

R1

1
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

.TITLE CNDMCAO DMV11 LINE UNIT DIAG1
.SBTTL PROGRAM DOCUMENT
.REM †

IDENTIFICATION

PRODUCT CODE: AC T830A MC
PRODUCT NAME: CNDMCAO DMV-11 LINE UNIT STATIC DIAGNOSTIC PART 1
PRODUCT DATE: APRIL 1984
MAINTAINER: ISS DIAGNOSTICS
AUTHORS: CHRIS BRIEMEN
 DAVE HOFFMAN
 RAY MARSHALL
MODIFIED BY: JAKI BERG 9-APR-1983
PURPOSE: THIS DIAGNOSTIC IS DESIGNED TO PERFORM STATIC LOGIC TESTS FOR
 THE M8053 OR M8064 (HEREAFTER REFERRED TO AS THE DMV OR DMV-11)

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL POP UNIBUS MASSBUS
DEC DECUS DECTAPE

PROGRAM DOCUMENT

50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

***** MODIFICATION HISTORY *****

REV A: ORIGINAL RELEASE	BRIENEN, HOFFMAN, MARSHALL	14 JAN-81
REV B: INSTALLED OUTSTANDING PATCHES		11 JUL-83
CVDMCB -> CNDMCA	IAKI BERG	9 APR 84

CHANGES WERE MADE TO CVDMCB TO PRODUCE CNDMCA FOR THE FALCON-PLUS PROJECT (SBC 11/21*). CHANGES, MARKED BY "JB REV A-0", ARE:
 - SET THE ODT BREAK VECTOR (LOCATION 140) TO THE STARTING ADDRESS OF FALCON'S ODT ROM (170000 OCTAL).



PROGRAM DOCUMENT

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

CONTENTS

1.0 INTRODUCTION

2.0 HARDWARE REQUIREMENTS

3.0 PRELIMINARY PROGRAM REQUIREMENTS

4.0 GENERAL PROGRAM CONSIDERATIONS

- 4.1 DIAGNOSTIC SUPERVISOR
- 4.2 EXECUTION TIME
- 4.3 XXOP.
- 4.4 ACT/SLIDE
- 4.5 APT
- 4.6 MEMORY MANAGEMENT
- 4.7 ERROR LOGGING

5.0 PROGRAM LOAD MEDIA

6.0 OPERATING INSTRUCTIONS

- 6.1 LOADING AND STARTING PROCEDURES
 - 6.1.1 LOADING PROCEDURES
 - 6.1.2 STARTING PROCEDURES
 - 6.1.3 ** STEPS FOR QUICK AND SIMPLE EXECUTION **
- 6.2 INITIAL DIALOGUE
- 6.3 PROGRAM OPTIONS
 - 6.3.1 START COMMAND
 - 6.3.2 RESTART COMMAND
 - 6.3.3 CONTINUE COMMAND
 - 6.3.4 PROCEED COMMAND
 - 6.3.5 ADD COMMAND
 - 6.3.6 DROP COMMAND
 - 6.3.7 PRINT COMMAND
 - 6.3.8 DISPLAY COMMAND
 - 6.3.9 FLAGS COMMAND
 - 6.3.10 ZFLAGS COMMAND
 - 6.3.11 CONTROL CHARACTERS
 - 6.3.12 HARDWARE PARAMETERS
 - 6.3.13 SOFTWARE PARAMETERS
 - 6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE

7.0 TEST DESCRIPTIONS

8.0 ERROR INFORMATION

- 8.1 ERROR REPORTING

PROGRAM DOCUMENT

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

1.0 INTRODUCTION

THE M8053 AND M8064 ARE SINGLE-LINE SYNCHRONOUS, MICRO-PROCESSOR BASED COMMUNICATIONS INTERFACES WHICH CAN SUPPORT BOTH CHARACTER-ORIENTED (DDCMP, BSC, ETC.) AND BIT-ORIENTED (SDLC, HDLC, ETC.) PROTOCOLS. THE PURPOSE OF THIS PROGRAM IS TO PERFORM BASIC DIAGNOSTIC TESTING OF THE VIF, FIFO, AND USYRT (BCP/BOP MODES) ON THESE BOARDS. THE FOLLOWING FUNCTIONS WILL BE PERFORMED: USYRT REGISTER ADDRESSING, USYRT REGISTER STATIC BIT INTERACTION AND READ/WRITE TESTING, AND BASIC BOP AND BCP TX TESTING (USING THE TSO STATUS BIT).

THE STATIC LOGIC TESTS WILL PROVIDE EXTENSIVE TROUBLESHOOTING CAPABILITIES, SUCH AS TIGHT SCOPE LOOPS, SWITCH OPTIONS, AND ABILITY TO "LOCK" ONTO INTERMITTENT ERRORS. IN ADDITION TESTS ARE DESIGNED AND STRUCTURED TO ACHIEVE MAXIMUM FAULT RESOLUTION AND FACILITATE REPLACEMENT OF THE SMALLEST FIELD REPLACEABLE UNIT.

THIS PROGRAM IS IMPLEMENTED USING THE DIAGNOSTIC SUPERVISOR AND A STRUCTURED PROGRAMMING APPROACH. BECAUSE THE DESIGN CONFORMS TO THE SUPERVISOR (STANDALONE VERSION) THE PROGRAM IS COMPATIBLE WITH ACT, APT, XXDP, AND SLIDE.

THROUGH DIALOGUE WITH THE OPERATOR, THE PROGRAM ALLOWS MODIFICATION OF DEVICE PARAMETERS, SUCH AS LSI-BUS ADDRESS, VECTOR ADDRESSES AND DEVICE PRIORITY. IN ADDITION, THE OPERATOR CAN SPECIFY PARTICULAR TESTS TO BE RUN AND A VARIETY OF LOOPING, RUNNING, AND REPORTING MODES.

DEVICE ERRORS WILL BE REPORTED AS THEY OCCUR. THE REPORT WILL INCLUDE A TEST NUMBER AND DESCRIPTION OF THE ERROR, GOOD AND BAD TEST DATA, AND APPLICABLE DEVICE REGISTER CONTENTS.

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE M8053/8064 STATIC LOGIC TESTS:

SBC-11/21+
16K WORDS OF MEMORY
CONSOLE TERMINAL
M8053 OR M8064 COMMUNICATIONS INTERFACE

3.0 PRELIMINARY PROGRAM REQUIREMENTS

THIS PROGRAM (CNDMC) SHOULD BE THE THIRD OF THE FIVE DMV-11 STATIC DIAGNOSTICS TO BE RUN (CNDMA/CNDMB SHOULD BE RUN FIRST). ERRORS FOUND IN THIS PROGRAM SHOULD BE CORRECTED BEFORE RUNNING ANY OF THE OTHER LINE UNIT DIAGNOSTICS (CNDMD OR CNDME).

PROGRAM DOCUMENT

170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 DIAGNOSTIC SUPERVISOR

THIS PROGRAM IS COMPATIBLE WITH THE STANDALONE DIAGNOSTIC SUPERVISOR, AND MUST BE LOADED TO BE CO-RESIDENT WITH THE SUPERVISOR, OR BE PREVIOUSLY COMBINED WITH THE SUPERVISOR AND LOADED AS A SINGLE FILE. IN EITHER CASE, THE COMBINED PROGRAM WILL NOT EXCEED 16K OF MEMORY.

4.2 EXECUTION TIME

THE MAXIMUM TIME REQUIRED TO RUN THIS PROGRAM IS ABOUT 15 SECONDS PER PASS FOR EACH UNIT.

4.3 XXDP.

THIS PROGRAM MAY BE LOADED UNDER XXDP., AND MAY BE RUN IN DUMP MODE OR CHAIN MODE.

4.4 ACT/SLIDE

THIS PROGRAM MAY BE LOADED UNDER ACT OR SLIDE AND MAY BE RUN IN DUMP MODE OR CHAIN MODE.

4.5 APT

THIS PROGRAM MAY BE LOADED BY THE APT SYSTEM (INCLUDING APT-RD) AND RUN IN PROGRAM MODE OR SCRIPT MODE.

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM.

4.7 ERROR LOGGING

AT THE END OF EACH PASS ON ALL UNITS, THE PROGRAM PRINTS OUT THE CUMULATIVE TOTAL NUMBER OF ERRORS SINCE THE LAST START OR RESTART COMMAND.

5.0 PROGRAM LOAD MEDIA

THIS PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR FROM ACT, SLIDE, OR APT SYSTEMS, OR FROM ANY MEDIA SUPPORTED BY XXDP.. WHEN USING THE PAPER TAPE ABSOLUTE LOADER, THE PROGRAM SHOULD BE LOADED FIRST, FOLLOWED BY THE DIAGNOSTIC SUPERVISOR. WHEN USING XXDP., THE DIAGNOSTIC SUPERVISOR SHOULD BE LOADED FIRST, FOLLOWED BY

PROGRAM DOCUMENT

227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283

THE DIAGNOSTIC PROGRAM.

6.0 OPERATING INSTRUCTIONS

6.1 LOADING AND STARTING PROCEDURES

6.1.1 LOADING PROCEDURES

THIS PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER. IT MAY ALSO BE LOADED FROM ANY XXDP* LOAD MEDIA. WHEN LOADED UNDER XXOP*, THE DIAGNOSTIC SUPERVISOR WILL BE LOADED AUTOMATICALLY.

6.1.2 STARTING PROCEDURES

THE PROGRAM STARTS AT LOCATION 200. USE STANDARD DEC PROCEDURES TO START THE PROGRAM.

6.1.3 STEPS FOR QUICK AND SIMPLE EXECUTION

THE DIAGNOSTIC CAN BE EXECUTED STANDALONE UNDER XXDP*, WITHOUT READING THE REMAINDER OF THIS DOCUMENT, AS FOLLOWS:

- A) LOAD AND START DIAGNOSTIC USING RUN COMMAND
- B) RECEIVE DIAGNOSTIC SUPERVISOR IDENTIFICATION AND PROMPT (DRS-C>)
- C) ENTER STA<CR>
- D) ANSWER HARDWARE AND SOFTWARE QUESTIONS
- E) GET END OF PASS MESSAGES OR ERROR MESSAGES
- F) TO END EXECUTION, ENTER CONTROL/C

6.2 INITIAL DIALOGUE

AFTER THE PROGRAM AND THE SUPERVISOR ARE LOADED AND THE PROGRAM IS STARTED, THE FOLLOWING IDENTIFICATION IS TYPED :

```
DRS LOADED  
DIAG. RUN-TIME SERVICES  
CNDMC-A-0  
DR: 11 LINE UNIT TESTS - PART 1 OF 3  
UNIT IS M8053 OR M8064  
DR>
```

THE OPERATOR THEN PROCEEDS BY TYPING ONE OR MORE OF THE COMMANDS DESCRIBED IN THE FOLLOWING SECTION 6.3. (FOR MORE DETAILED INFORMATION, REFER TO THE DIAGNOSTIC SUPERVISOR FUNCTIONAL SPECIFICATION).

6.3 PROGRAM OPTIONS

PROGRAM DOCUMENT

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
337
338
339
340

6.3.1 START COMMAND

```
*****
STAR(T)>TESTS:<TEST LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/EOP:<INCR>
*****
```

6.3.1.1 TESTS SWITCH (/TESTS:<TEST-LIST>)

<TEST-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (1:2 ETC.) OR RANGES OF DECIMAL NUMBERS (1-5:8 10 ETC.) THAT SPECIFY THE TESTS TO BE EXECUTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS RANGE FROM 1 TO THE LARGEST TEST NUMBER IN THE DIAGNOSTIC. THEY MAY BE SPECIFIED IN ANY ORDER. TESTS WILL BE EXECUTED IN NUMERICAL ORDER REGARDLESS OF THE ORDER OF SPECIFICATION. THE DEFAULT IS TO EXECUTE ALL TESTS. ON THIS AND ALL SWITCHES, THE ANGLE BRACKETS <> ARE PUNCTUATION USED IN THE DEFINITION ONLY, AND ARE NOT TO BE TYPED BY THE OPERATOR. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.2 PASS SWITCH (/PASS:<PASS-CNT>)

<PASS-CNT> IS A DECIMAL NUMBER INDICATING THE DESIRED NUMBER OF PASSES. A PASS IS DEFINED AS THE EXECUTION OF THE FULL DIAGNOSTIC (ALL SELECTED TESTS) AGAINST ALL UNITS SUBMITTED. THE DEFAULT IS NON-ENDING EXECUTION. IN THIS CASE EXIT FROM THE PROGRAM IS ACCOMPLISHED EITHER BY TYPING A CONTROL/C OR BY OCCURANCE OF AN ERROR WITH THE HALT ON ERROR FLAG BEING SET. THE EXIT IS A RETURN TO COMMAND MODE. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.3 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS A SEQUENCE OF ELEMENTS OF THE FORM <FLAG>, <FLAG=1>, OR <FLAG=0>, SEPARATED BY COLONS, WHERE <FLAG> HAS ONE OF THE FOLLOWING VALUES:

- MOE HALT ON ERROR, CAUSING COMMAND MODE TO BE ENTERED WHEN AN ERROR IS ENCOUNTERED
- LOE LOOP ON ERROR, CAUSING THE DIAGNOSTIC TO LOOP CONTINUOUSLY WITHIN THE SMALLEST DEFINED BLOCK OF CODING (SEGMENT, SUBTEST, OR TEST) CONTAINING THE ERROR
- IER INHIBIT ERROR REPORTING
- IBE INHIBIT BASIC ERROR REPORTS
- IXE INHIBIT EXTENDED ERROR REPORTS
- PRI DIRECT ALL MESSAGES TO A LINE PRINTER
- PNT PRINT NUMBER OF TEST BEING EXECUTED
- BOE BELL ON ERROR
- UAM RUN IN UNATTENDED MODE, BYPASSING MANUAL INTERVENTION TESTS
- ISR INHIBIT STATISTICAL REPORTS
- IDU INHIBIT DROPPING OF UNITS BY DIAGNOSTIC
- LOT LOOP ON TEST

PROGRAM DOCUMENT

341
 342 THE FLAGS NAMED OR EQUATED TO 1 ARE SET, THOSE EQUATED TO 0
 343 ARE CLEARED. A FLAG NOT SPECIFIED IS CLEARED. IF THE FLAGS
 344 SWITCH IS NOT GIVEN ALL FLAGS ARE CLEARED. SEE EXAMPLE AT
 345 END OF 6.3.1.5.
 346
 347
 348 6.3.1.4 END OF PASS SWITCH (/EOP:<INCR>)
 349
 350 <INCR> IS A DECIMAL NUMBER INDICATING HOW OFTEN (IN TERMS OF
 351 PASSES) IT IS DESIRED THAT THE END OF PASS MESSAGE BE
 352 PRINTED. THE DEFAULT IS AT THE END OF EVERY PASS. SEE
 353 EXAMPLE AT END OF 6.3.1.5.
 354
 355
 356 6.3.1.5 EFFECT OF START COMMAND
 357
 358 THE EFFECT OF THE START COMMAND IS TO INITIATE THE HARDWARE
 359 PARAMETER DIALOGUE, THE SOFTWARE PARAMETER DIALOGUE, AND
 360 THEN THE DIAGNOSTIC TESTS THEMSELVES.
 361
 362 THE HARDWARE PARAMETER DIALOGUE COMMENCES WITH THE QUESTION
 363 "N UNITS?" TO WHICH THE OPERATOR REPLIES WITH A DECIMAL
 364 NUMBER N FROM 1 TO 16. THE TERM "UNIT" REFERS TO THE DEVICE
 365 TO WHICH THIS SERIES OF DIAGNOSTICS IS DEDICATED. FOLLOWING
 366 THIS ARE THE QUESTIONS WHEREBY THE P-TABLES THEMSELVES WILL
 367 BE BUILT. EACH P-TABLE IS A CORE-RESIDENT TABLE CONTAINING
 368 ALL THE HARDWARE INFORMATION FOR ONE UNIT. THE OPERATOR
 369 MUST SUPPLY N (NUMBER OF UNITS) VALUES FOR EACH QUESTION.
 370 HE MAY DO THIS BY GIVING ONE ANSWER TO EACH QUESTION (IN
 371 WHICH CASE THE SERIES OF QUESTIONS WILL BE POSED N TIMES) OR
 372 BY GIVING N VALUES, SEPARATED BY COMMAS, TO EACH QUESTION
 373 (SERIES WILL BE POSED ONCE). EACH QUESTION IS FOLLOWED BY
 374 THE RESPONSE RADIX (D FOR DECIMAL, B FOR BINARY, O FOR
 375 OCTAL, L FOR YES/NO) IN PARENTHESES AND THE DEFAULT VALUE
 376 AFTER THE PARENTHESES.
 377
 378 FOLLOWING THE HARDWARE QUESTIONS ARE THE SOFTWARE QUESTIONS
 379 TO BUILD THE SOFTWARE TABLES, WHICH DEFINE THE MODE (QUICK
 380 VERIFY ETC.) THAT THE DIAGNOSTIC WILL EXECUTE IN.
 381
 382 WHEN THE QUESTION "N UNITS?" IS ANSWERED, MEMORY STORAGE IS
 383 ALLOCATED FOR THE P-TABLES, AND IF THERE IS NOT ENOUGH TO
 384 ACCOMMODATE THEM THE MESSAGE "TOO MANY UNITS" IS ISSUED. IN
 385 THIS CASE THE DIAGNOSTIC MUST BE EXECUTED MORE THAN ONCE TO
 386 TEST ALL UNITS.
 387
 388 EXAMPLE:
 389
 390 STA/TESTS:1:2-4:6:8-10/PASS:3/FLAGS:IER:HOE*1:UAM:LOE
 391
 392 THIS COMMAND WILL CAUSE THREE PASSES TO BE MADE, EACH PASS
 393 CONSISTING OF TESTS 1,2,3,4,6,8,9, AND 10 EXECUTED AGAINST
 394 ALL UNITS. THERE IS NO DIFFERENCE BETWEEN SAYING <FLAG> AND
 395 SAYING <FLAG=1>. THE NOTATION <FLAG=0> IS MEANINGFUL ONLY ON
 396 A COMMAND OTHER THAN START TO CLEAR A FLAG THAT WAS
 397 PREVIOUSLY SET. NOTE THAT ON ALL COMMANDS ONLY THE FIRST

PROGRAM DOCUMENT

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
449
450
451
452
453
454

THREE LETTERS ARE SCANNED.

6.3.2 RESTART COMMAND

RE(START)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/UNITS:<UNIT-LIST>

6.3.2.1 TESTS, PASS, AND FLAGS SWITCHES

<TEST-LIST>, <PASS-CNT>, AND <FLAG-LIST> ARE AS IN THE START COMMAND.

6.3.2.2 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (0,1 ETC.) OR RANGES OF DECIMAL NUMBERS (0-5, 8-10 ETC.) THAT SPECIFY THE UNITS TO BE TESTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS MAY RANGE FROM 0 THRU N-1 (N IS THE NUMBER OF UNITS SPECIFIED IN THE PREVIOUS START COMMAND). THE NUMBER INDICATES THE POSITION OF THE P-TABLE AS THE DATA WAS ENTERED DURING THE HARDWARE DIALOGUE. THE UNITS WHICH ARE SELECTED MUST NOT HAVE BEEN DROPPED BY THE DROP COMMAND. SEE THE DISCUSSION OF ADD AND DROP COMMANDS BELOW. DEFAULT IS TO TEST ALL UNITS WHICH HAVE NOT BEEN DROPPED BY A DROP COMMAND.

6.3.2.3 EFFECT OF RESTART COMMAND

THE RESTART COMMAND DIFFERS FROM THE START COMMAND IN THAT THE P-TABLES FROM THE PREVIOUS START COMMAND (THERE MUST HAVE BEEN ONE) ARE USED, INSTEAD OF NEW ONES BEING BUILT. THE UNITS SWITCH GIVES THE ABILITY TO SELECT A SUBSET OF THESE. THE SOFTWARE DIALOGUE MAY OPTIONALLY BE REEXECUTED (OPERATOR WILL BE ASKED). THE COMMAND CAN BE USED AFTER COMMAND MODE HAS BEEN REENTERED IN ANY OF THE THREE NORMAL WAYS: A) THE REQUESTED NUMBER OF PASSES HAVE BEEN MADE B) AN ERROR WAS ENCOUNTERED WITH THE HALT ON ERROR FLAG SET C) A CONTROL/C WAS ENTERED BY THE OPERATOR.

6.3.3 CONTINUE COMMAND

CON(TINUE)/PASS:<PASS-CNT>/FLAGS:<FLAG-LIST>

6.3.3.1 PASS SWITCH (/PASS:<PASS-CNT>)

<PASS-CNT> IS SAME AS IN START COMMAND, BUT THE DEFAULT IS THE UNSATISFIED PASS-CNT FROM THE PREVIOUS START OR RESTART.

PROGRAM DOCUMENT

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
505
506
507
508
509
510
511

IF NONE REMAINS, THE DEFAULT IS NON-ENDING EXECUTION.

6.3.3.2 FLAG SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS SAME AS IN START COMMAND, BUT UNSPECIFIED FLAGS RETAIN THEIR CURRENT VALUE.

6.3.3.3 EFFECT OF CONTINUE COMMAND

CONTINUE MUST FOLLOW A START OR RESTART, AND COMMAND MODE MUST HAVE BEEN ENTERED DUE TO A HALT ON ERROR OR A CONTROL/C. THE EFFECT OF THE COMMAND IS TO GO TO THE BEGINNING OF THE TEST THAT WAS BEING EXECUTED WHEN THE HALT OR CONTROL/C TOOK PLACE. SOFTWARE DIALOGUE MAY OPTIONALLY BE REEXECUTED. HARDWARE PARAMETERS MAY NOT BE CHANGED.

6.3.4 PROCEED COMMAND

PROCEED)/FLAGS:<FLAG-LIST>

6.3.4.1 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS AS IN THE START COMMAND, BUT UNSPECIFIED FLAGS RETAIN THEIR CURRENT VALUE.

6.3.4.2 EFFECT OF PROCEED COMMAND

PROCEED MUST FOLLOW A START, RESTART, OR CONTINUE. COMMAND MODE MUST HAVE BEEN ENTERED VIA A HALT ON ERROR. THE EFFECT OF THE COMMAND IS TO BEGIN EXECUTION AT THE LOCATION FOLLOWING THE ERROR CALL. NEITHER HARDWARE NOR SOFTWARE PARAMETERS MAY BE ALTERED.

6.3.5 ADD COMMAND

ADD/UNITS:<UNIT-LIST>

6.3.5.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.5.2 EFFECT OF ADD COMMAND

THE UNITS SPECIFIED ARE ADDED TO THE TEST SEQUENCE. EACH UNIT MUST HAVE A P-TABLE IN MEMORY DUE TO AN EARLIER

PROGRAM DOCUMENT

512 HARDWARE DIALOGUE. THIS COMMAND MUST BE FOLLOWED BY A
 513 RESTART OR CONTINUE. THE UNIT SWITCH MUST BE SPECIFIED.
 514 THE ADD COMMAND IS MEANINGFUL ONLY FOR UNITS THAT WERE
 515 PREVIOUSLY DROPPED.
 516
 517
 518 6.3.6 DROP COMMAND
 519
 520 *****
 521 DRO(P)/UNITS:<UNIT-LIST>
 522 *****
 523
 524
 525 6.3.6.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)
 526
 527 <UNIT-LIST> IS AS IN THE RESTART COMMAND.
 528
 529
 530 6.3.6.2 EFFECT OF DROP COMMAND
 531
 532 THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS
 533 WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START
 534 COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND
 535 MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.
 536
 537
 538 6.3.7 PRINT COMMAND
 539
 540 *****
 541 PRI(NT)
 542 *** *****
 543
 544
 545 6.3.7.1 EFFECT OF PRINT COMMAND
 546
 547 THE TOTAL NUMBER OF ERRORS FOR EACH UNIT SINCE THE LAST
 548 START OR RESTART COMMAND ARE PRINTED. THE ISR (INHIBIT
 549 STATISTICAL REPORTING) FLAG IS CLEARED.
 550
 551
 552 6.3.8 DISPLAY COMMAND
 553
 554 *****
 555 DIS(PLAY)/UNITS:<UNIT-LIST>
 556 *****
 557
 558
 559 6.3.8.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)
 560
 561 <UNIT-LIST> IS AS IN THE RESTART COMMAND.
 562
 563
 564 6.3.8.2 EFFECT OF DISPLAY COMMAND
 565
 566 THE HARDWARE P-TABLES FOR ALL UNITS UNDER TEST ARE PRINTED
 567 OUT IN THE FORMAT IN WHICH THEY WERE ENTERED. ANY UNITS
 568 THAT WERE DROPPED BY THE OPERATOR "DROP" COMMAND ARE SO

PROGRAM DOCUMENT

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
617
618
619
620
621
622
623
624
625

DESIGNATED.

6.3.9 FLAGS COMMAND

FLA(GS)

6.3.9.1 EFFECT OF FLAGS COMMAND

THE CURRENT SETTINGS OF ALL FLAGS ARE PRINTED.

6.3.10 ZFLAGS COMMAND

ZFL(AGS)

6.3.10.1 EFFECT OF ZFLAGS COMMAND

ALL FLAGS ARE CLEARED.

6.3.11 CONTROL CHARACTERS

A CONTROL C (C) ENTERED DURING THE EXECUTION OF A DIAGNOSTIC CAUSES A RETURN TO COMMAND MODE.

A CONTROL Z (Z) ENTERED DURING ONE OF THE THREE OPERATOR DIALOGUES- HARD CORE QUESTIONS (SEE 6.2), HARDWARE DIALOGUE (SEE 6.3.1.5), OR SOFTWARE DIALOGUE (SEE 6.3.1.5) CAUSES THE DEFAULTS TO BE TAKEN FOR THE REMAINDER OF THAT DIALOGUE.

A CONTROL O (O) ENTERED DURING THE EXECUTION OF A DIAGNOSTIC CAUSES ALL TELETYPE OUTPUT TO BE SUPPRESSED FOR THE REMAINDER OF THE DIAGNOSTIC OR UNTIL ANOTHER O IS TYPED, WHICH RESTORES NORMAL TELETYPE OUTPUT.

6.3.12 HARDWARE PARAMETERS

THE FOLLOWING 3 QUESTIONS WILL BE ASKED ON A START COMMAND. THE VALUE LOCATED TO THE LEFT OF THE QUESTION MARK IS THE DEFAULT VALUE THAT WILL BE TAKEN ON A CARRIAGE RETURN RESPONSE.

- 1. DEVICE CSR ADDRESS : (O) 160020?

THIS IS THE ADDRESS AT WHICH THE CSR REGISTERS (SELO) RESIDE ON THE LSI-BUS. THE ALLOWABLE RANGE IS 160020-177760 (OCTAL), AND THE DEFAULT VALUE IS 160020.

- 2. DEVICE VECTOR ADDRESS : (O) 300 ?

PROGRAM DOCUMENT

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
673
674
675
676
677
678
679
680
681
682

THIS IS THE ADDRESS OF THE INPUT INTERRUPT VECTOR FOR THIS DEVICE. THE ALLOWABLE RANGE IS 000-674 (OCTAL), AND THE DEFAULT VALUE IS 300.

3. DEVICE PRIORITY LEVEL : (0) 4 ?

THIS IS THE CPU PRIORITY AT WHICH THE INTERRUPT HANDLERS OF THIS DEVICE WILL BE EXECUTED. THE ALLOWABLE RANGE IS 0-7, AND THE DEFAULT VALUE IS 4.

4. SWITCH PACK # 1 (BOOT ADDRESS): (0) 0 ?

5. SWITCH PACK # 2 (DDCMF ADDRESS): (0) 0 ?

THESE REPRESENT THE TWO USER MODIFIABLE 8 POSITION DIP-SWITCHES. THE ALLOWABLE RANGES FOR BOTH ARE 000-377 (OCTAL), AND THE DEFAULTS ARE BOTH 0.

6. BOARD TYPE (0=M8064, 1=M8053-V35, 2=M8053-EIA) : (0) 0 ?

THIS IS THE TYPE OF DMV-11 CURRENTLY INSTALLED. NOTE THAT THE M8053 IS SWITCH SELECTABLE BETWEEN V.35 AND FIA.

7. BAUD RATE (0=LOW (19.2K), 1=HIGH (56K)):

THIS IS THE SPEED AT WHICH THE DMV TRANSMITS AND RECEIVES DATA. IN THE UNIT IS AN M8064, THE ANSWER IS IGNORED (DEFAULTS TO 56K). IF THE UNIT IS AN M8053, THEN THE ANSWER SHOULD BE BASED ON THE "SPEED SELECT SWITCH" LOCATED ON THE BOARD.

6.3.13 SOFTWARE PARAMETERS

NO SOFTWARE PARAMETER QUESTIONS ARE ASKED BY PART 1 OF THE STATIC LOGIC TESTS.

