER
2$: .WORD 0     ;THIS IS WHERE THE DELAY COUNT RESIDES
BR 1$         ;GO BACK TO ISSUE ANOTHER CHARACTER
RED01: TYPE ,SQUES ;TYPE A QUESTION MARK(?)
        JMP PRG2A ;REITERATE THE XDBR QUESTION TO USER
;THIS IS PROGRAM #3
;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
;
; A) SELECTION OF A 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
;
PRG3: NOP
TYPE ,PROG3M ;INDICATE THAT USER SELECTED
                ;PROGRAM #3
PRG3A: 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 USER RESPONSE WAS WITHIN LIMITS
MOV (SP)+,R2  ;GET THE ANSWER TYPED
CMP R2,#176176 ;IS THE NUMBER TOO HIGH?
BHI RED02     ;IF YES - GO TO RETRY SITUATION
CMP R2,#175616 ;IS THE NUMBER TOO LOW?
BLO RED02     ;IF YES - GO TO RETRY SITUATION
BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
                ;ON AN EVEN BOUNDARY?
BNE RED02     ;IF NOT - GO TO RETRY SITUATION
;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR DBR ADDRESS
MOV R2,R3    ;GET THE USER RESPONSE

```



```
2666 006674 142703 000370      BICB    #370,R3      ;MASK OFF LOWER BYTE EXCEPT FOR
2667                                ;LEAST SIGNIFICANT DIGIT
2668 006700 122703 000006      CMPB    #6,R3        ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2669                                ;USER RESPONSE EQUAL TO A TWO?
2670 006704 001055                BNE     REDO2        ;BRANCH IF NOT
2671 006706 010267 172270      MOV     R2,$TMP0    ;THE TRANSMITTER ADDRESS TYPED IS
2672                                ;OK - STORE FOR FUTURE USE
2673                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2674 006712 016746 171066      MOV     ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
2675 006716 012767 006730 171060 MOV     #2$,ERRVEC   ;SET UP TIMEOUT SERVICE ADDRESS
2676 006724 005712                TST     (R2)         ;IF PRESENT WE WILL EXECUTE THE
2677                                ;NEXT INSTRUCTION - IF NOT WE
2678                                ;GO TO 2$:
2679 006726 000412                BR      4$          ;BRANCH IF PRESENT
2680 006730 004767 005772 2$:   JSR     PC,SUERT2   ;GO SET UP FOR ERROR INFORMATION
2681 006734 012767 006744 172302 MOV     #3$, $ESCAPE ;POINT OF RETURN AFTER ERROR REPORT
2682 006742 104006                ERROR   +6          ;XDBR REFERENCE CAUSED TIMEOUT
2683 006744 022626 3$:   CMP     (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
2684 006746 012667 171032      MOV     (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
2685 006752 000432                BR      REDO2        ;GO TO RETRY SITUATION
2686 006754 012667 171024 4$:   MOV     (SP)+,ERRVEC ;DEVICE REGISTER IS PRESENT
2687                                ;RESTORE TIMEOUT VECTOR
2688                                ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
2689                                ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN SUCCESSIVE
2690                                ;CHARACTER TRANSFERS
2691 006760 104401 021163 PRG3B: TYPE    ,SELCAR ;ASK USER FOR THE CHARACTER
2692                                ;HE WISHES TO TRANSFER
2693 006764 104412                RDOCT              ;ACCEPT THE ANSWER TYPED BY USER
2694                                ;AND STORE ON TOP OF STACK
2695 006766 012667 172212      MOV     (SP)+,$TMP1 ;GET THE ANSWER TYPED
2696                                ;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
2697                                ;OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. B-102
2698 006772 104401 021271      TYPE    ,SELDLY   ;ASK THE USER FOR THE DELAY
2699                                ;IN MSEC (OCTAL NO.) BETWEEN
2700                                ;CHARACTER TRANSFERS
2701 006776 104412                RDOCT              ;ACCEPT THE ANSWER TYPED BY
2702                                ;USER AND STORE ON TOP OF STACK
2703 007000 012667 172202      MOV     (SP)+,$TMP2 ;GET THE ANSWER TYPED
2704 007004 162702 000002      SUB     #2,R2      ;GET THE CORRESPONDING XCSR
2705                                ;ADDRESS FOR TRANSMITTER UNDER-
2706                                ;GOING TEST
2707 007010 052712 000004 1$:   BIS     #BIT2,(R2) ;SET MAINTENANCE BIT IN XCSR
2708 007014 116767 172166 000012 MOVB    $TMP2,2$   ;SET THE DELAY COUNT ARGUMENT
2709                                ;FOR TIMER ROUTINE
2710 007022 116777 172156 172152 MOVB    $TMP1,@$TMP0 ;LOAD THE TRANSMITTER DATA BUFFER
2711                                ;WITH THE CHARACTER
2712 007030 004767 005252      JSR     PC,DELAY   ;GO OFF TO WAIT THE SPECIFIED
2713                                ;NO. OF MSEC. BEFORE ISSUING
2714                                ;ANOTHER CHARACTER
2715 007034 000000 2$:   .WORD 0 ;THIS IS WHERE THE DELAY COUNT RESIDES
2716 007036 000764                BR      1$         ;GO BACK TO ISSUE ANOTHER CHARACTER
2717 007040 104401 001252 REDO2: TYPE    ,SQUES ;TYPE A QUESTION MARK(?)
2718 007044 000167 17757C      JMP     PRG3A     ;REITERATE THE XDBR QUESTION TO
2719                                ;USER
2720
2721                                ;THIS IS PROGRAM #4
```

```

2722 :THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
2723 : A) SELECTION OF A TRANSMITTER DATA BUFFER
2724 : B) SELECTION OF A SINGLE CHARACTER TO BE SENT, RECEIVED
2725 : AND CHECKED WITH MAINTENANCE BIT SET
2726 :
2727 :
2728 007050 000240 PRG4: NOP
2729 007052 104401 017570 TYPE ,PROG4M ;INDICATE THAT USER SELECTED
2730 ;PROGRAM #4
2731 007056 104401 021102 PRG4A: TYPE ,LINTAD ;ASK USER FOR THE TRANSMITTER
2732 ;DATA BUFFER ADDRESS OF THE
2733 ;DEVICE HE WISHES TO TEST
2734 007062 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY
2735 ;USER AND STORE ON TOP OF STACK
2736 ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
2737 007064 012602 MOV (SP)+,R2 ;GET THE ANSWER TYPED
2738 007066 020227 176176 CMP R2,#176176 ;IS THE NUMBER TOO HIGH?
2739 007072 101136 BHI RED03 ;IF YES - GO TO RETRY SITUATION
2740 007074 020227 175616 CMP R2,#175616 ;IS THE NUMBER TOO LOW?
2741 007100 103533 BLO RED03 ;IF YES - GO TO RETRY SITUATION
2742 007102 132702 000001 BITB #BIT0,R2 ;NUMBER IS IN RANGE BUT IS IT
2743 ;ON AN EVEN BOUNDARY?
2744 007106 001130 BNE RED03 ;IF NO - GO TO RETRY SITUATION
2745 ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
2746 007110 010203 MOV R2,R3 ;GET THE USER RESPONSE
2747 007112 142703 000370 BICB #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
2748 ;LEAST SIGNIFICANT DIGIT
2749 007116 122703 000006 CMPB #6,R3 ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2750 ;USER RESPONSE EQUAL TO A SIX?
2751 007122 001122 BNE RED03 ;BRANCH IF NOT
2752 007124 010267 172052 MOV R2,$TMP0 ;THE TRANSMITTER ADDRESS TYPED
2753 ;IS OK - STORE FOR FUTURE USE
2754 ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2755 007130 016746 170650 MOV ERRVEC,-(SP) ;SAVE THE TIMEOUT VECTOR
2756 007134 012767 007146 170642 MOV #2$,ERRVEC ;SET UP TIMEOUT SERVICE ADDRESS
2757 007142 005712 TST (R2) ;IF PRESENT WE WILL EXECUTE THE
2758 ;NEXT INSTRUCTION - IF NOT WE
2759 ;GO TO 2$:
2760 007144 000412 BR 4$ ;BRANCH IF PRESENT
2761 007146 004767 005554 2$: JSR PC,SUERT2 ;GO SET UP FOR ERROR INFORMATION
2762 007152 012767 007162 172064 MOV #3$, $ESCAPE ;POINT OF RETURN AFTER ERROR REPORT
2763 007160 104006 ERROR +6 ;XDBR REFERENCE CAUSED TIMEOUT
2764 007162 022626 3$: CMP (SP)+,(SP)+ ;CLEAN STACK FROM TIMEOUT
2765 007164 012667 170614 MOV (SP)+,ERRVEC ;RESTORE TIMEOUT VECTOR
2766 007170 000477 BR RED03 ;GO TO RETRY SITUATION
2767 007172 012667 170606 4$: MOV (SP)+,ERRVEC ;DEVICE REGISTER IS PRESENT
2768 ;RESTORE TIMEOUT VECTOR
2769 007176 104401 021364 TYPE ,RSTALL ;ASK THE USER IF HE DESIRES SOME
2770 ;RANDOM NO. OF MSEC. WAIT TIME
2771 ;BEFORE CHECKING FOR XCSR DONE
2772 ;FLAG
2773 007202 104412 RDOCT ;ACCEPT THE ANSWER TYPED BY USER
2774 ;AND STORE ON TOP OF STACK
2775 007204 012667 171776 MOV (SP)+,$TMP2 ;GET THE ANSWER TYPED
2776 ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED
2777 007210 104401 021163 PRG4B: TYPE ,SELCAR ;ASK USER FOR THE CHARACTER HE

```

```

2778                                     ;WISHES TO TRANSFER
2779 007214 104412 RDOCT                ;ACCEPT THE ANSWER TYPED BY USER
2780                                     ;AND STORE ON TOP OF STACK
2781 007216 012667 171762 MOV (SP)+,$TMP1 ;GET THE ANSWER TYPED
2782 ;NOTE: THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE OCTAL
2783 ; ASCII EQUIVALENT OF THE CHARACTER E.G. C=103
2784
2785 007222 104401 020013 PRG4C: TYPE, LENGTH ;ASK USER FOR THE CHARACTER LENGTH
2786                                     ;FOR WHICH HIS DEVICE IS SET
2787 007226 104413 RDDEC                ;ACCEPT THE ANSWER TYPED BY USER
2788 ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2789 007230 012600 MOV (SP)+,R0 ;GET THE ANSWER TYPED
2790 007232 020027 000010 CMP R0,#8 ;IS THE NUMBER TOO HIGH?
2791 007236 101060 BHI REDO3A ;IF YES - GO TO RETRY SITUATION
2792 007240 020027 000005 CMP R0,#5 ;IS THE NUMBER TOO LOW?
2793 007244 103455 BLO REDO3A ;IF YES - GO TO RETRY SITUATION
2794 007246 010067 171762 MOV R0,$TMP15 ;THE VALUE TYPED IS OK
2795                                     ;STORE FOR FUTURE USE
2796 007252 016767 171724 171730 MOV $TMP0,$TMP3 ;GET THE XDBR ADDRESS
2797 007260 162767 000002 171722 SUB #2,$TMP3 ;FORM THE XCSR ADDRESS
2798 007266 005767 171714 1$: TST $TMP2 ;DO WE RANDOM STALL?
2799 007272 001402 BEQ 2$ ;BRANCH IF IT WASN'T DESIRED
2800 007274 004767 005052 JSR PC,STALL ;GO STALL RANDOM VALUE OF MSEC.
2801 007300 004767 005156 2$: JSR PC,TIMETX ;GO WAIT FOR TRANSMITTER DONE
2802                                     ;BIT TO SET
2803 007304 104401 017700 TYPE ,XDB ;TYPE TRANSMITTER DONE BIT MESSAGE
2804 007310 104000 ERROR +0 ;XCSR DONE BIT NEVER SET
2805 007312 052777 000004 171670 BIS #BIT2,@$TMP3 ;SET THE MAINTENANCE BIT IN THE
2806                                     ;TRANSMITTER CONTROL STATUS REGISTER
2807 007320 016777 171660 171654 MOV $TMP1,@$TMP0 ;LOAD TRANSMITTER DATA BUFFER
2808                                     ;WITH SELECTED CHARACTER
2809 007326 004767 005112 JSR PC,TIMERX ;GO WAIT FOR RECEIVER DONE BIT
2810                                     ;TO SET
2811 007332 104401 017747 TYPE ,RDB ;TYPE RECEIVER DONE BIT MESSAGE
2812 007336 104000 ERROR +0 ;RCSR DONE BIT NEVER SET
2813 007340 016767 171644 171644 MOV $TMP3,$TMP4 ;GET THE TRANSMITTER CONTROL
2814                                     ;STATUS REGISTER ADDRESS
2815 007346 162767 000002 171636 SUB #2,$TMP4 ;FORM THE RECEIVER DATA BUFFER
2816                                     ;ADDRESS
2817 007354 017767 171632 171632 MOV @$TMP4,$TMP5 ;STORE THE CHARACTER FROM THE
2818                                     ;RECEIVER BUFFER + REST OF CONTENTS
2819 007362 004767 005132 JSR PC,DATCHK ;GO TO COMPARE EXPECTED & RECEIVED
2820                                     ;DATA
2821 007366 000737 BR 1$ ;GO BACK TO ISSUE ANOTHER CHARACTER
2822 007370 104401 001252 REDO3: TYPE ,SQUES ;TYPE A QUESTION MARK(?)
2823 007374 000167 177456 JMP PRG4A ;REITERATE THE XDBR QUESTION TO USER
2824 007400 104401 001252 REDO3A: TYPE, SQUES ;TYPE '?' INDICATING USER TYPED
2825                                     ;SOMETHING WRONG FOR CHARACTER LENGTH
2826 007404 000167 177612 JMP PRG4C ;GO BACK TO REISSUE QUESTION
2827
2828 ;THIS IS PROGRAM #5
2829 ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
2830 ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
2831 ;
2832
2833 007410 000240 PRG5: NOP

```