6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE

THE FULL CAPABILITY OF THE HARDWARE DIALOGUE IS REVEALED BY THE FOLLOWING DISCUSSION OF WHAT HAPPENS INTERNALLY.

AS SOON AS THE QUESTION "# UNITS?" IS ANSWERED (WITH THE NUMBER N, SAY) SPACE IN CORE IS ALLOCATED FOR N P-TABLES. ALL OF THE P-TABLES ARE OF THE SAME FORMAT, AND THERE IS A ONE-TO-ONE CORRESPONDENCE BETWEEN THE HARDWARE PARAMETER QUESTIONS AND THE SLOTS IN THE P-TABLE FORMAT.

ON THE FIRST TRIP THRU THE QUESTIONS, ALL OF THE SLOTS IN ALL OF THE P-TABLES ARE FILLED. IF THE OPERATOR TYPES IN LESS THAN N EXPLICIT VALUES IN RESPONSE TO A PARTICULAR QUESTION, THESE VALUES ARE PLACED IN THE P-TABLES (ONE VALUE GOING INTO THE PROPER SLOT OF EACH P-TABLE BEGINNING WITH THE FIRST P-TABLE) UNTIL THE STRING OF VALUES IS EXHAUSTED. THE LAST VALUE IN THE STRING BECOMES THE NEW DEFAULT AND IS USED TO FILL THAT SLOT IN THE REMAINING P-TABLES.

PROGRAM DOCUMENT

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
729
730
731
732
733
734
735
736
737
738
739

ON SUBSEQUENT TRIPS THRU THE QUESTIONS, THE SAME PROCESS IS CARRIED OUT, EXCEPT THAT THE EARLIEST P-TABLE NOT TO HAVE RECEIVED AN EXPLICIT VALUE IN ANY OF ITS SLOTS NOW ASSUMES THE ROLE THAT TABLE NUMBER ONE PLAYED IN THE FIRST TRIP.

THE SERIES OF QUESTIONS IS REISSUED UNTIL AT LEAST ONE QUESTION HAS RECEIVED N EXPLICIT VALUES FROM THE OPERATOR.

IN GIVING A STRING OF VALUES, COMMAS WITHOUT INTERVENING VALUES MAY BE USED TO INDICATE A REPETITION OF THE LAST NAMED VALUE.

A STRING OF VALUES MAY BE GIVEN AS A RANGE (6-10 FOR EXAMPLE). IF THE VALUES REPRESENT PURE NUMERICAL DATA, THIS SAMPLE RANGE TRANSLATES TO THE STRING 6,7,8,9,10 (AN INCREMENT OF 1). IF THE VALUES ARE ADDRESSES, THE SAMPLE RANGE TRANSLATES TO THE STRING 6,8,10 (AN INCREMENT OF 2).

NOW LET US SEE HOW WE COULD USE THESE CAPABILITIES TO CONSTRUCT A SET OF P-TABLES. ASSUME THAT WE HAVE 16 UNITS, AND THAT THERE ARE THREE HARDWARE PARAMETERS FOR EACH (THREE SLOTS IN THE P-TABLE, THREE HARDWARE QUESTIONS IN THE DIALOGUE). LET THE DESIRED VALUE FOR THE FIRST PARAMETER BE THE NUMBER 75 FOR ALL 16 TABLES. LET THE DESIRED VALUE FOR THE SECOND PARAMETER BE EQUAL TO THE UNIT NUMBER (0,1,2,...,15) EXCEPT FOR UNIT 12, WHICH SHOULD RECEIVE THE VALUE 11. LET THE DESIRED VALUE FOR THE THIRD PARAMETER BE THE NUMBER 76 FOR THE FIRST 7 UNITS AND THE NUMBER 77 FOR THE LAST 9 UNITS.

THE FOLLOWING DIALOGUE WOULD ACCOMPLISH THIS GOAL:

```

* UNITS (0) ? 16
UNIT 0
<QUESTION 1> ? 75
<QUESTION 2> ? 0-6
<QUESTION 3> ? 76

UNIT 7
<QUESTION 1> ?
<QUESTION 2> ? 7-11,,13-15
<QUESTION 3> ? 77
    
```

THE FIRST TIME THE SERIES IS ASKED, SLOT ONE RECEIVES A 75 IN ALL 16 TABLES. SLOT TWO RECEIVES THE VALUES 0,1,2,...,6 IN TABLES 0 THRU 6 AND A CONSTANT 6 IN TABLES 7 THRU 15. SLOT THREE RECEIVES A CONSTANT 76 IN ALL 16 TABLES.

THE SECOND TIME THRU THE SERIES, TABLES 7 THRU THE END ARE GOING TO BE AFFECTED (NOTE THAT THIS PIECE OF INFORMATION IS PRINTED OUT FOR THE THE OPERATOR IN THE FORM "UNIT XX" AT THE BEGINNING OF EACH SERIES). QUESTION 1 IS RESPONDED TO BY A <CR>, SO SLOT ONE STAYS AT CONSTANT 75 IN TABLES 7 THRU 15, SINCE NO NEW EXPLICIT VALUES ARE TYPED IN. SLOT TWO GETS THE VALUES 7,8,9,10,11 IN TABLES 7 THRU 11, AND GETS AN 11 IN SLOT 12, AND GETS THE VALUES 13,14,15 IN TABLES 13 THRU 15. SLOT THREE GETS THE VALUE 77 IN TABLES 7

PROGRAM DOCUMENT

740
741
742
743
744
745
746

THRU 15.

THE DIALOGUE IS TERMINATED WHEN THE SOFTWARE RECOGNIZES THAT
16 EXPLICIT VALUES HAVE BEEN GIVEN FOR AT LEAST ONE QUESTION
(NAMESLY QUESTION 2).

PROGRAM DOCUMENT

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
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804

7.0 TEST DESCRIPTIONS

```

;*****
;*      TEST 1 <TBMT MICROCODE INTERRUPT TEST>
;*
;*      THIS TEST CHECKS THE OPERATION OF THE TBMT (IRQ) INTERRUPT.
;*      THIS IS DONE BY ISSUING THE "SET MAINTENANCE INTERRUPT FLAG AND CLEAR
;*      INTERRUPT DISABLE IN PROCESSOR STATUS" COMMAND WHILE IN THE MAINTENANCE
;*      LOOP AND THEN CHECKING FOR BIT 7 OF BSEL3 TO BE SET (THE BIT IS SET
;*      BY THE MICROCODE WHEN THE TBMT INTERRUPT OCCURS).
;*****

;*****
;*      TEST 2 <SWITCH SETTING TEST>
;*
;*      SUBTEST #1:
;*      THE TWO READABLE SWITCH PACKS WILL BE SAMPLED AND COMPARED AGAINST THE 2
;*      VALUES IN THE P-TABLE. AN ERROR IS REPORTED ON A MISMATCH.
;*
;*      SUBTEST #2:
;*      THE SPEED SELECT SWITCH (SPDSEL) IS READ VIA THE VIAORA REGISTER (BIT PA4)
;*      AND COMPARED AGAINST THE BAUD RATE VALUE IN THE P-TABLE. IF A MISMATCH
;*      OCCURS IT WILL BE REPORTED. NOTE: THIS SUBROUTINE IS NOT RUN IF AN M8064
;*      BOARD IS BEING TESTED (IT ONLY RUNS 56K... MAKING A SPEED SWITCH USELESS).
;*
;*      THIS TEST IS ONLY RUN ON THE FIRST PASS AFTER A "START" OR "RESTART".
;*      ALL SUCCESSIVE PASSES WILL SKIP THIS TEST.
;*****

;*****
;*      TEST 3 <USYRT MASTER CLEAR TEST>
;*
;*      ALL REGISTERS ARE LOADED WITH PATTERN E IN THE SAME SEQUENCE AS FOR
;*      PATTERN F BELOW. THE USYRT IS THEN CLEARED BY A MASTER CLEAR
;*      (BIT 6 OF BSEL 1). ALL REGISTERS ARE THEN CHECKED FOR THE PROPER CONTENTS.
;*      THE INITIALIZED STATE OF THE REGISTERS IS CHECKED AGAINST DATA PATTERN F.
;*
;*      PATTERN E: 377, 377, 377, 377, 377, 377, 377, 366.
;*      PATTERN F: 000, 000, 000, 000, 000, 000, 000, 110.
;*
;*      SEQUENCE OF REGISTERS AS USED WITH PATTERNS E & F:
;*
;*      RDSRL, RDSRH, TDSRL, TDSRH, PCSARL, PCSARH, PCR, USYRT STATUS REG
;*****

;*****
;*      TEST 4 <USYRT PROGRAM RESET TEST>
;*
;*      ALL REGISTERS ARE LOADED WITH PATTERN E IN THE SAME SEQUENCE AS FOR
;*      PATTERN F BELOW. THE USYRT IS THEN RESET BY ASSERTING PROGRAM RESET
;*      (BIT 0 @ A000) IN THE 6522 VIA. ALL REGISTERS ARE THEN CHECKED FOR

```


PROGRAM DOCUMENT

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
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861

```

: * THE PROPER CONTENTS. THE INITIALIZED STATE OF THE REGISTERS IS CHECKED
: * AGAINST DATA PATTERN F.
: *
: * PATTERN E: 377, 377, 377, 377, 377, 377, 377, 366.
: * PATTERN F: 000, 000, 000, 000, 000, 000, 000, 110.
: *
: * SEQUENCE OF REGISTERS AS USED WITH PATTERNS E & F:
: *
: * RDSRL, RDSRH, TDSRL, TDSRH, PCSARL, PCSARH, PCR, USYRT STATUS REG
: *
: *****
: *****
: * TEST 5 <USYRT REGISTER ADDRESSING TEST>
: *
: * FIRST, A MASTER CLEAR IS ISSUED, TO INITIALIZE THE USYRT REGS TO
: * PATTERN F. THEN, EACH REGISTER IS WRITTEN WITH A BYTE OF PATTERN J,
: * AND AFTER EACH IS WRITTEN, ALL ARE READ AND COMPARED TO THE CURRENT
: * EXPECTED VALUES. THIS IS PERFORMED FOR ALL REGISTERS -- INCLUDING THE
: * READ ONLY REGS -- IN ORDER TO MAKE SURE THAT EACH REGISTER ONLY RESPONDS
: * TO ITS OWN ADDRESS.
: * PATTERN F: 000, 000, 000, 000, 000, 000, 000, 110
: * PATTERN J: 000, 000, 001, 002, 004, 020, 040, 010
: *
: * SEQUENCE OF REGISTERS AS USED WITH PATTERNS F & J:
: * RDSRL, RDSRH, TDSRL, TDSRH, PCSARL, PCSARH, PCR, USYRT STATUS REG
: *
: *****
: *****
: * TEST 6 <R/W BIT TEST OF PCSAR HIGH BYTE>
: *
: * WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN G.
: *
: * PATTERN G: 000, 001, 003, 004, 005, 007, 100, 101, 103, 104, 105,
: * 107, 000, 017, 027, 041, 200, 277, 103, 144, 115, 157, 000.
: *
: *****
: *****
: * TEST 7 <R/W BIT TEST OF S/AR REGISTER>
: *
: * WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN H.
: *
: * PATTERN H: 125, 252, 000, 377, 000, 001, 002, 004, 010, 020, 040, 100,
: * 200, 000, 377, 376, 375, 373, 367, 357, 337, 277, 177, 377,
: * 000
: *
: *****
: *****
: * TEST 8 <R/W BIT TEST OF PCR REGISTER>
: *
: * PATTERN I IS LOADED INTO PCR (HIGH) AND THE DATA READ BACK AND
: * CHECKED.
: *

```

PROGRAM DOCUMENT

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
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918

```

;* PATTERN I: 000, 041, 102, 143, 204, 245, 306, 347, 000, 001, 002,
;* 004, 040, 100, 200, 000, 346, 345, 343, 307, 247, 147, 347, 242,
;* 105, 347, 010, 020, 367, 357, 030, 027, 377.
;*****
;*****
;* TEST 9 <R/W BIT TEST OF TDSR REGISTER'S HIGH BYTE>
;*****
;* PATTERN K IS LOADED INTO TDSR (HIGH) AND THE DATA READ BACK IS
;* COMPARED AGAINST PATTERN L. (UNPREDICTABLE BITS ARE MASKED OFF TO 0
;* WHEN READING FOR COMPARISON.)
;*
;* PATTERN K: 000, 377, 376, 375, 373, 376, 177, 377, 000, 001, 002,
;* 004, 010, 200, 125, 252, 000.
;*
;* PATTERN L: 000, 017, 016, 015, 013, 016, 017, 017, 000, 001, 002,
;* 004, 010, 000, 005, 012, 000.
;*
;* NOTE THAT THE UNDEFINED BITS (12, 13, & 14) ARE MASKED OFF TO 0'S
;* FOR THE COMPARISON. ALSO THAT BIT 15 IS A READ/ONLY BIT AND CAN'T BE
;* SET -- THEREFORE SHOULD ALWAYS BE READ AS A 0 BY THIS TEST.
;*****
;*****
;* TEST 10 <R/W BIT TEST OF TXDB REGISTER>
;*****
;* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN H.
;*
;* PATTERN H: 000, 001, 002, 004, 010, 020, 040, 100, 200, 000, 377,
;* 376, 375, 373, 367, 357, 337, 277, 177, 377, 000
;*****
;*****
;* TEST 11 <PSEUDO R/W BIT TEST OF RXDB>
;*****
;* WRITE, READ (BUT NO COMPARE) OF EACH WORD IN DATA PATTERN H. THIS IS
;* PRIMARILY TO PROVIDE A SCOPE LOOP FUNCTION ON THIS REGISTER.
;*
;* PATTERN H: 000, 001, 002, 004, 010, 020, 040, 100, 200, 000, 377,
;* 376, 375, 373, 367, 357, 337, 277, 177, 377, 000
;*****
;*****
;* TEST 12 <PSEUDO R/W BIT TEST OF RDSR'S HIGH BYTE>
;*****
;* WRITE, READ (BUT NO COMPARE) OF EACH WORD IN DATA PATTERN H. THIS IS
;* PRIMARILY TO PROVIDE A SCOPE LOOP FUNCTION ON THIS REGISTER.
;*
;* PATTERN H: 000, 001, 002, 004, 010, 020, 040, 100, 200, 000, 377,
;* 376, 375, 373, 367, 357, 337, 277, 177, 377, 000
;*****

```

PROGRAM DOCUMENT

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
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975

```

*****
; * TEST 13 <NULL CLOCK TEST>
; *
; * FIRST, A MASTER CLEAR IS DONE TO INIT THE DMV. THEN, THE T1 TIMER ON THE
; * VIA CHIP IS PROGRAMMED FOR SQUARE WAVE CLOCK GENERATION ON PB7 (BIT 7
; * OF VIA OUTPUT REG B), WITH A BAUD RATE = 56 KBAUD. THIS IS THE MODE OF
; * VIA OPERATION WHICH IS USED TO GENERATE THE NULL CLOCK. THEN, THE PROGRAM
; * SCANS ORB REPEATEDLY TO MONITOR THE NULL CLOCK BIT, IN THE FOLLOWING
; * SEQUENCE :
; * - THE PROGRAM REPEATEDLY CHECKS THE NULL CLOCK BIT FOR THE 1 STATE, AND
; * IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MICRO-SEC (A GROSS TIMEOUT
; * INTERVAL), AN ERROR IS REPORTED. (AT 56 KBAUD, THE CLOCK SHOULD
; * HAVE A PERIOD OF ABOUT 18 MICRO-SEC.)
; * - THE PROGRAM NEXT REPEATEDLY CHECKS THE NULL CLOCK BIT FOR THE 0 STATE,
; * AND IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MICRO-SEC, AN ERROR IS
; * REPORTED.
; * - THE PROGRAM NEXT REPEATEDLY CHECKS THE NULL CLOCK BIT FOR THE 1 STATE
; * AGAIN, AND IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MICRO-SEC,
; * AN ERROR IS REPORTED.
*****

```

```

*****
; * TEST 14 <BCP TX RESET W/IDLE = 0>
; *
; * THE USYRT IS INITIALIZED FOR "BYTE-CONTROL PROTOCOL" (BCP) WITH IDLE
; * SET TO ZERO AND A 125 SYNC CHARACTER IS LOADED INTO S/AR. A 226 SYNC
; * CHARACTER IS LOADED INTO TXDB SO THAT THE SOURCE OF SYNC CHARACTERS
; * CAN BE LATER DETERMINED. THE VALID STATE OF THE USYRT REGISTERS IS
; * READ AND CHECKED. TXE IS ASSERTED TO ENABLE THE TRANSMITTER LOGIC.
; * THEN, TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE OBSERVING
; * TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
; * (TXBE SHOULD GO HIGH; AT THIS TIME THE S/AR'S SYNC CHARACTER SHOULD
; * BE LOADED INTO TXSO AND TSOM IS AGAIN SET -- DRIVING TXBE LOW.)
; * THREE SYNC CHARACTERS ARE SENT/RECEIVED: THE FIRST TWO SYNCHRONIZE
; * THE RECEIVER, THE THIRD IS DIRECTLY READ (STRIP SYNC IS OFF) AND
; * COMPARED AGAINST 125 (THE S/AR SYNC CHARACTER).
; * IF VALUE READ IS 226, THEN TXDB PROVIDED THE SYNC (IE: ERROR).
; * THE USYRT IS THEN RESET AND REGISTERS ARE AGAIN READ AND CHECKED.
; * THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY
; * ONE MARK AND THREE SYNC CHARACTERS (FROM THE S/AR) IS TRANSMITTED.
; * ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN THE
; * SEQUENCE.
*****

```

```

*****
; * TEST 15 <BCP TX RESET W/IDLE = 1>
; *
; * THE USYRT IS INITIALIZED FOR "BYTE-CONTROL PROTOCOL" (BCP) WITH IDLE
; * SET TO ONE AND A 226 SYNC CHARACTERS LOADED INTO S/AR AND TXDB.
; * THE VALID STATE OF THE USYRT REGISTERS IS READ AND CHECKED. TXE IS
; * ASSERTED TO ENABLE THE TRANSMITTER LOGIC.
; * THEN, TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE OBSERVING
; * TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
; * (TXBE SHOULD GO HIGH; AT THIS TIME THE TXDB SYNC CHARACTER SHOULD

```

PROGRAM DOCUMENT

```

976 ;* BE LOADED INTO TXSO AND TSOM IS AGAIN SET -- DRIVING TXBE LOW.)
977 ;* AFTER THE RECEIVER IS SYNCHRONIZED (TWO SYNC CHARACTERS), TXDB IS
978 ;* LOADED WITH A 125 AND, WITH TSOM STILL = 1 (SYNC SOURCE = TXDB), THE
979 ;* USYRT IS AGAIN CLOCKED.
980 ;* AT THIS POINT, IF THE IDLE BIT WORKED, THE VALUE 125 WILL BE READ
981 ;* BY THE RECEIVER. OTHERWISE A 226 WILL BE READ, INDICATING TXDB WASN'T
982 ;* PROVIDING THE SYNC CHARACTERS.
983 ;* WHEN TXBE GOES HIGH AGAIN, THE USYRT IS RESET.
984 ;* ALL REGISTERS ARE AGAIN READ AND CHECKED.
985 ;* THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY
986 ;* ONE MARK AND THREE SYNCs (226,226,125 FROM TXDB) ARE TRANSMITTED.
987 ;* ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN THE
988 ;* SEQUENCE.
989 ;*****
990
991
992 ;*****
993 ;* TEST 16 <BCP TX UNDERRUN W/TSOM TERMINATION>
994 ;*
995 ;* THE USYRT IS INITIALIZED FOR BCP WITH IDLE = 1 AND TXC IS MANUALLY
996 ;* CONTROLLED UNTIL TWO SYNC CHARACTERS AND ONE DATA CHARACTER (000)
997 ;* HAVE BEEN TRANSMITTED. AT THIS TIME WHEN TXBE IS ASSERTED BY THE
998 ;* USYRT, NO DATA IS LOADED INTO TXDB -- FORCING AN UNDER RUN
999 ;* CONDITION. TXU AND TERR ARE CHECKED BOTH BEFORE AND AFTER THEIR
1000 ;* EXPECTED ASSERTIONS. AFTER THE FIRST NON-DATA CHARACTER (WHICH
1001 ;* SHOULD BE THE MARK CHARACTER) HAS BEEN STARTED, IDLE IS SET TO 0.
1002 ;* THIS SHOULD FORCE THE NEXT NON-DATA CHARACTER TO BE A SYNC CHARACTER
1003 ;* FROM S/AR. WHILE THIS SYNC CHARACTER IS BEING TRANSMITTED, TSOM IS
1004 ;* ASSERTED (CLEARING TXU AND TERR) -- IDLE IS LEFT AT 0. TXBE IS THEN
1005 ;* CYCLED THROUGH AT LEAST ONE MORE SYNC CHARACTER AND THE TEST IS
1006 ;* ABORTED. ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS
1007 ;* WITHIN THE SEQUENCE.
1008 ;* NOTE: BITS SHIFT OUT OF TX LSB FIRST.
1009 ;*****
1010
1011
1012 ;*****
1013 ;* TEST 17 <BCP TX UNDERRUN W/RESET TERMINATION>
1014 ;*
1015 ;* THE USYRT IS INITIALIZED FOR BCP WITH IDLE = 1 AND TXC IS MANUALLY
1016 ;* CONTROLLED UNTIL TWO SYNC CHARACTERS AND ONE DATA CHARACTER HAVE
1017 ;* BEEN TRANSMITTED. AT THIS TIME WHEN TXBE IS ASSERTED BY THE USYRT,
1018 ;* NO DATA IS LOADED INTO TXDB -- FORCING AN UNDER RUN CONDITION. TXU
1019 ;* AND TERR ARE CHECKED BOTH BEFORE AND AFTER THEIR EXPECTED
1020 ;* ASSERTIONS. AFTER THE FIRST NON-DATA CHARACTER (WHICH SHOULD BE THE
1021 ;* MARK CHARACTER) HAS BEEN STARTED, IDLE IS SET TO 0. THIS SHOULD
1022 ;* FORCE THE NEXT NON-DATA CHARACTER TO BE A SYNC CHARACTER.
1023 ;* IMMEDIATELY AFTER THIS SYNC CHARACTER HAS BEING TRANSMITTED, A
1024 ;* PROGRAM RESET IS ISSUED AND ALL REGISTERS ARE CHECKED. ERROR
1025 ;* LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN THE
1026 ;* SEQUENCE.
1027 ;*****
1028
1029
1030 ;*****
1031 ;* TEST 18 <BCP TX DISABLE TEST>
1032 ;*

```

PROGRAM DOCUMENT

1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089

```

; * THE USYRT IS INITIALIZED FOR BCP AND A MESSAGE IS STARTED. ONCE THE
; * SECOND DATA CHARACTER IS LOADED INTO TXDB TXE IS DROPPED. TXSO IS
; * WATCHED TO ASSURE THAT THE CHARACTER BEING TRANSMITTED IS COMPLETED.
; * WHEN IT IS, THE USYRT SHOULD DROP TXA AND STOP TRANSMITTING -- THE
; * LAST CHARACTER LOADED INTO TXDB SHOULD BE LOST.
; *
; * CHARACTERS LOADED: 125 252
; * CHARACTERS TRANSMITTED: 125
; *****
; *****
; * TEST 19 <FIFO STACKING CHARACTERS TEST>
; *
; * THE USYRT IS SETUP FOR BCP MODE WITH NO ERROR DETECTION.
; * THIS TEST BEGINS BY SYNCHRONIZING THE RECEIVER AND THEN PROCEEDS
; * TO FILL THE 8 CHARACTER RECEIVER FIFO WITH THE CHARACTERS:
; * 1/2(SYNCH),000,377,125,252,347,030,303,1/2(074).
; * THESE CHARACTERS ARE THEN READ OFF OF THE FIFO AND CHECKED. NOTE
; * THAT NO CLOCKS ARE PROVIDED WHEN RECEIVING THE CHARACTERS SINCE THEY
; * ARE SUPPLIED BY THE FIFO SUPPORT LOGIC IN GROUPS OF 4 TICKS (WHEN
; * RDA = 0).
; * ALSO NOTE THAT DUE TO FIFO TIMING, TWO 'HALF CHARACTERS' ARE LOADED
; * INTO THE FIFO (THE 1ST AND LAST CHARACTERS).
; *
; *****
; *****
; * TEST 20 <BCP CHARACTER LENGTH TEST>
; *
; * THE USYRT IS INITIALIZED FOR BCP WITH NO ERROR CHECKING. TXC IS MANUALLY
; * CONTROLLED UNTIL TWO SYNC CHARACTERS HAVE BEEN TRANSMITTED. THEN 3
; * SUBTESTS FOLLOW, EACH ONE USING A DIFFERENT TRANSMIT CHARACTER LENGTH
; * STARTING AT FIVE (5) AND ENDING WITH SEVEN (7).
; *
; * TEST PATTERN: 111 222 333 044 155 266 377
; *****
; *****
; * TEST 21 <BOP TX TABORT/(IDLE = 0) TEST>
; *
; * THE USYRT IS INITIALIZED FOR "BIT-ORIENTED PROTOCOL" (BOP) WITH IDLE
; * SET TO ZERO. TXE AND TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE
; * OBSERVING TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
; * NEXT, TXBE SHOULD GO HIGH; AT THIS TIME AN ALL ZEROS CHARACTER WILL BE
; * LOADED INTO TXDB DRIVING TXBE LOW. THE TRANSMITTER IS CLOCKED THROUGH
; * ONE CHARACTER. WHEN TXBE GOES HIGH AGAIN, TABORT IS ASSERTED CAUSING
; * ABORT TO BE TRANSMITTED. ALL CHARACTERS ARE CHECKED AT TXSO.
; * THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY TWO
; * FLAGS, ONE ZERO CHARACTER, AND ONE ABORT CHARACTER IS SENT (INTO THE BIT
; * BUCKET). ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN
; * THE SEQUENCE.
; *****
; *****

```

PROGRAM DOCUMENT

1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146

```

*****
*      TEST 22 <BOP TX TABORT/(IDLE = 1) TEST>
*
* THE USYRT IS INITIALIZED FOR "BIT-ORIENTED PROTOCOL" (BOP) WITH IDLE
* SET TO ONE. TXE AND TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE
* OBSERVING TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
* NEXT, TXBE SHOULD GO HIGH; AT THIS TIME AN ALL ZEROS CHARACTER WILL BE
* LOADED INTO TXDB DRIVING TXBE LOW. THE TRANSMITTER IS CLOCKED THROUGH
* ONE CHARACTER. WHEN TXBE GOES HIGH AGAIN, TABORT IS ASSERTED CAUSING
* ABORT TO BE TRANSMITTED. ALL CHARACTERS ARE CHECKED AT TXSO.
* THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY TWO
* FLAGS, ONE ZERO CHARACTER, AND ONE FLAG CHARACTER IS SENT (INTO THE BIT
* BUCKET). ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN
* THE SEQUENCE.
*****
    
```

```

*****
*      TEST 23 <BOP TX TXGA (TRANSMIT GO-AHEAD) TEST>
*
* THE USYRT IS INITIALIZED FOR BOP AND TXE IS ASSERTED. TSOM IS ASSERTED
* AND TXA OBSERVED -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
* NEXT, TXBE SHOULD GO HIGH; AT THIS TIME AN ALL ZEROS CHARACTER WILL BE
* LOADED INTO TXDB DRIVING TXBE LOW. WHEN TXBE GOES HIGH AGAIN, TXGA
* IS ASSERTED CAUSING GA TO BE TRANSMITTED. THE SEQUENCE OF EVENTS IS
* CONTINUALLY MONITORED WHILE TXC IS MANUALLY CONTROLLED AND ALL
* CHARACTERS ARE CHECKED AT TXSO. THIS TEST WILL GO NO FURTHER INTO
* THE TRANSMIT SEQUENCE SO THAT ONLY TWO FLAGS, ONE ZERO CHARACTER
* AND ONE GA CHARACTER IS SENT (INTO THE BIT BUCKET).
* ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN
* THE SEQUENCE.
*****
    
```

```

*****
*      TEST 24 <BOP TX MESSAGE WITHOUT CRC>
*
* THE USYRT IS INITIALIZED FOR BOP MODE WITH NO ERROR DETECTION. TXC IS THEN
* MANUALLY CONTROLLED UNTIL TWO FLAG CHARACTERS HAVE BEEN TRANSMITTED. THEN
* A 5 CHARACTER MESSAGE IS TRANSMITTED, RECEIVED, CHECKED, AND TERMINATED
* (WITH TEOM). A CHECK IS MADE TO ASCERTAIN THAT NO CRC OR VRC IS GENERATED
* -- FLAG CHARACTERS SHOULD FOLLOW THE DATA.
*      (NOTE: NO BIT STUFFING OCCURS IN THIS TEST)
*
*      TEST MESSAGE: FLAG FLAG 000 307 125 252 201 FLAG
*****
    
```

```

*****
*      TEST 25 <BOP RX CHARACTER LENGTH TEST>
*
* THE USYRT IS INITIALIZED FOR BOP WITH CRC-CCITT PRESET TO 1'S. TXC
* IS MANUALLY CONTROLLED UNTIL TWO FLAG CHARACTERS HAVE BEEN
* TRANSMITTED. THEN 6 SUBTESTS FOLLOW, EACH ONE USING A DIFFERENT
* TRANSMIT CHARACTER LENGTH STARTING AT TWO (2) AND ENDING WITH SEVEN
* (?). IN EACH SUBTEST, TWO 8 BIT CHARACTERS WILL BE TRANSMITTED
* BEFORE TXCL IS CHANGED TO THE CHARACTER LENGTH BEING TESTED. THIS
    
```