```

2890 007574 012667 171406      MOV      (SP)+,$TMP2      ;GET THE ANSWER TYPED
2891                          ;WE ARE NOW READY TO INITIALIZE THE BINARY COUNT AND GET
2892                          ;THE BINARY CHARACTER
2893                          ;
2894 007600 012767 177777 171416  MOV      #-1,$TMP11      ;SET LEAD IN VARIABLE TO -1
2895 007606 016767 171412 171412  PRG5B:  MOV      $TMP11,$TMP12  ;STORE PREVIOUS BINARY CHARACTER
2896 007614 005267 171406      INC      $TMP12          ;FLIP BINARY CHARACTER AGAIN
2897 007620 042767 177400 171400  BIC      #177400,$TMP12  ;MASK TO 8 BITS
2898 007626 016767 171374 171370  MOV      $TMP12,$TMP11  ;STORE BINARY CHARACTER
2899 007634 016767 171366 171342  MOV      $TMP12,$TMP1   ;STORE BINARY CHARACTER
2900                          ;FOR FUTURE USE
2901 007642 016767 171334 171340  MOV      $TMP0,$TMP3    ;GET THE XDBR ADDRESS
2902 007650 162767 000002 171332  SUB      #2,$TMP3       ;FORM THE XCSR ADDRESS
2903 007656 005767 171324      1$:    TST      $TMP2       ;DO WE RANDOM STALL?
2904 007662 001402      BEQ      2$           ;BRANCH IF IT WASN'T DESIRED
2905 007664 004767 004462      JSR      PC,$STALL    ;GO STALL RANDOM VALUE OF MSEC.
2906 007670 004767 004566      2$:    JSR      PC,$TIMETX  ;GO WAIT FOR TRANSMITTER DONE
2907                          ;BIT TO SET
2908 007674 104401 017700      TYPE    ,XDB         ;TYPE TRANSMITTER DONE BIT MESSAGE
2909 007700 104000      ERROR  +0          ;XCSR DONE BIT NEVER SET
2910 007702 052777 000004 171300  BIS      #BIT2,@$TMP3  ;SET THE MAINTENANCE BIT IN THE
2911                          ;TRANSMITTER CONTROL STATUS REGISTER
2912 007710 016777 171270 171264  MOV      $TMP1,@$TMP0  ;LOAD TRANSMITTER DATA BUFFER
2913                          ;WITH SELECTED CHARACTER
2914 007716 004767 004522      JSR      PC,$TIMERX   ;GO WAIT FOR RECEIVER DONE BIT TO SET
2915 007722 104401 017747      TYPE    ,RDB         ;TYPE RECEIVER DONE BIT MESSAGE
2916 007726 104000      ERROR  +0          ;PCSR DONE BIT NEVER SET
2917 007730 016767 171254 171254  MOV      $TMP3,$TMP4   ;GET THE TRANSMITTER CONTROL
2918                          ;STATUS REGISTER ADDRESS
2919 007736 162767 000002 171246  SUB      #2,$TMP4     ;FORM THE RECEIVER DATA BUFFER
2920                          ;ADDRESS
2921 007744 017767 171242 171242  MOV      @$TMP4,$TMP5  ;STORE THE CHARACTER FROM THE
2922                          ;RECEIVER BUFFER + REST OF CONTENTS
2923 007752 004767 004542      JSR      PC,$DATCHK  ;GO TO COMPARE EXPECTED & RECEIVED
2924                          ;DATA
2925 007756 000713      BR      PRG5B       ;GO BACK TO ISSJE ANOTHER BINARY
2926                          ;CHARACTER
2927 007760 104401 001252      RED04: TYPE    ,SQUES  ;TYPE A QUESTION MARK(?)
2928 007764 000167 177426      JMP      PRG5A       ;GO BACK TO REITERATE XDBR QUESTION
2929 007770 104401 001252      RED04A: TYPE    ,SQUES ;TYPE '?' INDICATING USER TYPED
2930                          ;SOMETHING WRONG FOR CHARACTER LENGTH
2931 007774 000167 177536      JMP      PRG5C       ;GO BACK TO REISSUE QUESTION
2932
2933                          ;THIS ROUTINE WILL SET UP:
2934                          ; RCSR - RECEIVER STATUS REGISTER
2935                          ; RBUF - RECEIVER BUFFER REGISTER
2936                          ; XCSR - TRANSMITTER STATUS REGISTER
2937                          ; XBUF - TRANSMITTER BUFFER REGISTER
2938                          ;INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS
2939                          ;TESTED, AND THEREAFTER, AT THE END OF A PROGRAM TO CYCLE THRU
2940                          ;ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED)
2941 010000 016767 171254 171402  DLADDR: MOV      DLBASE,DLRCSR ;STORE RECEIVER STATUS REGISTER
2942                          ;OF CURRENT DEVICE
2943 010006 062767 000002 171244  ADD      #2,DLBASE     ;FORM RECEIVER BUFFER REGISTER
2944                          ;OF CURRENT DEVICE
2945 010014 016767 171240 171370  MOV      DLBASE,DLRDBR ;STORE RECEIVER BUFFER REGISTER

```



```

3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071 010406
3072 010406 104407
3073 010410 032777 040000 170522
3074 010416 001111
3075
3076 010420 000416
3077
3078 010422 013746 000004
3079 010426 012737 010446 000004
3080 010434 005737 177060
3081 010440 012637 000004
3082 010444 000463
3083 010446 022626
3084 010450 012637 000004
3085 010454 000423
3086 010456
3087 010456 032777 000400 170454
3088 010464 001404
3089 010466 127767 170446 170406
3090 010474 001462
3091 010476 105767 170401
3092 010502 001421
3093 010504 126767 170405 170371
3094 010512 101015
3095 010514 032777 001000 170416
3096 010522 001404
3097 010524 016767 170360 170354
3098 010532 000443
3099 010534 105067 170343
3100 010540 005067 170476
3101 010544 000415
3102 010546 032777 004000 170364
3103 010554 001011
3104 010556 005767 170316
3105 010562 001406
3106 010564 005267 170314
3107 010570 026767 170446 170306
3108 010576 002021
3109 010600 012767 000001 170276
3110 010606 016767 000044 170426
3111 010614 105267 170262
3112 010620 011667 170262
3113 010624 011667 170260

;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($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

$SCOPE:
3073 1$: BIT #BIT14,@SWR ;:TEST FOR CHANGE IN SOFT-SWR
;:LOOP ON PRESENT TEST?
3074 BNE $OVER ;:YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
3076 $XTSTR: BR 6$ ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
;:THIS INSTRUCTION TO A 'NOP' (NOP-240)
3078 MOV @WERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
3079 MOV #5$,@WERRVEC ;:SET FOR TIMEOUT
3080 TST @#177060 ;:TIME OUT ON XOR?
3081 MOV (SP)+,@WERRVEC ;:RESTORE THE ERROR VECTOR
3082 BR $SVLAD ;:GO TO THE NEXT TEST
3083 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
3084 MOV (SP)+,@WERRVEC ;:RESTORE THE ERROR VECTOR
3085 BR 7$ ;:LOOP ON THE PRESENT TEST
3086 6$: ;*****END OF CODE FOR THE XOR TESTER*****
3087 BIT #BIT08,@SWR ;:LOOP ON SPEC. TEST?
3088 BEQ 2$ ;:BR IF NO
3089 CMPB @SWR,$TSTNM ;:ON THE RIGHT TEST? SWR<7:0>
3090 BEQ $OVER ;:BR IF YES
3091 2$: TSTB $ERFLC ;:HAS AN ERROR OCCURRED?
3092 BEQ 3$ ;:BR IF NO
3093 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED
3094 BHI 3$ ;:BR IF NO
3095 BIT #BIT09,@SWR ;:LOOP ON ERROR?
3096 BEQ 4$ ;:BR IF NO
3097 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
3098 BR $OVER
3099 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
3100 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
3101 BR 1$ ;:ESCAPE TO THE NEXT TEST
3102 3$: BIT #BIT11,@SWR ;:INHIBIT ITERATIONS?
3103 BNE 1$ ;:BR IF YES
3104 TST $PASS ;:IF FIRST PASS OF PROGRAM
3105 BEQ 1$ ;: INHIBIT ITERATIONS
3106 INC $ICNT ;:INCREMENT ITERATION COUNT
3107 CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
3108 BGE $OVER ;:BR IF MORE ITERATION REQUIRED
3109 1$: MOV #1,$ICNT ;:REINITIALIZE THE ITERATION COUNTER
3110 MOV $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
3111 $SVLAD: INCB $TSTNM ;:COUNT TEST NUMBERS
3112 MOV (SP),$LPADR ;:SAVE SCOPE LOOP ADDRESS
3113 MOV (SP),$LPERR ;:SAVE ERROR LOOP ADDRESS

```



```

3114 010630 005067 170410 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3115 010634 112767 000001 170253 MOV #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3116 010642 016777 170234 170272 $OVER: MOV $STSTM,@DISPLAY ;;DISPLAY TEST NUMBER
3117 010650 016716 170232 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
3118 010654 000002 RTI ;;FIXES PS
3119 010656 000100 $MXCNT: 100 ;;MAX. NUMBER OF ITERATIONS
  
```

3120
 3121 .SBTTL ERROR HANDLER ROUTINE

```

3122  

3123 *****  

3124 *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,  

3125 *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL  

3126 *AND GO TO $ERRTYP ON ERROR  

3127 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  

3128 *SW15 1 HALT ON ERROR  

3129 *SW13 1 INHIBIT ERROR TYPEOUTS  

3130 *SW10=1 BELL ON ERROR  

3131 *SW09-1 LOOP ON ERROR  

3132 *CALL  

3133 * ERROR N ;;ERROR EMT AND N-ERROR ITEM NUMBER
  
```

```

3134  

3135 010660 $ERROR:  

3136 010660 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR  

3137 010662 105267 170215 7$: INCB $ERFLG ;;SET THE ERROR FLAG  

3138 010666 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO  

3139 010670 016777 170206 170244 MOV $STSTM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG  

3140 010676 032777 002000 170234 BIT #BIT10,@SWR ;;BELL ON ERROR?  

3141 010704 001402 BEQ 1$ ;;NO - SKIP  

3142 010706 104401 001246 TYPE ,SBELL ;;RING BELL  

3143 010712 005267 170174 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS  

3144 010716 011667 170174 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION  

3145 010722 162767 000002 170166 SUB #2,$ERRPC  

3146 010730 117767 170162 170156 MOV @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE  

3147 010736 032777 020000 170174 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET  

3148 010744 001004 BNE 20$ ;;SKIP TYPEOUTS  

3149 010746 004767 000046 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE  

3150 010752 104401 001253 TYPE ,SRLF  

3151 010756 20$:  

3152 010756 005777 170156 2$: TST @SWR ;;HALT ON ERROR  

3153 010762 100002 BPL 3$ ;;SKIP IF CONTINUE  

3154 010764 000000 HALT ;;HALT ON ERROR!  

3155 010766 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR  

3156 010770 032777 001000 170142 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?  

3157 010776 001402 BEQ 4$ ;;BR IF NO  

3158 011000 016716 170104 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING  

3159 011004 005767 170234 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS  

3160 011010 001402 BEQ 5$ ;;BR IF NONE  

3161 011012 016716 170226 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE  

3162 011016 5$:  

3163 011016 000002 RTI ;;RETURN
  
```

3164
 3165 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

3166  

3167 *****  

3168 *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH  

3169 *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
  
```

```

3170
3171
3172 011020
3173 011020 104401 001253
3174 011024 010046
3175 011026 005000
3176 011030 153700 001114
3177 011034 001004
3178
3179 011036 016746 170054
3180
3181 011042 104402
3182 011044 000426
3183 011046 005300
3184 011050 006300
3185 011052 006300
3186 011054 006300
3187 011056 062700 001310
3188 011062 012067 000004
3189 011066 001404
3190 011070 104401
3191 011072 000000
3192 011074 104401 001253
3193 011100 012067 000004
3194 011104 001404
3195 011106 104401
3196 011110 000000
3197 011112 104401 001253
3198 011116 011000
3199 011120 001004
3200 011122 012600
3201 011124 104401 001253
3202 011130 000207
3203 011132
3204 011132 013046
3205 011134 104402
3206 011136 005710
3207 011140 001770
3208 011142 104401 011150
3209 011146 000771
3210 011150 020040 000
3211 011154
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225

```

```

;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
$ERRTYP:
TYPE ,SCLRF ;:'CARRIAGE RETURN' & 'LINE FEED'
MOV R0,-(SP) ;:SAVE R0
CLR R0 ;:PICKUP THE ITEM INDEX
BISB @($ITEMB,R0
BNE 1$ ;:IF ITEM NUMBER IS ZERO, JUST
;:TYPE THE PC OF THE ERROR
MOV $ERRPC,-(SP) ;:SAVE $ERRPC FOR TYPEOUT
;:ERROR ADDRESS
;:GO TYPE--OCTAL ASCII(ALL DIGITS)
BR 6$ ;:GET OUT
1$: DEC R0 ;:ADJUST THE INDEX SO THAT IT WILL
ASL R0 ;: WORK FOR THE ERROR TABLE
ASL R0
ASL R0
ADD # $ERRTB,R0 ;:FORM TABLE POINTER
MOV (R0)+,2$ ;:PICKUP 'ERROR MESSAGE' POINTER
BEQ 3$ ;:SKIP TYPEOUT IF NO POINTER
TYPE ;:TYPE THE 'ERROR MESSAGE'
;:'ERROR MESSAGE' POINTER GOES HERE
2$: .WORD 0
TYPE ,SCLRF ;:'CARRIAGE RETURN' & 'LINE FEED'
3$: MOV (R0)+,4$ ;:PICKUP 'DATA HEADER' POINTER
BEQ 5$ ;:SKIP TYPEOUT IF 0
TYPE ;:TYPE THE 'DATA HEADER'
;:'DATA HEADER' POINTER GOES HERE
4$: .WORD 0
TYPE ,SCLRF ;:'CARRIAGE RETURN' & 'LINE FEED'
5$: MOV (R0),R0 ;:PICKUP 'DATA TABLE' POINTER
BNE 7$ ;:GO TYPE THE DATA
6$: MOV (SP)+,R0 ;:RESTORE R0
TYPE ,SCLRF ;:'CARRIAGE RETURN' & 'LINE FEED'
RTS PC ;:RETURN
7$: MOV @ (R0)+,-(SP) ;:SAVE @ (R0)+ FOR TYPEOUT
;:GO TYPE--OCTAL ASCII(ALL DIGITS)
;:IS THERE ANOTHER NUMBER?
TST (R0)
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
;* TYPOS ;:CALL FOR TYPEOUT
;* .BYTE N ;:N-1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;* .BYTE M ;:M 1 OR 0
;* ;:1=TYPE LEADING ZEROS
;* ;:0=SUPPRESS LEADING ZEROS

```

```
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238 011154 017646 000000
3239 011160 116667 000001 000211
3240 011166 112667 000207
3241 011172 062716 000002
3242 011176 000406
3243 011200 112767 000001 000171
3244 011206 112767 000006 000165
3245 011214 112767 000005 000154
3246 011222 010346
3247 011224 010446
3248 011226 010546
3249 011230 116704 000145
3250 011234 005404
3251 011236 062704 000006
3252 011242 110467 000132
3253 011246 116704 000125
3254 011252 016605 000012
3255 011256 005003
3256 011260 006105 1$:
3257 011262 000404 BR 3$
3258 011264 006105 2$:
3259 011266 006105 ROL R5
3260 011270 006105 ROL R5
3261 011272 010503 MOV R5,R3
3262 011274 006103 3$:
3263 011276 105367 000076 DECB $OMODE
3264 011302 100016 BPL 7$
3265 011304 042703 177770 BIC #177770,R3
3266 011310 001002 BNE 4$
3267 011312 005704 TST R4
3268 011314 001403 BEQ 5$
3269 011316 005204 4$:
3270 011320 052703 000060 INC R4
3271 011324 052703 000040 5$:
3272 011330 110367 000040 BIS #'0,R3
3273 011334 104401 011374 MOV R3,8$
3274 011340 105367 000032 7$:
3275 011344 003347 BGT 2$
3276 011346 002402 BLT 6$
3277 011350 005204 INC R4
3278 011352 000744 BR 2$
3279 011354 012605 6$:
3280 011356 012604 MOV (SP)+,R5
3281 011360 012603 MOV (SP)+,R4
MOV (SP)+,R3
```

```

3282 011362 016666 000002 000004      MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
3283 011370 012616                    MOV      (SP)+,(SP)
3284 011372 000002                    RTI                          ;;RETURN
3285 011374 000      8$: .BYTE 0      ;;STORAGE FOR ASCII DIGIT
3286 011375 000      .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
3287 011376 000      $OCNT: .BYTE 0      ;;OCTAL DIGIT COUNTER
3288 011377 000      $OFILL: .BYTE 0      ;;ZERO FILL SWITCH
3289 011400 000000      $OMODE: .WORD 0      ;;NUMBER OF DIGITS TO TYPE
3290
3291      .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3292
3293      ;*****
3294      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3295      ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3296      ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3297      ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3298      ;*REPLACED WITH SPACES.
3299      ;*CALL:
3300      ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
3301      ;*      TYPDS      ;;GO TO THE ROUTINE
3302
3303      $TYPDS:
3304      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
3305      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
3306      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
3307      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
3308      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
3309      MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
3310      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
3311      BPL      1$      ;;BR IF INPUT IS POS.
3312      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
3313      MOV      #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
3314      CLR      R0      ;;ZERO THE CONSTANTS INDEX
3315      MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
3316      MOV      #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
3317      CLR      R2      ;;CLFAR THE BCD NUMBER
3318      MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
3319      SUB      R1,R5      ;;FORM THIS BCD DIGIT
3320      BLT      4$      ;;BR IF DONE
3321      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
3322      BR      3$
3323      ADD      R1,R5      ;;ADD BACK THE CONSTANT
3324      TST      R2      ;;CHECK IF BCD DIGIT 0
3325      BNE      5$      ;;FALL THROUGH IF 0
3326      TSTB      (SP)      ;;STILL DOING LEADING 0'S?
3327      BMI      7$      ;;BR IF YES
3328      ASLB      (SP)      ;;MSD?
3329      BCC      6$      ;;BR IF NO
3330      MOV      1(SP),-1(R3)      ;;YES--SET THE SIGN
3331      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
3332      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
3333      MOV      R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3334      TST      (R0)+      ;;JUST INCREMENTING
3335      CMP      R0,#10      ;;CHECK THE TABLE INDEX
3336      BLT      2$      ;;GO DO THE NEXT DIGIT
3337      BGT      8$      ;;GO TO EXIT
  
```

```
3338 011536 010502          MOV      R5,R2          ;;GET THE LSD
3339 011540 000764          BR       6$             ;;GO CHANGE TO ASCII
3340 011542 105726          8$:     TSTB      (SP)+  ;;WAS THE LSD THE FIRST NON-ZERO?
3341 011544 100003          BPL     9$             ;;BR IF NO
3342 011546 116663 177777 177776  MOVB    -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
3343 011554 105013          9$:     LRB      (R3)   ;;SET THE TERMINATOR
3344 011556 012605          MOV     (SP)+,R5      ;;POP STACK INTO R5
3345 011560 012603          MOV     (SP)+,R3      ;;POP STACK INTO R3
3346 011562 012602          MOV     (SP)+,R2      ;;POP STACK INTO R2
3347 011564 012601          MOV     (SP)+,R1      ;;POP STACK INTO R1
3348 011566 012600          MOV     (SP)+,R0      ;;POP STACK INTO R0
3349 011570 104401 011616  TYPE     $DBLK        ;;NOW TYPE THE NUMBER
3350 011574 016666 000002 000004  MOV     2(SP),4(SP)   ;;ADJUST THE STACK
3351 011602 012616          MOV     (SP)+,(SP)
3352 011604 000002          RTI                      ;;RETURN TO USER
3353 011606 023420          $DTBL: 10000.
3354 011610 001750          1000.
3355 011612 000144          100.
3356 011614 000012          10.
3357 011616 000004          $DBLK: .BLKW 4
3358
3359          .SBTTL  TTY INPUT ROUTINE
3360
3361          ;*****
3362          .ENABL  LSB
3363 011626 000000          $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
3364 011630 000000          $TKQIN: .WORD 0     ;;INPUT POINTER
3365 011632 000000          $TKQOUT: .WORD 0    ;;OUTPUT POINTER
3366 011634 000010          $TKQSRT: .BLKB 8.   ;;TTY KEYBOARD QUEUE
3367          $TKQEND=.
3368
3369          ;*TK INITIALIZE ROUTINE
3370          ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
3371          ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
3372
3373          ;*CALL:
3374          ;*      JSR      PC,$TKINT
3375          ;*      RETURN
3376
3377 011644 005067 177756          $TKINT: CLR      $TKCNT  ;;CLEAR COUNT OF ITEMS IN QUEUE
3378 011650 012767 011634 177752  MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
3379 011656 016767 177746 177746  MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
3380 011664 012737 011714 000060  MOV     #$TKSRV,@TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
3381 011672 012737 000200 000062  MOV     #200,@TKVEC+2  ;;'BR' LEVEL 4
3382 011700 005777 167242          TST     @TKB           ;;CLEAR DONE FLAG
3383 011704 012777 000100 167232  MCV     #100,@TKS     ;;ENABLE TTY KEYBOARD INTERRUPT
3384 011712 000207          RTS      PC           ;;RETURN TO CALLER
3385
3386          ;*TK SERVICE ROUTINE
3387          ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
3388          ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
3389          ;*IT IN THE QUEUE.
3390          ;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
3391          ;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (200)
3392
3393 011714 117746 167226          $TKSRV: MOVB    @TKB,-(SP) ;;PICKUP THE CHARACTER
```

```
3394 011720 042716 177600      BIC    #^C177,(SP)    ;;STRIP THE JUNK
3395 011724 021627 000003      CMP    (SP),#3       ;;IS IT A CONTROL C?
3396 011730 001007              BNE    1$            ;;BRANCH IF NO
3397 011732 104401 013030      TYPE   ,SCNTLC       ;;TYPE A CONTROL-C (^C)
3398 011736 004767 177702      JSR    PC,$TKINT     ;;INIT THE KEYBOARD
3399 011742 005726              TST    (SP)+         ;;CLEAN UP STACK
3400 011744 000167 166230      JMP    200           ;;CONTROL C RESTART
3401 011750 021627 000007      1$:    CMP    (SP),#7       ;;IS IT A CONTROL G?
3402 011754 001004              BNE    2$            ;;BRANCH IF NO
3403 011756 022767 000176 167154  CMP    #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
3404 011764 001500              BEQ    6$            ;;GO TO SWR CHANGE
3405
3406 011766
3407 011766 022767 000010 177632      2$:    CMP    #8,$TKCNT     ;;IS THE QUEUE FULL?
3408 011774 001004              BNE    3$            ;;BRANCH IF NO
3409 011776 104401 001246      TYPE   ,SBELL        ;;RING THE TTY BELL
3410 012002 005726              TST    (SP)+         ;;CLEAN CHARACTER OFF OF STACK
3411 012004 000451              BR     5$            ;;EXIT
3412 012006 021627 000023      3$:    CMP    (SP),#23     ;;IS IT A CONTROL-S?
3413 012012 001021              BNE    32$           ;;BRANCH IF NO
3414 012014 005077 167124      CLR    @TKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
3415 012020 005726              TST    (SP)+         ;;CLEAN CHAR OFF STACK
3416 012022 105777 167116      31$:   TSTB   @TKS          ;;WAIT FOR A CHAR
3417 012026 100375              BPL    31$           ;;LOOP UNTIL ITS THERE
3418 012030 117746 167112      MOVB   @TKB,-(SP)    ;;GET THE CHARACTER
3419 012034 042716 177600      BIC    #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
3420 012040 022627 000021      CMP    (SP)+,#21     ;;IS IT A CONTROL-Q?
3421 012044 001366              BNE    31$           ;;BRANCH IF NO
3422 012046 012777 000100 167070      MOV    #100,@TKS    ;;REENABLE TTY KEYBOARD INTERRUPTS
3423 012054 000002              RTI
3424 012056 005267 177544      32$:   INC    $TKCNT       ;;COUNT THIS CHARACTER
3425 012062 021627 000140      CMP    (SP),#140    ;;IS IT UPPER CASE?
3426 012066 002405              BLT    4$            ;;BRANCH IF YES
3427 012070 021627 000175      CMP    (SP),#175    ;;IS IT A SPECIAL CHAR?
3428 012074 003002              BGT    4$            ;;BRANCH IF YES
3429 012076 042716 000040      PIC    #40,(SP)     ;;MAKE IT UPPER CASE
3430 012102 112677 177522      4$:    MOVB   (SP)+,@TKQIN ;;AND PUT IT IN QUEUE
3431 012106 005267 177516      INC    $TKQIN        ;;UPDATE THE POINTER
3432 012112 026727 177512 011644      CMP    $TKQIN,#$TKQEND ;;GO OFF THE END?
3433 012120 001003              BNE    5$            ;;BRANCH IF NO
3434 012122 012767 011634 177500      MOV    #$TKQSRRT,$TKQIN ;;RESET THE POINTER
3435 012130 000002      5$:    RTI
3436
3437
3438
3439
3440
3441
3442 012132 022767 000176 167000  $CKSWR: CMP    #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED?
3443 012140 001124              BNE    15$           ;;EXIT IF NOT
3444 012142 105777 166776      TSTB   @TKS          ;;IS A CHAR WAITING?
3445 012146 100121              BPL    15$           ;;IF NOT, EXIT
3446 012150 117746 166772      MOVB   @TKB,-(SP)    ;;YES
3447 012154 042716 177600      BIC    #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
3448 012160 021627 000007      CMP    (SP),#7       ;;IS IT A CONTROL-G?
3449 012164 001300              BNE    2$            ;;IF NOT, PUT IT IN THE TTY QUEUE
```

```

3450                                     ;;AND EXIT
3451
3452                                     ;*****
3453                                     ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
3454                                     ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
3455                                     ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
3456 012166 126727 166742 000001 6$:   CMPB   $AUTOB,#1   ;;ARE WE RUNNING IN AUTO-MODE?
3457 012174 001674           BEQ     2$           ;;BRANCH IF YES
3458 012176 005726           TST    (SP)+       ;;CLEAR CONTROL-G OFF STACK
3459 012200 004767 177440     JSP    PC,$TKINT   ;;FLUSH THE TTY INPUT QUEUE
3460 012204 005077 166734     CLR    @STKS       ;;DISABLE TTY KEYBOARD INTERRUPTS
3461 012210 112767 000001 166717     MOVB  #1,$INTAG    ;;SET INTERRUPT MODE INDICATOR
3462
3463 012216 104401 013042           TYPE  ,$CNTLG      ;;ECHO THE CONTROL-G (^G)
3464 012222 104401 013047     $GTSWR: TYPE  ,$MSWR      ;;TYPE CURRENT CONTENTS
3465 012226 016746 165744     MOV   SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
3466 012232 104402           TYPOC           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3467 012234 104401 013060     TYPE  ,$MNEW      ;;PROMPT FOR NEW SWR
3468 012240 005046           19$: CLR  -(SP)      ;;CLEAR COUNTER
3469 012242 005046           CLR  -(SP)      ;;THE NEW SWR
3470 012244 105777 166674     7$:  TSTB  @STKS    ;;CHAR THERE?
3471 012250 100375           BPI   7$        ;;IF NOT TRY AGAIN
3472
3473 012252 117746 166670     MOVB  @STKB,-(SP) ;;PICK UP CHAR
3474 012256 042716 177600     BIC   #^C177,(SP) ;;MAKE IT 7-BIT ASCII
3475
3476 012262 021627 000003           CMP   (SP),#3     ;;IS IT A CONTROL-C?
3477 012266 001015           BNE   9$          ;;BRANCH IF NOT
3478 012270 104401 013030     TYPE  ,$CNTLC     ;;YES, ECHO CONTROL-C (^C)
3479 012274 062706 000006     ADD   #6,SP       ;;CLEAN UP STACK
3480 012300 126727 166631 000001     CMPB  $INTAG,#1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
3481 012306 001003           BNE   8$          ;;BRANCH IF NO
3482 012310 012777 000100 166626     MOV   #100,@STKS ;;ALLOW TTY KEYBOARD INTERRUPTS
3483 012316 000167 165656     8$:  JMP   200      ;;CONTROL-C RESTART
3484
3485
3486 012322 021627 000025     9$:  CMP   (SP),#25  ;;IS IT A CONTROL-U?
3487 012326 001005           BNE   10$         ;;BRANCH IF NOT
3488 012330 104401 013035     TYPE  ,$CNTLU     ;;YES, ECHO CONTROL-U (^U)
3489 012334 062706 000006     20$: ADD   #6,SP    ;;IGNORE PREVIOUS INPUT
3490 012340 000737           BR   19$         ;;LET'S TRY IT AGAIN
3491
3492
3493 012342 021627 000015     10$: CMP   (SP),#15   ;;IS IT A <CR>?
3494 012346 001022           BNE   16$         ;;BRANCH IF NO
3495 012350 005766 000004     TST   4(SP)       ;;YES, IS IT THE FIRST CHAR?
3496 012354 001403           BEQ   11$         ;;BRANCH IF YES
3497 012356 016677 000002 166554     MOV   2(SP),@SWR  ;;SAVE NEW SWR
3498 012364 062706 000006     11$: ADD   #6,SP    ;;CLEAR UP STACK
3499 012370 104401 001253     14$: TYPE  ,$CRLF   ;;ECHO <CR> AND <LF>
3500 012374 126727 166535 000001     CMPB  $INTAG,#1   ;;RE-ENABLE TTY KBD INTERRUPTS?
3501 012402 001003           BNE   15$         ;;BRANCH IF NOT
3502 012404 012777 000100 166532     MOV   #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
3503 012412 000002           15$: RTI          ;;RETURN
3504 012414 004767 000762     16$: JSR   PC,$TYPEC ;;ECHO CHAR
3505 012420 021627 000060     CMP   (SP),#60   ;;CHAR < 0?