PROGRAM DOCUMENT

1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203

```

; * CORRESPONDS TO NORMAL USAGE WHERE EITHER:
; *
; *   1 -- A MESSAGE OF CHARACTERS WHICH ARE LESS THEN 8 BITS IS
; *        SENT AS A STREAM OF 8 BIT CHARACTERS AND THE REMAINING
; *        BITS ARE SENT AS A CHARACTER OF LESS THEN 8 BITS OR
; *
; *   2 -- A HEADER OF TWO 8 BIT CHARACTERS IS SENT FOLLOWED BY A
; *        DATA STREAM OF DATA CHARACTERS WHICH MAY BE LESS THEN 8
; *        BITS IN LENGTH (I.E. 2, 3, 4, 5, 6, OR 7 BIT
; *        CHARACTERS).
; *
; *        THE TEST PATTERN IS: 123 321 111 222 333 044 155 266 377
; *****
; *****
; * TEST 26 <TX "SPACING SEQUENCE">
; *
; * THE TRANSMITTER IS INITIALIZED AND THE "SPACING SEQUENCE" IS FORCED
; * BY ASSERTING BOTH TSOM & TEOM AT THE SAME TIME -- CHECK THE BIT
; * STREAM FOR ACCURACY (SPACES) AND COMPLETNESS (16 OF THEM). WHEN TXBC
; * GOES HIGH (= 1) A SMALL MESSAGE IS SENT.
; *****
; *****
; * TEST 27 <FIFO OVERRUN INTEGRITY TEST>
; *
; * THIS TEST BEGINS BY SYNCHRONIZING THE RECEIVER AND THEN PROCEEDS TO FILL
; * THE 8 CHARACTER RECEIVER FIFO UNTIL RXOR WITH THE CHARACTERS:
; * (SYNCH),000,377,125,252,347,030,303,074,125.
; * THESE CHARACTERS ARE THEN READ OFF OF THE FIFO AND CHECKED. OF IMPORTANCE
; * IS THE INTEGRITY OF THE LAST OVERRUN-CAUSING FIFO CHARACTER (IT SHOULD
; * REMAIN INTACT).
; * NOTE THAT NO CLOCKS ARE PROVIDED WHEN RECEIVING THE CHARACTERS SINCE THEY
; * ARE SUPPLIED BY THE FIFO SUPPORT LOGIC IN GROUPS OF 4 TICKS (WHEN
; * RDA = 0).
; *****
; *****
; * TEST 28 <BCP RX OVERRUN SET AND CLEAR TEST>
; *
; * THE USYRT IS INITIALIZED AND THREE SUBTESTS ARE PERFORMED.
; *
; *   1 -- AN OVERRUN CONDITION IS FORCED, RECEIVER STATUS REGISTER IS
; *        READ TWICE: ONCE TO VERIFY ROR BIT = 1, AND AGAIN TO VERIFY
; *        THAT THE FIRST READ CLEARED ROR .
; *
; *   2 -- AN OVERRUN CONDITION IS FORCED (BY THE SAME TECHNIQUE USED IN
; *        (2), THE USYRT IS RESET AND THE PROPER STATE OF ALL REGISTERS
; *        IS VERIFIED.
; *
; *   3 -- AN OVERRUN CONDITION IS FORCED (AS ABOVE). RXE IS THEN DROPPED
; *        AND A DELAY IS PROVIDED TO ALLOW TIME FOR THE FIFO TO FLUSH
; *        (CAUSED BY RDA GOING LOW). RXE IS THEN RE-INITIALIZED AND ROR

```

PROGRAM DOCUMENT

1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251

:* IS CHECKED = 0. THE RECEIVER IS THEN RE-SYNCHED AND THE TEST IS
:* TERMINATED.
:*

:* TEST 29 <BCP RX SYNC CHARACTER RECOGNITION>
:*

:* THE FOLLOWING MESSAGE IS INITIATED WITHOUT ASSERTING RXE AND ONCE
:* THE DATA IS BEING TRANSMITTED, RXE IS ASSERTED (IE: *):

:* SYNC * SYNC DATA DATA DATA SYNC SYNC SYNC SYNC SYNC DATA DATA DATA
:* SYNC SYNC DATA DATA DATA SYNC SYNC

:* THE RECEIVER SHOULD IGNORE THE FIRST STRING OF DATA CHARACTERS, USE
:* THE NEXT TWO SYNC CHARACTERS FOR SYNCHRONIZATION, THEN PASS THE REST
:* OF THE MESSAGE (7 SYNC AND 6 DATA CHARACTERS) THROUGH RXDB REGISTER.

:* TEST 30 <BCP RX STRIP-SYNC TEST>
:*

:* THE USYRT IS INITIALIZED WITH THE STRIP-SYNC CONTROL BIT ASSERTED.
:* THE FOLLOWING MESSAGE IS THEN INITIATED WITHOUT ASSERTING RXE AND
:* ONCE THE DATA IS BEING TRANSMITTED, RXE IS ASSERTED (IE: *):

:* SYNC * SYNC DATA DATA DATA SYNC SYNC SYNC SYNC SYNC DATA DATA DATA
:* SYNC SYNC DATA DATA DATA SYNC SYNC

:* THE RECEIVER SHOULD IGNORE THE FIRST STRING OF DATA CHARACTERS, USE
:* THE NEXT TWO SYNC CHARACTERS FOR SYNCHRONIZATION, IGNORE THE NEXT
:* THREE SYNC CHARACTERS, AND PASS THE REST OF THE MESSAGE (4 SYNC AND
:* 6 DATA CHARACTERS) THROUGH RXDB REGISTER.

:* TEST 31 <BCP RX LOST RXE TEST>
:*

:* THE USYRT IS INITIALIZED (CRC16,STRIPS,BCP MODE) AND A MESSAGE IS STARTED.
:* WHILE IN THE MIDDLE OF TEXT, RXE IS DROPPED AND THE REACTION OF THE
:* RECEIVER IS MONITORED.

PROGRAM DOCUMENT

1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294

8.0 ERROR INFORMATION

8.1 ERROR REPORTING

ERRORS ARE REPORTED BY THE PROGRAM AS THEY OCCUR (IF NOT INHIBITED). THE REPORT CONFORMS TO THE DIAGNOSTIC SUPERVISOR ERROR REPORT FORMAT, AND CONSISTS OF A DESCRIPTION OF THE ERROR, THE TEST NUMBER, SUBTEST NUMBER, PC OF THE ERROR CALL, DEVICE ADDRESS, AND BASIC AND EXTENDED ERROR INFORMATION.

THE FOLLOWING EXAMPLE PROVIDES A TYPICAL ERROR REPORT, WHICH DESCRIBES A "MASTER CLEAR FAILURE" ERROR, AND PROVIDES THE PC OF THE ERROR CALL AND THE DEVICE REGISTER CONTENTS :

CNDMB DVC FTL ERR 00001 ON UNIT 00 TST 002 SUB 000 PC: 021122
MASTER CLEAR FAILURE

THE CONTENTS OF ALL BYTE SELECT REG'S ARE:

BSEL0	BSEL1	BSEL2	BSEL3
000	000	000	000
BSEL4	BSEL5	BSEL6	BSEL7
000	000	121	000
BSEL10	BSEL11	BSEL12	BSEL13
000	000	000	000
BSEL14	BSEL15	BSEL16	BSEL17
000	000	000	000

FOR OTHER ERRORS, THE REPORT MAY BE MORE EXTENSIVE, AND REQUIRE ADDITIONAL DATA TO BE REPORTED.

IF EXTENDED ERROR INFORMATION HAD BEEN INHIBITED USING THE IXE FLAG PRIOR TO RUNNING THE TEST, THE ABOVE ERROR WOULD HAVE BEEN REPORTED IN THE FOLLOWING SHORTENED FORM :

CNDMB DVC FTL ERR 00001 ON UNIT 00 TST 002 SUB 000 PC: 021122
MASTER CLEAR FAILURE

†

GENERAL EQUATES AND DS INVOCATION & SETUP

```

1296          .SBTTL GENERAL EQUATES AND DS INVOCATION & SETUP
1297
1298
1299          000000      HELP=0          ; CONTROL LISTING OF HELP INFORMATION
1300
1301
1302
1303
1309          002000      .=2000
1310
1311          .MCALL SVC
1312 002000      SVC          ; INITIALIZE SUPERVISOR MACROS
1313
1314
1315 002000      BGNMOD  LU1MOD
1316
1317
1318          000001      $LSTIN= 1
1319          000001      $LSTTAG= 1
1320          000001      SVCINS= 1      ; LIST INSTRUCTIONS, SHIFTED RIGHT
1321          000001      SVCTST= 1      ; LIST TEST TAGS, SHIFTED RIGHT
1322          000001      SVCS'IB= 1     ; LIST SUBTEST TAGS, SHIFTED RIGHT
1323          000001      SVCGBL= 1     ; LIST GLOBAL TAGS, SHIFTED RIGHT
1324          000001      SVCTAG= 1     ; LIST OTHER TAGS, SHIFTED RIGHT
1325
1326          ;          CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
1327          ;          TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS.  CHANGE THE
1328          ;          SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS.  YOU MAY
1329          ;          CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.

```

PROGRAM HEADER

```

1331
1332
1333
1334
1335
1336
1337 002000
1338
1346
1347 002000
002000
002000 103
002001 116
002002 104
002003 115
002004 103
002005 000
002006 000
002007 000
002010
002010 101
002011
002011 060
002012
002012 000000
002014
002014 000012
002016
002016 036674
002020
002020 000000
002022
002022 002224
002024
002024 000000
002026
002026 037462
002030
002030 000000
002032
002032 000000
002034
002034 000000
002036
002036 000000
002040
002040 002124
002042
002042 000000
002044
002044 000000
002046
002046 000000
002050
002050 003
002051 003
002052

```

```

.SBTTL PROGRAM HEADER
;
; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;
        POINTER  BGNAU,BGNDU,ERRTRL

        HEADER  CNDMC,A,0,10.,0

```

```

L$NAME::
        .ASCII /C/
        .ASCII /N/
        .ASCII /D/
        .ASCII /M/
        .ASCII /C/
        .BYTE 0
        .BYTE 0
        .BYTE 0

L$REV::
        .ASCII /A/

L$DEPU::
        .ASCII /0/

L$UNIT::
        .WORD 0

L$TIML::
        .WORD 10.

L$MPCP::
        .WORD L$HARD

L$SPCP::
        .WORD 0

L$HPTP::
        .WORD L$HW

L$SPTP::
        .WORD 0

L$LADP::
        .WORD L$LAST

L$STA::
        .WORD 0

L$CO::
        .WORD 0

L$DTYP::
        .WORD 0

L$APT::
        .WORD 0

L$DTP::
        .WORD L$DISPATCH

L$PRIO::
        .WORD 0

L$ENVI::
        .WORD 0

L$EXPI::
        .WORD 0

L$MREV::
        .BYTE C$REVISION
        .BYTE C$EDIT

L$EF::

```

PROGRAM HEADER

002052 000000
 002054 000000
 002056 000000
 002060 003306
 002062 000000
 002064 000000
 002066 000000
 002070 021602
 002072 021576
 002074 000000
 002076 003326
 002100 104035
 002102 002246
 002104 021100
 002106 021574
 002110 021450
 002112 021072
 002114 000000
 002116 000000
 002120 000000

.WORD 0
 .WORD 0
 L\$SPC:: .WORD 0
 L\$DEVP:: .WORD L\$DVTYP
 L\$REPP:: .WORD 0
 L\$EXP4:: .WORD 0
 L\$EXP5:: .WORD 0
 L\$AUT:: .WORD L\$AU
 L\$DUT:: .WORD L\$DU
 L\$LUN:: .WORD 0
 L\$DESP:: .WORD L\$DESC
 L\$(LOAD):: EMT E\$LOAD
 L\$ETP:: .WORD L\$ERRTBL
 L\$ICP:: .WORD L\$INIT
 L\$CCP:: .WORD L\$CLEAN
 L\$ACP:: .WORD L\$AUTC
 L\$PRT:: .WORD L\$PROT
 L\$TEST:: .WORD 0
 L\$DLY:: .WORD 0
 L\$HIME:: .WORD 0

1348
 1354
 1355

.EVEN

DISPATCH TABLE

1357
1358
1359 002122

1360
1361
1362 002122

.SBTTL DISPATCH TABLE

SLASH
:;///
:; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
:; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
SLASH
:;///

1363
1364 002122

DISPATCH 31.

002122 000037
002124 021604
002126 021764
002130 022764
002132 023150
002134 023234
002136 023460
002140 023552
002142 023644
002144 023736
002146 024066
002150 024160
002152 024234
002154 024310
002156 024632
002160 024764
002162 025146
002164 025654
002166 026254
002170 026462
002172 026740
002174 027450
002176 027644
002200 030040
002202 030234
002204 030612
002206 033534
002210 034112
002212 034476
002214 035100
002216 035556
002220 036312

1365

.WORD 31
L\$DISPATCH:;
.WORD T1
.WORD T2
.WORD T3
.WORD T4
.WORD T5
.WORD T6
.WORD T7
.WORD T8
.WORD T9
.WORD T10
.WORD T11
.WORD T12
.WORD T13
.WORD T14

.WORD T15

.WORD T16

.WORD T17

.WORD T18

.WORD T19

.WORD T20

.WORD T21

.WORD T22

.WORD T23

.WORD T24

.WORD T25

.WORD T26

.WORD T27

.WORD T28

.WORD T29

.WORD T30

.WORD T31

DEFAULT HARDWARE P-TABLE

```

1373          .SBTTL  DEFAULT HARDWARE P-TABLE
1374
1375          ;////////////////////////////////////////////////////////////////////
1376          ;/ THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
1377          ;/ THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
1378          ;/ IS IDENTICAL TO THE STRUCTURE OF THE RUN-TIME P-TABLE.
1379          ;////////////////////////////////////////////////////////////////////
1380
1381          002222          BGNHW  DFPTBL
1382          002222          000010
1383          002224          160020          .WORD 160020          ;DMV11 CSR UNIBUS ADDRESS
1384          002226          000300          .WORD 300          ;DMV11 INTERRUPT VECTOR
1385          002230          004000          .WORD 4000          ;DMV11 INTERRUPT PRIORITY LEVEL = 4
1386          002232          000000          .WORD 000          ;SWITCH REG. #1 (BOOT ADDRESS)
1387          002234          000000          .WORD 000          ;SWITCH REG. #2 (DDCMP ADDRESS)
1388          002236          000000          .WORD 0          ;MODULE IS M8064
1389          002240          000000          .WORD 0          ;H3254&H3255 USED
1390          002242          000001          .WORD 1          ;BAUD RATE = 56 K
1391          ;          0 = 19.2 K
1392          ;          1 = 56 K
1393
1394          002244          ENDDHW
1395          002244

```

```

          .WORD  L10000-L$HW/2
L$HW::
DFPTBL::
L10000:

```

SOFTWARE P-TABLE

1396
1397
1398
1399
1400
1401
1402
1403 002244
002244 000000
002246
002246
1404
1405 002246
002246

```
.SBTTL SOFTWARE P-TABLE
;/////////////////////////////////////////////////////////////////
; THE SOFTWARE P-TABLE CONTAINS THE VALUES OF THE PROGRAM
; PARAMETERS THAT CAN BE CHANGED BY THE OPERATOR.
;/////////////////////////////////////////////////////////////////
          BGNSW SFPTBL
                                     .WORD L10001-L$SW/2
L$SW: :
SFPTBL: :
L10001: :
```

GLOBAL EQUATES SECTION -- BASIC EQUATES

1407
1408
1409
1410
1411
1412
1413
1414
1415 002246

```
.SBTTL GLOBAL EQUATES SECTION -- BASIC EQUATES

;
; // THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
; // ARE USED IN MORE THAN ONE TEST.
;
; //////////////////////////////////////////////////////////////////
```

EQUALS

; BIT DIFINITIONS

```
100000 BIT15== 100000
040000 BIT14== 40000
020000 BIT13== 20000
010000 BIT12== 10000
004000 BIT11== 4000
002000 BIT10== 2000
001000 BIT09== 1000
000400 BIT08== 400
000200 BIT07== 200
000100 BIT06== 100
000040 BIT05== 40
000020 BIT04== 20
000010 BIT03== 10
000004 BIT02== 4
000002 BIT01== 2
000001 BIT00== 1
```

```
001000 BIT9== BIT09
000400 BIT8== BIT08
000200 BIT7== BIT07
000100 BIT6== BIT06
000040 BIT5== BIT05
000020 BIT4== BIT04
000010 BIT3== BIT03
000004 BIT2== BIT02
000002 BIT1== BIT01
000001 BIT0== BIT00
```

; EVENT FLAG DEFINITIONS
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

```
000040 EF.START== 32. ; BIT POSITION IN SECOND STATUS WORD
000037 EF.RESTART== 31. ; (100000) START COMMAND WAS ISSUED
000036 EF.CONTINUE== 30. ; (040000) RESTART COMMAND WAS ISSUED
000035 EF.NEW== 29. ; (020000) CONTINUE COMMAND WAS ISSUED
000034 EF.PWR== 28. ; (010000) A NEW PASS HAS BEEN STARTED
; ; (004000) A POWER-FAIL/POWER-UP OCCURRED
```

; PRIORITY LEVEL DEFINITIONS

```
000340 PRI07== 340
000300 PRI06== 300
000240 PRI05== 240
000200 PRI04== 200
```


GLOBAL EQUATES SECTION -- BASIC EQUATES

000140	PRI03**	140
000100	PRI02**	100
000040	PRI01**	40
000000	PRI00**	0
	:	
	;OPERATOR FLAG BITS	
	:	
000004	EVL**	4
000010	LOT**	10
000020	ADR**	20
000040	IDU**	40
000100	ISR**	100
000200	UAM**	200
000400	BOE**	400
001000	PNT**	1000
002000	PRI**	2000
004000	IXE**	4000
010000	IBE**	10000
020000	IER**	20000
040000	LOE**	40000
100000	HOE**	100000

REGISTER DEFINITIONS -- MAINTENANCE REGISTERS -- SELN & BSELN

```

1417          .SBTTL REGISTER DEFINITIONS -- MAINTENANCE REGISTERS -- SELN & BSELN
1418
1419          ;;*****
1420          ;* MAINTENANCE REGISTER # 0 - BSEL0
1421          ;;*****
1422          000020 IEO      = BIT4      ;"INTERRUPT ENABLE OUT"
1423          000001 IEI      = BIT0      ;"INTERRUPT ENABLE IN"
1424
1425          ; BIT 7 IS ALSO USED BY THE MICROCODE. ITS LABEL IS "RQI" WHICH STANDS FOR
1426          ; "REQUIST IN". IT'S PART OF THE HANDSHAKING FOR USING THE SEL & BSFL REG'S.
1427          ; HOWEVER, THE MAINT. LOOP DOES NOT MAKE USE OF THIS BIT AND IT IS THEREFORE
1428          ; UNNECESSARY TO DEFINE IT HERE.
1429
1430          ;;*****
1431          ;* MAINTENANCE REGISTER # 1 - BSEL1
1432          ;;*****
1433          000200 RUN      = BIT7      ;"RUN" & ALSO CONTROLS 6502 MICROPROCESSOR'S RDY STATE
1434          000100 MCLR     = BIT6      ;MASTER CLEAR
1435          000001 MREQ     = BIT0      ;M-LOOP ACCESS
1436          000301 STRTMLOP- RUN!MCLR!MREQ ;INITIATE M-LOOP
1437
1438          ;;*****
1439          ;* MAINTENANCE REGISTER # 2 - BSEL2
1440          ;;*****
1441          000200 MRDY     = BIT7      ;M-LOOP READY
1442
1443          ;*****
1444          ;* MAINTENANCE LOOP COMMAND DEFINITIONS
1445          ;*****
1446          000001 REDLOC   = 1        ;READ LOC. W/IN DMV-11 ---- (SEL4) ==> BSEL6
1447          000002 WRILOC   = 2        ;WRITE LOC. W/IN DMV-11 --- BSEL6 ==> (SEL4)
1448          000003 REDPAG   = 3        ;READ BLOCK W/IN DMV-11 --- (SEL6) ==> (SEL4)
1449          000004 WRIPAG   = 4        ;WRITE BLOCK W/IN DMV-11 -- (SEL4) ==> (SEL6)
1450          000005 EXECUT   = 5        ;SET 6502'S PC AND EXECUTE -- SEL6 ==> PC
1451          000007 DOTBMT   = 7        ;SET MAINTENANCE INTERRUPT DISABLE IN PROCESSOR
1452          ;STATUS --- [KB7] ==> BSEL3
1453

```

REGISTER DEFINITIONS - USYRT

```

1455      .SBTTL REGISTER DEFINITIONS -- USYRT
1456
1457
1458      120400      USYRT = 120400      ;USYRT BASE ADDRESS = A100 (HEX)
1459
1460      ;*****
1461      ;* USYRT "RECEIVER DATA BUFFER" REGISTER -- READ ONLY
1462      ;*****
1463
1464      120400      RDSRL  = 120400      ;ADDRESS OF THIS REG
1465
1466      ;*****
1467      ;* USYRT "RECEIVER STATUS" REGISTER -- READ ONLY
1468      ;*****
1469
1470      120401      RDSRH  = 120401      ;ADDRESS OF THIS REG
1471
1472      ;BIT DEFINITIONS ON BYTE BASIS :
1473      000200      RERR   = BIT7      ;ERROR CHECK
1474      000160      ABC    = BIT6:BIT5:BIT4 ;ASSEMBLED BIT COUNT
1475      000010      ROR    = BIT3      ;RECEIVER OVER RUN
1476      000004      RABGA  = BIT2      ;RECEIVED ABORT/GO CHARACTER
1477      000002      REOM   = BIT1      ;RECEIVED END-OF-MESSAGE
1478      000001      RSOM   = BIT0      ;RECEIVED START-OF-MESSAGE
1479
1480      ;BIT DEFINITIONS ON WORD BASIS :
1481      100000      RXERR  = BIT15     ;RECEIVED CRC/VRC ERROR
1482      004000      RXOR   = BIT11     ;RECEIVER OVER RUN
1483      002000      RXABGA = BIT10     ;RECEIVED ABORT/GO AHEAD CHARACTER
1484      001000      RXEOM  = BIT9      ;RECEIVED END-OF-MESSAGE
1485      000400      RXSOM  = BIT8      ;RECEIVED START-OF-MESSAGE
1486
1487      000001      RERCHK = BIT0      ;FLAG TO INVOKE RERR CHK IN SUBROUTINE RXCHAR
1488
1489      ;*****
1490      ;* USYRT "TRANSMITTER DATA BUFFER" REGISTER
1491      ;*****
1492
1493      120402      TDSRL  = 120402      ;ADDRESS OF THIS REG
1494
1495      ;*****
1496      ;* USYRT "TX STATUS AND CONTROL" REGISTER
1497      ;*****
1498
1499      120403      TDSRH  = 120403      ;ADDRESS OF THIS REG
1500
1501      ;BIT DEFINITIONS ON BYTE BASIS :
1502      000200      TERR   = BIT7      ;TRANSMITTER UNDERRUN ERROR
1503      000010      TGA    = BIT3      ;TRANSMIT GO AHEAD
1504      000004      TAB    = BIT2      ;TRANSMIT ABORT
1505      000002      TEOM   = BIT1      ;TRANSMIT END-OF-MESSAGE
1506      000001      TSOM   = BIT0      ;TRANSMIT START-OF-MESSAGE
1507
1508      ;BIT DEFINITIONS ON WORD BASIS :
1509      100000      TXERR  = BIT15     ;TRANSMITTER UNDERRUN ERROR
1510      004000      TXGA   = BIT11     ;TRANSMIT GO AHEAD
1511      002000      TXAB   = BIT10     ;TRANSMIT ABORT
    
```

REGISTER DEFINITIONS - - USYRT

```

1512      001000      TXEOM   = BIT9           ; TRANSMIT END-OF-MESSAGE
1513      000400      TXSOM   = BIT8           ; TRANSMIT START-OF-MESSAGE
1514
1515      ;*****
1516      ;* USYRT "SYNC/SECONDARY ADDRESS" REGISTER
1517      ;*****
1518
1519      120404      PCSARL  = 120404       ; ADDRESS OF THIS REG
1520      000226      SYNCH   = 226         ; STANDARD SYNCH CHARACTER
1521
1522      ;*****
1523      ;* USYRT "MODE CONTROL"
1524      ;*****
1525
1526      120405      PCSARH  = 120405       ; ADDRESS OF THIS REG
1527
1528      ;BIT DEFINITIONS ON BYTE BASIS:
1529
1530      000200      APA     = BIT7         ; "ALL PARTIES ADDRESS" ENABLE
1531      000100      PROTO  = BIT6         ; SPECIFIES BOP/CCP PROTOCOL -- 0 = BOP
1532      000040      STRIP  = BIT5         ; STRIP EXTRA SYNC'S IN CCP MODE, SEE GA CHARS IN BOP
1533      000020      SECAD  = BIT4         ; SECONDARY ADDRESS MODE -- BOP MODE ONLY
1534      000010      IDLE   = BIT3         ; IDLE & SYNC CHAR. TRANSMISSION CONTROL
1535      000007      XYZ    = BIT2:BIT1:BIT0 ; CRC/PARITY SELECTION CONTROL
1536
1537      ;BIT DEFINITIONS ON WORD BASIS:
1538
1539      100000      APAD    = BIT15        ; "ALL PARTIES ADDRESS" ENABLE
1540      040000      DDCMP  = BIT14        ; CODE FOR DDCMP MODE
1541      020000      STRIPS  = BIT13        ; STRIP EXTRA SYNC'S IN CCP MODE, SEE GA CHARS IN BOP
1542      010000      SECADR  = BIT12        ; SECONDARY ADDRESS MODE -- BOP MODE ONLY
1543      004000      IDLES   = BIT11        ; IDLE & SYNC CHAR. TRANSMISSION CONTROL
1544      001400      CRC16  = BIT9:BIT8     ; CODE FOR CRC-16 SELECTION
1545      003400      NOCHK  = BIT10:BIT9:BIT8 ; CODE FOR NO ERROR CHECKING
1546      002400      EVRC   = BIT10:BIT8    ; CODE FOR VRC EVEN CHECK
1547      002000      OVRC   = BIT10        ; CODE FOR VRC ODD CHECK
1548
1549      ;*****
1550      ;* USYRT "DATA LENGTH SELECT" REGISTER
1551      ;*****
1552
1553      120407      PCR     = 120407       ; ADDRESS OF THIS REG
1554
1555      ;BIT DEFINITIONS:
1556
1557      000340      TXDL    = BIT7:BIT6:BIT5 ; TRANSMIT DATA LENGTH SELECTION
1558      000020      EXADD  = BIT4         ; EXTENDED ADDRESS FIELD -- NOT USED OR TESTED
1559      000010      EXCON  = BIT3         ; EXTENDED CONTROL FIELD -- NOT USED OR TESTED
1560      000007      RXDL   = BIT2:BIT1:BIT0 ; RECEIVER DATA LENGTH SELECTION
1561
1562      ;*****
1563      ;* USYRT STATUS REGISTER (ADDR. A400)
1564      ;*****
1565      122000      USTATR  = 122000       ; USYRT STATUS REGISTER ADDRESS = A400 (HEX)
1566
1567      ;BIT DEFINITIONS:
1568

```

REGISTER DEFINITIONS USYRI

1569	000200	RDA	=	BIT7	;RECEIVER DATA AVAILABLE
1570	000100	TBMT	=	BIT6	;TRANSMITTER BUFFER EMPTY
1571	000040	RXACT	=	BIT5	;RECEIVER ACTIVE
1572	000020	RSA	=	BIT4	;RECEIVER STATUS AVAILABLE
1573	000010	TSO	=	BIT3	;TRANSMITTER SERIAL OUTPUT
1574	000004	TXACT	=	BIT2	;TRANSMITTER ACTIVE
1575	000002	TXU	=	BIT1	;TRANSMITTER UNDERRUN
1576	000001	SFR	=	BIT0	;SYNC/FLAG RECEIVED