```

```
3506 012424 002420          BLT      18$          ;;BRANCH IF YES
3507 012426 021627 000067    CMP      (SP),#67    ;;CHAR > 7?
3508 012432 003015          BGT      18$          ;;BRANCH IF YES
3509 012434 042726 000060    BIC      #60,(SP)+   ;;STRIP-OFF ASCII
3510 012440 005766 000002    TST      2(SP)       ;;IS THIS THE FIRST CHAR
3511 012444 001403          BEQ      17$          ;;BRANCH IF YES
3512 012446 006316          ASL      (SP)        ;;NO, SHIFT PRESENT
3513 012450 006316          ASL      (SP)        ;;CHAR OVER TO MAKE
3514 012452 006316          ASL      (SP)        ;;ROOM FOR NEW ONE.
3515 012454 005266 000002    17$: INC      2(SP)       ;;KEEP COUNT OF CHAR
3516 012460 056616 177776    BIS      -2(SP),(SP) ;;SET IN NEW CHAR
3517 012464 000667          BR       7$          ;;GET THE NEXT ONE
3518 012466 104401 001252    18$: TYPE  $QUES     ;;TYPE ?<CR><LF>
3519 012472 000720          BR       20$        ;;SIMULATE CONTROL-U
3520          .DSABL  LSB
3521
3522
3523          ;;*****
3524          ;;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3525          ;;CALL:
3526          ;; RDCHR          ;;GET A CHARACTER FROM THE QUEUE
3527          ;; RETURN HERE   ;;CHARACTER IS ON THE STACK
3528          ;;              ;;WITH PARITY BIT STRIPPED OFF
3529          ;;
3530
3531 012474 011646          $RDCHR: MOV      (SP),-(SP) ;;PUSH DOWN THE PC AND
3532 012476 016666 000004 000002  MOV      4(SP),2(SP) ;;THE PS
3533 012504 005066 000004          CLR      4(SP)       ;;GET READY FOR A CHARACTER
3534 012510 005046          CLR      -(SP)      ;;PUT NEW PS ON STACK
3535 012512 012746 012520          MOV      #64$,-(SP) ;;PUT NEW PC ON STACK
3536 012516 000002          RTI             ;;POP NEW PC AND PS
3537 012520          64$:
3538 012520 005767 177102    1$: TST      $STKCNT   ;;WAIT ON A CHARACTER
3539 012524 001775          BEQ      1$
3540 012526 005367 177074          DEC      $STKCNT   ;;DECREMENT THE COUNTER
3541 012532 117766 177074 000004  MOVB     @STKQOUT,4(SP) ;;GET ONE CHARACTER
3542 012540 005267 177066          INC      $STKQOUT  ;;UPDATE THE POINTER
3543 012544 026727 177062 011644  CMP      $STKQOUT,#$STKQEND ;;DID IT GO OFF OF THE END?
3544 012552 001003          BNE      2$        ;;BRANCH IF NO
3545 012554 012767 011634 177050  MOV      #$STKQRT,$STKQOUT ;;RESET THE POINTER
3546 012562 000002          2$: RTI             ;;RETURN
3547          ;;*****
3548          ;;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3549          ;;CALL:
3550          ;; RDLIN          ;;INPUT A STRING FROM THE TTY
3551          ;; RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3552          ;;              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3553
3554 012564 010346          $RDLIN: MOV      R3,-(SP) ;;SAVE R3
3555 012566 005046          CLR      -(SP)     ;;CLEAR THE RUBOUT KEY
3556 012570 012703 013020    1$: MOV      #$TTYIN,R3 ;;GET ADDRESS
3557 012574 022703 013030    2$: CMP      #$TTYIN+8.,R3 ;;BUFFER FULL?
3558 012600 101456          BLOS     4$         ;;BR IF YES
3559 012602 104410          RDCHR     ;;GO READ ONE CHARACTER FROM THE TTY
3560 012604 112613          MOVB     (SP)+,(R3) ;;GET CHARACTER
3561 012606 122713 000177    10$: CMPB    #177,(R3) ;;IS IT A RUBOUT
```



```

3562 012612 001022      BNE      5$      ;;BR IF NO
3563 012614 005716      TST      (SP)    ;;IS THIS THE FIRST RUBOUT?
3564 012616 001007      BNE      6$      ;;BR IF NO
3565 012620 112767 000134 000170  MOVB     #' \ ,9$  ;;TYPE A BACK SLASH
3566 012626 104401 013016      TYPE     ,9$
3567 012632 012716 177777      MOV      #-1,(SP) ;;SET THE RUBOUT KEY
3568 012636 005303      DEC      R3      ;;BACKUP BY ONE
3569 012640 020327 013020 6$:  CMP      R3,#$TTYIN ;;STACK EMPTY?
3570 012644 103434      BLO      4$      ;;BR IF YES
3571 012646 111367 000144  MOVB     (R3),9$  ;;SETUP TO TYPEOUT THE DELETED CHAR.
3572 012652 104401 013016      TYPE     ,9$      ;;GO TYPE
3573 012656 000746      BR       2$      ;;GO READ ANOTHER CHAR.
3574 012660 005716      TST      (SP)    ;;RUBOUT KEY SET?
3575 012662 001406      BEQ      7$      ;;BR IF NO
3576 012664 112767 000134 000124  MOVB     #' \ ,9$  ;;TYPE A BACK SLASH
3577 012672 104401 013016      TYPE     ,9$
3578 012676 005016      CLR      (SP)    ;;CLEAR THE RUBOUT KEY
3579 012700 122713 000025 7$:  CMPB     #25,(R3) ;;IS CHARACTER A CTRL U?
3580 012704 001003      BNE      8$      ;;BR IF NO
3581 012706 104401 013035      TYPE     , $CNTLU ;;TYPE A CONTROL 'U'
3582 012712 000726      BR       1$      ;;GO START OVER
3583 012714 122713 000022 8$:  CMPB     #22,(R3) ;;IS CHARACTER A '^R'?
3584 012720 001011      BNE      3$      ;;BRANCH IF NO
3585 012722 105013      CLRB     (R3)    ;;CLEAR THE CHARACTER
3586 012724 104401 001253      TYPE     , $CRLF ;;TYPE A 'CR' & 'LF'
3587 012730 104401 013020      TYPE     , $TTYIN ;;TYPE THE INPUT STRING
3588 012734 000717      BR       2$      ;;GO PICKUP ANOTHER CHARACTER
3589 012736 104401 001252 4$:  TYPE     , $QUES ;;TYPE A '?'
3590 012742 000712      BR       1$      ;;CLEAR THE BUFFER AND LOOP
3591 012744 111367 000046 3$:  MOVB     (R3),9$  ;;ECHO THE CHARACTER
3592 012750 104401 013016      TYPE     ,9$
3593 012754 122723 000015      CMPB     #15,(R3)+ ;;CHECK FOR RETURN
3594 012760 001305      BNE      2$      ;;LOOP IF NOT RETURN
3595 012762 105063 177777      CLRB     -1(R3)  ;;CLEAR RETURN (THE 15)
3596 012766 104401 001254      TYPE     , $LF   ;;TYPE A LINE FEED
3597 012772 005726      TST      (SP)+  ;;CLEAN RUBOUT KEY FROM THE STACK
3598 012774 012603      MOV      (SP)+,R3 ;;RESTORE R3
3599 012776 011646      MOV      (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3600 013000 016666 000004 000002  MOV      4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
3601 013006 012766 013020 000004  MOV      #$TTYIN,4(SP)
3602 013014 000002      RTI
3603 013016 000      9$:  .BYTE   0      ;;STORAGE FOR ASCII CHAR. TO TYPE
3604 013017 000      .BYTE   0      ;;TERMINATOR
3605 013020 000010  $TTYIN: .BLKB   8.  ;;RESERVE 8 BYTES FOR TTY INPUT
3606 013030 041536 005015 000  $CNTLC: .ASCIZ  / ^C / <15><12>  ;;CONTROL 'C'
3607 013035 0136 006525 000012  $CNTLU: .ASCIZ  / ^U / <15><12>  ;;CONTROL 'U'
3608 013042 043536 005015 000  $CNTLG: .ASCIZ  / ^G / <15><12>  ;;CONTROL 'G'
3609 013047 015 051412 051127  $MSWR:  .ASCIZ  <15><12> / SWR /
3610 013054 036440 000040
3611 013060 020040 042516 020127  $MNEW:  .ASCIZ  / NEW /
3612 013066 020075 000
3613 013072      .EVEN
3614
3615      .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
3616
3617      ;:*****

```

3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673

013072 011646
013074 016666 000004 000002
013102 010046
013104 010146
013106 010246
013110 104411
013112 012600
013114 010067 000100
013120 005001
013122 005002
013124 112046
013126 001420
013130 122716 000060
013134 003026
013136 122716 000067
013142 002423
013144 006301
013146 006102
013150 006301
013152 006102
013154 006301
013156 006102
013160 042716 177770
013164 062601
013166 000756
013170 005726
013172 010166 000012
013176 010267 000026
013202 012602
013204 012601
013206 012600
013210 000002
013212 005726
013214 105010
013216 104401
013220 000000
013222 104401 001252
013226 000730
013230 000000

```

;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;*CHANGE IT TO BINARY.
;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
;*CALL:
;*      RDOCT          ;;READ AN OCTAL NUMBER
;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
;*                  ;;HIGH ORDER BITS ARE IN $HIOCT

$RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
        MOV      4(SP),2(SP)    ;;INPUT NUMBER
        MOV      R0,-(SP)       ;;PUSH R0 ON STACK
        MOV      R1,-(SP)       ;;PUSH R1 ON STACK
        MOV      R2,-(SP)       ;;PUSH R2 ON STACK
1$:     RD_LIN      ;;READ AN ASCII LINE
        MOV      (SP)+,R0       ;;GET ADDRESS OF 1ST CHARACTER
        MOV      R0,5$         ;;AND SAVE IT
        CLR      R1            ;;CLEAR DATA WORD
        CLR      R2
2$:     MOVB      (R0)+,-(SP)    ;;PICKUP THIS CHARACTER
        BEQ      3$           ;;IF ZERO GET OUT
        CMPB     #'0,(SP)      ;;MAKE SURE THIS CHARACTER
        BGT      4$           ;;IS AN OCTAL DIGIT
        CMPB     #'7,(SP)
        BLT      4$
        ASL      R1            ;;*2
        ROL      R2
        ASL      R1            ;;*4
        ROL      R2
        ASL      R1            ;;*8
        ROL      R2
        BIC      #'C7,(SP)     ;;STRIP THE ASCII JUNK
        ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
        BR       2$           ;;LOOP
3$:     TST      (SP)+         ;;CLEAN TERMINATOR FROM STACK
        MOV      R1,12(SP)     ;;SAVE THE RESULT
        MOV      R2,$HIOCT
        MOV      (SP)+,R2     ;;POP STACK INTO R2
        MOV      (SP)+,R1     ;;POP STACK INTO R1
        MOV      (SP)+,R0     ;;POP STACK INTO R0
        RTI                ;;RETURN
4$:     TST      (SP)+         ;;CLEAN PARTIAL FROM STACK
        CLRB     (R0)         ;;SET A TERMINATOR
        TYPE     ;;TYPE UP THRU THE BAD CHAR.
5$:     .WORD    0
        TYPE     ,SQUES      ;; '?' 'CR' & 'LF'
        BR      1$           ;;TRY AGAIN
$HIOCT: .WORD    0          ;;HIGH ORDER BITS GO HERE

.SBTTL  TYPE ROUTINE

;*****
;*ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.

```

```

3674      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3675      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3676      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3677      ;*
3678      ;*CALL:
3679      ;*1) USING A TRAP INSTRUCTION
3680      ;*      TYPE      ,MESADR      ;;MFSADR IS FIRST ADDRESS OF AN ASCIZ STRING
3681      ;*OR
3682      ;*      TYPE
3683      ;*      MESADR
3684      ;*
3685
3686 013232 105767 165721      $TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
3687 013236 100002      BPL      1$      ;;BR IF YES
3688 013240 000000      HALT      ;;HALT HERE IF NO TERMINAL
3689 013242 000407      BR      3$      ;;LEAVE
3690 013244 010046      1$:  MOV      R0,-(SP)      ;;SAVE R0
3691 013246 017600 000002      MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
3692 013252 112046      2$:  MOVVB   (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3693 013254 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
3694 013256 005726      TST      (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
3695 013260 012600      60$:  MOV      (SP)+,R0      ;;RESTORE R0
3696 013262 062716 000002      3$:  ADD      #2,(SP)      ;;ADJUST RETURN PC
3697 013266 000002      RTI      ;;RETURN
3698 013270 122716 000011      4$:  CMPB     #HT,(SP)      ;;BRANCH IF <HT>
3699 013274 001430      BEQ      8$
3700 013276 122716 000200      CMPB     #CRLF,(SP)      ;;BRANCH IF NOT <CR> <LF>
3701 013302 001006      BNE      5$
3702 013304 005726      TST      (SP)+      ;;POP <CR><LF> EQUIV
3703 013306 104401      TYPE      ;;TYPE A CR AND LF
3704 013310 001253      $CRLF
3705 013312 105067 000130      CLRB     $CHARCNT      ;;CLEAR CHARACTER COUNT
3706 013316 000755      BR      2$      ;;GET NEXT CHARACTER
3707 013320 004767 000056      5$:  JSR      PC,$TYPEPC      ;;GO TYPE THIS CHARACTER
3708 013324 126726 165626      6$:  CMPB     $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
3709 013330 001350      BNE      2$      ;;IF NO GO GET NEXT CHAR.
3710 013332 016746 165616      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
3711      ;;AND THE NULL CHAR.
3712 013336 105366 000001      7$:  DECB     1(SP)      ;;DOES A NULL NEED TO BE TYPED?
3713 013342 002770      BLT      6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
3714 013344 004767 000032      JSR      PC,$TYPEPC      ;;GO TYPE A NULL
3715 013350 105367 000072      DECB     $CHARCNT      ;;DO NOT COUNT AS A COUNT
3716 013354 000770      BR      7$      ;;LOOP
3717
3718      ;HORIZONTAL TAB PROCESSOR
3719
3720 013356 112716 000040      8$:  MOVVB   #' ,(SP)      ;;REPLACE TAB WITH SPACE
3721 013362 004767 000014      9$:  JSR      PC,$TYPEPC      ;;TYPE A SPACE
3722 013366 132767 000007 000052      BITB     #7,$CHARCNT      ;;BRANCH IF NOT AT
3723 013374 001372      BNE      9$      ;;TAB STOP
3724 013376 005726      TST      (SP)+      ;;POP SPACE OFF STACK
3725 013400 000724      BR      2$      ;;GET NEXT CHARACTER
3726 013402 105777 165542      $TYPEPC: TSTB     @2$TPS      ;;WAIT UNTIL PRINTER IS READY
3727 013406 100375      BPL      $TYPEPC
3728 013410 116677 000002 165534      MOVVB   2(SP),@2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3729 013416 122766 000015 000002      CMPB     #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?

```

```

3730 013424 001003          BNE      1$          ;;BRANCH IF NO
3731 013426 105067 000014  CLRB    $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
3732 013432 000406          BR      $TYPEX      ;;EXIT
3733 013434 122766 000012 000002 1$:  CMPB   #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
3734 013442 001402          BEQ     $TYPEX      ;;BRANCH IF YES
3735 013444 105227          INCB   (PC)+        ;;COUNT THE CHARACTER
3736 013446 000000          $CHARCNT: .WORD 0   ;;CHARACTER COUNT STORAGE
3737 013450 000207          $TYPEX: RTS        PC
3738
3739
3740
3741
3742

```

.SBTTL READ A DECIMAL NUMBER FROM THE TTY

```

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

```

```

*CALL:
*      RDDEC          ;;READ A DECIMAL NUMBER
*      RETURN HERE   ;;NUMBER IS ON TOP OF THE STACK

```

```

3753
3754 013452 011646          $RDDEC: MOV    (SP),-(SP)  ;;PROVIDE SPACE FOR
3755 013454 016666 000004 000002  MOV    4(SP),2(SP)    ;;THE INPUT NUMBER
3756 013462 010046          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
3757 013464 010146          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
3758 013466 010246          MOV    R2,-(SP)      ;;PUSH R2 ON STACK
3759 013470 104411          1$:  RDLIN          ;;READ AN ASCII LINE
3760 013472 012600          MOV    (SP)+,R0      ;;ADDRESS OF 1ST CHAR.
3761 013474 010067 000120  MOV    R0,6$         ;;SAVE INCASE OF BAD INPUT
3762 013500 005046          CLR    -(SP)         ;;CLEAR DATA WORD
3763 013502 005002          CLR    R2           ;;SIGN SET POSITIVE
3764 013504 122710 000055  CMPB   #'-(R0)       ;;SEE IF A MINUS SIGN WAS TYPED
3765 013510 001001          BNE    2$           ;;BR IF NO MINUS SIGN
3766 013512 112002          MOVB  (R0)+,R2      ;;SAVE FOR LATER USE
3767 013514 112001          2$:  MOVB  (R0)+,R1      ;;PICKUP THIS CHARACTER
3768 013516 001424          BEQ    3$           ;;GET OUT IF ZERO
3769 013520 122701 000060  CMPB   #'0,R1       ;;MAKE SURE THIS CHARACTER
3770 013524 003032          BGT    5$           ;;IS A DIGIT BETWEEN 0 & 9
3771 013526 122701 000071  CMPB   #'9,R1
3772 013532 002427          BLT    5$
3773 013534 032716 170000  BIT    #'C777,(SP)  ;;DON'T LET NUMBER GET TO BIG
3774 013540 001024          BNE    5$           ;;BR IF NUMBER WOULD OVERFLOW
3775 013542 006316          ASL    (SP)         ;;*2
3776 013544 011646          MOV    (SP),-(SP)  ;;SAVE FOR LATER
3777 013546 006316          ASL    (SP)         ;;*4
3778 013550 006316          ASL    (SP)         ;;*8
3779 013552 062616          ADD    (SP)+,(SP)  ;;*10
3780 013554 102416          BVS    5$           ;;OVERFLOW ISN'T ALLOWED
3781 013556 162701 000060  SUB    #'0,R1       ;;STRIP AWAY THE ASCII JUNK
3782 013562 060116          ADD    R1,(SP)     ;;ADD IN THIS DIGIT
3783 013564 102412          BVS    5$           ;;OVERFLOW ISN'T ALLOWED
3784 013566 000752          BR     2$           ;;LOOP
3785 013570 005702          3$:  TST    R2       ;;CHECK IF NUMBER IS NEG

```

```
3786 013572 001401          BEQ      4$          ;;BR IF NO
3787 013574 005416          NEG      (SP)        ;;YES--NEGATE THE NUMBER
3788 013576 012666 0000 2  4$:  MOV      (SP)+,12(SP) ;;SAVE THE RESULT
3789 013602 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
3790 013604 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
3791 013606 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
3792 013610 000002          RTI                    ;;RETURN
3793
3794 013612 005726          5$:  TST      (SP)+      ;;CLEAN PARTIAL NUMBER FROM STACK
3795 013614 105010          CLRB    (R0)        ;;SET A TERMINATOR
3796 013616 104401          TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
3797 013620 000000          6$:  .WORD    0          ;;POINTER GOES HERE
3798 013622 104401 001252  TYPE      ,SQUES      ;;'?' 'CR' & 'LF'
3799 013626 000720          BR      1$          ;;TRY AGAIN
```

.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.

```
3809 013630 010046          $TRAP: MOV     R0,-(SP)    ;;SAVE R0
3810 013632 016600 000002  MOV     2(SP),R0     ;;GET TRAP ADDRESS
3811 013636 005740          TST     -(R0)       ;;BACKUP BY 2
3812 013640 111000          MOVB    (R0),R0     ;;GET RIGHT BYTE OF TRAP
3813 013642 006300          ASL     R0          ;;POSITION FOR INDEXING
3814 013644 016000 013664  MOV     $TRPAD(R0),R0 ;;INDEX TO TABLE
3815 013650 000200          RTS     R0          ;;GO TO ROUTINE
```