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1578 .SBTTL REGISTER DEFINITIONS -- 6522 VIA CHIP
1579
1580 120000 VIA = 120000 ;VIA BASE ADDRESS = A000 (HEX)
1581
1582 ;*****
1583 ;* MODEM & MAINTENANCE CONTROL -- "ORB" 8 BIT PORT B -- WRITE ONLY
1584 ;*****
1585
1586 120000 VIAORB = 120000 ;ADDRESS OF THIS REGISTER -- HEX = A0X0
1587
1588 000200 NULCLK = BIT7 ;"NULL CLK L" -- NULL CLOCK
1589 000100 RXEN = BIT6 ;"RXENL" -- USYRT RECEIVER ENABLE
1590 000040 TXEN = BIT5 ;"TXENL" -- USYRT TRANSMITTER ENABLE
1591 000020 DTR = BIT4 ;"DTR" -- DATA TERMINAL READY
1592 000010 RTSND = BIT3 ;"RTSND" -- REQUEST TO SEND
1593 000004 HDX = BIT2 ;"HDX" -- HALF DUPLEX
1594 000002 TTLLOOP = BIT1 ;"SELECT TTL LEVEL LOOPBACK"
1595 000001 PRESET = BIT0 ;"PRESET H" --
1596 000000 DTRL = 0 ;DTR IS ASSERTED LOW
1597
1598 ;*****
1599 ;* MODEM STATUS REGISTER -- "ORA" 8 BIT PORT A -- READ ONLY
1600 ;*****
1601
1602 120001 VIAMS = 120001 ;ADDRESS OF THIS REGISTER -- HEX = A0X1
1603
1604 000200 RING = BIT7 ;"RING H" --
1605 000100 CARRIER = BIT6 ;"CARRIER H" --
1606 000040 MDMRDY = BIT5 ;"MODEM RDY H" --
1607 000020 SPEED = BIT4 ;"BAUD RATE SWITCH -- (19.2K/56K)
1608 000010 CTS = BIT3 ;"CTS H -- CLEAR TO SEND
1609 000004 TM = BIT2 ;"TEST MODE H" --
1610 000002 RCVDAT = BIT1 ;"RCV DATA H" --
1611 000001 UMAINT = BIT0 ; SELECT USYRT INT LOOPBACK **SELECT BIT**
1612
1613
1614 ;*****
1615 ;* DATA DIRECTION FOR PORT B -- "ODRB" -- READ/WRITE
1616 ;*****
1617
1618 120002 VIADPB = 120002 ;ADDRESS OF THIS REGISTER -- HEX = A0X2
1619
1620 ; ALL BITS ARE DEFINED THE SAME:
1621 ; THE BIT SETTING DEFINED THE DIRECTION OF ITS RELATED BIT IN BIT PORT B
1622
1623 ; INITIALIZED TO 377 (HEX = FF) -- PORT B IS READ/WRITE
1624
1625
1626 ;*****
1627 ;* DATA DIRECTION FOR PORT A -- "DDRA" -- READ/WRITE
1628 ;*****
1629
1630 120003 VIADPA = 120003 ;ADDRESS OF THIS REGISTER -- HEX = A0X3
1631
1632 ; ALL BITS ARE DEFINED THE SAME:
1633 ; THE BIT SETTING DEFINED THE DIRECTION OF ITS RELATED BIT IN BIT PORT A
1634

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1635 ;           INITIALIZED TO 001 (HEX = 01) -- PORT A IS READ ONLY (EXCEPT FOR
1636 ;           BIT0 WHICH ENABLES USYRT INTERNAL LOOPBACK).
1637 ;
1638
1639
1640 ;*****
1641 ;* TIMER 1 LOW ORDER (LATCH & COUNTER) -- "T1L-L" & "T1C-L" -- WRITE & READ
1642 ;*****
1643
1644 120004 VIAT1A = 120004 ;ADDRESS OF THIS REGISTER -- HEX = A0X4
1645
1646 ; WHEN WRITING, LOW ORDER LATCH IS LOADED.
1647 ; WHEN READING, LOW ORDER COUNTER IS READ.
1648
1649
1650
1651 ;*****
1652 ;* TIMER 1 HIGH ORDER COUNTER & TRIGGER -- "T1L-H AND TRIGGER" & "T1C-H"
1653 ;* -- WRITE & READ
1654 ;*****
1655
1656 120005 VIAT1B = 120005 ;ADDRESS OF THIS REGISTER -- HEX = A0X5
1657
1658 ; WHEN WRITING; HIGH ORDER LATCH IS LOADED, BOTH LOW & HIGH ORDER LATCHES
1659 ; ARE LOADED INTO THE COUNTER, AND THE COUNTER IS STARTED.
1660
1661 ; WHEN READING, THE HIGH ORDER COUNTER IS READ.
1662
1663
1664
1665 ;*****
1666 ;* TIMER 1 LOW ORDER LATCH -- "T1L-L" -- READ/WRITE
1667 ;*****
1668
1669 120006 VIAT1C = 120006 ;ADDRESS OF THIS REGISTER -- HEX = A0X6
1670
1671 ; THE LOW ORDER LATCH IS READ OR LOADED. THIS LATCH IS USED TO LOAD THE
1672 ; COUNTER WHEN T1MODE (IN VIAACR) = 3
1673
1674
1675
1676 ;*****
1677 ;* TIMER 1 HIGH ORDER LATCH -- "T1L-H" -- READ/WRITE
1678 ;*****
1679
1680 120007 VIAT1D = 120007 ;ADDRESS OF THIS REGISTER -- HEX = A0X7
1681
1682 ; THE HIGH ORDER LATCH IS READ OR LOADED. THIS LATCH IS USED TO LOAD THE
1683 ; COUNTER WHEN T1MODE (IN VIAACR) = 3
1684
1685
1686
1687 ;*****
1688 ;* TIMER 2 LOW ORDER (LATCH & COUNTER) -- "T2L-L" & "T2C-L" -- WRITE & READ
1689 ;*****
1690
1691 120010 VIAT2A = 120010 ;ADDRESS OF THIS REGISTER -- HEX = A0X8

```

REGISTER DEFINITIONS - 6522 VIA CHIP

```

1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748

```

```

; WHEN WRITING, LOW ORDER LATCH IS LOADED.
; WHEN READING, LOW ORDER COUNTER IS READ.

;*****
;* TIMER 2 HIGH ORDER COUNTER & TRIGGER -- "T2L-H AND TRIGGER" & "T2C H"
;* -- WRITE & READ
;*****

120011
VIAT2B = 120011 ;ADDRESS OF THIS REGISTER -- HEX = A0X9

; WHEN WRITING; HIGH ORDER LATCH IS LOADED, BOTH LOW & HIGH ORDER LATCHES
; ARE LOADED INTO THE COUNTER, AND THE COUNTER IS STARTED.

; WHEN READING, THE HIGH ORDER COUNTER IS READ.

;*****
;* SHIFT REGISTER -- "SR" -- READ/WRITE
;*****

120012
VIASR = 120012 ;ADDRESS OF THIS REGISTER -- HEX = A0XA

; SHIFTING IS CONTROLLED BY THE SETTING OF VIASRC (ACR2 ---> ACR4) IN VIAACR

;*****
;* AUXILIARY CONTROL REGISTER -- "ACR" -- READ/WRITE
;*****

120013
VIAACR = 120013 ;ADDRESS OF THIS REGISTER -- HEX = A0XB

000300
T1MODE = BIT7!BIT6 ;CONTROL THE MODE OF TIMER # 1

;BIT 7:
; 0 PB7 DISABLED -- ONLY T1T0 IN VIAIFR REFLECTS TIMEOUT
; 1 PB7 & T1T0 REFLECT TIMEOUT

;BIT 6:
; 0 TIMER 1 IN ONE SHOT MODE
; 1 TIMER 1 IN CONTINUOUS SQUARE WAVE MODE

000040
T2MODE = BIT5 ;CONTROLS THE MODE OF TIMER # 1

; 0 PULSE COUNTING MODE
; 1 INTERVAL TIMER MODE

000034
SRMODE = BIT4!BIT3!BIT2 ;CONTROLS THE MODE OF THE SHIFT REGISTER

; 0 SR DISABLED
; 1 SHIFT IN UNDER CONTROL OF T2, SHFT PULSES GEN'D ON CB1
; 2 SHIFT IN AT SYS. CLOCK RATE, SHFT PULSES GEN'D ON CB1
; 3 SHIFT IN UNDER CONTROL OF EXTERNAL INPUT PULSES
; 4 SHIFT OUT -- FREE RUNNING -- RATE CONTROLLED BY T2
; 5 SHIFT OUT -- RATE CONTROLLED BY T2 -- PULSES ON CB1

```


REGISTER DEFINITIONS - 6502 VIA CHIP

```

1749                                     ; 6   SHIFT OUT -- SYS. CLOCK RATE -- PULSES ON CB1
1750                                     ; 7   SHIFT OUT -- UNDER CONTROL OF PULSES APPLIED TO CB1
1751
1752      000002      PBLN8  = BIT1      ;PB LATCH CONTROL -- 1 ENABLES LATCH
1753      000001      PALN8  = BIT0      ;PA LATCH CONTROL -- 1 ENABLES LATCH
1754
1755
1756
1757
1758      ;*****
1759      ;* PERIPHERAL CONTROL REGISTER -- "PCR" -- READ/WRITE
1760      ;*****
1761
1762      120014      VIAPCR = 120014      ;ADDRESS OF THIS REGISTER -- HEX = A0XC
1763
1764      000340      CB2CTL = BIT7!BIT6!BIT5 ;CB2 MODE SELECT
1765      000020      CB1CTL = BIT4        ;CB1 MODE SELECT
1766      000016      CA2CTL = BIT3!BIT2!BIT1 ;CA2 MODE SELECT
1767      000001      CA1CTL = BIT0        ;CA1 MODE SELECT
1768
1769
1770
1771      ;*****
1772      ;* INTERRUPT FLAG REGISTER -- "IFR" -- READ ONLY
1773      ;*****
1774
1775      120015      VIAIFR = 120015      ;ADDRESS OF THIS REGISTER -- HEX = A0XD
1776
1777      000200      FLGIRQ = BIT7        ;SET WHEN A FLAG IN THIS REG. GOES HIGH AND
1778                                     ;ITS CORRESPONDING BIT IN VIAIER IS SET.
1779                                     ;(I.E. VIAIER IS THE ENABLE REGISTER FOR THE
1780                                     ;FOR THE SETTING OF IRQ AND THE ISSUANCE OF
1781                                     ;AN INTERRUPT TO THE 6502 WHEN IRQ IS SET.)
1782
1783      000100      FLGT1  = BIT6        ;TIMEOUT OF TIMER 1
1784      000040      FLGT2  = BIT5        ;TIMEOUT OF TIMER 2
1785      000020      FLGCB1 = BIT4        ;ACTIVE TRANSITION OF PIN 18 (CB1)
1786      000010      FLGCB2 = BIT3        ;ACTIVE TRANSITION OF PIN 19 (CB2)
1787      000004      FLGSR  = BIT2        ;COMPLETION OF 8 SHIFTS
1788      000002      FLGCA1 = BIT1        ;ACTIVE TRANSITION OF PIN 40 (CA1)
1789      000001      FLGCA2 = BIT0        ;ACTIVE TRANSITION OF PIN 39 (CA2)
1790
1791
1792
1793      ;*****
1794      ;* INTERRUPT ENABLE REGISTER -- "IER" -- READ/WRITE
1795      ;*****
1796
1797      120016      VIAIER = 120016      ;ADDRESS OF THIS REGISTER -- HEX = A0XE
1798
1799      000200      INTSC  = BIT7        ;CONTROLS THE SETTING OR CLEARING OF BITS IN
1800                                     ;THE REST OF IER. IF = 0 THE OTHER BITS IN
1801                                     ;THIS REG., IF SET, WILL CLEAR THEIR RESPECTIVE
1802                                     ;BITS IN THE INT. ENAB. REG.. IF = 1, THE
1803                                     ;RESPECTIVE BITS WILL BE SET.
1804
1805      ; WHEN WRITING THIS REG., THE COMMENT ABOVE HOLDS.

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826

120017

```

; WHEN READING THIS REG., THE CURRENT STATE OF THE INT. ENABLE REG. IS RETURNED.
; THE BIT ASSIGNMENTS ARE THE SAME AS FOR VIAIFR AS DEFINED ABOVE.

;*****
;* OUTPUT REGISTER A -- "ORA" -- READ ONLY (OR READ/WRITE UNDER CONTROL OF "DDPA")
;*****

VIAORA = 120017 ;ADDRESS OF THIS REGISTER -- HEX = A0XF

; THIS ADDRESS ACCESSES THE SAME DATA AS "VIAMS" EXCEPT THAT NO "HANDSHAKING"
; WILL TAKE PLACE (I.E. THERE IS NO CHANGE IN IRQ OR CA2 AS A RESULT OF
; READING ORA THROUGH THIS ADDRESS)

;THE BIT ASSIGNMENTS ARE THE SAME AS FOR "VIAMS" ABOVE.

```

REGISTER DEFINITIONS -- MISC

```

1828          .SBTTL REGISTER DEFINITIONS -- MISC
1829
1830          ;*****
1831          ;* SWITCH PACKS
1832          ;*****
1833
1834          121000 SWPBOT      = 121000          ;"BOOT ADDRESS" SWITCH PACK [A200]
1835          121400 SWPDDCMP  = 121400          ;"DDCMP ADDRESS" SWITCH PACK [A300]
1836
1837          ;MISCELLANEOUS EQUATES
1838
1839          100000 TCCHEK    = BIT15          ;FLAG TO REQUEST H3254,5 CHECK
1840          001000 RAMADR   = 001000          ;STARTING ADRS OF RAM PAGE 2 (ADRS 0200 HEX)
1841
1842          000002 EIAV35   = BIT1          ;SELECT V.35 OR EIA 423/232C
1843          000001 INTGRL   = BIT0          ;SELECT INTEGRAL MODEM
1844
1845          040000 NCRXEN   = BIT14          ;KILL RXEN DURING "INITRN"
1846          001000 NOLLOOP  = BIT9          ;KILL TTLOOP DURING "INITRN"
1847
1848          000200 NCTBMT   = BIT7          ;DISABLE INITIAL TBMT=0 CHECK IN TXCHAR
1849
1850          100000 NOCRDA   = BIT15          ;DISABLE INITIAL RDA=0 CHECK IN RXCHAR
1851          040000 NFCRDA   = BIT14          ;DISABLE FINAL RDA=1 CHECK IN RXCHAR
1852          020000 NCRACT   = BIT13          ;DISABLE RXACT=1 CHECK AFTER CLOCKING (RXCHAR)
1853

```

GLOBAL DATA SECTION

2113
 2114
 2115
 2116
 2117
 2118
 2119
 2120
 2121
 2122
 2123
 2124 002246
 002246
 002246 000000
 002250 000000
 002252 000000
 002254 000000
 2125
 2126
 2127
 2128
 2129 002256
 2130 002256 000000
 2131 002260
 2132 002260 000000
 2133 002262
 2134 002262 000000
 2135 002264
 2136 002264 000000
 2137 002266
 2138 002266 000000
 2139 002270
 2140 002270 000000
 2141 002272
 2142 002272 000000
 2143 002274
 2144 002274 000000
 2145 002276 000000
 2146 002300 000000
 2147 002302 000000
 2148 002304 000000
 2149 002306 000000
 2150 002310 000000
 2151 002312 000000
 2152 002314 000000
 2153
 2154 002316
 2155
 2156
 2157 002336

```

.SBTTL GLOBAL DATA SECTION
;/////////////////////////////////////////////////////////////////
; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
; IN MORE THAN ONE TEST.
;/////////////////////////////////////////////////////////////////
;+*****
; CONTROL BLOCK FOR STACKED ERROR MESSAGES
;-*****

                ERRTBL
                L$ERRTBL:
ERRTYP:        .WORD 0
ERRNBR:        .WORD 0
ERRMSG:        .WORD 0
ERRBLK:        .WORD 0

;+*****
; STORAGE FOR DEVICE REGISTERS
;+*****
WSR0:          ; STORAGE FOR DEVICE CSR REGISTERS
BSR0:          .WORD 0
WSR2:
BSR2:          .WORD 0
WSR4:
BSR4:          .WORD 0
WSR6:
BSR6:          .WORD 0
WSR10:
BSR10:         .WORD 0
WSR12:
BSR12:         .WORD 0
WSR14:
BSR14:         .WORD 0
WSR16:
BSR16:         .WORD 0
BSR7:          .WORD 0
BSR10:         .WORD 0
BSR11:         .WORD 0
BSR12:         .WORD 0
BSR13:         .WORD 0
BSR14:         .WORD 0
BSR15:         .WORD 0
BSR16:         .WORD 0
BSR17:         .WORD 0

UREGS:         .BLKW 8.
VREGS:         .BLKW 16.

; THE FIRST 7 ARE FOR THE USYRT'S ACTUAL
; REGISTERS. THE LAST ONE IS FOR THE STATUS
; REG. (USTATR).
; STORAGE FOR VIA REGISTERS FOR PRINTOUT

```

GLOBAL DATA SECTION

```

2159 ;*****
2160 ;* MISCELLANEOUS STORAGE
2161 ;*****
2162 002370 000000 TDATA: .WORD 0 ;TEST DATA
2163 002400 000000 GDATA: .WORD 0 ;GOOD DATA
2164 002402 000000 BDATA: .WORD 0 ;BAD DATA
2165 002404 000000 XDATA: .WORD 0 ;EXCLUSIVE-OR BETWEEN GOOD AND BAD DATA
2166 002406 000000 SCRACH: .WORD 0 ;GEN'L PURPOSE SCRATCH WORD
2167 002410 000000 LOGDEV: .WORD 0 ;LOGICAL DEVICE NUMBER
2168 002412 000000 REGNUM: .WORD 0 ;CONTAINS A DEVICE REGISTER NUMBER
2169 002414 000000 PSTACK: .WORD 0 ;CONTAINS BASE LEVEL PROGRAM STACK POINTER
2170 002416 000000 PRIOR: .WORD 0 ;CPU PRIORITY FOR PRINTOUT
2171 002420 000000 SUBRPC: .WORD 0 ;PC OF SUBR CALL FOR ERROR REPORTS
2172 002422 000000 INTFLG: .WORD 0 ;INTERRUPT RECEIVED FLAGS
2173 ; CIT 0 FOR TX, BIT 1 FOR RCV
2174 002424 000000 ERRFLG: .WORD 0 ;SUBROUTINE ERROR FLAG
2175 002426 000000 TIMFLG: .WORD 0 ;EVENT TIME-OUT FLAG
2176 002430 000000 RETADR: .WORD 0 ;SUBR ERROR RETURN ADDRESS
2177 002432 000000 REDBYT: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM LU REG
2178 002434 000000 WRIBYT: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO LU REG
2179 002436 000000 LOADAT: .WORD 0 ;CONTAINS TEST DATA LOADED INTO REG
2180 002440 000000 GOODAT: .WORD 0 ;STORAGE FOR EXPECTED DATA
2181 002442 000000 BADDAT: .WORD 0 ;STORAGE FOR ACTUAL DATA
2182 002444 000000 FRSTIM: .WORD 0 ;FLAG=0 IF PROGRAM JUST LOADED
2183 002446 000000 SAVE4: .WORD 0 ;SAVE LOC 4 HERE (ERROR TRAP VECTOR)
2184 002450 000000 SAVE6: .WORD 0 ;SAVE LOC 6 HERE (ERROR TRAP VECTOR)
2185 002452 000000 ERROR1: .WORD 0 ;SUBR ERR. BIT FLAGS (DEF'D IN GLOBAL EQUATES)
2186 002454 000000 CHPTYP: .WORD 0 ;USYRT CHIP TYPE, =0 FOR SMC, ELSE =1
2187 002456 000000 SAVLEN: .WORD 0 ;SAVED TX AND RCV CHAR LENGTHS
2188 002460 000000 DEVMAP: .WORD 0 ;BIT MAP OF ACTIVE DEVICES
2189 002462 000000 DEVPTR: .WORD 0 ;DEVICE MAP BIT POINTER
2190 002464 000000 UNIT: .WORD 0 ;CONTAINS UNIT NO. (1 TO N)
2191 002466 000000 STARES: .WORD 0 ;FLAG TO SHOW NO. OF PASSES SINCE STA OR RES
2192 002470 000000 TSTNUM: .WORD 0 ;NO. OF CURRENT TEST (FOR SOME TESTS)
2193

```

GLOBAL DATA SECTION

```

2195          ;;***** CURRENT DEVICE PARAMETERS *****
2196 002472    BSEL0:
2197 002472    SEL0:
2198 002472    160020    MPCSR: .WORD      160020          ;POINTER TO DMV11 CSR'S
2199 002474    160021    BSEL1: .WORD      160021          ;POINTER TO BSEL1
2200 002476    BSEL2:
2201 002476    160022    SEL2: .WORD      160022          ;POINTER TO SEL2
2202 002500    160023    BSEL3: .WORD      160023          ;POINTER TO BSEL3
2203 002502    BSEL4:
2204 002502    160024    SEL4: .WORD      160024          ;POINTER TO SEL4
2205 002504    160025    BSEL5: .WORD      160025          ;POINTER TO BSEL5
2206 002506    BSEL6:
2207 002506    160026    SEL6: .WORD      160026          ;POINTER TO SEL6
2208 002510    160027    BSEL7: .WORD      160027          ;POINTER TO BSEL7
2209 002512    BSEL10:
2210 002514    160030    SEL10: .WORD     160030          ;POINTER TO SEL10
2211 002514    160031    BSEL11: .WORD     160031          ;POINTER TO BSEL11
2212 002516    BSEL12:
2213 002516    160032    SEL12: .WORD     160032          ;POINTER TO SEL12
2214 002520    160033    BSEL13: .WORD     160033          ;POINTER TO BSEL13
2215 002522    BSEL14:
2216 002522    160034    SEL14: .WORD     160034          ;POINTER TO SEL14
2217 002524    160035    BSEL15: .WORD     160035          ;POINTER TO BSEL15
2218 002526    BSEL16:
2219 002526    160036    SEL16: .WORD     160036          ;POINTER TO SEL16
2220 002530    160037    BSEL17: .WORD     160037          ;POINTER TO BSEL17
2221
2222 002532    000300    MPIVEC: .WORD      300          ;DMV11 INPUT INTERRUPT VECTOR
2223 002534    000304    MPOVEC: .WORD      304          ;DMV11 OUTPUT INTERRUPT VECTOR
2224 002536    000240    MPRIOR: .WORD      240          ;DMV11 DEVICE PRIORITY
2225 002540    000000    LUSWI1: .WORD       0          ;LINE UNIT SWITCH PACK #1
2226 002542    000000    LUSWI2: .WORD       0          ;LINE UNIT SWITCH PACK #2
2227 002544    000000    BRDTYP: .WORD       0          ;0=M8064, 1=M8053/V.35,2=M8053/EIA
2228 002546    000000    TSTCON: .WORD       0          ;TEST CONNECTOR INDICATOR
2229 002550    000001    BDRATE: .WORD       1          ;BAUD RATE = 56 K
2230          ;           0 = 19.2 K
2231          ;           1 = 56 K

```

GLOBAL DATA SECTION

```

2233      ;TABLE OF USYRT REGISTER ADDRESSES
2234 002552 120400  USYREG: .WORD 120400      ;ADDRESS OF RDSRL
2235 002554 120401      .WORD 120401      ;ADDRESS OF RDSRH
2236 002556 120402      .WORD 120402      ;ADDRESS OF TDSRL
2237 002560 120403      .WORD 120403      ;ADDRESS OF TDSRH
2238 002562 120404      .WORD 120404      ;ADDRESS OF PCSARL
2239 002564 120405      .WORD 120405      ;ADDRESS OF PCSARH
2240 002566 120407      .WORD 120407      ;ADDRESS OF PCR
2241 002570 122000      .WORD 122000      ;ADDRESS OF USYRT STATUS REG
2242
2243      ;***** STORAGE FOR DATA READ IN ADDRESS TESTS *****
2244 002572  REDDAT: .BLKB 8
2245
2246      ;***** GEN'L PURPOSE SCRATCH STORAGE *****
2247 002602 000000  REG0: .WORD 0
2248 002604 000000  REG1: .WORD 0
2249 002606 000000  REG2: .WORD 0
2250 002610 000000  REG3: .WORD 0
2251 002612 000000  REG4: .WORD 0
2252 002614 000000  REG5: .WORD 0
2253 002616 000000  REG6: .WORD 0
2254 002620 000000  REG7: .WORD 0
2255
2256      ;***** SCRATCH STORAGE FOR MESSAGE REPORTING *****
2257 002622 000000  TMP0: .WORD 0
2258 002624 000000  TMP1: .WORD 0
2259 002626 000000  TMP2: .WORD 0
2260 002630 000000  TMP3: .WORD 0
2261 002632 000000  TMP4: .WORD 0
2262 002634 000000  TMP5: .WORD 0
2263 002636 000000  TMP6: .WORD 0
2264 002640 000000  TMP7: .WORD 0
2265
2266      ;***** INBUS LU REG BIT MASKS FOR UNPREDICTABLE BITS *****
2267 002642  UPBITS:
2268 002642 377      .BYTE 377      ;MASK FOR RDBR
2269 002643 000      .BYTE 000      ;MASK FOR RDSR
2270 002644 000      .BYTE 000      ;MASK FOR TDBR
2271 002645 360      .BYTE 360      ;MASK FOR TDSR
2272 002646 000      .BYTE 000      ;MASK FOR SSAR
2273 002647 000      .BYTE 000      ;MASK FOR PCSAR
2274 002650 347      .BYTE 347      ;MASK FOR PCR
2275
2276 002651 200      TDSRNRW: .BYTE 200      ;TDSR NON-R/W BITS

```

DATA TEST PATTERNS

```

2278          .SBTTL DATA TEST PATTERNS
2279          ;***** DATA PATTERN E *****
2280 002652          PATE:
2281 002652          .BYTE 377
2282 002653          .BYTE 377
2283 002654          .BYTE 377
2284 002655          .BYTE 377
2285 002656          .BYTE 377
2286 002657          .BYTE 377
2287 002660          .BYTE 377
2288 002661          .BYTE 366
2289
2290          ;***** DATA PATTERN F *****
2291 002662          PATF:
2292 002662          .BYTE 000
2293 002663          .BYTE 000
2294 002664          .BYTE 000
2295 002665          .BYTE 000
2296 002666          .BYTE 000
2297 002667          .BYTE 000
2298 002670          .BYTE 000
2299 002671          .BYTE 110
2300
2301          ;***** DATA PATTERN G *****
2302 002672          PATG:
2303 002672          .BYTE 000
2304 002673          .BYTE 001
2305 002674          .BYTE 003
2306 002675          .BYTE 004
2307 002676          .BYTE 005
2308 002677          .BYTE 007
2309 002700          .BYTE 100
2310 002701          .BYTE 101
2311 002702          .BYTE 103
2312 002703          .BYTE 104
2313 002704          .BYTE 105
2314 002705          .BYTE 107
2315 002706          .BYTE 000
2316 002707          .BYTE 017
2317 002710          .BYTE 027
2318 002711          .BYTE 041
2319 002712          .BYTE 200
2320 002713          .BYTE 277
2321 002714          .BYTE 103
2322 002715          .BYTE 144
2323 002716          .BYTE 115
2324 002717          .BYTE 157
2325 002720          .BYTE 000
2326
2327          ;***** DATA PATTERN H *****
2328 002721          PATH:
2329 002721          .BYTE 125
2330 002722          .BYTE 252
2331 002723          .BYTE 000
2332 002724          .BYTE 377
2333 002725          .BYTE 000
2334 002726          .BYTE 001

```


DATA TEST PATTERNS

2335	002727	002	.BYTE	002
2336	002730	004	.BYTE	004
2337	002731	010	.BYTE	010
2338	002732	020	.BYTE	020
2339	002733	040	.BYTE	040
2340	002734	100	.BYTE	100
2341	002735	200	.BYTE	200
2342	002736	000	.BYTE	000
2343	002737	377	.BYTE	377
2344	002740	376	.BYTE	376
2345	002741	375	.BYTE	375
2346	002742	373	.BYTE	373
2347	002743	367	.BYTE	367
2348	002744	357	.BYTE	357
2349	002745	337	.BYTE	337
2350	002746	277	.BYTE	277
2351	002747	177	.BYTE	177
2352	002750	377	.BYTE	377
2353	002751	000	.BYTE	000

2354

2355

***** DATA PATTERN I *****
PATTI:

2356	002752	000	.BYTE	000
2357	002752	000	.BYTE	000
2358	002753	041	.BYTE	041
2359	002754	102	.BYTE	102
2360	002755	143	.BYTE	143
2361	002756	204	.BYTE	204
2362	002757	245	.BYTE	245
2363	002760	306	.BYTE	306
2364	002761	347	.BYTE	347
2365	002762	000	.BYTE	000
2366	002763	001	.BYTE	001
2367	002764	002	.BYTE	002
2368	002765	004	.BYTE	004
2369	002766	040	.BYTE	040
2370	002767	100	.BYTE	100
2371	002770	200	.BYTE	200
2372	002771	000	.BYTE	000
2373	002772	346	.BYTE	346
2374	002773	345	.BYTE	345
2375	002774	343	.BYTE	343
2376	002775	307	.BYTE	307
2377	002776	247	.BYTE	247
2378	002777	147	.BYTE	147
2379	003000	347	.BYTE	347
2380	003001	242	.BYTE	242
2381	003002	105	.BYTE	105
2382	003003	347	.BYTE	347
2383	003004	010	.BYTE	010
2384	003005	020	.BYTE	020
2385	003006	367	.BYTE	367
2386	003007	357	.BYTE	357
2387	003010	030	.BYTE	030
2388	003011	027	.BYTE	027
2389	003012	377	.BYTE	377

2390

2391

***** DATA PATTERN J *****

DATA TEST PATTERNS

2392	003013		PATJ:		
2393	003013	000		.BYTE	000
2394	003014	000		.BYTE	000
2395	003015	001		.BYTE	001
2396	003016	002		.BYTE	002
2397	003017	004		.BYTE	004
2398	003020	020		.BYTE	020
2399	003021	040		.BYTE	040
2400	003022	010		.BYTE	010

***** DATA PATTERN K *****

2401			PATK:		
2402					
2403	003023			.BYTE	000
2404	003023	000		.BYTE	000
2405	003024	377		.BYTE	377
2406	003025	376		.BYTE	376
2407	003026	375		.BYTE	375
2408	003027	373		.BYTE	373
2409	003030	376		.BYTE	376
2410	003031	177		.BYTE	177
2411	003032	377		.BYTE	377
2412	003033	000		.BYTE	000
2413	003034	001		.BYTE	001
2414	003035	002		.BYTE	002
2415	003036	004		.BYTE	004
2416	003037	010		.BYTE	010
2417	003040	200		.BYTE	200
2418	003041	125		.BYTE	125
2419	003042	252		.BYTE	252
2420	003043	000		.BYTE	000

***** DATA PATTERN L *****

2421			PATL:		
2422					
2423	003044			.BYTE	000
2424	003044	000		.BYTE	000
2425	003045	017		.BYTE	017
2426	003046	016		.BYTE	016
2427	003047	015		.BYTE	015
2428	003050	013		.BYTE	013
2429	003051	016		.BYTE	016
2430	003052	017		.BYTE	017
2431	003053	017		.BYTE	017
2432	003054	000		.BYTE	000
2433	003055	001		.BYTE	001
2434	003056	002		.BYTE	002
2435	003057	004		.BYTE	004
2436	003060	010		.BYTE	010
2437	003061	000		.BYTE	000
2438	003062	005		.BYTE	005
2439	003063	012		.BYTE	012
2440	003064	000		.BYTE	000

DATA TEST PATTERNS

2442
 2443
 2444 003065 000
 2445 003066 003
 2446 003067 014
 2447 003070 060
 2448 003071 001
 2449 003072 007
 2450 003073 037
 2451 003074 177
 2452
 2453
 2454 003075 000
 2455 003076 140
 2456 003077 030
 2457 003100 006
 2458 003101 100
 2459 003102 160
 2460 003103 174
 2461 003104 177
 2462 003105
 2463

***** DATA PATTERN Q *****

PATQ: .BYTE 000
 .BYTE 003
 .BYTE 014
 .BYTE 060
 .BYTE 001
 .BYTE 007
 .BYTE 037
 .BYTE 177

***** DATA PATTERN INVERTED Q *****

PATQB: .BYTE 000 ; INVERTED 000 (7 BIT)
 .BYTE 140 ; INVERTED 003 (7 BIT)
 .BYTE 030 ; INVERTED 014 (7 BIT)
 .BYTE 006 ; INVERTED 060 (7 BIT)
 .BYTE 100 ; INVERTED 001 (7 BIT)
 .BYTE 160 ; INVERTED 007 (7 BIT)
 .BYTE 174 ; INVERTED 037 (7 BIT)
 .BYTE 177 ; INVERTED 177 (7 BIT)

ENDPAT:
 .EVEN

DATA TEST PATTERNS

2465
2466
2467
2468
2469 003106
2470
2471
2472
2473

;*** RECEIVED DATA BUFFER (64. WORDS) ***
RCVBUF: .BLKW 64.