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

```
3820 013652 011646          $TRAP2: MOV    (SP),-(SP) ;;MOVE THE PC DOWN
3821 013654 016666 000004 000002 MOV    4(SP),2(SP)    ;;MOVE THE PSW DOWN
3822 013662 000002          RTI                    ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

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

```
3829 : ROUTINE
3830 : -----
3831 013664 013652          $TRPAD: .WORD    $TRAP2
3832 013666 013232          $TYPE   ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
3833 013670 011200          $TYPOC  ;;CALL=TYPOC   TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3834 013672 011154          $TYPOS  ;;CALL=TYPOS   TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3835 013674 011214          $TYPON  ;;CALL= YPON    TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
3836 013676 011402          $TYPDS  ;;CALL=1/PDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
3837
3838 013700 012222          $GTSWR  ;;CALL=GTSWR   TRAP+6(104406) GET SOFT-SWR SETTING
3839
3840 013702 012132          $CKSWR  ;;CALL CKSWR   TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
3841 013704 012474          $RDCHR  ;;CALL-RDCHR  TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
```

3842 013706 012564 \$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
3843 013710 013072 \$RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3844 013712 013452 \$RDECC ;;CALL=RDECC TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY

3845
3846 .SBTTL POWER DOWN AND UP ROUTINES
3847

3848
3849 :*****
:POWER DOWN ROUTINE

3850 013714 012737 014060 000024 \$PWRDN: MOV #SILLUP,@PWRVEC ;;SET FOR FAST UP
3851 013722 012737 J00340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
3852 013730 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
3853 013732 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
3854 013734 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
3855 013736 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
3856 013740 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
3857 013742 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
3858 013744 017746 165170 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
3859 013750 010667 000110 MOV SP,\$SAVR6 ;;SAVE SP
3860 013754 012737 013766 000024 MOV #SPWRUP,@PWRVEC ;;SET UP VECTOR
3861 013762 000000 HALT
3862 013764 000776 BR -2 ;;HANG UP

3863
3864 :*****

3865 :POWER UP ROUTINE

3866 013766 012737 014060 000024 \$PWRUP: MOV #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
3867 013774 016706 000064 MOV \$SAVR6,SP ;;GET SP
3868 014000 005067 000060 CLR \$SAVR6 ;;WAIT LOOP FOR THE TTY
3869 014004 005267 000054 1\$: INC \$SAVR6 ;;WAIT FOR THE INC
3870 014010 001375 BNE 1\$;;OF WORD
3871 014012 012677 165122 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
3872 014016 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
3873 014020 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
3874 014022 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
3875 014024 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
3876 014026 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
3877 014030 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
3878 014032 012737 013714 000024 MOV #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
3879 014040 012737 000340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
3880 014046 104401 TYPE ;;REPORT THE POWER FAILURE
3881 014050 014046 \$PWRMG: .WORD \$POWER ;;POWER FAIL MESSAGE POINTER
3882 014052 012736 MOV (PC)+,(SP) ;;RESTART AT RESTR
3883 014054 002036 \$PWRAD: .WORD RESTR ;;RESTART ADDRESS
3884 014056 000002 RTI
3885 014060 000000 \$SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
3886 014062 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
3887 014064 000000 \$SAVR6: 0 ;;PUT THE SP HERE
3888 014066 005015 047520 042527 \$POWER: .ASCIZ <15><12>'POWER'
3889 014074 000122

3890 .EVEN
3891

3892 :*****
:TRANSMIT INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
3893 :*****

3894
3895
3896 014076 105777 165312 XINT: TSTB @DLXCSR ;'READY' SET ??
3897 014102 100416 BMI 1\$;BR IF YES

```

3898 014104 013767 177776 165070      MOV    @#PSW,$TMP0    ;SAVE THE ERROR PSW
3899 014112 010667 165060      MOV    SP,$REG6      ;SAVE THE ERROR STACK POINTER
3900 014116 005167 165300      COM    XFLGO         ;SET XMIT SOFTWARE ERROR FLAG
3901 014122 042777 000100 165260      BIC    #100,@DLRCSR  ;TURN OFF THE INTERRUPT ENABLES
3902 014130 042777 000100 165256      BIC    #100,@DLXCSR
3903 014136 000411              BR     2$            ;GO TO EXIT
3904 014140 022767 022440 165264 1$:    CMP    #DLBUFI,OPTR  ;XMITTED 256. BYTES YET ??
3905 014146 001405              BEQ    2$            ;BR IF YES
3906 014150 117777 165256 165240      MOVB  @OPTR,@DLXDBR  ;OUTPUT A BYTE
3907 014156 005267 165250      INC   OPTR           ;UPDATE BUFFER POINTER
3908 014162 000002 2$:    RTI                ;RETURN TO MAINLINE TEST
  
```

```

3909
3910
3911
3912
3913
  
```

```

:*****
:RECEIVER INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
:*****
  
```

```

3914 014164 105777 165220      RINT:  TSTB   @DLRCSR ;'DONE' SET ??
3915 014170 100410              BMI    1$            ;BR IF YES
3916 014172 013767 177776 165002      MOV    @#PSW,$TMP0    ;SAVE THE ERROR PSW
3917 014200 010667 164772      MOV    SP,$REG6      ;SAVR THE ERROR STACK POINTER
3918 014204 005167 165214      COM    RFLGO         ;SET HARD RCVR ERROR FLAG
3919 014210 000415              BR     2$            ;GO EXIT
3920 014212 005777 165174 1$:    TST    @DLRDBR      ;ANY SOFT ERRORS ??
3921 014216 100021              BPL    3$            ;BR IF NOT
3922 014220 013767 177776 164754      MOV    @#PSW,$TMP0    ;SAVE THE ERROR PSW
3923 014226 010667 164744      MOV    SP,$REG6      ;SAVE THE ERROR STACK POINTER
3924 014232 017767 165154 164744      MOV    @DLRDBR,$TMP1  ;SAVE THE ERROR REGISTER IN TMP1
3925 014240 005167 165162      COM    RFLG1         ;SET THE SOFT ERROR FLAG
3926 014244 042777 000100 165142 2$:    BIC    #100,@DLXCSR  ;TURN OFF THE INTR. ENABLES
3927 014252 042777 000100 165130      BIC    #100,@DLRCSR
3928 014260 000411              BR     4$            ;GO TO EXIT
3929 014262 022767 023040 165144 3$:    CMP    #BUFEND,IPTR  ;RECEIVED 256. BYTES YET ??
3930 014270 001405              BEQ    4$            ;BR IF YES
3931 014272 117777 165114 165134      MOVB  @DLRDBR,@IPTP  ;INPUT A BYTE FROM THE DL11
3932 014300 005267 165130      INC   IPTP           ;UPDATE BUFFER POINTER
3933 014304 000002 4$:    RTI                ;RETURN TO MAINLINE TEST
  
```

```

;THE FOLLOWING ROUTINE IS USED BY THE USER UTILITY PROGRAMS TO WAIT
;A SPECIFIED NO. OF MILLISECONDS BETWEEN CHARACTER TRANSFERS
  
```

```

3934
3935
3936
3937
3938 014306 017667 000000 000034 DELAY: MOV    @ (R6),DELCNT ;GET THE NO. OF MSEC. DELAY COUNT
3939
3940 014314 062716 000002      ADD    #2,(R6)       ;TYPED IN BY USER
3941 014320 005767 000024      TST   DELCNT        ;SET UP THIS ROUTINE'S EXIT ADDRESS
3942 014324 001410              BEQ    3$            ;IS THE DELAY COUNT ZERO?
3943 014326 012746 000226 1$:    MOV    #226,-(SP)    ;BRANCH IF YES
3944 014332 005316 2$:    DEC    (SP)         ;PUSH A 1 MSEC. COUNT TO STACK
3945 014334 001376              BNE    2$            ;DECREMENT THE 1 MSEC. COUNT BY 1
3946
3947 014336 005726              TST   (SP)+         ;BRANCH IF 1 MSEC. NOT EATEN
3948 014340 005367 000004      DEC    DELCNT        ;AWAY YET
3949
3950 014344 001370              BNE    1$            ;RESET STACK AFTER 1 MSEC. TIME UP
3951
3952 014346 000207 3$:    RTS    PC           ;DECREMENT THE TOTAL NO. OF
3953 014350 000000 DELCNT: .WORD 0        ;MSECS. COUNT
  
```

```

3954                                     ;TRANSPIRE RESIDES HERE
3955                                     ;THE FOLLOWING ROUTINE IS USED BY USER PROGRAM #4 AND WILL ALLOW
3956                                     ;A RANDOM NUMBER OF MILLISECONDS BEFORE TRANSMISSION OF CHARACTER
3957
3958 014352 016700 000062      STALL: MOV     NUMONE,RO      ;GET THE LOW LIMIT
3959 014356 006100              ROL     RO                ;MULTIPLY BY 4
3960 014360 006100              ROL     RO                ;
3961 014362 066700 000054      ADD     NUMTWO,RO         ;ADD IN THE HIGH LIMIT
3962 014366 010067 000046      MOV     RO,NUMONE        ;STORE THIS AS NEW LOW LIMIT
3963 014372 006100              ROL     RO                ;MULTIPLY NEW LOW LIMIT BY 4
3964 014374 006100              ROL     RO                ;
3965 014376 066700 000040      ADD     NUMTWO,RO         ;ADD IN THE HIGH LIMIT
3966 014402 006100              ROL     RO                ;MULTIPLY BY 4 AGAIN
3967 014404 006100              ROL     RO                ;
3968 014406 010067 000030      MOV     RO,NUMTWO        ;STORE THIS AS NEW HIGH LIMIT
3969 014412 016700 000022      MOV     NUMONE,RO        ;SAVE THE RANDOMLY GENERATED NO.
3970 014416 046700 164664      BIC     ST,MSK,RO        ;STRIP ALL BUT 1ST 5 BITS SO AS
3971                                     ;NOT TO ALLOW THE STALL TO BE TOO
3972                                     ;LARGE
3973 014422 001405              BEQ     2$,              ;BRANCH IF RESULT WAS ZERO
3974 014424 010067 000004      MOV     RO,1$            ;SET STALL TIME FOR DELAY ROUTINE
3975 014430 004767 177652      JSR     PC,DELAY         ;GO OFF TO STALL
3976 014434 000000              1$:   .WORD 0            ;THIS IS WHERE STALL TIME RESIDES
3977 014436 000207              2$:   RTS     PC         ;RETURN TO ISSUE CHARACTER
3978 014440 001233              NUMONE: 1233            ;LOW LIMIT FOR RANDOM NO.
3979 014442 007622              NUMTWO: 7622           ;HIGH LIMIT FOR RANDOM NO.
3980                                     ;THE FOLLOWING ROUTINE CHECKS THE 'DONE' BIT FOR BOTH THE RECEIVER
3981                                     ;AND TRANSMITTER. THIS ROUTINE IS USED BY PROGRAM #4
3982
3983 014444 016767 164540 000044  TIMERX: MOV     $TMP3,DUT      ;GET THE TRANSMITTER CONTROL
3984                                     ;STATUS REGISTER ADDRESS
3985 014452 162767 000004 000036      SUB     #4,DUT           ;FORM THE RECEIVER CONTROL
3986                                     ;STATUS REGISTER ADDRESS
3987 014460 000403              BR      TCONT           ;GO TO TIME OUT THE RECEIVERS'
3988                                     ;DONE BIT
3989 014462 016767 164522 000026  TIMETX: MOV     $TMP3,DUT      ;GET THE TRANSMITTER CONTROL
3990                                     ;STATUS REGISTER ADDRESS
3991 014470 005067 164522      TCONT: CLR     $TMP6      ;INITIALIZE A TIME COUNT
3992 014474 005267 164516      1$:   INC     $TMP6      ;INCREMENT THE TIME COUNT
3993 014500 001405              BEQ     2$,              ;BRANCH IF TIME COUNTER OVERFLOWED
3994                                     ;INDICATING DONE BIT NEVER SET
3995                                     ;WITH PLENTY OF TIME ELAPSED
3996 014502 105777 000010              TSTB   @DUT            ;SEE IF DONE BIT IS SET YET
3997 014506 100372              BPL     1$,              ;WAIT SOME MORE IF IT ISN'T
3998 014510 062716 000006      ADD     #6,@R6          ;DONE BIT IS SET - SET UP EXIT
3999                                     ;RETURN TO SKIP ERROR REPORT
4000                                     ;RETURN TO PROGRAM #4
4001 014516 000000              2$:   RTS     PC         ;THIS IS WHERE THE RCSR OR XCSR
4002                                     ;ADDRESS RESIDES
4003                                     ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5, AND WILL CHECK FOR CORRECT
4004                                     ;EXPECTED AND RECEIVED DATA, IN ADDITION TO ANY ERROR BITS
4005
4006 014520 016767 164470 164470  DATCHK: MOV     $TMP5,$TMP6    ;GET THE CONTENTS OF THE RECEIVER
4007                                     ;BUFFER
4008 014526 016767 164460 164432      MOV     $TMP4,$REG2      ;STORE THE ADDRESS OF THE RECEIVER
4009                                     ;DATA BUFFER

```