GLOBAL TEXT SECTION

```

2475          .SBTTL  GLOBAL TEXT SECTION
2476
2477          ;*****
2478          ;*      THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
2479          ;*      MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
2480          ;*      MORE THAN ONE TEST.
2481          ;*****
2482
2483          ;*****
2484          ;*      NAMES OF DEVICES SUPPORTED BY PROGRAM
2485          ;*****
2486          003306          DEVTYP  <M8053 OR M8064>
                003306          L#DVTYP::
                003306          115          070          060          .ASCIZ  #M8053 OR M8064#
                003311          065          063          040
                003314          117          122          040
                003317          115          070          060
                003322          066          064          000
                .EVEN
2487
2488          ;*****
2489          ;*      TITLE OF PROGRAM
2490          ;*****
2491
2492          000012          .RADIX  10.
2493          003326          DESCRIPT  <DMV-11 LINE UNIT TESTS - PART 1 OF 3>
                003326          L#DESC::
                003326          104          115          126          .ASCIZ  /DMV-11 LINE UNIT TE
STS - PART 1 OF 3/
                003331          055          061          061
                003334          040          114          111
                003337          116          105          040
                003342          125          116          111
                003345          124          040          124
                003350          105          123          124
                003353          123          040          055
                003356          040          120          101
                003361          122          124          040
                003364          061          040          117
                003367          106          040          063
                003372          000
                .EVEN
2494          000010          .RADIX  8.
2495
2496

```

GLOBAL SUBROUTINE SECTION

```

2504          ,SBTTL GLOBAL SUBROUTINE SECTION
2505
2506
2507
2508
2509
2510          ,SBTTL ....M-LOOP -- MSTCLR -- MASTER CLEAR AND ENTER M LOOP
2511          ;*****
2512          ; MSTCLR -- MASTER CLEAR & ENTER M-LOOP
2513          ;
2514          ; CALLING SEQUENCE:
2515          ;
2516          ;     JSR     PC,MSTCLR
2517          ;     BCC     N$          ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2518          ;     ERROR   ;AN ERROR MESSAGE HAS BEEN STACKED; PRINT IT
2519          ;     <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2520          ;
2521          ; N$:  <RESUMPTION OF NORMAL PROCESSING>
2522          ;
2523          ;-----*****
2524
2525 003574 112777 000301 177072 MSTCLR: MOVB  #RUN!MCLR!MREQ,#SEL1 ;INITIATE M-LOOP
2526
2527 003402 010346          MOV    R3,-(SP)
2528 003404 012703 000030          MOV    #24.,R3          ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2529 003410 077301          SOB   R3,1#
2530 003412 012603          MOV    (SP)+,R3
2531
2532 003414 132777 000200 177054          BITB  #MRDY,#SEL2      ;DID THE M-LOOP FINISH
2533 003422 001023          BNE   5#              ;YES, GOOD. RETURN
2534 003424 004737 004210          JSR   PC,GETWSR       ;GET BYTE SELECT REGISTERS
2535 003430 012737 000301 002400          MOV    #RUN!MCLR!MREQ,GDATA ;IDENTIFY REQUESTED FUNCTION
2536 003436          GTDF  EM3,ERR4      ;"MRDY" TIMEOUT
2537          ;          QUEUE "DEVICE FATAL" ERROR # 1
2538          ;          MOV    #T.EDF,ERRTYP
2539          ;          MOV    #1,ERRNBR
2540          ;          MOV    #EM3,ERRMSG
2541          ;          MOV    #ERR4,ERRBLK
2542
2543          ;SET CARRY TO INDICATE ERROR
2544          ;EXIT WITH THE "ERROR" FLAG (CARRY BIT) SET
2545          ;CLEAR C BIT FOR NO ERRORS
2546          ;RETURN
2547
2548          5#:  CLC
2549          9#:  RTS    PC

```

....M-LOOP -- READ

```

2546 .SBTTL ....M-LOOP -- READ
2547 ;*****
2548 ; READ - READ THE SPECIFIED ADDRESS WITHIN THE DMV 11 (M8053)
2549 ;
2550 ; CALLING SEQUENCE:
2551 ;
2552 ;     JSR     R5,READ
2553 ;     .WORD  <ADDRESS OF REGISTER WITHIN DMV-11>
2554 ;     .WORD  <DESTINATION ADDRESS WITHIN LSI-11>
2555 ;     BCC    N$ ; IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2556 ;     ERROR  ; AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2557 ;     <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2558 ;
2559 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2560 ;
2561 ;*****
2562
2563 003476 012577 177000 READ:  MOV     (R5)+,RSEL4 ;SETUP SOURCE POINTER
2564 003502 112777 000001 176766  MOVB   @REDLOC,RSEL2 ;TELL M-LOOP TO GIVE US THE REQUESTED DATA
2565
2566 003510 010346          MOV     R3,(SP)
2567 003512 012703 000050          MOV     @40,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2568 003516 077301          SOB    R3,1$
2569 003520 012603          MOV     (SP)+,R3
2570
2571 003522 132777 000200 176746  BITE   @MRDY,RSEL2 ;DID THE M-LOOP FINISH
2572 003530 001023          BNE    5$ ;YES, GOOD. RETURN
2573
2574 003532 004737 004210          JSR    PC,GETWSR ;GET BYTE SELECT REGISTERS
2575 003536 012737 000001 002400  MOV    @REDLOC,GDATA ;IDENTIFY REQUESTED FUNCTION
2576 003544          GTDF   EM4,ERR4 ;"MRDY" TIMEOUT
2577          ;          QUEUE "DEVICE FATAL" ERROR # 2
2578          MOV    @T,EDF,ERR1P
2579          MOV    @2,ERRNBR
2580          MOV    @EM4,ERRMSG
2581          MOV    @ERR4,ERRBLK
2582          SEC
2583          BR    6$ ;INDICATE AN ERROR HAS BEEN STACKED
2584          ;RETURN WITH THAT INDICATION
2585
2586          5$: CLC
2587          6$: MOVB  @RSEL6,@(R5)+ ;INDICATE "NO ERROR"
2588          RTS   R5 ;PUT DATA WHERE CALLER WANTS IT
2589          ;RETURN

```

....M-LOOP -- READ IMMEDIATE

```

2588          .SBTTL ....M-LOOP -- READ IMMEDIATE
2589          ;*****
2590          ; READI - READ IMMEDIATE THE SPECIFIED ADDRESS WITHIN THE DMV-11 (M8053)
2591          ;
2592          ; CALLING SEQUENCE:
2593          ;
2594          ;     JSR     R5,READI
2595          ;     .WORD  <ADDRESS OF REGISTER WITHIN DMV-11>
2596          ;     .WORD  <DESTINATION -- CONTENTS OF REG. IS PUT HERE>
2597          ;     BCC     N$          ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2598          ;     ERROR   ;AN ERROR MESSAGE HAS BEEN STACKED; PRINT IT
2599          ;     <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2600          ;
2601          ; N$:  <RESUMPTION OF NORMAL PROCESSING>
2602          ;
2603          ;-----*****
2604
2605          READI:
2606          003610 012577 176666      MOV     (R5)+,RSEL4      ;SETUP SOURCE POINTER
2607          003614 112777 000001 176654  MOVB   @REDLOC,RSEL2    ;TELL M-LOOP TO GIVE US THE REQUESTED DATA
2608
2609          003622 010346                MOV     R3,-(SP)
2610          003624 012703 000050        MOV     @40,R3          ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2611          003630 077301                SOB     R3,1$
2612          003632 012603                MOV     (SP)+,R3
2613
2614          003634 132777 000200 176634  BITB   @MRDY,RSEL2     ;DID THE M-LOOP FINISH
2615          003642 001023                BNE     5$             ;YES. GOOD. RETURN
2616
2617          003644 004737 004210        JSR     PC,GETWSR      ;GET BYTE SELECT REGISTERS
2618          003650 012737 000001 002400  MOV     @REDLOC,GDATA  ;IDENTIFY REQUESTED FUNCTION
2619          003656                GTDF   EM4,ERR4       ;"MRDY" TIMEOUT
2620
2621                ;     QUEUE "DEVICE FATAL" ERROR # 3
2622                ;
2623                ;     MOV     @T,EDF,ERRTYP
2624                ;     MOV     @3,ERRNBR
2625                ;     MOV     @EM4,ERRMSG
2626                ;     MOV     @ERR4,ERRBLK
2627
2628          003706 000261                SEC
2629          003710 000401                BR     6$             ;INDICATE AN ERROR HAS BEEN STACKED
2630
2631                ;RETURN WITH THAT INDICATION
2632
2633          003712 000241                5$:   CLC
2634          003714 017725 176566        6$:   MOV     @SEL6,(R5)+ ;INDICATE "NO ERROR"
2635          003720 000205                RTS    R5             ;PUT DATA WHERE CALLER WANTS IT
2636
2637                ;RETURN
2638
2639

```


....M-LOOP -- WRITE

2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654

003722 012577 176554
003726 113577 176554
003732 000404

```

.SBTTL ....M-LOOP -- WRITE
;*****
; WRITE - WRITE THE SPECIFIED DATA INTO THE SPECIFIED DMV-11 ADDRESS
;
; CALLING SEQUENCE:
;
;     JSR     R5,WRITE
;     .WORD  <ADDRESS OF REGISTER WITHIN DMV-11>
;     .WORD  <ADDRESS OF DATA BYTE>
;     BCC    N$                ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
;     ERROR  ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
;           <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
;
; N$:  <RESUMPTION OF NORMAL PROCESSING>
;
;*****
WRITE:  MOV     (R5)+,0SFL4      ;SETUP SOURCE POINTER
        MOVB   0(R5)+,0SEL6    ;MAKE DATA AVAILABLE TO M-LOOP
        BR     MLWRI           ;THE REST OF THIS ROUTINE IS THE SAME AS "WRITEI"
    
```

...M-LOOP -- WRITE IMMEDIATE

```

2656 .SBTTL ...M-LOOP -- WRITE IMMEDIATE
2657 ;*****
2658 ; WRITEI - WRITE IMMEDIATE THE SPECIFIED DATA INTO THE SPECIFIED DMV-11 ADDRESS
2659 ;
2660 ; CALLING SEQUENCE:
2661 ;
2662 ; JSR R5,WRITEI
2663 ; .WORD <ADDRESS OF REGISTER WITHIN DMV-11>
2664 ; .WORD <DATA FIELD -- DATA TO BE WRITTEN IN DMV-11>
2665 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2666 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2667 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2668 ;
2669 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2670 ;
2671 ;-----
2672
2673 003734 WRITEI:
2674 003734 012577 176542 MOV (R5)+,0SEL4 ;SETUP SOURCE POINTER
2675 003740 012577 176542 MOV (R5)+,0SEL6 ;MAKE DATA AVAILABLE TO M-LOOP
2676 003744 112777 000002 176524 MLWRI: MOVB 0WRILOC,0BSEL2 ;TELL M-LOOP TO WRITE THE DATA
2677
2678 003752 010346 MOV R3,-(SP)
2679 003754 012703 000050 MOV 040.,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2680 003760 077301 1$: SOB R3,1$
2681 003762 012603 MOV (SP)+,R3
2682
2683 003764 132777 000200 176504 BITB 0MRDY,0BSEL2 ;DID THE M-LOOP FINISH
2684 003772 001023 BNE 5$ ;YES, GOOD. RETURN
2685 003774 004737 004210 JSR PC,GETWSR ;GET BYTE SELECT REGISTERS
2686 004000 012737 000002 002400 MOV 0WRILOC,GDATA ;IDENTIFY REQUESTED FUNCTION
2687 004006 GTDF EM4,ERR4 ;"MRDY" TIME OUT
; QUEUE "DEVICE FATAL" ERROR 4
MOV 0T,EDF,ERR4TYP
MOV 04,ERRNBR
MOV 0EM4,ERRMSG
MOV 0ERR4,ERRBLK
2688 004036 000261 SEC ;INDICATE AN ERROR HAS BEEN STACKED
2689 004040 000401 BR 6$ ;RETURN WITH THAT INDICATION
2690
2691 004042 000241 5$: CLC ;INDICATE "NO ERROR"
2692 004044 000205 6$: RTS R5 ;RETURN
2693
2694
2695
2696

```

....GETBSR -- GET BYTE SELECT REGISTERS

2693
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713

```
.SBTTL ....GETBSR -- GET BYTE SELECT REGISTERS
:*****
:
:   GET THE CONTENTS OF ALL CONTROL AND STATUS REGISTERS
:
:   FUNCTION - THIS SUBROUTINE COLLECTS THE CONTENTS OF THE
:             BYTE SELECT REGISTERS FOR THE PURPOSE OF DISPLAY.
:
:   ENTRY CONDITIONS - NONE      00 0 0000 0 00 0
:   EXIT CONDITIONS - NONE      0 0 0 0 0 0 0 0
:   REGISTERS DESTROYED - NONE  00 0000 0000 0 0 0 0
:*****
```

2714 004046 117737 176420 002256
2715 004054 117737 176414 002260
2716 004062 117737 176410 002262
2717 004070 117737 176404 002264
2718 004076 117737 176400 002266
2719 004104 117737 176374 002270
2720 004112 117737 176370 002272
2721 004120 117737 176364 002274
2722 004126 117737 176360 002276
2723 004134 117737 176354 002300
2724 004142 117737 176350 002302
2725 004150 117737 176344 002304
2726 004156 117737 176340 002306
2727 004164 117737 176334 002310
2728 004172 117737 176330 002312
2729 004200 117737 176324 002314
2730 004206 000207

```
GETBSR: MOVB  @BSSEL0,BSR0      ;PUT THE CURRENT CSR VALUES INTO THE PRINT-OUT
:          MOVB  @BSSEL1,BSR1      ;TABLE
:          MOVB  @BSSEL2,BSR2
:          MOVB  @BSSEL3,BSR3
:          MOVB  @BSSEL4,BSR4
:          MOVB  @BSSEL5,BSR5
:          MOVB  @BSSEL6,BSR6
:          MOVB  @BSSEL7,BSR7
:          MOVB  @BSSEL10,BSR10
:          MOVB  @BSSEL11,BSR11
:          MOVB  @BSSEL12,BSR12
:          MOVB  @BSSEL13,BSR13
:          MOVB  @BSSEL14,BSR14
:          MOVB  @BSSEL15,BSR15
:          MOVB  @BSSEL16,BSR16
:          MOVB  @BSSEL17,BSR17
:          RTS    PC              ;RETURN TO CALLER
```

2731
2732
2733
2734

```
.SBTTL ....GETWSR -- GET WORD SELECT REGISTERS
; "WORD" VERSION OF ABOVE SUBROUTINE
```

2735 004210 017737 176256 002256
2736 004216 017737 176254 002260
2737 004224 017737 176252 002262
2738 004232 017737 176250 002264
2739 004240 017737 176246 002266
2740 004246 017737 176244 002270
2741 004254 017737 176242 002272
2742 004262 017737 176240 002274
2743 004270 000207

```
GETWSR: MOV   @WSEL0,WSR0      ;MOVE THE 4 WORD REGISTERS TO THE OTHERWISE
:           MOV   @WSEL2,WSR2      ;BYTE TABLE
:           MOV   @WSEL4,WSR4
:           MOV   @WSEL6,WSR6
:           MOV   @WSEL10,WSR10
:           MOV   @WSEL12,WSR12
:           MOV   @WSEL14,WSR14
:           MOV   @WSEL16,WSR16
:           RTS    PC              ;RETURN TO CALLER
```

...STUREG -- STATIC TEST OF SPECIFIED USYRT REGISTER

```

2745 .SBTTL ....STUREG -- STATIC TEST OF SPECIFIED USYRT REGISTER
2746 ;*****
2747 ; STUREG -- PERFORM A STATIC TEST OF THE SPECIFIED USYRT REGISTER
2748 ;
2749 ; CALLING SEQUENCE:
2750 ;
2751 ; <R0 CONTAINS THE ADDRESS OF THE REGISTER TO BE TESTED>
2752 ; <"TDATA" CONTAINS THE TEST BYTE>
2753 ; <"GDATA" CONTAINS THE EXPECTED DATA>
2754 ; <"REGNUM" CONTAINS REG INDEX FOR POSSIBLE ERRORS>
2755 ;
2756 ; JSR PC,STUREG
2757 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2758 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED; PRINT IT
2759 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2760 ;
2761 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2762 ;
2763 ;-----*****
2764
2765 004272 010037 004306 STUREG: MOV R0,2$ ;PUT SPECIFIED REGISTER'S ADDRESS IN I/O CALLS
2766 004276 010037 004324 MOV R0,4$
2767
2768 004302 004537 003722 JSR R5,WRITE ;WRITE IT
2769 004306 000000 2$: .WORD 0 ;*** MODIFIED FROM ABOVE ***
2770 004310 002376 .WORD TDATA ;
2771 004312 103431 BCS 10$ ;ON ERROR, EXIT
2772
2773 004314 005037 002402 CLR BDATA ;CLEAR BOTH BYTES -- JUST IN CASE....
2774 004320 004537 003476 JSR R5,READ ;READ IT BACK AGAIN
2775 004324 000000 4$: .WORD 0 ;*** MODIFIED FROM ABOVE ***
2776 004326 002402 .WORD BDATA ;
2777 004330 103422 BCS 10$ ;ON ERROR, EXIT
2778
2779 004332 123737 002400 002402 CMPB GDATA,BDATA ;DID WE READ WHAT WE WROTE?
2780 004340 000241 CLC ; (THIS ISN'T NEEDED FOR THE ERROR TEST BUT
2781 ; MUST BE CLEARED ON EXIT IF NO ERROR OCCURED)
2782 004342 001415 BEQ 10$ ;YES, EXIT FROM SUBTEST
2783 004344 GTDF EM25,ERR7A ;REPORT READ/WRITE ERROR
; QUEUE "DEVICE FATAL" ERROR # 5
; MOV @T.EDF,ERRTYP
; MOV @5,ERRNBR
; MOV @EM25,ERRMSG
; MOV @ERR7A,ERRBLK
004344 012737 000001 002246
004352 012737 000005 002250
004360 012737 013366 002252
004366 012737 017252 002254
2784 004374 000261 10$: SEC ;INDICATE THAT AN ERROR WAS DETECTED
2785 004376 000207 RTS PC
2786
2787
2788 .SBTTL ....STALL - DELAY FOR 10.5 MICRO-SEC'S (ON LSI-11)
2789 ;*****
2790 ; STALL -- THIS SUBROUTINE STALLS FOR ABOUT 10.5 MICRO-SECONDS
2791 ;-----*****
2792
2793 004400 000207 STALL: RTS PC

```

```

2795          .SBTTL
2796
2797          ;*****
2798          ;* GETURS - LOAD INTO THE 8 WORD STORAGE AREA (UREGS) THE CONTENTS OF THE
2799          ;*   VARIOUS USYRT REGISTERS
2800          ;*
2801          ;*   CALLING SEQUENCE:
2802          ;*
2803          ;*****
2804 004402 012737 002316 004444 GETURS: MOV    @UREGS,5$      ;INIT POINTER TO REG STORAGE TABLE
2805 004410 012737 120400 004442      MOV    @USYRT,4$      ;INIT POINTER TO REGISTER ADDRESSES
2806
2807 004416 005037 002334          CLR    UREGS+14.      ;CLEAR STORAGE WORD
2808 004422 004537 003476          JSR    R5,READ        ;READ THE USYRT STATUS REGISTER
2809 004426 122000          .WORD  USTATR        ;STATUS REGISTER'S ADDRESS WITHIN DMV-11
2810 004430 002334          .WORD  UREGS+14.      ;ADDRESS ALLOCATED TO THAT REG. W/IN "UREGS"
2811
2812 004432 005077 000006 3$:   CLR    @5$          ;CLEAR STORAGE WORD
2813 004436 004537 003476          JSR    R5,READ        ;READ A LINE UNIT REG
2814 004442 000000          .WORD  0              ;REGISTER ADDRESS GOES HERE
2815 004444 000000          .WORD  0              ;STORAGE ADRS IN TABLE GOES HERE
2816
2817 004446 005237 004442 6$:   INC    4$            ;INCREMENT REG NO.
2818 004452 023727 004442 120406  CMP    4$,@USYRT-6     ;THIS IS NOT A VALID REGISTER ADDRESS
2819 004460 001772          BEQ    6$            ;SO IT MUST BE BYPASSED
2820
2821 004462 062737 000002 004444  ADD    @2,5$          ;ADVANCE ADDRESS OF STORAGE AREA POINTER
2822 004470 023727 004442 120410  CMP    4$,@USYRT+10    ;SEE IF ALL REGS READ YET
2823 004476 001355          BNE    3$            ;BR IF NOT
2824
2825 004500 000207          RTS     PC            ;RETURN
2826
2827
2828
2829          ;*****
2830          ;* GETVRS: - LOAD INTO THE 16 WORD STORAGE AREA (VREGS) THE CONTENTS OF THE
2831          ;*   VARIOUS VIA REGISTERS.
2832          ;*
2833          ;*   CALLING SEQUENCE:
2834          ;*
2835          ;*****
2835 004502 012737 002336 004530 GETVRS: MOV    @VREGS,5$  ;INIT POINTER TO REG STORAGE TABLE
2836 004510 012737 120000 004526      MOV    @VIA,4$        ;INIT POINTER TO REGISTER ADDRESSES
2837 004516 005077 000006          3$:   CLR    @5$          ;CLEAR STORAGE WORD
2838 004522 004537 003476          JSR    R5,READ        ;READ A VIA REG
2839 004526 000000          4$:   .WORD  0              ;REGISTER ADDRESS GOES HERE
2840 004530 000000          5$:   .WORD  0              ;STORAGE ADRS IN TABLE GOES HERE
2841 004532 005237 004526          6$:   INC    4$            ;INCREMENT REG NO.
2842 004536 062737 000002 004530  ADD    @2,5$          ;INCREMENT STORAGE ADRS
2843 004544 023727 004526 120020  CMP    4$,@VIA+16.    ;SEE IF ALL VIA REGS READ YET
2844 004552 001361          BNE    3$            ;BR IF NOT
2845 004554 000207          RTS     PC            ;RETURN

```

....INITT1 -- INITIALIZE TIMER #1

```

2847 .SBITL ....INITT1 -- INITIALIZE TIMER #1
2848 :*****
2849 :* INITT1 - INITIALIZE TIMER # 1
2850 :*
2851 :*          CALLING SEQUENCE:
2852 :*
2853 :*          JSR      R5,INITT1
2854 :*          .WORD   <VALUE LOADED INTO THE T1 LATCH @ VIAT1C & VIAT1D>
2855 :*          .WORD   <VALUE LOADED INTO "T1L-L" & "T1C-H">
2856 :*          .BYTE   <BITS 6 & 7 WILL BE LOADED INTO "ACR", BIT 5 WILL BE
2857 :*                   USED TO SET OR CLEAR BIT 6 ("T1") OF THE INTERRUPT
2858 :*                   ENABLE REGISTER ("IER")>
2859 :*          .BYTE   <UNUSED>
2860 :*
2861 :*
2862 :* NOTE:
2863 :*
2864 :* BEFORE LOADING AND STARTING THE COUNTER, THE LATCH REGISTER (ACCESSED THRU
2865 :* "VIAT1C") IS LOADED. THEN, T1L-L IS LOADED AND NEXT, T1C-H. THIS LAST
2866 :* LOAD WILL RESET THE TIMEOUT BIT AND COUNTER LOGIC. IT IS EXPECTED AT THIS
2867 :* TIME (5/25/79) THAT THE INTERRUPT FACILITY OF THE VIA CHIP WILL NOT BE USED
2868 :* -- HOWEVER, ACCESS TO THE INTERRUPT ENABLE BIT IS GIVEN THROUGH THE THIRD
2869 :* PARAMETER IN THE CALLING SEQUENCE (BIT 5 - 0 WILL CAUSE THIS ROUTINE TO
2870 :* CLEAR THE ENABLE BIT ("T1") IN "IER".)
2871 :*
2872 :*****
2873
2874 004556 010146          INITT1: MOV      R1,-(SP)          ;SAVE THE REGISTER WE WILL BE USING
2875 004560 012537 004702      MOV      (R5)+,7$        ;SETUP VALUE TO BE WRITTEN IN LATCH
2876 004564 012537 004730      MOV      (R5)+,10$       ;SETUP VALUE TO BE WRITTEN IN COUNTER
2877 004570 111501          MOVVB   (R5),R1          ;GET & PROCESS BITS FOR ACR 6 & 7
2878 004572 143701 000077      BICB   077,R1
2879 004576 010137 004672      MOV      R1,4$
2880 004602 112501          MOVVB   (R5)+,R1
2881
2882 004604 106301          ASLB   R1
2883 004606 106301          ASLB   R1
2884
2885
2886
2887
2888 004610 143701 000177      BICB   177,R1
2889 004614 153701 000100      BISB   100,R1
2890 004620 010137 004632      MOV      R1,2$
2891
2892 004624 004537 003734          JSR      R5,WRITEI
2893 004630 120016          VIAIER
2894 004632 000000          2$: .WORD  0
2895
2896 004634 004537 003610          JSR      R5,READI
2897 004640 120013          VIAACR
2898 004642 000000          3$: .WORD  0
2899
2900 004644 013701 004642          MOV      3$,R1
2901 004650 143701 000300      BICB   300,R1
2902 004654 053701 004672      BIS    4$,R1
2903 004660 010137 004672      MOV      R1,4$
;GET THAT VALUE
;CLEAR THE CURRENT SETTING OF BITS 6 & 7
;SET THEM ACCORDING TO THE PASSED VALUES
;PASS THE NEW REG. SETTING TO APPROPRIATE CALL

```

....INITI1 -- INITIALIZE TIMER #1

```

2904
2905 004664 004537 003734      JSR    R5,WRITEI      ;WRITE TO
2906 004670 120013              VIAACR                ;THE VIA'S ACR
2907 004672 000000      4$:  .WORD    0      ;THE NEW REGISTER SETTING
2908
2909 004674 004537 003734      JSR    R5,WRITEI      ;WRITE TO
2910 004700 120006              VIAT1C                ;LOW ORDER LATCH REGISTER (T1L-L)
2911 004702 000000      7$:  .WORD    0      ;THE VALUE PASSED
2912
2913 004704 113737 004703 004720  MOVB   7$+1,8$        ;SETUP FOR AND
2914 004712 004537 003734      JSR    R5,WRITEI      ;WRITE TO
2915 004716 120007              VIAT1D                ;HIGH ORDER LATCH REGISTER (T1L-H)
2916 004720 000000      8$:  .WORD    0      ;THE VALUE PASSED
2917
2918 004722 004537 003734      JSR    R5,WRITEI      ;WRITE TO
2919 004726 120004              VIAT1A                ;LOW ORDER LATCH & COUNTER (T1L-L & T1C-L)
2920 004730 000000      10$: .WORD    0      ;THE VALUE PASSED
2921
2922 004732 113737 004731 004746  MOVB   10$+1,11$     ;SETUP FOR AND
2923 004740 004537 003734      JSR    R5,WRITEI      ;WRITE TO
2924 004744 120005              VIAT1B                ;HIGH ORDER COUNTER (T1C-H) <ALSO STARTS CTR>
2925 004746 000000      11$: .WORD    0      ;THE VALUE PASSED
2926
2927      ; DON'T WAIT AROUND FOR ANYTHING TO HAPPEN -- JUST RETURN!
2928
2929 004750 012601              MOV    (SP)+,R1      ;BUT FIRST RESTORE R1
2930 004752 005205              INC    R5             ;AND PUT R5 BACK ON A WORD BOUNDARY (THE LAST
2931      ;PASSED PARAM. WAS A BYTE, NOT A WORD!)
2932
2933 004754 000205              RTS    R5             ;NOW, RETURN
2934
2935

```


....INITT2 -- INITIALIZE TIMER #2

```

2994 005066 120013          VIAACR          ;THE VIA'S ACR
2995 005070 000000          4$: .WORD 0          ;THE NEW REGISTER SETTING
2996
2997 005072 004537 003734    JSR R5,WRITEI      ;WRITE TO
2998 005076 120010          VIAT2A          ;LOW ORDER LATCH & COUNTER (T2L-L & T2C-L)
2999 005100 000000          10$: .WORD 0      ;THE VALUE PASSED
3000
3001 005102 113737 005101 005116  MOVB 10$,1,11$    ;SETUP FOR AND
3002 005110 004537 003734    JSR R5,WRITEI      ;WRITE TO
3003 005114 120011          VIAT2B          ;HIGH ORDER COUNTER (T2C-H) <ALSO STARTS CTR>
3004 005116 000000          11$: .WORD 0      ;THE VALUE PASSED
3005
3006          ; DON'T WAIT AROUND FOR ANYTHING TO HAPPEN -- JUST (JEST) RETURN!
3007
3008 005120 012601          MOV (SP)+,R1      ;BUT FIRST RESTORE R1
3009 005122 005205          INC R5           ;AND PUT R5 BACK ON A WORD BOUNDARY (THE LAST
3010                                     ;PASSED PARAM. WAS A BYTE, NOT A WORD!)
3011
3012 005124 000205          RTS R5           ;THEN RETURN
3013

```

...RSTCHK -- RESET USYRT/VERIFY ALL USYRT REGS @ RESET STATE

```

3015 .SBTTL .. RSTCHK -- RESET USYRT/VERIFY ALL USYRT REGS @ RESET STATE
3016 ;*****
3017 ; RSTCHK - MANUALLY RESET THE USYRT AND VERIFY THAT ALL USYRT REGISTERS
3018 ; ARE IN THEIR RESET STATE. AN ERROR MESSAGE IDENTIFYING THE
3019 ; FAILING REGISTER IS STACKED IF ONE IS ENCOUNTERED.
3020 ;
3021 ; CALLING SEQUENCE:
3022 ; JSR R5,RSTCHK
3023 ;*****
3024
3025 RSTCHK:
3026 005126 010146 MOV R1,-(SP) ;SAVE R1
3027 005130 010246 MOV R2,-(SP) ;SAVE R2
3028
3029 005132 004537 003734 JSR R5,WRITEI ;SET PROGRAM RESET BIT IN VIA ORB REG
3030 005136 120000 VIAORB
3031 005140 000031 DTR!RTSND!PRESET
3032 005142 004537 003734 JSR R5,WRITET ;CLEAR PROGRAM RESET BIT IN VIA ORB REG
3033 005146 120000 VIAORB
3034 005150 000030 DTR!RTSND
3035
3036 005152 005001 CLR R1 ;INIT USYRT REG ADRS PTR
3037 005154 012702 002662 MOV #PATF,62 ;INIT DATA PATTERN POINTER
3038 005160 016137 002552 005172 6#: MOV USYREG(R1),7# ;SET USYRT READ ADDRESS
3039 005166 004537 003610 JSR R5,READI ;READ A USYRT REG
3040 005172 000000 7#: .WORD 0 ;USYRT REG ADRS GOES HERE
3041 005174 000000 8#: .WORD 0 ;DATA READ IS RETURNED HERE
3042 005176 125722 005174 CMPB 8#,(R2)+ ;SEE IF REG CONTAINS EXPECTED DATA
3043 005202 001432 BEQ 9# ;BR IF MATCH
3044
3045 005204 010137 002412 MOV R1,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
3046 005210 006237 002412 ASR REGNUM ;GET WORD OFFSET
3047 005214 005037 002400 CLR GDATA ;GET EXPECTED DATA
3048 005220 116237 177777 002400 MOVB -1(R2),GDATA
3049 005226 013737 005174 002402 MOV 8#,BDATA ;GET ACTUAL DATA
3050 ;STACK "USYRT NOT CLEARED BY PROGRAM RESET" MSG
3051 005234 GTDF EM2,ERR10
;
; QUEUE "DEVICE FATAL" ERROR # 6
;
; T.EDF,ERRTYP
; 6,ERRNBR
; EM2,ERRMSG,
; ERR10,ERRBLK
;
3052 005234 012737 000001 002246 MOV
3053 005242 012737 000006 002250 MOV
3054 005250 012737 013142 002252 MOV
3055 005256 012737 017372 002254 MOV
3056 005264 000261 SEC
3057 005266 000406 BR 10# ;SET C BIT TO FLAG ERROR
;TAKE ERROR EXIT
3058
3059 005270 062701 000002 9#: ADD #2,R1 ;INCR USYRT REG ADRS PTR
3060 005274 020127 000020 CMP R1,#16. ;SEE IF ALL REGS READ YET
3061 005300 002727 BLT 6# ;BR IF NOT
3062 005302 000241 CLC ;** CLEAR C BIT FOR NO ERRORS
3063 005304 012602 10#: MOV (SP)+,R2 ;RESTORE R2
3064 005306 012601 MOV (SP)+,R1 ;RESTORE R1
3065 005310 000205 RTS R5 ;** RETURN
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100

```

...RSTCHK -- RESET USYRT/VERIFY ALL USYRT REGS & RESET STATE

```

3065
3066
3067
3068 005312 010146
3069 005314 012701 000005
3070 005320 077101
3071 005322 012601
3072 005324 000207
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091 005326
3092 005326 004537 003734
3093 005332 120002
3094 005334 000377
3095 005336 004537 003734
3096 005342 120003
3097 005344 000001
3098 005346 004537 003734
3099 005352 120017
3100 005354 000000
3101 005356 004537 003734
3102 005362 120000
3103 005364 000030
3104 005366 004537 003734
3105 005372 120013
3106 005374 000350
3107 005376 004537 003734
3108 005402 120014
3109 005404 000022
3110 005406 004537 003734
3111 005412 120016
3112 005414 000177
3113 005416 000207
3114
3115

;*****
;* WAIT50 THIS SUBROUTINE STALLS FOR AT LEAST 50 MICRO-SEC, AND THEN RETURNS.
;*****
WAIT50: MOV R1, (SP) ;SAVE R1
        MOV #5,R1 ;INIT COUNTER
3068: SOB R1,#4 ;DELAY HERE FOR 25.8 MICRO-SEC'S
        MOV (SP)+,R1 ;RESTORE R1
        RTS PC ;RETURN

; OVERHEAD (JSR, MOV, MOV, MOV, & RTS) ADD UP TO 25.25 MICRO-SEC'S
; THEREFORE, ACTUAL TOTAL DELAY IS 49.35 MICRO-SECONDS

.SBTTL ....SETVIA -- SET UP VIA REGISTERS
;*****
;* SETVIA - SET UP THE VIA REGISTERS
;*
;* THIS SUBROUTINE PROGRAMS THE VIA REGISTERS FOR NORMAL OPERATION, BY
;* LOADING THE DDRB, DDRA, ORB, ACR, PCR, IER.
;*
;* CALLING SEQUENCE :
;* JSR PC,SETVIA
;*****
SETVIA: JSR R5,WRITEI ;SET PORT B FOR OUTPUT MODE
        VIAOPB
        377
        JSR R5,WRITEI ;SET PORT A FOR INPUT MODE
        VIADPA ; (BIT0 IS ONLY OUTPUT BIT)
        001
        JSR R5,WRITEI ;DISABLE USYRT INTERNAL LOOPBACK
        VIAORA
        000
        JSR R5,WRITEI ;INIT PORT B
        VIAORB
        DTR!RTSND
        JSR R5,WRITEI ;SET ACR FOR : T1 SQUARE WAVE OUTPUT MODE,
        VIAACR ; T2 ONE-SHOT OUTPUT MODE,
        350 ; SR AT SYS CLOCK RATE ON CB1
        JSR R5,WRITEI ;SET PCR FOR : CB1 NEG TRANS INPUT MODE,
        VIAPCR ; CA2 NEG TRANS INPUT MODE,
        022 ; CA1 NEG TRANS INPUT MODE
        JSR R5,WRITEI ;DISABLE ALL MICRO-INTRPTS
        VIAIER
        177
        RTS PC ;RETURN
    
```

(D)

....INIDMV -- INIT DMV (MCLR, VIA SETUP)

```

3117 .SBTTL ....INIDMV -- INIT DMV (MCLR, VIA SETUP)
3118 ;*****
3119 ;* INIDMV - THIS SUBROUTINE INITIALIZES THE DMV-11, BY DOING A MASTER CLEAR,
3120 ;* ENTERING THE M-LOOP, AND PROGRAMMING THE VIA REGS FOR DEFAULT
3121 ;* OPERATION.
3122 ;*
3123 ;* CALLING SEQUENCE :
3124 ;* JSR PC,INIDMV
3125 ;*****
3126 005420 004737 003374 INIDMV: JSR PC,MSTCLR ;MASTER CLR, M-LOOP
3127 005424 004737 005326 JSR PC,SETVIA ;PROGRAM VIA
3128 005430 000207 RTS PC ;RETURN
3129
3130
3131
3132
3133 .SBTTL ....CKUSTS -- CHECK USYRT STATUS REGISTERS
3134 ;*****
3135 ;* CKUSTS - THIS SUBROUTINE CHECKS THE USYRT STATUS BY READING THE USYRT
3136 ;* STATUS REGISTER AND COMPARING IT TO THE LOW BYTE OF THE WORD FOLLOWING
3137 ;* THE CALL. IF THERE IS A MISMATCH, THE SUBROUTINE STACKS THE ERROR
3138 ;* INFORMATION, AND SETS THE "C" BIT AND RETURNS.
3139 ;*****
3140 CKUSTS:
3141 005432 004537 003610 JSR R5,READI ;READ USYRT STATUS REGISTER
3142 005436 122000 USTATR
3143 005440 000000 1$: .WORD 0
3144 005442 122537 005440 CMPB (R5),1$ ;SEE IF STATUS MATCHES EXPECTED
3145 005446 000241 CLC ;CLEAR C BIT
3146 005450 001430 BEQ 2$ ;BI: IF STATUS OK
3147 005452 012737 000007 002412 MOV #7,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
3148 005460 016537 177777 002400 MOV -1(R5),GDATA ;GET EXPECTED DATA
3149 005466 005037 002402 CLR BDATA ;GET ACTUAL DATA
3150 005472 113737 005440 002402 MOVB 1$,BDATA
3151 ;STACK "USYRT STATUS INCORRECT" ERROR
3152 005500 GDF EM68,ERR10
; QUEUE "DEVICE FATAL" ERROR # 7
MOV #1,EDF,ERRTYP
MOV #7,ERRNR
MOV #EM68,ERRMSG
MOV #ERR10,ERRBLK
3153 005530 000261 2$: SEC ;SET C BIT FOR ERROR
3154 005532 005205 INC R5 ;INCREMENT R5 PAST ARGUMENT
3155 005534 000205 RTS R5 ;RETURN
3156
3157
3158
3159

```

....CKTACT -- CHECK TRANSMITTER ACTIVE (TXACT)

```

3161 .SBTTL ....CKTACT -- CHECK TRANSMITTER ACTIVE (TXACT)
3162 ;*****
3163 ;* CKTACT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TXACT IN THE USYRT
3164 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
3165 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3166 ;*
3167 ;* CALLING SEQUENCE :
3168 ;* JSR R5,CKTACT
3169 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TXACT>
3170 ;*****
3171 005536 CKTACT:
3172 005536 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3173 005544 004537 003610 JSR R5,READI ;READ USYRT STATUS
3174 005550 122000 USTATR
3175 005552 000000 1$: .WORD 0
3176 005554 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TXACT
3177 005560 001422 BEQ 2$ ;BR IF EXPECTED TXACT = 0
3178 005562 132737 000004 005552 BITB #TXACT,1$ ;SEE IF TXACT = 1
3179 005570 001040 BNE 3$ ;BR IF TXACT = 1
3180 ;STACK "TXACT NOT SET" MSG
3181 005572 GTDF EM69,ERR12
; QUEUE "DEVICE FATAL" ERROR # 8
; MOV #T.EDF,ERRTYP
; MOV #8,ERRNBR
; MOV #EM69,ERRMSG
; MOV #ERR12,ERRBLK
005572 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
005600 012737 000010 002250 BR 4$ ;TAKE ERROR EXIT
005606 012737 014007 002252 2$: BITB #TXACT,1$ ;SEE IF TXACT = 0
005614 012737 017722 002254 BEQ 3$ ;BR IF TXACT = 0
3182 005622 000261 ;STACK "TXACT NOT CLEARED" MSG
3183 005624 000423 GTDF EM70,ERR12
; QUEUE "DEVICE FATAL" ERROR # 9
; MOV #T.EDF,ERRTYP
; MOV #9,ERRNBR
; MOV #EM70,ERRMSG
; MOV #ERR12,ERRBLK
005636 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
005644 012737 000011 002250 BR 4$ ;TAKE ERROR EXIT
005652 012737 014025 002252 3$: CLC ;CLEAR C BIT FOR NO ERRORS
005660 012737 017722 002254 4$: RTS R5 ;RETURN
3188 005666 000261
3189 005670 000401
3190 005672 000241
3191 005674 000205
3192
3193
3194
3195

```

....CKRACT -- CHECK RECEIVER ACTIVE (RXACT)

```

3197 .SBTTL ....CKRACT -- CHECK RECEIVER ACTIVE (RXACT)
3198 ;*****
3199 ;* CKRACT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RXACT IN THE USYRT
3200 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
3201 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3202 ;*
3203 ;* CALLING SEQUENCE :
3204 ;* JSR R5,CKRACT
3205 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RXACT>
3206 ;*****
3207 CKRACT:
3208 005676 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3209 005704 004537 003610 JSR R5,READI ;READ USYRT STATUS
3210 005710 122000 USTATR
3211 005712 000000 1$: .WORD 0
3212 005714 032725 000001 BIT #BIT0,(R5)+ ;GET EXPECTED STATE OF RXACT
3213 005720 001422 BEQ 2$ ;BR IF EXPECTED RXACT = 0
3214 005722 132737 000040 005712 BITB #RXACT,1$ ;SEE IF RXACT = 1
3215 005730 001040 BNE 3$ ;BR IF RXACT = 1
3216 ;STACK "RXACT NOT SET" MSG
3217 005732 GTDF EM71,ERR12
; QUEUE "DEVICE FATAL" ERROR # 10
MOV #T.EDF,ERRTYP
MOV #10,ERRNBR
MOV #EM71,ERRMSG
MOV #ERR12,ERRBLK
3218 005762 000261 SEC ;SET C BIT TO FLAG ERROR
3219 005764 000423 BR 4$ ;TAKE ERROR EXIT
3220 005766 132737 000040 005712 2$: BITB #RXACT,1$ ;SEE IF RXACT = 0
3221 005774 001416 BEQ 3$ ;BR IF RXACT = 0
3222 ;STACK "RXACT NOT CLEARED" MSG
3223 005776 GTDF EM72,ERR12
; QUEUE "DEVICE FATAL" ERROR # 11
MOV #T.EDF,ERRTYP
MOV #11,ERRNBR
MOV #EM72,ERRMSG
MOV #ERR12,ERRBLK
3224 006026 000261 SEC ;SET C BIT TO FLAG ERROR
3225 006030 000401 BR 4$ ;TAKE ERROR EXIT
3226 006032 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3227 006034 000205 4$: RTS R5 ;RETURN
3228
3229
3230
3231

```

...CKTBMT -- CHECK TRANSMIT BUFFER EMPTY

```

3233 .SBTTL ...CKTBMT -- CHECK TRANSMIT BUFFER EMPTY
3234 ;*****
3235 ;* CKTBMT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TBMT IN THE USYRT
3236 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
3237 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3238 ;*
3239 ;* CALLING SEQUENCE :
3240 ;* JSR R5,CKTBMT
3241 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TBMT>
3242 ;*****
3243 006036 CKTBMT:
3244 006036 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3245 006044 004537 003610 JSR R5,READI ;READ USYRT STATUS
3246 006050 122000 USTATR
3247 006052 000000 1$: .WORD 0
3248 006054 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TBMT
3249 006060 001422 BEQ 2$ ;BR IF EXPECTED TBMT = 0
3250 006062 132737 000100 006052 BITB #TBMT,1$ ;SEE IF TBMT = 1
3251 006070 001040 BNE 3$ ;BR IF TBMT = 1
3252 ;STACK "TBMT NOT SET" MSG
3253 006072 GTDF EM73,ERR12
; QUEUE "DEVICE FATAL" ERROR # 12
; MOV #T.EDF,ERRTYP
; MOV #12,ERRNBR
; MOV #EM73,ERRMSG
; MOV #ERR12,ERRBLK
006072 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
006100 012737 000014 002250 BR 4$ ;TAKE ERROR EXIT
006106 012737 014107 002252 2$: BITB #TBMT,1$ ;SEE IF TBMT = 0
006114 012737 017722 002254 BEQ 3$ ;BR IF TBMT = 0
3254 006122 000261 ;STACK "TBMT NOT CLEARED" MSG
3255 006124 000423 GTDF EM74,ERR12
3256 006126 132737 000100 006052 3$: SEC ;SET C BIT TO FLAG ERROR
3257 006134 001416 BR 4$ ;TAKE ERROR EXIT
3258 4$: CLC ;CLEAR C BIT FOR NO ERRORS
3259 006136 RTS R5 ;RETURN
006136 012737 000001 002246 MOV #T.EDF,ERRTYP
006144 012737 000015 002250 MOV #13,ERRNBR
006152 012737 014124 002252 MOV #EM74,ERRMSG
006160 012737 017722 002254 MOV #ERR12,ERRBLK
3260 006166 000261 SEC ;SET C BIT TO FLAG ERROR
3261 006170 000401 BR 4$ ;TAKE ERROR EXIT
3262 006172 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3263 006174 000205 4$: RTS R5 ;RETURN
3264
3265
3266
3267

```

....CKRDA -- CHECK RECEIVE DATA AVAILABLE

```

3269 .SBTTL ....CKRDA -- CHECK RECEIVE DATA AVAILABLE
3270 ;*****
3271 ;* CKRDA - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RDA IN THE USYRT
3272 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
3273 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3274 ;*
3275 ;* CALLING SEQUENCE :
3276 ;* JSR R5,CKRDA
3277 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RDA>
3278 ;*****
3279 006176 CKRDA:
3280 006176 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3281 006204 004537 003610 JSR R5,READI ;READ USYRT STATUS
3282 006210 122000 USTATR
3283 006212 000000 1$: .WORD 0
3284 006214 032725 000001 BIT #BIT0,(R5)+ ;GET EXPECTED STATE OF RDA
3285 006220 001422 BEQ 2$ ;BR IF EXPECTED RDA = 0
3286 006222 132737 000200 006212 BITB #RDA,1$ ;SEE IF RDA = 1
3287 006230 001040 BNE 3$ ;BR IF RDA = 1
3288 ;STACK "RDA NOT SET" MSG
3289 006232 GTDF EM75,ERR12
; QUEUE "DEVICE FATAL" ERROR # 14
MOV #T.EDF,ERRTYP
MOV #14,ERRNBR
MOV #EM75,ERRMSG
MOV #ERR12,ERRBLK
006232 012737 000001 002246
006240 012737 000016 002250
006246 012737 014145 002252
006254 012737 017722 002254
3290 006262 000261 SEC ;SET C BIT TO FLAG ERROR
3291 006264 000423 BR 4$ ;TAKE ERROR EXIT
3292 006266 132737 000200 006212 2$: BITB #RDA,1$ ;SEE IF RDA = 0
3293 006274 001416 BEQ 3$ ;BR IF RDA = 0
3294 ;STACK "RDA NOT CLEARED" MSG
3295 006276 GTDF EM76,ERR12
; QUEUE "DEVICE FATAL" ERROR # 15
MOV #T.EDF,ERRTYP
MOV #15,ERRNBR
MOV #EM76,ERRMSG
MOV #ERR12,ERRBLK
006276 012737 000001 002246
006304 012737 000017 002250
006312 012737 014161 002252
006320 012737 017722 002254
3296 006326 000261 SEC
3297 006330 000401 BR 4$
3298 006332 000241 3$: CLC
3299 006334 000205 4$: RTS R5
;RETURN
3300
3301
3302
3303

```


....CKRSA -- CHECK RECEIVER STATUS AVAILABLE

```

3305 .SBTTL ....CKRSA -- CHECK RECEIVER STATUS AVAILABLE
3306 ;*****
3307 ;* CKRSA - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RSA IN THE USYRT
3308 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
3309 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3310 ;*
3311 ;* CALLING SEQUENCE :
3312 ;* JSR R5,CKRSA
3313 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RSA>
3314 ;*****
3315 006336 CKRSA:
3316 006336 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3317 006344 004537 003610 JSR R5,READI ;READ USYRT STATUS
3318 006350 122000 USTATR
3319 006352 000000 1$: .WORD 0
3320 006354 032725 000001 BIT #BIT0,(R5)+ ;GET EXPECTED STATE OF RSA
3321 006360 001422 BEQ 2$ ;BR IF EXPECTED RSA = 0
3322 006362 132737 000020 006352 BITB #RSA,1$ ;SEE IF RSA = 1
3323 006370 001040 BNE 3$ ;BR IF RSA = 1
3324 ;STACK "RSA NOT SET" MSG
3325 006372 GTDF EM77,ERR12
; QUEUE "DEVICE FATAL" ERROR # 16
MOV #T.EDF,ERRTYP
MOV #16,ERRNBR
MOV #EM77,ERRMSG
MOV #ERR12,ERRBLK
3326 006422 000261 SEC ;SET C BIT TO FLAG ERROR
3327 006424 000423 BR 4$ ;TAKE ERROR EXIT
3328 006426 132737 000020 006352 2$: BITB #RSA,1$ ;SEE IF RSA = 0
3329 006434 001416 BEQ 3$ ;BR IF RSA = 0
3330 ;STACK "RSA NOT CLEARED" MSG
3331 006436 GTDF EM78,ERR12
; QUEUE "DEVICE FATAL" ERROR # 17
MOV #T.EDF,ERRTYP
MOV #17,ERRNBR
MOV #EM78,ERRMSG
MOV #ERR12,ERRBLK
3332 006466 000261 SEC ;SET C BIT TO FLAG ERROR
3333 006470 000401 BR 4$ ;TAKE ERROR EXIT
3334 006472 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3335 006474 000205 4$: RTS R5 ;RETURN
3336
3337

```

....CKROR -- CHECK RECEIVER OVERRUN

```

3339 .SBTTL ....CKROR -- CHECK RECEIVER OVERRUN
3340 ;*****
3341 ;* CKROR - THIS SUBROUTINE CHECKS FOR THE OCCURANCE OF RECEIVER OVERRUN IN THE
3342 ;* USYRT RECEIVER STATUS REGISTER (RDSRH), AND REPORTS AN ERROR IF IT IS
3343 ;* NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3344 ;*
3345 ;* CALLING SEQUENCE :
3346 ;* JSR R5,CKROR
3347 ;* .WORD <BIT 0 IS EXPECTED VALUE OF ROR>
3348 ;*****
3349 006476 CKROR:
3350 006476 012737 000001 002412 MOV #1,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3351 006504 004537 003610 JSR R5,READI ;READ RECEIVER STATUS
3352 006510 120401 RDSRH
3353 006512 000000 1$: .WORD 0
3354 006514 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF ROR
3355 006520 001422 BEQ 2$ ;BR IF EXPECTED ROR = 0
3356 006522 132737 000010 006512 BITB #ROR,1$ ;SEE IF ROR = 1
3357 006530 001040 BNE 3$ ;BR IF ROR = 1
3358 ;STACK "RECEIVER OVRN NOT SET" MSG
3359 006532 GTDF EM90,ERR12
; QUEUE "DEVICE FATAL" ERROR # 18
MOV #T,EDF,ERRTYP
MOV #18,ERRNBR
MOV #EM90,ERRMSG
MOV #ERR12,ERRBLK
006532 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
006540 012737 000022 002250 BR 4$ ;TAKE ERROR EXIT
006546 012737 014235 002252 2$: BITB #ROR,1$ ;SEE IF ROR = 0
006554 012737 017722 002254 BEQ 3$ ;BR IF ROR = 0
3360 006562 000261 ;STACK "ROR NOT CLEARED" MSG
3361 006564 000423 GTDF EM91,ERR12
; QUEUE "DEVICE FATAL" ERROR # 19
MOV #T,EDF,ERRTYP
MOV #19,ERRNBR
MOV #EM91,ERRMSG
MOV #ERR12,ERRBLK
006576 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
006604 012737 000023 002250 BR 4$ ;TAKE ERROR EXIT
006612 012737 014266 002252 3$: CLC ;CLEAR C BIT FOR NO ERRORS
006620 012737 017722 002254 4$: RTS R5 ;RETURN
3366 006626 000261
3367 006630 000401
3368 006632 000241
3369 006634 000205
3370
3371
3372