```

4010 014534 016767 164426 164422      MOV      $REG2,$REG1      ;GET THE ADDRESS OF THE RECEIVER
4011                                     ;DATA BUFFER
4012 014542 162767 000002 164414      SUB      #2,$REG1        ;FORM THE ADDRESS OF THE RECEIVER
4013                                     ;STATUS REGISTER FROM IT
4014 014550 016767 164440 164412      MOV      $TMP5,$REG3     ;STORE THE CONTENTS OF THE RECEIVER
4015                                     ;DATA BUFFER
4016 014556 032767 170000 164432      BIT      #170000,$TMP6   ;ARE ANY ERROR BITS SET?
4017 014564 001013                       BNE     '$              ;BRANCH IF YES
4018 014566 004767 000720                       JSR     PC,UPMASK       ;GO TO MASK OFF BITS AS A FUNCTION OF
4019                                     ;CHARACTER LENGTH( 5, 6, 7, OR 8 BITS)
4020 014572 026767 164416 164432      CMP      $TMP5,$TMP14    ;WAS RECEIVED CHARACTER THE
4021                                     ;SAME AS THE ONE TRANSMITTED?
4022 014600 001406                       BEQ     2$              ;BRANCH IF YES
4023 014602 016767 164424 164362      MOV      $TMP14,$REG4    ;STORE WHAT THE CONTENTS OF THE
4024                                     ;RECEIVER DATA BUFFER SHOULD BE
4025 014610 104010                       ERROR   +10            ;DATA RECEIVED WRONG.
4026 014612 000401                       BR      2$              ;GET SET TO RETURN AFTER ERROR REPORT
4027 014614 104007                       1$:    ERROR   +7      ;ERROR BIT/S SET FROM TRANSMISSION
4028 014616 000207                       2$:    RTS      PC      ;RETURN TO PROGRAM #4
4029
4030                                     ;:*****
4031                                     ;:SUBROUTINE TO SETUP ERROR INFORMATION FOR ERROR MESSAGES
4032                                     ;:*****
4033
4034

```

```

4035 014620 013767 177776 164354      SUER2:  MOV      @#PSW,$TMP0    ;SAVE THE [PSW]
4036 014626 016701 164556                       MOV      DLRCR,R1        ;PUT DEVADR IN R1
4037 014632 011203                       MOV      (R2),R3        ;PUT WAS INFO IN .3
4038 014634 010667 164336                       MOV      SP,$REG6       ;SAVE THE [SP]
4039 014640 062767 000002 164330      SUERR1: ADD      #2,$REG6        ;CORRECT FOR CALLING JSR
4040 014646 116700 164230                       MOVNB   $STNM,R0        ;PUT TEST NO. IN R0
4041 014652 010067 164304                       MOV      R0,$REG0       ;SAVE [R0] THRU [R4]
4042 014656 010167 164302                       MOV      R1,$REG1
4043 014662 010267 164300                       MOV      R2,$REG2
4044 014666 010367 164276                       MOV      R3,$REG3
4045 014672 010467 164274                       MOV      R4,$REG4
4046 014676 000207                       RTS      PC              ;RETURN TO CALLING TEST
4047
4048 014700 013767 177776 164274      SUERT1: MOV      @#PSW,$TMP0    ;SAVE THE [PSW]
4049 014706 116700 164170                       MOVNB   $STNM,R0        ;PUT TEST NO. IN R0
4050 014712 016701 164472                       MOV      DLRCR,R1        ;PUT DEVADR IN R1
4051 014716 010067 164240                       MOV      R0,$REG0       ;SAVE [R0]
4052 014722 010167 164236                       MOV      R1,$REG1       ;SAVE [R1]
4053 014726 013767 177776 164250      SUERT2: MOV      @#PSW,$TMP1    ;SAVE THE [PSW]
4054 014734 010667 164236                       MOV      SP,$REG6       ;SAVE THE [SP]
4055 014740 062767 000002 164230      ADD      #2,$REG6        ;CORRECT FOR CALLING JSR
4056 014746 010267 164214                       MOV      R2,$REG2       ;SAVE [R2]
4057 014752 000207                       RTS      PC              ;RETURN
4058
4059                                     ;SUBROUTINE TO SETUP VECTORS FOR 256. BYTE BLOCK TRANSFER TESTS
4060

```

```

4061 014754 016705 164440      SUVEC:  MOV      DLVECT,R5      ;GET FIRST VECTOR ADDRESS
4062 014760 012725 014164      MOV      #RINT,(R5)+      ;SET UP RCVR VECTOR
4063 014764 016725 164312      MOV      DLPRI,(R5)+
4064 014770 012725 014076      MOV      #XINT,(R5)+      ;SET UP XMIT VECTOR
4065 014774 016715 164302      MOV      DLPRI,(R5)

```

```

4066 015000 000207          RTS      PC          ;RETURN TO CALLER
4067
4068          ;SUBROUTINE TO PRIME DATA BUFFERS AND DEVICE FOR 256. BYTE TRANSFER
4069
4070 015002 005077 164406    PRIME:  CLR      @DLXCSR      ;CLEAR XMIT AND RCVR CSR'S
4071 015006 005077 164376    CLR      @DLRCSR
4072 015012 005067 164404    CLR      XFLG0      ;INITIALIZE ERROR FLAGS
4073 015016 005067 164402    CLR      RFLG0
4074 015022 005067 164400    CLR      RFLG1
4075 015026 012767 022040 164376    MOV      #DLBUFO,OPTR ;SET UP OUTPUT POINTER
4076 015034 012767 022440 164372    MOV      #DLBUFI,IPTR ;SET UP INPUT POINTER
4077 015042 004767 000044    JSR      PC,CLDLBF   ;GO CLEAR THE BUFFERS
4078 015046 004777 164364    JSR      PC,@LDOUT   ;GO SET UP THE PATTERN
4079 015052 005067 164362    CLR      TIMR1      ;INIT TIMEOUT COUNTERS
4080 015056 012767 000036 164356    MOV      #30,TIMR2
4081 015064 005777 164322    TST      @DLRDBR    ;FLUSH 'DONE' BIT IN RCVR CSR
4082 015070 005777 164316    TST      @DLRDBR
4083 015074 052777 000100 164306    BIS      #100,@DLRCSR ;ENABLE RCVR INTR.
4084 015102 052777 000104 164304    BIS      #104,@DLXCSR ;ENABLE XMIT INTR. AND MAINT MODE
4085 015110 000207          RTS      PC
4086
4087
4088
4089
4090          ;THIS ROUTINE IS CALLED TO CLEAR THE INPUT AND OUTPUT BUFFERS
4091
4092 015112 012705 022040    CLDLBF: MOV      #DLBUFO,R5 ;R5 POINTS TO BEGINNING OF BUFFER AREA
4093 015116 005025          1$:      CLR      (R5)+      ;CLEAR A WORD
4094 015120 022705 023040    CMP      #BUFEND,R5    ;DONE ALL WORDS ??
4095 015124 001374          BNE      1$            ;BR IF NOT
4096 015126 000207          RTS      PC          ;RETURN TO CALLER
4097
4098          ;THIS ROUTINE IS CALLED TO SET UP THE NULL-DEL-NULL PATTERN
4099
4100 015130 012705 022040    LDOUT1: MOV      #DLBUFO,R5 ;R5 POINTS TO OUTPUT BUFFER
4101 015134 105025          1$:      CLRB      (R5)+      ;MOVE A NULL CHAR
4102 015136 112725 000377    MOV      #377,(R5)+    ;MOV A DEL CHAR
4103 015142 022705 022440    CMP      #DLBUFI,R5    ;ALL DONE ??
4104 015146 001372          BNE      1$            ;BR IF NOT
4105 015150 000207          RTS      PC          ;RETURN TO CALLER
4106
4107          ;THIS ROUTINE IS USED TO LOAD AN ASCENDING BINARY COUNT PATTERN
4108
4109 015152 005005          LDOUT2: CLR      R5        ;START WITH 000
4110 015154 110565 022040    1$:      MOV      R5,DLBUFO(R5) ;LOAD ONE BYTE
4111 015160 005205          INC      R5          ;INCREMENT BYTE
4112 015162 022705 000400    CMP      #400,R5      ;DONE 000 THRU 377 ??
4113 015166 001372          BNE      1$            ;BR IF NOT
4114 015170 000207          RTS      PC          ;RETURN TO CALLER
4115
4116          ;THIS ROUTINE IS USED TO LOAD A DESCENDING BINARY COUNT PATTERN
4117
4118 015172 112767 000377 164020 LDOUT3: MOV      #377,$TMP7 ;START WITH A 377 BYTE
4119 015200 012705 022040    MOV      #DLBUFO,R5    ;R5 POINTS TO OUTPUT BUFFER
4120 015204 116725 164010    1$:      MOV      $TMP7,(R5)+ ;LOAD ONE BYTE
4121 015210 022705 022440    CMP      #DLBUFI,R5    ;ALL DONE ??

```

```

4122 015214 001403          BEQ      2$          ;BR IF YES
4123 015216 105367 163776  DECB    $TMP7      ;GENERATE NEXT BYTE
4124 015222 000770          BR       1$          ;GO MOVE IT
4125 015224 000207          2$:     RTS        PC ;RETURN TO CALLER
4126
4127
4128          ;THIS ROUTINE LOADS A COMPLEMENTING WORST CASE PATTERN
4129
4130 015226 012705 022040  LDOUT4: MOV      #DLBUFO,R5 ;R5 POINTS TO OUTPUT BUFFER
4131 015232 005067 163762  CLR      $TMP7      ;INIT. BYTE GENERATOR
4132 015236 116725 163756  1$:     MOVVB   $TMP7,(R5)+ ;MOVE A BYTE
4133 015242 105167 163752  COMB    $TMP7      ;COMPLEMENT IT
4134 015246 116725 163746  MOVVB   $TMP7,(R5)+ ;NOW LOAD THE 1'S COMPLEMENT
4135 015252 105267 163743  INCB    $TMP7+1    ;INCREMENT THE BYTE
4136 015256 116767 163737 163734  MOVVB   $TMP7+1,$TMP7 ;SET UP TO LOAD NEXT TWO
4137 015264 022705 022440  CMP     #DLBUF1,R5 ;ALL DONE ??
4138 015270 001362          BNE     1$          ;BR IF NOT
4139 015272 000207          RTS     PC          ;RETURN TO CALLER
4140
4141          ;THIS ROUTINE CHECKS FOR DATA COMPARE ERRORS IN 256. BYTE BLOCK TRANSFERS
4142
4143 015274 042777 000104 164112  CHKDAT: BIC     #104,@DLXCSR ;DISABLE BOTH XMIT AND RCVR INTR. ENAB.
4144 015302 042777 000100 164100  BIC     #100,@DLRCSR
4145 015310 012702 022040  MOV     #DLBUFO,R2 ;R2 POINTS TO S/B DATA IN OUTPUT BUFFER
4146 015314 004767 000070  JSR     PC,MASKING ;GO TO MASK OFF BITS AS A FUNCTION OF
4147          ;CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
4148 015320 012701 022440  MOV     #DLBUF1,R1 ;R1 POINTS TO WAS DATA IN RCVR. BUFFER
4149 015324 122221  CMPB   (R2)+,(R1)+ ;DID S/B = WAS ??
4150 015326 001004  BNE     3$          ;BR IF NOT
4151 015330 022701 023040  2$:     CMP     #BUFEND,R1 ;CHECKED ALL BYTES ??
4152 015334 001373  BNE     1$          ;BR IF NOT
4153 015336 000207  RTS     PC          ;RETURN TO CALLER
4154 015340 013767 177776 163634  3$:     MOV     @#PSW,$TMP0 ;SAVE THE [PSW]
4155 015346 010667 163624  MOV     SP,$REG6   ;SAVE THE [SP]
4156 015352 114204  MOVVB  -(R2),R4    ;GET THE S/B DATA
4157 015354 042704 177400  BIC     #177400,R4 ;CLEAR JUNK FROM HI BYTE
4158 015360 114103  MOVVB  -(R1),R3    ;GET THE WAS DATA
4159 015362 042703 177400  BIC     #177400,R3 ;CLEAR JUNK FROM HI BYTE
4160 015366 004767 177254  JSR     PC,SUERR1  ;GO SET UP ERROR INFO.
4161 015372 012767 015402 163644  MOV     #4$,$ESCAPE ;RETURN TO 4$ AFTER ERROR PRINT
4162 015400 104003  ERROR+3 ;DATA COMPARE ERROR
4163 015402 005202  4$:     INC     R2      ;REPOSITION BUFFER POINTERS
4164 015404 005201  INC     R1
4165 015406 000750  BR      2$          ;GO CHECK NEXT BYTE
4166
4167          ;THIS ROUTINE IS USED BY THE PATTERN TESTS
4168          ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE XMITTER
4169          ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS TRANSMITTED
4170          ;IS DONE. THE MASKING IS DONE AS A FUNCTION OF CHARACTER LENGTH WHICH
4171          ;CAN BE EITHER 5, 6, 7, OR 8 BITS .
4172
4173 015410 005205  MASKING: CLR     R5      ;INITIALIZE TABLE OFFSET
4174          ;FOR PICKING UP MASK WORD

```