```

....CKSEOM -- CHECK RSOM, REOM

```

3374 .SBTTL ....CKSEOM -- CHECK RSOM, REOM
3375 :*****
3376 :* CKSEOM - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF RSOM, REOM IN THE
3377 :* USYRT RECEIVER STATUS REG (RDSRH) AND REPORTS AN ERROR IF THEY ARE NOT
3378 :* PROPERLY SET TO THE STATES OF BITS 0,1 IN THE WORD FOLLOWING THE CALL.
3379 :* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3380 :* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3381 :* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3382 :*
3383 :* CALLING SEQUENCE :
3384 :* JSR R5,CKSEOM
3385 :* <BIT 0 IS EXPECTED VALUE OF RSOM, BIT 1 IS VALUE OF REOM>
3386 :*****
3387 006636 CKSEOM:
3388 006636 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3389 006644 004537 003610 JSR R5,READI ;READ USYRT RECEIVER STATUS
3390 006650 120401 RDSRH
3391 006652 000000 1$: .WORD 0
3392 006654 032725 000001 BIT #BIT0,(R5)+ ;GET EXPECTED STATE OF RSOM
3393 006660 001422 BEQ 2$ ;BR IF EXPECTED RSOM = 0
3394 006662 132737 000001 006652 BITB #RSOM,1$ ;SEE IF RSOM = 1
3395 006670 001040 BNE 3$ ;BR IF RSOM = 1
3396 ;STACK "RSOM NOT SET" MSG
3397 006672 GTDF EM29,ERR12
; QUEUE "DEVICE FATAL" ERROR # 20
MOV #T.EDF,ERRTYP
MOV #20,ERRNBR
MOV #EM29,ERRMSG
MOV #ERR12,ERRBLK
006672 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
006700 012737 000024 002250 BR 6$ ;TAKE ERROR EXIT
006706 012737 013520 002252 BITB #RSOM,1$ ;SEE IF RSOM = 0
006714 012737 017722 002254 BEQ 3$ ;BR IF RSOM = 0
3398 006722 000261 SEC ;SET C BIT TO FLAG ERROR
3399 006724 000473 BR 6$ ;TAKE ERROR EXIT
3400 006726 132737 000001 006652 2$: BITB #RSOM,1$ ;SEE IF RSOM = 0
3401 006734 001416 BEQ 3$ ;BR IF RSOM = 0
3402 ;STACK "RSOM NOT CLEARED" MSG
3403 006736 GTDF EM28,ERR12
; QUEUE "DEVICE FATAL" ERROR # 21
MOV #T.EDF,ERRTYP
MOV #21,ERRNBR
MOV #EM28,ERRMSG
MOV #ERR12,ERRBLK
006736 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
006744 012737 000025 002250 BR 6$ ;TAKE ERROR EXIT
006752 012737 013477 002252 BIT #BIT1,-2(R5) ;GET EXPECTED STATE OF REOM
006760 012737 017722 002254 BEQ 4$ ;BR IF EXPECTED REOM = 0
3404 006766 000261 SEC ;SET C BIT TO FLAG ERROR
3405 006770 000451 BR 6$ ;TAKE ERROR EXIT
3406 006772 032765 000002 177776 3$: BIT #BIT1,-2(R5) ;GET EXPECTED STATE OF REOM
3407 007000 001422 BEQ 4$ ;BR IF EXPECTED REOM = 0
3408 007002 132737 000002 006652 BITB #REOM,1$ ;SEE IF REOM = 1
3409 007010 001040 BNE 5$ ;BR IF REOM = 1
3410 ;STACK "REOM NOT SET" MSG
3411 007012 GTDF EM31,ERR12
; QUEUE "DEVICE FATAL" ERROR # 22
MOV #T.EDF,ERRTYP
MOV #22,ERRNBR
MOV #EM31,ERRMSG
MOV #ERR12,ERRBLK
007012 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
007020 012737 000026 002250 BR 6$ ;TAKE ERROR EXIT
007026 012737 013556 002252 BITB #REOM,1$ ;SEE IF REOM = 0
007034 012737 017722 002254 BEQ 5$ ;BR IF REOM = 0
3412 007042 000261 SEC ;SET C BIT TO FLAG ERROR
3413 007044 000423 BR 6$ ;TAKE ERROR EXIT
3414 007046 132737 000002 006652 4$: BITB #REOM,1$ ;SEE IF REOM = 0
3415 007054 001416 BEQ 5$ ;BR IF REOM = 0

```

....CKSEOM -- CHECK RSOM, REOM

```

3416
3417 007056          ;STACK "REOM NOT CLEARED" MSG
                          GTDF      EM30,ERR12
          007056 012737 000001 002246
          007064 012737 000027 002250
          007072 012737 013535 002252
          007100 012737 017722 002254
3418 007106 000261          SEC
3419 007110 000401          BR      6$
3420 007112 000241          5$:    CLC
3421 007114 000205          6$:    RTS      R5
3422
3423

```

```

; QUEUE "DEVICE FATAL" ERROR # 23
MOV      #T.EDF,ERR12P
MOV      #23,ERRNBR
MOV      #EM30,ERRMSG
MOV      #ERR12,ERRBLK
;SET C BIT TO FLAG ERROR
;TAKE ERROR EXIT
;CLEAR C BIT FOR NO ERRORS
;RETURN

```

....CHKTSO -- CHECK TRANSMIT SERIAL OUT BIT

```

3425 .SBTTL ....CHKTSO -- CHECK TRANSMIT SERIAL OUT BIT
3426 ;*****
3427 ;* CHKTSO - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TSO IN THE USYRT
3428 ;* STATUS REGISTER, AND SETS THE "C" BIT IF IT IS NOT SET TO THE STATE
3429 ;* OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3430 ;*
3431 ;* CALLING SEQUENCE :
3432 ;* JSR R5,CHKTSO
3433 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TSO>
3434 ;*****
3435 007116 CHKTSO:
3436 007116 012737 000007 002412 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3437 007124 004537 003610 JSR R5,READI ;READ USYRT STATUS
3438 007130 122000 USTATR
3439 007132 000000 1$: .WORD 0
3440 007134 032725 000001 BIT #BIT0,(R5)+ ;GET EXPECTED STATE OF TSO
3441 007140 001422 BEQ 2$ ;BR IF EXPECTED TSO = 0
3442 007142 132737 000010 007132 BITB #TS0,1$ ;SEE IF TSO = 1
3443 007150 001040 BNE 3$ ;BR IF TSO = 1
3444 ;*** STACK "TSO NOT SET" ERROR ***
3445 007152 GTDF EM100,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 24
; MOV #T.EDF,ERRTYP
; MOV #24,ERRNBR
; MOV #EM100,ERRMSG
; MOV #ERR12,ERRBLK
007152 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
007160 012737 000030 002250 BR 4$ ;TAKE ERROR EXIT
007166 012737 014425 002252
007174 012737 017722 002254
3446 007202 000261 SEC ;SET C BIT TO FLAG ERROR
3447 007204 000423 BR 4$ ;TAKE ERROR EXIT
3448
3449 007206 132737 000010 007132 2$: BITB #TS0,1$ ;SEE IF TSO = 0
3450 007214 001416 BEQ 3$ ;BR IF TSO = 0
3451 ;*** STACK "TSO NOT CLEARED" ERROR ***
3452 007216 GTDF EM101,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 25
; MOV #T.EDF,ERRTYP
; MOV #25,ERRNBR
; MOV #EM101,ERRMSG
; MOV #ERR12,ERRBLK
007216 012737 000001 002246 SEC ;SET C BIT TO FLAG ERROR
007224 012737 000031 002250 BR 4$ ;TAKE ERROR EXIT
007232 012737 014445 002252
007240 012737 017722 002254
3453 007246 000261 SEC ;SET C BIT TO FLAG ERROR
3454 007250 000401 BR 4$ ;TAKE ERROR EXIT
3455 007252 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3456 007254 000205 4$: RTS R5 ;RETURN
3457
3458

```

....SERIAL -- READ/CHECK TX CHARACTER VIA TSO BIT

```

3460 .SBTTL ....SERIAL -- READ/CHECK TX CHARACTER VIA TSO BIT
3461 ;*****
3462 ;* SERIAL - THIS SUBROUTINE SERIALY READS/CLOCKS/CHECKS A CHARACTER FROM
3463 ;* THE TRANSMIT SERIAL OUT (TSO) BIT OF THE USYRT STATUS REGISTER,
3464 ;* AND STACKS MESSAGE/SETS "C" BIT IF AN INCORRECT CHARACTER IS READ.
3465 ;* NOTE: "EXPECTED VALUE" ARGUMENT IS ALWAYS READ RIGHT-TO-LEFT.
3466 ;*
3467 ;* CALLING SEQUENCE :
3468 ;* JSR R5,SERIAL
3469 ;* .WORD <# OF BITS TO BE READ>
3470 ;* .WORD <EXPECTED VALUE OF SERIAL BIT STREAM>
3471 ;*****
3472 SERIAL:
3473 MOV R1,-(SP) ;SAVE R1
3474 MOV R2,-(SP) ;SAVE R2 (TICKS)
3475 MOV R3,-(SP) ;SAVE R3 (EXPECTED_WORD)
3476
3477 CLR R1 ;CLEAR ASSEMBLED_WORD
3478 MOV (R5)+,R2 ;GET # OF TICKS
3479
3480 1$: ASL R1 ;SHIFT ASSEMBLED_WORD
3481 JSR R5,STEPLU ;CLOCK USYRT ONCE
3482 1
3483
3484 JSR R5,CHKTSO ;CHECK FOR TSO=1
3485 1
3486 BCS 2$ ;BR IF TSO=0
3487 INC R1 ;TSO=1: SET LSB OF ASSEMBLED_WORD
3488 2$: SOB R2,1$ ;LOOP UNTIL NO MORE TICKS
3489
3490 MOV (R5)+,R3 ;GET EXPECTED_WORD
3491 CMP R1,R3 ;COMPARE EXPECTED_ AND ASSEMBLED_WORD
3492 BEQ 3$ ;BR IF CORRECT VALUE READ
3493
3494 MOV R3,GDATA ;EXPECTED_WORD => GDATA
3495 MOV R1,BDATA ;ASSEMBLED_WORD => BDATA
3496 ;*** STACK "TRANSMISSION ERROR" MSG ***
3497 GTDF EM102,ERR13
; QUEUE "DEVICE FATAL" ERROR # 26
; MOV #T,EDF,ERRTYP
; MOV #26,ERRNBR
; MOV #EM102,ERRMSG
; MOV #ERR13,ERRBLK
3498 007332 012737 000001 002246
3499 007340 012737 000032 002250
007346 012737 014471 02252
007354 012737 020036 002254
3498 007362 000261 ;SET C BIT TO FLAG ERROR
3499 007364 0004C1 BR 4 ;TAKE ERROR EXIT
3500
3501 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3502 MOV (SP)+,R3 ;RESTORE REGISTERS
3503 MOV (SP)+,R2
3504 MOV (SP)+,R1
3505 4$: RTS R5 ;RETURN
3506
3507

```

....INITRN -- INIT TRANSMISSION OF A MESSAGE

3509
 3510
 3511
 3512
 3513
 3514
 3515
 3516
 3517
 3518
 3519
 3520
 3521
 3522
 3523
 3524
 3525
 3526 007400
 3527 007400 010146
 3528 007402 004537 003734
 3529 007406 120000
 3530 007410 000031
 3531 007412 004537 003734
 3532 007416 120000
 3533 007420 000030
 3534 007422 112537 007434
 3535 007426 004537 003734
 3536 007432 120404
 3537 007434 000000
 3538 007436 112537 007450
 3539 007442 004537 003734
 3540 007446 120405
 3541 007450 000000
 3542 007452 112537 007476
 3543 007456 005037 002456
 3544 007462 113737 007476 002456
 3545 007470 004537 003734
 3546 007474 120407
 3547 007476 000000
 3548 007500 004537 003734
 3549 007504 120013
 3550 007506 000200
 3551 007510 004537 003734
 3552 007514 120006
 3553 007516 000300
 3554 007520 004537 003734
 3555 007524 120007
 3556 007526 000000
 3557 007530 004537 005432
 3558 007534 000110
 3559 007536 103454
 3560
 3561 007540 013737 007674 007560
 3562 007546 142537 007560
 3563
 3564 007552 004537 003734
 3565 007556 120000

```

.SBTTL ....INITRN -- INIT TRANSMISSION OF A MESSAGE
*****
; * INITRN - THIS SUBROUTINE INITIATES TRANSMISSION OF A MESSAGE, BY LOADING
; * THE USYRT PCSARL,H AND THE PCR WITH THE DATA PASSED IN THE 2 WORDS
; * FOLLOWING THE CALL ; LOADING AND CLOCKING 1 SOM UNTIL THE FIRST
; * SYNCH OR FLAG HAS BEEN SERIALIZED IN THE USYRT. THE PROGRAM MONITOR'S
; * ALL THE FLAGS IN THE USYRT STATUS REGISTER THROUGHOUT THE PROCESS.
; * IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION IS STACKED
; * AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE DISCRETION
; * OF THE CALLING ROUTINE OR SUBROUTINE.
; *
; * CALLING SEQUENCE :
; * JSR R5,INITRN
; * .WORD <VALUE TO LOAD INTO USYRT PCSARL,H>
; * .WORD <VALUE TO LOAD INTO USYRT PCR (PASSED IN LO BYTE)>
; * <SPECIAL VIAORB MASKING VALUE (PASSED IN HI BYTE)>
*****
INITRN:
MOV R1,(SP) ;SAVE R1
JSR R5,WRITEI ;RESET THE USYRT
VIAORB
RTSND!DTR!PRESET
JSR R5,WRITEI ;CLEAR USYRT RESET BIT
VIAORB
RTSND!DTR
MOVB (R5),.1# ;GET VALUE TO LOAD INTO USYRT PCSARL
JSR R5,WRITEI ;LOAD USYRT PCSARL
PCARL
1#: .WORD 0
MOVB (R5),.2# ;GET VALUE TO LOAD INTO PCSARH
JSR R5,WRITEI ;LOAD USYRT PCSARH
PC SARH
2#: .WORD 0
MOVB (R5),.3# ;GET VALUE TO LOAD INTO PCR
CLR SAVLEN
MOVB 3#,SAVLEN ;SAVE CHAR LENGTH BITS
JSR R5,WRITEI ;LOAD USYRT PCR
PCR
3#: .WORD 0
JSR R5,WRITEI ;SET ACR FOR T1 ONE-SHOT MODE
VIAACR
200
JSR R5,WRITEI ;LOAD VIA T1L-L
VIAT1C
300
JSR R5,WRITEI ;LOAD VIA T1L-H
VIAT1D
000
JSR R5,CKUSTS ;CHK USYRT STATUS FOR INIT'D STATE
110 ; TBMT = 1, TSO = 1
BCS 7# ;IF ERROR, EXIT SUBROUTINE

MOV 20#,13# ;* SET UP DEFAULT VIAORB PARAMETERS
BICB (R5),.13# ;* CLEAR ANY SPECIFIED VIAORB BITS.

JSR R5,WRITEI ;SET UP USYRT
VIAORB
    
```

....INITRN -- INIT TRANSMISSION OF A MESSAGE

```

3566 007560 000142          13#:  TXEN!RXEN!TTLOOP          ;* THIS VALUE MIGHT BE MODIFIED ABOVE
3567
3568 007562 004537 003734    JSR      R5,WRITEI          ;SET TSOM IN USYRT
3569 007566 120403          TDSRH
3570 007570 000001          TSOM
3571 007572 004537 003734    JSR      R5,WRITEI          ;LOAD SYNCH CHAR INTO TX BUF
3572 007576 120402          TDSRL
3573 007600 000226          SYNCH
3574 007602 004537 006036    JSR      R5,CKTBMT          ;CHK FOR TBMT = 0
3575 007606 000000          0
3576 007610 103427          BCS      7#                ;IF ERROR, EXIT SUBROUTINE
3577 007612 005001          CLR      R1                ;INIT CYCLE COUNTER
3578 007614 004537 011614    4#:     JSR      R5,STEPLU          ;CLOCK LU FOR 1 CYCLE
3579 007620 000001          1
3580 007622 004537 003610    JSR      R5,READI          ;READ USYRT STATUS REG
3581 007626 122000          USTATR
3582 007630 000000          .WORD   0
3583 007632 132737 000100 007630 5#:     BITB     @TBMT,5#          ;SEE IF TBMT IS SET YET
3584 007640 001010          BNE     6#                ;BR IF YES
3585 007642 005201          INC     R1                ;INCR CYCLE COUNTER
3586 007644 020127 000005    CMP     R1,#3             ;SEE IF 3 CYCLES DONE YET
3587 007650 002761          BLT     4#                ;BR IF LESS THAN 3 CYCLES
3588 007652 004537 006036    JSR      R5,CKTBMT          ;GO STACK "TBMT NOT SET" MSG
3589 007656 000001          1
3590 007660 103403          BCS     7#                ;IF ERROR, EXIT SUBROUTINE
3591 007662 004537 005536    6#:     JSR      R5,CKTACT          ;CHK FOR TXACT = 1
3592 007666 000001          1
3593 007670 012601          7#:     MOV     (SP)+,R1          ;RESTORE R1
3594 007672 000205          RTS     R5                ;RETURN (IF C = 1, WE HAD AN ERROR)
3595
3596 007674 000142          20#:   TXEN!RXEN!TTLOOP          ;DEFAULT VALUE FOR VIAORB: ENABLE
3597
3598

```


....TXCHAR -- TRANSMIT A CHARACTER

3600
 3601
 3602
 3603
 3604
 3605
 3606
 3607
 3608
 3609
 3610
 3611
 3612
 3613
 3614
 3615
 3616
 3617 007676
 3618 007676 010146
 3619 007700 010246
 3620 007702 012537 007714
 3621 007706 004537 003734
 3622 007712 120402
 3623 007714 000000
 3624 007716 005001
 3625 007720 005002
 3626 007722 112502
 3627 007724 001425
 3628 007726 004537 005536
 3629 007732 000001
 3630 007734 103421
 3631 007736 020102
 3632 007740 001414
 3633
 3634 007742 131527 000200
 3635 007746 001004
 3636
 3637 007750 004537 006036
 3638 007754 000000
 3639 007756 103410
 3640 007760 004537 011614
 3641 007764 000001
 3642 007766 005201
 3643 007770 000756
 3644 007772 004537 006036
 3645 007776 000001
 3646 010000 012602
 3647 010002 012601
 3648 010004 005205
 3649 010006 000205
 3650
 3651
 3652
 3653

```

.SBTTL ....TXCHAR -- TRANSMIT A CHARACTER
;*****
;* TXCHAR - THIS SUBROUTINE INITIATES TRANSMISSION OF A CHAR BY LOADING
;* THE USYRT TDSRL WITH THE DATA PASSED IN THE LO BYTE OF THE WORD
;* FOLLOWING THE CALL, AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES
;* PASSED IN THE SECOND WORD FOLLOWING THE CALL. THE PROGRAM CONTINUALLY
;* MONITORS TBMT AND TXACT THROUGHOUT THE PROCESS.
;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
;*
;* CALLING SEQUENCE :
;* JSR R5,TXCHAR
;* .WORD <DATA FOR TDSRL IN LO BYTE>
;* .WORD <NUMBER OF CYCLES TO CLOCK (IN LO BYTE)>
;* <SWITCH TO DISABLE INITIAL TBMT=0 CHECK (MSB IN HI BYTE)>
;*****
TXCHAR:
MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
MOV (R5),1# ;GET DATA FOR TDSRL
JSR R5,WRITEI ;LOAD DATA INTO TDSRL
TDSRL
1#: .WORD 0
CLR R1 ;INIT CYCLE COUNT AND CLEAR C BIT
CLR R2 ;CLEAR REQ'D CYCLE COUNT
MOVB (R5),R2 ;GET DESIRED NO. OF CYCLES
BEQ 6# ;BR IF NO CLOCKING DONE
3#: JSR R5,CKTACT ;CHECK TXACT = 1
1
BCS 6# ;BR TO EXIT IF ERROR
CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
BEQ 5# ;BR IF YES

BITB (R5),#NCTBMT ;* CHECK FOR "TBMT=0 CHECK" DISABLE
BNE 7# ;* BR IF MSB IS NOT SET

JSR R5,CKTBMT ;CHECK FOR TBMT = 0
0
BCS 6# ;BR TO EXIT IF ERROR
7#: JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
1
INC R1 ;INCR CYCLE COUNT
BR 3# ;KEEP CLOCKING
5#: JSR R5,CKTBMT ;CHK TBMT = 1
1
6#: MOV (SP),R2 ;RESTORE R2
MOV (SP),R1 ;RESTORE R1
INC R5 ;ADJUST R5 FOR SAME RETURN
RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)
    
```

....TXCTRL -- CONTROL MESSAGE TRANSMISSION (TDSRH)

```

3655 .SBITL ....TXCTRL -- CONTROL MESSAGE TRANSMISSION (TDSRH)
3656 ;*****
3657 ;* TXCTRL - THIS SUBROUTINE ALLOWS CONTROL OF MESSAGE TRANSMISSION BY LOADING
3658 ;* THE USYRT TDSRH WITH THE DATA PASSED IN THE LO BYTE OF THE WORD
3659 ;* FOLLOWING THE CALL, AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES
3660 ;* PASSED IN THE SECOND WORD FOLLOWING THE CALL. THE PROGRAM CONTINUALLY
3661 ;* MONITORS TBMT AND TXACT THROUGHOUT THE PROCESS.
3662 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3663 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3664 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3665 ;*
3666 ;* CALLING SEQUENCE :
3667 ;* JSR R5,TXCTRL
3668 ;* .WORD <DATA FOR TDSRH IN LO BYTE>
3669 ;* .WORD <NUMBER OF CYCLES TO CLOCK>
3670 ;*****
3671 010010 TXCTRL:
3672 010010 010146 MOV R1,-(SP) ;SAVE R1
3673 010012 010246 MOV R2,-(SP) ;SAVE R2
3674 010014 012537 010026 MOV (R5)+,2# ;GET DATA FOR TDSRH
3675 010020 004537 003734 JSR R5,WRITEI ;LOAD DATA INTO TDSRH
3676 010024 120403 TDSRH
3677 010026 000000 .WORD 0
3678 010030 005001 2#: CLR R1 ;INIT CYCLE COUNT AND CLEAR C BIT
3679 010032 012502 MOV (R5)+,R2 ;GET DESIRED NO. OF CYCLES
3680 010034 001422 BEQ 6# ;BR IF NO CLOCKING DONE
3681 010036 004537 005536 3#: JSR R5,CKTACT ;CHECK TXACT = 1
3682 010042 000001 1
3683 010044 103416 BCS 6# ;BR TO EXIT IF ERROR
3684 010046 020102 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
3685 010050 001411 BEQ 5# ;BR IF YES
3686 010052 004537 006036 JSR R5,CKTBMT ;CHECK FOR TBMT = 0
3687 010056 000000 0
3688 010060 103410 BCS 6# ;BR TO EXIT IF ERROR
3689 010062 004537 011614 JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
3690 010066 000001 1
3691 010070 005201 INC R1 ;INCR CYCLE COUNT
3692 010072 000761 BR 3# ;KEEP CLOCKING
3693 010074 004537 006036 5#: JSR R5,CKTBMT ;CHK TBMT = 1
3694 010100 000001 1
3695 010102 012602 6#: MOV (SP)+,R2 ;RESTORE R2
3696 010104 012601 MOV (SP)+,R1 ;RESTORE R1
3697 010106 000205 RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)
3698

```

....RXCHAR -- RECEIVE A CHARACTER

```

3700 .SBTTL ....RXCHAR -- RECEIVE A CHARACTER
3701 ;*****
3702 ;* RXCHAR - THIS SUBROUTINE READS THE USYRT RDSR AND CHECKS THE CONTENTS
3703 ;* AGAINST THE DATA PASSED IN THE WORD FOLLOWING THE CALL.
3704 ;* IF BIT0 = 0 IN THE SECOND WORD FOLLOWING THE CALL, THE RERR BIT IS
3705 ;* NOT CHECKED AGAINST THE EXPECTED VALUE. THEN, IT CLOCKS
3706 ;* THE LINE UNIT FOR THE NO. OF CYCLES PASSED IN THE THIRD WORD
3707 ;* FOLLOWING THE CALL. THE PROGRAM CONTINUALLY MONITORS RDA AND RXACT.
3708 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3709 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3710 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3711 ;*
3712 ;* CALLING SEQUENCE :
3713 ;* JSR R5,RXCHAR
3714 ;* .WORD <EXPECTED RDSRL IN LO BYTE, RDSRH IN HI BYTE>
3715 ;* .WORD <=0 FOR NO RERR CHK, =1 FOR RERR CHK>
3716 ;* .WORD <NUMBER OF CYCLES TO CLOCK (IN LO BYTE)>
3717 ;* <SPECIAL DISABLE SWITCHES: NOCRDA,NFCRDA,NCRACK(IN HI BYTE)>
3718 ;*****
3719 010110 RXCHAR:
3720 010110 010146 MOV R1,-(SP) ;SAVE R1
3721 010112 010246 MOV R2,-(SP) ;SAVE R2
3722 010114 004537 003610 JSR R5,READI ;READ RDSRH
3723 010120 120401 RDSRH
3724 010122 000000 2$: .WORD 0
3725 010124 004537 003610 JSR R5,READI ;READ RDSRL
3726 010130 120400 RDSRL
3727 010132 000000 1$: .WORD 0
3728 010134 111501 MOVB (R5),R1 ;GET EXPECTED RDSRL
3729 010136 042701 177400 BIC #177400,R1 ;MASK OFF UNUSED BITS
3730 010142 023727 002456 000347 CMP SAVLEN,#TXDL!RXDL ;SEE IF 7-BIT CHARS BEING USED
3731 010150 001005 3$ SNE 3$ ;BR IF NOT 7-BIT CHARS
3732 010152 142737 000200 010132 BICB #BIT7,1$ ;CLEAR 8TH BIT FOR COMPARE
3733 010160 142701 000200 BICB #BIT7,R1
3734 010164 123701 010132 3$: CMPB 1$,R1 ;COMPARE RCV'D CHAR TO EXPECTED
3735 010170 001462 BEQ 6$ ;BR IF MATCH
3736 010172 004537 003610 JSR R5,READI ;READ USYRT STATUS REG
3737 010176 122000 USTATR
3738 010200 000000 4$: .WORD 0
3739 010202 132737 000002 010200 BITB #TXU,4$ ;SEE IF TX UNDERRUN OCCURRED
3740 010210 001421 BEQ 5$ ;BR IF NOT
3741 010212 012737 000007 002412 MOV #7,REGNUM ;SET USYRT REG NO. FOR STATUS REG
3742 ;STACK "TX UNDERRUN" ERROR
3743 010220 GTDF EM54,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 27
; MOV #T.EDF,ERRTYP
; MOV #27,ERRNBR
; MOV #EM54,ERRMSG
; MOV #ERR12,ERRBLK
010220 012737 000001 002246
010226 012737 000033 002250
010234 012737 013717 002252
010242 012737 017722 002254
3744 010250 000137 011350 JMP 20$ ;TAKE ERROR EXIT
3745 010254 005037 002412 5$: CLR REGNUM ;SET USYRT REG NO. FOR RDSRL
3746 010260 005037 002400 CLR GOATA ;SET EXPECTED DATA
3747 010264 110137 002400 MOVB R1,GDATA
3748 010270 005037 002402 CLR BDATA ;SET ACTUAL DATA
3749 010274 113737 010132 002402 MOVB 1$,BDATA
3750 ;STACK "RCV'D DATA MISCOMPARE" ERROR
3751 010302 GTDF EM34,ERR10

```

....RXCHAR -- RECEIVE A CHARACTER

```

010302 012737 000001 002246
010310 012737 000034 002250
010316 012737 013573 002252
010324 012737 017372 002254
3752 010332 000137 011350
3753 010336 116501 000001
3754 010342 042701 177400
3755 010346 123701 010122
3756 010352 001016
3757 010354 000137 011234
3758 010360 012737 000001 002412
3759 010366 005037 002400
3760 010372 110137 002400
3761 010376 005037 002402
3762 010402 113737 010122 002402
3763 010410 012737 000001 002412
3764 010416 032765 000001 000002
3765 010424 001447
3766
3767 010426 132701 000200
3768 010432 001022
3769 010434 132737 000200 010122
3770 010442 001440
3771
3772 010444
;
; QUEUE "DEVICE FATAL" ERROR # 28
;
; TAKE ERROR EXIT
; GET RDSRH
; MASK OFF UNUSED BITS
; COMPARE RCVD STATUS TO EXPECTED
; BR IF MISMATCH
; CONTINUE
; SET USYRT REG NO. FOR RDSRH
; SET EXPECTED DATA
; SET ACTUAL DATA
; SET REG NO. FOR PRINTOUT
; SEE IF RCVD ERROR BIT SHOULD BE IGNORED
; BR IF YES
; SEE IF EXPECTED BIT = 1
; BR IF YES
; SEE IF ACTUAL BIT = 0
; BR IF YES
; STACK "RERR NOT CLEARED" MSG
; GTDF EM35,ERR12
;
; QUEUE "DEVICE FATAL" ERROR # 29
;
; TAKE ERROR EXIT
; SEE IF ACTUAL BIT = 1
; BR IF YES
;
; QUEUE "DEVICE FATAL" ERROR # 30
;
; TAKE ERROR EXIT
; SEE IF EXPECTED BIT = 1
; BR IF YES
; SEE IF ACTUAL BIT = 0
; BR IF YES
; STACK "ROR NOT CLEARED" MSG
; GTDF EM16,ERR12
;
; QUEUE "DEVICE FATAL" ERROR # 31
;
; TAKE ERROR EXIT
; SEE IF ACTUAL BIT = 1
; BR IF YES

010444 012737 000001 002246
010452 012737 000035 002250
010460 012737 013621 002252
010466 012737 017722 002254
3773 010474 000137 011350
3774 010500 132737 000200 010122
3775 010506 001016
3776
3777 010510
;
; CHECK RERR BIT
; BITB #RERR,R1
; BNE 8#
; BITB #RERR,2#
; BEQ 9#
; STACK "RERR NOT SET" MSG
; GTDF EM36,ERR12
;
; CHECK ROR BIT
; BITB #ROR,R1
; BNE 10#
; BITB #ROR,2#
; BEQ 11#
; STACK "ROR NOT CLEARED" MSG
; GTDF EM16,ERR12

010510 012737 000001 002246
010516 012737 000036 002250
010524 012737 013642 002252
010532 012737 017722 002254
3778 010540 000137 011350
3779
3780 010544 132701 000010
3781 010550 001022
3782 010552 132737 000010 010122
3783 010560 001440
3784
3785 010562
;
; CHECK ROR BIT
; BITB #ROR,R1
; BNE 10#
; BITB #ROR,2#
; BEQ 11#
; STACK "ROR NOT CLEARED" MSG
; GTDF EM16,ERR12

010562 012737 000001 002246
010570 012737 000037 002250
010576 012737 013346 002252
010604 012737 017722 002254
3786 010612 000137 011350
3787 010616 132737 000010 010122
3788 010624 001016
;
; CHECK ROR BIT
; BITB #ROR,R1
; BNE 10#
; BITB #ROR,2#
; BEQ 11#
; STACK "ROR NOT CLEARED" MSG
; GTDF EM16,ERR12