```

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                                BNE      2$              ;BRANCH IF NO TO MASK NEXT BYTE
4193 015472 001374                      MOV      #DLBUFO,R2     ;RESTORE R2 BEFORE RETURNING
4194 015474 012702 022040      RTS      PC              ;RETURN TO MAINLINE CODE
4195                                3$:
4196 015502 000377      ;TABLE OF MASK WORDS
4197 015504 000177      (CHARL: .WORD 377      ;8. BITS IN LENGTH
4198 015506 000077      .WORD 177              ;7. BITS IN LENGTH
4199 015510 000037      .WORD 77               ;6. BITS IN LENGTH
                                .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    $STNM,R0          ;PUT TEST NO. IN R0
4246 015662 010067 163274      MOV     R0,$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     @#RESTRT         ;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  DH1:    .ASCIZ  '(PC) (PS) (SP) TEST DEVADR REGADR'
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                .EVEN
4277 016042 001116 001202 001176  DT1:    .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,0
4278 016050 001162 001164 001166
4279 016056 000000
4280
4281                ;INFORMATION FOR ERROR MESSAGE 2
4282
4283 016060 046104 030461 051040  EM2:    .ASCIZ  'DL11 REGISTER ERROR'
4284 016066 043505 051511 042524
4285 016074 020122 051105 047522
4286 016102 000122
  
```

```

CZDLCCO DL11-C,D,E OF LNE TST MACY11 30A(1052) 21-SEP-79 09:11 H 7 PAGE 86
CZDLCC.P11 21-SEP-79 09:07 POWER DOWN AND UP ROUTINES SEQ 0085

4287 016104 024040 041520 020051 DH2: .ASCIZ ' (PC) (PS) (SP) TEST DEVADR REGADR WAS S/B'
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 016202 .EVEN
4299 016202 001116 001202 001176 DT2: .WORD $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
4300 016210 001162 001164 001166
4301 016216 001170 001172 000000
4302
4303 ;INFORMATION FOR MESSAGE 3
4304
4305 016224 046104 030461 042040 EM3: .ASCIZ 'DL11 DATA COMPARE ERROR'
4306 016232 052101 020101 047503
4307 016240 050115 051101 020105
4308 016246 051105 047522 000122
4309 016254 024040 041520 020051 DH3: .ASCIZ ' (PC) (PS) (SP) TEST WASADR SHBADR WAS S/B'
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
4321 016352 001116 001202 001176 .EVEN
4322 016360 001162 001164 001166 DT3: .WORD $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
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
4340 016504 001200 001202 001176 .EVEN
4341 016512 001162 000000 DT4: .WORD $REG7,$TMP0,$REG6,$REG0,0
4342

```

4343

:ERROR INFORMATION FOR ERROR MESSAGE 5

4344

4345 016516 046104 030461 051440

EMS: .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
DT5: .WORD \$ERRPC,\$TMP0,\$REG6,\$REG0,\$REG1,\$REG2,\$REG3,0

4364 016662 001116 001202 001176

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
DT6: .WORD \$ERRPC,\$TMP1,\$REG6,\$REG2,0

4377 016742 001116 001204 001176

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
DT7: .WORD \$ERRPC,\$REG1,\$REG2,\$REG3,0

4389 017014 001116 001164 001166

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 | | | | | .EVFN |
| 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-NULL 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 | STMES: .ASCIZ <15><12>'CZDLCCO DL11-C,D,E OFLNE TST'<15><12> |
| 4443 | 017424 | 041514 | 030103 | 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 | PROG2M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 2'<15><12> |
| 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 PROG3M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 3'<15><12>
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 PROG4M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 4'<15><12>
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 PROG5M: .ASCIZ <15><12>'YOU HAVE SELECTED PROGRAM NO. 5'<15><12>
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 XDB: .ASCIZ <15><12>'TRANSMITTER DONE BIT NEVER SET PC= '
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 RDB: .ASCIZ <15><12>'RECEIVER DONE BIT NEVER SET PC= '
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 ;MESSAGES SEEKING USER RESPONSE
4488
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
  
```

CZDLCCO DL11-C,D,E OFLNE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11

30A(1052) 21-SEP-79 09:11 L 7
POWER DOWN AND UP ROUTINES PAGE 90

SEQ 0089

| | | | | | |
|------|--------|--------|--------|--------|---|
| 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 | .ASCIIZ <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: .ASCIIZ <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: .ASCIIZ <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 | SELDLY: .ASCIIZ <15><12>'WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G. 10=8(10))? ' |
| 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 | RSTALL: .ASCIIZ <15><12>'IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO? ' |
| 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 0464
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
4657 021674 044524 052516 020105
4658 021702 053523 052111 044103
4659 021710 005015 043111 04704SG:
4676 022036 000040
4677
4678

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'

.ASCII ' - PC'
.ASCIIZ ' '

.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      BUFBND: 0                ;TAG MARKS END OF BUFFERS
4689
4690          .END
```

CZDLCCO DL11-C D.E OFLNE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 25
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0093

| | | | | | | | | | | | | |
|---------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|------|
| ACTREG | 001274 | 1459# | 1827* | 1859* | 1860* | 1870* | 2974 | 2998 | | | | |
| BASEAD | 001264 | 1445# | 1779* | 1865* | 1866 | 1873* | 1881* | 2986* | 3002 | 3021* | 3023 | |
| BASEIV | 001270 | 1452# | 1799* | 2989* | 3004 | 3022* | 3024 | | | | | |
| BEGIN | 001452 | 1243 | 1588# | | | | | | | | | |
| BIT0 = | 000001 | 1345# | 1766 | 1789 | 1847 | 2301 | 2307 | 2320 | 2577 | 2661 | 2742 | 2847 |
| BIT00 = | 000001 | 1335# | 1345 | | | | | | | | | |
| BIT01 = | 000002 | 1334# | 1344 | | | | | | | | | |
| BIT02 = | 000004 | 1333# | 1343 | | | | | | | | | |
| BIT03 = | 000010 | 1332# | 1342 | | | | | | | | | |
| BIT04 = | 000020 | 1331# | 1341 | | | | | | | | | |
| BIT05 = | 000040 | 1330# | 1340 | | | | | | | | | |
| BIT06 = | 000100 | 1329# | 1339 | | | | | | | | | |
| BIT07 = | 000200 | 1328# | 1338 | | | | | | | | | |
| BIT08 = | 000400 | 1327# | 1337 | 3087 | | | | | | | | |
| BIT09 | 001000 | 1326# | 1336 | 3095 | 3156 | | | | | | | |
| BIT1 | 000002 | 1344# | 2212 | 2226 | 2271 | | | | | | | |
| BIT10 = | 002000 | 1325# | 3140 | | | | | | | | | |
| BIT11 | 004000 | 1324# | 3102 | | | | | | | | | |
| BIT12 = | 010000 | 1323# | | | | | | | | | | |
| BIT13 | 020000 | 1322# | 3147 | | | | | | | | | |
| BIT14 = | 040000 | 1321# | 3073 | | | | | | | | | |
| BIT15 = | 100000 | 1320# | 2146 | 2158 | 2177 | 2189 | 2191 | 2213 | 2225 | 2227 | 2285 | 2910 |
| BIT2 | 000004 | 1343# | 2018 | 2024 | 2091 | 2102 | 2115 | 2145 | 2157 | 2707 | 2805 | |
| BIT3 | 000010 | 1342# | 2176 | 2190 | | | | | | | | |
| BIT4 = | 000020 | 1341# | | | | | | | | | | |
| BIT5 | 000040 | 1340# | 2246 | 2252 | 2270 | | | | | | | |
| BIT6 = | 000100 | 1339# | 2067 | 2073 | 2090 | 2117 | | | | | | |
| BIT7 = | 000200 | 1338# | | | | | | | | | | |
| BIT8 | 000400 | 1337# | | | | | | | | | | |
| BIT9 = | 001000 | 1336# | | | | | | | | | | |
| BPTVEC | 000014 | 1352# | | | | | | | | | | |
| BUFEND | 023040 | 2342 | 2399 | 2456 | 2513 | 3929 | 4094 | 4151 | 4688# | | | |
| BUSERR | 015602 | 1692 | 4230# | | | | | | | | | |
| CHARL | 015502 | 4187 | 4196# | 4223 | | | | | | | | |
| CHKDAT | 015274 | 2344 | 2401 | 2458 | 2515 | 4143# | | | | | | |
| CKSWR = | 104407 | 3072 | 3136 | 3155 | 3840# | | | | | | | |
| CLDLBF | 015112 | 4077 | 4092# | | | | | | | | | |
| CONQUE | 002672 | 1832 | 1876 | 1888# | 1929 | | | | | | | |
| CR = | 000015 | 1260# | 3729 | 3739 | | | | | | | | |
| CRLF | 000200 | 1261# | 3700 | 3739 | | | | | | | | |
| DATCHK | 014520 | 2819 | 2923 | 4006# | | | | | | | | |
| DDISP = | 177570 | 1267# | 1389 | 1615 | | | | | | | | |
| DEFAULT | 020075 | 1737 | 4498# | | | | | | | | | |
| DELAY | 014306 | 2625 | 2712 | 3938# | 3975 | | | | | | | |
| DELCNT | 014350 | 3938* | 3941 | 3948* | 3953# | | | | | | | |
| DH1 | 015763 | 1510 | 4268# | | | | | | | | | |
| DH10 | 017026 | 1559 | 4394# | | | | | | | | | |
| DH2 | 016104 | 1517 | 4287# | | | | | | | | | |
| DH3 | 016254 | 1524 | 4309# | | | | | | | | | |
| DH4 | 016446 | 1531 | 4334# | | | | | | | | | |
| DH5 | 016573 | 1538 | 4353# | | | | | | | | | |
| DH6 | 016702 | 1545 | 4370# | | | | | | | | | |
| DH7 | 016754 | 1552 | 4382# | | | | | | | | | |
| DISPLA | 001142 | 1389# | 1615* | 1623* | 3116* | 3139* | | | | | | |
| DISPRE | 000174 | 1232# | 1623 | | | | | | | | | |
| DLADDR | 010000 | 1686 | 1777 | 2941# | 3006 | 3026 | | | | | | |

CZDLCCO DL11-C.D.E OFLNE TST
CZDLCC.P11 21-SEP-79 09:07

MACY11 30A(1052) 21-SEP-79 09:11 PAGE 105
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0102

| | | | |
|--------|-------|-------|------|
| .SETUP | 1# | 1199# | 1587 |
| .SWRHI | 1# | 1199# | 1275 |
| .SWRLO | 1199# | 1227# | 1228 |
| .SACT1 | 1# | | |
| .SAPT8 | 1# | | |
| .SAPTH | 1# | | |
| .SAPTY | 1# | | |
| .SASTA | 1# | | |
| .SCATC | 1# | 1199# | |
| .SCMTA | 1# | 1199# | 1361 |
| .SDB2D | 1# | | |
| .SDB2U | 1# | | |
| .SDIV | 1# | | |
| .SEOP | 1# | 1199# | 2957 |
| .SERRO | 1# | 1199# | 3121 |
| .SERRT | 1# | 1199# | 3165 |
| .SMULT | 1# | | |
| .SPOWE | 1# | 1199# | 3846 |
| .SRAND | 1# | | |
| .SRDDE | 1# | 1199# | 3740 |
| .SRDOC | 1# | 1199# | 3615 |
| .SREAD | 1# | 1199# | 3359 |
| .SR2AZ | 1# | | |
| .SSAVE | 1# | | |
| .SSB2D | 1# | | |
| .SSB2U | 1# | | |
| .SSCOP | 1# | 1199# | 3057 |
| .SSIZE | 1# | | |
| .SSUPR | 1# | | |
| .STRAP | 1# | 1199# | 3801 |
| .STYPB | 1# | | |
| .STYPD | 1# | 1199# | 3291 |
| .STYPE | 1# | 1199# | 3669 |
| .STYPO | 1# | 1199# | 3213 |
| .S4OCA | 1# | | |
| .1170 | 1# | | |

. ABS. 023042 000

ERRORS DETECTED: 0

CZDLCC.BIN,CZDLCC.LST/CRF/SOL/NL:TOC=CZDLCC.SML,CZDLCC.P11
RUN-TIME: 44 59 3 SECONDS
RUN-TIME RATIO: 383/107-3.5
CORE USED: 34K (67 PAGE)