```

...RXCHAR -- RECEIVE A CHARACTER

```

3789                ;STACK "ROR NOT SET" MSG
3790 010626          GTDF      EM14,ERR12
                               ;
                               ;   QUEUE "DEVICE FATAL" ERROR # 32
                               ;
                               ;   MOV      #T.EDF,ERRTYP
                               ;   MOV      #32,ERRNBR
                               ;   MOV      #EM14,ERRMSG
                               ;   MOV      #ERR12,ERRBLK
                               ;
                               ;   TAKE ERROR EXIT
                               ;
                               ;   SEE IF EXPECTED BIT = 1
                               ;   BR IF YES
                               ;
                               ;   SEE IF ACTUAL BIT = 0
                               ;   BR IF YES
                               ;
010626 012737 000001 002246
010634 012737 000040 002250
010642 012737 013332 002252
010650 012737 017722 002254
3791 010656 000137 011350                JMP      20$
3792                ;CHECK RABGA BIT
3793 010662 132701 000004 11$: BITB     #RABGA,R1
3794 010666 001022                BNE     12$
3795 010670 132737 000004 010122  BITB     #RABGA,2$
3796 010676 001440                BEQ     13$
3797                ;STACK "RABGA NOT CLEARED" MSG
3798 010700          GTDF      EM39,ERR12
                               ;
                               ;   QUEUE "DEVICE FATAL" ERROR # 33
                               ;
                               ;   MOV      #T.EDF,ERRTYP
                               ;   MOV      #33,ERRNBR
                               ;   MOV      #EM39,ERRMSG
                               ;   MOV      #ERR12,ERRBLK
                               ;
                               ;   TAKE ERROR EXIT
                               ;
                               ;   SEE IF ACTUAL BIT = 1
                               ;   BR IF YES
                               ;
010700 012737 000001 002246
010706 012737 000041 002250
010714 012737 013657 002252
010722 012737 017722 002254
3799 010730 000137 011350                JMP      20$
3800 010734 132737 000004 010122 12$: BITB     #RABGA,2$
3801 010742 001016                BNE     13$
3802                ;STACK "RABGA NOT SET" MSG
3803 010744          GTDF      EM40,ERR12
                               ;
                               ;   QUEUE "DEVICE FATAL" ERROR # 34
                               ;
                               ;   MOV      #T.EDF,ERRTYP
                               ;   MOV      #34,ERRNBR
                               ;   MOV      #EM40,ERRMSG
                               ;   MOV      #ERR12,ERRBLK
                               ;
                               ;   TAKE ERROR EXIT
                               ;
                               ;   SEE IF EXPECTED BIT = 1
                               ;   BR IF YES
                               ;
                               ;   SEE IF ACTUAL BIT = 0
                               ;   BR IF YES
                               ;
010744 012737 000001 002246
010752 012737 000042 002250
010760 012737 013701 002252
010766 012737 017722 002254
3804 010774 000137 011350                JMP      20$
3805                ;CHECK REOM BIT
3806 011000 132701 000002 13$: BITB     #REOM,R1
3807 011004 001022                BNE     14$
3808 011006 132737 000002 010122  BITB     #REOM,2$
3809 011014 001440                BEQ     15$
3810                ;STACK "REOM NOT CLEARED" MSG
3811 011016          GTDF      EM30,ERR12
                               ;
                               ;   QUEUE "DEVICE FATAL" ERROR # 35
                               ;
                               ;   MOV      #T.EDF,ERRTYP
                               ;   MOV      #35,ERRNBR
                               ;   MOV      #EM30,ERRMSG
                               ;   MOV      #ERR12,ERRBLK
                               ;
                               ;   TAKE ERROR EXIT
                               ;
                               ;   SEE IF ACTUAL BIT = 1
                               ;   BR IF YES
                               ;
011016 012737 000001 002246
011024 012737 000043 002250
011032 012737 013535 002252
011040 012737 017722 002254
3812 011046 000137 011350                JMP      20$
3813 011052 132737 000002 010122 14$: BITB     #REOM,2$
3814 011060 001016                BNE     15$
3815                ;STACK "REOM NOT SET" MSG
3816 011062          GTDF      EM31,ERR12
                               ;
                               ;   QUEUE "DEVICE FATAL" ERROR # 36
                               ;
                               ;   MOV      #T.EDF,ERRTYP
                               ;   MOV      #36,ERRNBR
                               ;   MOV      #EM31,ERRMSG
                               ;   MOV      #ERR12,ERRBLK
                               ;
                               ;   TAKE ERROR EXIT
                               ;
                               ;   SEE IF EXPECTED BIT = 1
                               ;   BR IF YES
                               ;
011062 012737 000001 002246
011070 012737 000044 002250
011076 012737 013556 002252
011104 012737 017722 002254
3817 011112 000137 011350                JMP      20$
3818                ;CHECK RSOM BIT
3819 011116 132701 000001 15$: BITB     #RSOM,R1
3820 011122 001022                BNE     16$

```

....RXCHAR -- RECEIVE A CHARACTER

```

3821 011124 132737 000001 010122      BITB   4RSOM,2$      ;SEE IF ACTUAL BIT = 0
3822 011132 001440                      BEQ    17$          ;BR IF YES
3823                                     ;STACK "RSOM NOT CLEARED" MSG
3824 011134                      GTDF   EM28,ERR12

                                ;      QUEUE "DEVICE FATAL" ERROR # 37
                                MOV    0T.EDF,ERRTYP
                                MOV    037,ERRNBR
                                MOV    0EM28,ERRMSG
                                MOV    0ERR12,ERRBLK

011134 012737 000001 002246
011142 012737 000045 002250
011150 012737 013477 002252
011156 012737 017722 002254
3825 011164 000137 011350      JMP    20$          ;TAKE ERROR EXIT
3826 011170 132737 000001 010122 16$:  BITB   0RSOM,2$      ;SEE IF ACTUAL BIT = 1
3827 011176 001016                      BNE   17$          ;BR IF YES
3828                                     ;STACK "RSOM NOT SET" MSG
3829 011200                      GTDF   EM29,ERR12

                                ;      QUEUE "DEVICE FATAL" ERROR # 38
                                MOV    0T.EDF,ERRTYP
                                MOV    038,ERRNBR
                                MOV    0EM29,ERRMSG
                                MOV    0ERR12,ERRBLK

011200 012737 000001 002246
011206 012737 000046 002250
011214 012737 013520 002252
011222 012737 017722 002254
3830 011230 000137 011350      JMP    20$          ;TAKE ERROR EXIT
3831
3832 011234 116502 000004          17$:  MOVB   4(R5),R2      ;GET DESIRED NO. OF CYCLES
3833 011240 005001                      CLR   R1           ;INIT CYCLE COUNT
3834
3835 011242 136527 000005 000040 18$:  BITB   5(R5),0BIT5    ;* IS RXACT CHECK TO BE DISABLED ?
3836 011250 001004                      BNE   31$          ;* BR IF YES
3837 011252 004537 005676          JSR   R5,CKRACT    ;CHK FOR RACT = 1
3838 011256 000001                      1
3839 011260 103433                      BCS   20$          ;BR TO EXIT IF ERROR
3840
3841 011262 020102          31$:  CMP    R1,R2       ;SEE IF REQUIRED CYCLES DONE YET
3842 011264 001415                      BEQ   19$          ;BR IF YES
3843
3844 011266 136527 000005 000200      BITB   5(R5),0BIT7    ;* SEE IF INITIAL RDA CHECK DESIRED
3845 011274 001004                      BNE   22$          ;* BR IF NO
3846 011276 004537 006176          JSR   R5,CKRDA     ;CHK FOR RDA = 0
3847 011302 000000                      0
3848 011304 103421                      BCS   20$          ;BR TO EXIT IF ERROR
3849
3850 011306 004537 011614          22$:  JSR   R5,STEPLU    ;CLOCK LU FOR 1 CYCLE
3851 011312 000001                      1
3852 011314 005201                      INC   R1           ;INCR CYCLE COUNT
3853 011316 000751                      BR    18$          ;CONTINUE CLOCKING
3854
3855 011320 136527 000005 000100 19$:  BITB   5(R5),0BIT6    ;* IS FINAL RDA CHECK TO BE SKIPPED ?
3856 011326 001004                      BNE   30$          ;* BR IF YES
3857 011330 004537 006176          JSR   R5,CKRDA     ;CHK RDA = 1
3858 011334 000001                      1
3859 011336 103404                      BCS   20$          ;BR IF ERROR
3860
3861 011340 062705 000006          30$:  ADD    06,R5       ;FIX UP RETURN ADRS
3862 011344 000241                      CLC
3863 011346 000403                      BR    21$          ;SET C = 0 FOR NO ERROR
3864 011350 062705 000006          20$:  ADD    06,R5       ;TAKE ERROR-FREE EXIT
3865 011354 000261                      SEC          ;FIX UP RETURN ADDRESS
3866 011356 012602          21$:  MOV    (SP)+,R2    ;SET C BIT FOR ERROR
3867 011360 012601                      MOV    (SP)+,R1    ;RESTORE R2
                                ;RESTORE R1

```

....RXCHAR -- RECEIVE A CHARACTER

3868 011362 000205

RTS R5

;RETURN

....RCV1ST -- RECEIVE FIRST CHARACTER OF MESSAGE

```

3870 .SBTTL ....RCV1ST -- RECEIVE FIRST CHARACTER OF MESSAGE
3871 ;*****
3872 ;* RCV1ST - THIS SUBROUTINE RECEIVES THE FIRST CHAR OF A MESSAGE AND MONITORS
3873 ;* THE STATUS OF THE RECEIVER. FIRST, A CHECK IS MADE FOR RXACT = 0,
3874 ;* RDA = 0, RSA = 0, RSOM = 0. THEN, THE LINE UNIT IS CLOCKED UNTIL
3875 ;* RDA = 1. THE PROGRAM CHECKS FOR THIS TO OCCUR WITHIN 3 CYCLES AFTER
3876 ;* THE NO. OF CYCLES PASSED IN THE SECOND WORD FOLLOWING THE CALL.
3877 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3878 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3879 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3880 ;*
3881 ;* CALLING SEQUENCE :
3882 ;* JSR R5,RCV1ST
3883 ;* .WORD <EXPECTED RECEIVER CYCLE COUNT>
3884 ;*****
3885 RCV1ST:
3886 011364 MOV R1,-(SP) ;SAVE R1
3887 011366 MOV R2,-(SP) ;SAVE R2
3888 011370 CLR R1 ;INIT CYCLE COUNT
3889 011372 MOV (R5)+,R2 ;GET CYCLE COUNT LIMIT
3890 011374 062702 000003 ADD #3,R2
3891 011400 004537 005676 JSR R5,CKRACT ;CHK FOR RXACT = 0
3892 011404 000000 0
3893 011406 103446 BCS 6# ;BR TO EXIT IF ERROR
3894 011410 004537 006176 JSR R5,CKRDA ;CHK FOR RDA = 0
3895 011414 000000 0
3896 011416 103442 BCS 6# ;BR TO EXIT IF ERROR
3897 011420 004537 006636 JSR R5,CKSEOM ;CHK FOR RSOM = 0, REOM = 0
3898 011424 000000 0
3899 011426 103436 BCS 6# ;BR TO EXIT IF ERROR
3900 011430 004537 011614 1$: JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
3901 011434 000001 1
3902 011436 005201 INC R1 ;INCREMENT CYCLE COUNT
3903 011440 004537 003610 JSR R5,READI ;READ USYRT STATUS REG
3904 011444 122000 USTATR
3905 011446 000000 2$: .WORD 0
3906 011450 132737 000200 011446 BITB #RDA,2# ;SEE IF RDA SET YET
3907 011456 001006 BNE 3# ;BR IF YES
3908 011460 020102 CMP R1,R2 ;SEE IF LIMIT EXCEEDED
3909 011462 002762 BLT 1# ;BR IF NOT YET
3910 011464 004537 006176 JSR R5,CKRDA ;GO STACK "RDA NOT SET" MSG
3911 011470 000001 1
3912 011472 103414 BCS 6# ;BR TO EXIT IF ERROR
3913 011474 020165 177776 3$: CMP R1,-2(R5) ;SEE IF LESS THAN REQUIRED CYCLES
3914 011500 002004 BGE 4# ;BR IF NOT
3915 011502 004537 006176 JSR R5,CKRDA ;GO STACK "RDA NOT CLEARED" MSG
3916 011506 000000 0
3917 011510 103405 BCS 6# ;BR TO EXIT IF ERROR
3918 011512 004537 005676 4$: JSR R5,CKRACT ;CHK FOR RXACT = 1
3919 011516 000001 1
3920 011520 103401 BCS 6# ;BR TO EXIT IF ERROR
3921 011522 000241 5$: CLC ;CLEAR C BIT FOR NO ERRORS
3922 011524 012602 6$: MOV (SP)+,R2 ;RESTORE R2
3923 011526 012601 MOV (SP)+,R1 ;RESTORE R1
3924 011530 000205 RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)

```


....ENDTRN -- SHUT DOWN TRANSMITTER/RECEIVER

```

3926 .SBTTL ....ENDTRN -- SHUT DOWN TRANSMITTER/RECEIVER
3927 ;*****
3928 ;* ENDTRN - THIS SUBROUTINE TERMINATES A MESSAGE BY CLEARING TXEN AND RXEN,
3929 ;* CLOCKING THE LINE UNIT FOR THE NUMBER OF CYCLES PASSED IN THE WORD
3930 ;* FOLLOWING THE CALL, AND CHECKING FOR THE USYRT TRANSMITTER AND
3931 ;* RECEIVER TO BE SHUT DOWN.
3932 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3933 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3934 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3935 ;* NOTE: THIS ROUTINE ASSUMES THAT TTLOOP MODE SHOULD BE ENABLED.
3936 ;*
3937 ;* CALLING SEQUENCE :
3938 ;* JSR R5,ENDTRN
3939 ;* <NO. OF CYCLES TO CLOCK>
3940 ;*****
3941 011532 ENDTRN:
3942 011532 012537 011572 MOV (R5)+,2$ ;GET DESIRED NO. OF CYCLES TO CLOCK
3943 011536 004537 005536 JSR R5,CKTACT ;CHK FOR TXACT = 1
3944 011542 000001 1
3945 011544 103422 BCS 6$ ;BR IF ERROR
3946 011546 004537 005676 JSR R5,CKRACT ;CHK FOR RXACT = 1
3947 011552 000001 1
3948 011554 103416 BCS 6$
3949 011556 004537 003734 JSR R5,WRITEI ;CLEAR TXEN AND RXEN IN USYRT
3950 011562 120000 VIAORB ;** AND KEEP TTLOOP ENABLED **
3951 011564 000002 TTLOOP
3952 011566 004537 011614 JSR R5,STEPLU ;CLOCK LU FOR DESIRED NO. OF CYCLES
3953 011572 000000 2$: .WORD 0
3954 011574 004537 005536 JSR R5,CKTACT ;CHK FOR TXACT = 0
3955 011600 000000 0
3956 011602 103403 BCS 6$ ;BR IF ERROR
3957 011604 004537 005676 JSR R5,CKRACT ;CHK FOR RXACT = 0
3958 011610 000000 0
3959 011612 000205 6$: RTS R5
3960
3961

```

....STEPLU -- CLOCK THE USYRT N TIMES

3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976 011614
3977 011614 010146
3978 011616 012501
3979 011620 004537 003734
3980 011624 120005
3981 011626 000000
3982 011630 005301
3983 011632 001372
3984 011634 012601
3985 011636 000205
3986
3987
3988
3989
3990
3991

```

.SBTTL ....STEPLU -- CLOCK THE USYRT N TIMES
;*****
;* STEPLU - THIS SUBROUTINE CLOCKS THE LINE UNIT FOR THE NUMBER OF CYCLES
;* PASSED IN THE WORD FOLLOWING THE CALL. THE VIA ACR MUST BE PREVIOUSLY
;* SET UP FOR T1 ONE-SHOT MODE, AND THE T1 LATCHES MUST BE PREVIOUSLY SET
;* TO CONTROL THE WIDTH OF THE CLOCK PULSE. ALL THAT THIS SUBROUTINE
;* DOES IS TO LOAD 000 INTO THE HI BYTE OF THE T1 COUNTER, FOR THE
;* DESIRED NUMBER OF TIMES.
;*
;* CALLING SEQUENCE :
;* JSR R5,STEPLU
;* .WORD <NUMBER OF CYCLES TO CLOCK>
;*****
STEPLU:
MOV R1,-(SP) ;SAVE R1
MOV (R5)+,R1 ;INIT CYCLE COUNTER
1$: JSR R5,WRITEI ;LOAD TIC-H, START COUNTER, CLOCK 1 CYCLE
VIAT1B
000
DEC R1 ;DECR CYCLE COUNTER
BNE 1$ ;BR IF ALL CYCLES NOT DONE YET
MOV (SP)+,R1 ;RESTORE R1
RTS R5 ;RETURN

```

GLOBAL ERROR REPORT SECTION

```

3993          .SBTTL GLOBAL ERROR REPORT SECTION
3994
3995          ;////////////////////////////////////
3996          ;/ THE GLOBAL ERROR REPORT SECTION CONTAINS ERROR MESSAGES
3997          ;/ THAT ARE USED IN MORE THAN ONE TEST.
3998          ;////////////////////////////////////
3999
4000          .NLIST BEX
4001 011640      045      116      045 ENDEMB: .ASCIZ /#N#N/
4002 011645      045      116      000 NEWLIN: .ASCIZ /#N/          ;USED TO TERMINATE ERROR MESSAGES
4003
4004 011650      045      116      045 FMT2:  .ASCIZ /#N#AFAILING REG = #T#ASEL#01/
4005 011705      045      116      045 FMT3:  .ASCIZ /#N#A EXPECTED: #03#A ACTUAL: #03#A XOR: #03/
4006 011771      045      116      045 FMT4:  .ASCIZ /#N#ATHE CONTENTS OF ALL#T#N#T/
4007 012027      045      116      045 FMT4A: .ASCIZ /#N#S1#03#S5#03#S5#03#S5#03/
4008 012062      045      116      045 FMT4B: .ASCIZ /#N#T/
4009 012067      045      116      045 FMT4C: .ASCIZ /#N#S5#03#S5#03#S5#03#S5#03/
4010 012122      045      116      045 FMT5:  .ASCIZ /#N#A WHEN #03#A LOADED INTO BSEL1/
4011 012165      045      116      045 FMT5A: .ASCIZ /#N#A ATTEMPTING "M-LOOP" FUNCTION CODE #02#A (#T#A)/
4012
4013 012252      045      116      045 FMT10: .ASCIZ /#N#A EXPECTED:#08#A ACTUAL:#08#A XOR:#08/
4014 012326      045      101      040 FMT10A: .ASCIZ /#A LSI ADDR:#08/
4015 012347      045      116      045 FMT11: .ASCIZ /#N#08#08#08#08/
4016 012366      045      101      040 FMT15: .ASCIZ /#A DETECTED IN #T#T#A --/
4017 012420      045      101      040 FMT15A: .ASCIZ /#A DETECTED @ TEST PATTERN ELEMENT # #D2/
4018
4019 012472      045      116      045 FMT19: .ASCIZ /#N#ATEST #D2#A NOT RUN#N/
4020 012523      045      124      045 FMT21: .ASCIZ /#T#06#N/
4021 012533      045      116      045 FMT22: .ASCIZ /#N#AFAILING REG: /
4022 012555      045      101      105 FMT23: .ASCIZ /#AEXPECTED: #03#S5#AACTUAL: #03#S5#AXOR: #03#N/
4023 012634      045      116      045 FMT24: .ASCIZ /#N#T#N#T#N/
4024 012647      045      117      063 FMT25: .ASCIZ /#03#S5#03#S5#03#S5#03#N/
4025 012677      045      123      064 FMT26: .ASCIZ /#S4#03#S5#03#S5#03#S5#03#N/
4026 012732      045      124      045 FMT27: .ASCIZ /#T#T#N/
4027 012741      045      124      045 FMT29: .ASCIZ /#T#N/
4028 012746      045      116      045 FMT30: .ASCIZ /#N#AFOR BAUD RATE SPECIFIED,/
4029 013003      045      116      045 FMT31: .ASCIZ /#N#AIMPROPER CONNECTOR TYPE SPECIFIED/
4030 013051      045      116      045 FMT32: .ASCIZ /#N#AFOR OPTION SPECIFIED,/
4031
4032 013103      122      105      107 EM1:  .ASCIZ /REG NOT INITIALIZED BY MST CLR/
4033 013142      125      123      131 EM2:  .ASCIZ /USYRT NOT INITIALIZED BY PROGRAM RESET/
4034 013211      115      111      103 EM3:  .ASCIZ /MICRO-DIAG. FAILURE/
4035 013235      115      122      104 EM4:  .ASCIZ /MRDY TIMEOUT/
4036 013252      116      125      114 EM5:  .ASCIZ /NULL CLK BIT STUCK AT 0/
4037 013302      116      125      114 EM6:  .ASCIZ /NULL CLK BIT STUCK AT 1/
4038 013332      122      117      122 EM14: .ASCIZ /ROR NOT SET/
4039 013346      122      117      122 EM16: .ASCIZ /ROR NOT CLEARED/
4040 013366      122      105      101 EM25: .ASCIZ 'READ/WRITE DATA ERROR'
4041 013414      111      116      103 EM26: .ASCIZ /INCORRECT DATA CHAR RCV'D/
4042 013446      111      116      103 EM27: .ASCIZ /INCORRECT CRC BYTE RCV'D/
4043 013477      122      123      117 EM28: .ASCIZ /RSOM NOT CLEARED/
4044 013520      122      123      117 EM29: .ASCIZ /RSOM NOT SET/
4045 013535      122      105      117 EM30: .ASCIZ /REOM NOT CLEARED/
4046 013556      122      105      117 EM31: .ASCIZ /REOM NOT SET/
4047 013573      122      103      126 EM34: .ASCIZ /RCV'D DATA MISCOMPARE/
4048 013621      122      105      122 EM35: .ASCIZ /RERR NOT CLEARED/
4049 013642      122      105      122 EM36: .ASCIZ /RERR NOT SET/

```

GLOBAL ERROR REPORT SECTION

4050	013657	122	101	102	EM39:	.ASCIZ	/RABGA NOT CLEARED/
4051	013701	122	101	102	EM40:	.ASCIZ	/RABGA NOT SET/
4052	013717	124	130	040	EM54:	.ASCIZ	/TX UNDERRUN ERROR/
4053	013741	122	105	107	EM66:	.ASCIZ	/REG MISCOMPARE/
4054	013760	125	123	131	EM68:	.ASCIZ	/USYR1 STATUS INCORRECT/
4055	014007	124	130	101	EM69:	.ASCIZ	/TXACT NOT SET/
4056	014025	124	130	101	EM70:	.ASCIZ	/TXACT NOT CLEARED/
4057	014047	122	130	101	EM71:	.ASCIZ	/RXACT NOT SET/
4058	014065	122	130	101	EM72:	.ASCIZ	/RXACT NOT CLEARED/
4059	014107	124	102	115	EM73:	.ASCIZ	/TBMT NOT SET/
4060	014124	124	102	115	EM74:	.ASCIZ	/TBMT NOT CLEARED/
4061	014145	122	104	101	EM75:	.ASCIZ	/RDA NOT SET/
4062	014161	122	104	101	EM76:	.ASCIZ	/RDA NOT CLEARED/
4063	014201	122	123	101	EM77:	.ASCIZ	/RSA NOT SET/
4064	014215	122	123	101	EM78:	.ASCIZ	/RSA NOT CLEARED/
4065	014235	122	105	103	EM90:	.ASCIZ	/RECEIVER OVERRUN NOT SET/
4066	014266	122	105	103	EM91:	.ASCIZ	/RECEIVER OVERRUN NOT CLEARED/
4067	014323	124	102	115	EM92:	.ASCIZ	/TBMT INTERRUPT TEST FAILURE/
4068	014357	124	105	122	EM98:	.ASCIZ	/TERR BIT NOT CLEARED/
4069	014404	124	105	122	EM99:	.ASCIZ	/YERR BIT NOT SET/
4070	014425	124	123	117	EM100:	.ASCIZ	/TSO BIT NOT SET/
4071	014445	124	123	117	EM101:	.ASCIZ	/TSO BIT NOT CLEARED/
4072	014471	124	122	101	EM102:	.ASCIZ	/TRANSMISSION ERROR (AS READ BY TSO BIT)/

4073
4074
4075
4076
4077
4078
4079
4080

.SBTTLTEXT STRINGS FOR ERROR HANDLERS -- "TXT....."

----- TEXT USED BY ERROR HANDLERS -----

4081	014541	102	123	105	TXT1:	.ASCIZ	/BSEL0 BSEL1 BSEL2 BSEL3/
4082	014577	040	040	040	TXT2:	.ASCIZ	/ BSEL4 BSEL5 BSEL6 BSEL7/
4083	014641	102	123	105	TXT2A:	.ASCIZ	/BSEL10 BSEL11 BSEL12 BSEL13/
4084	014700	040	040	040	TXT2B:	.ASCIZ	/ BSEL14 BSEL15 BSEL16 BSEL17/
4085	014743	040	102	131	TXT3:	.ASCIZ	/ BYTE SELECT REG'S ARE:/
4086	014773	040	040	040	TXT4:	.ASCIZ	/ SEL0 SEL2 SEL4 SEL6/
4087	015033	040	040	040	TXT4A:	.ASCIZ	/ SEL10 SEL12 SEL14 SEL16/
4088	015074	102	000		TXT5:	.ASCIZ	/B/
4089	015076	040	123	105	TXT6:	.ASCIZ	/ SELECT REG'S ARE:/
4090	015121	040	122	105	TXT7:	.ASCIZ	/ REGISTERS ORB ORA DORB DORA TICL TICH TILL TILH /
4091	015211	040	040	040	TXT7A:	.ASCIZ	/ T2CL T2CH SR ACR PCR IFR IER ORA /
4092	015301	040	105	130	TXT8:	.ASCIZ	/ EXPECTED: /
4093	015321	040	101	103	TXT9:	.ASCIZ	/ ACTUAL: /
4094	015341	040	130	117	TXT10:	.ASCIZ	/ XOR: /
4095	015361	040	040	116	TXT11:	.ASCIZ	/ N P R R E G I S T E R S:/
4096	015433	040	040	040	TXT11A:	.ASCIZ	/ CONTROL DATA/
4097	015471	040	040	040	TXT11B:	.ASCIZ	/ OUT ADDR. IN ADDR./
4098	015541	104	105	126	TXT12:	.ASCIZ	/DEVICE CSR ADDRESS : /
4099	015567	125	123	131	TXT13:	.ASCIZ	/USYR1 REGS :/
4100	015604	122	104	123	TXT14:	.ASCIZ	/RDSRL RDSRH TDSRL TDSRH/
4101	015642	040	040	040	TXT15:	.ASCIZ	/ PCSARL PCSARH PCR USTAT/
4102	015704	126	111	101	TXT16:	.ASCIZ	/VIA REGS :/
4103	015717	117	122	102	TXT17:	.ASCIZ	/ORB ORA DORB DORA/
4104	015754	040	040	040	TXT18:	.ASCIZ	/ TICL TICH TILL TILH/
4105	016015	124	062	103	TXT19:	.ASCIZ	/T2CL T2CH SR ACR/
4106	016051	040	040	040	TXT20:	.ASCIZ	/ PCR IFR IER ORA/

08

....TEXT STRINGS FOR ERROR HANDLERS -- "TXT..."

```

4107
4108 016111      021      000          TXTNUM: .BYTE      21,0          ;CTL-Q -- THIS (WE HOPE) IS HARMLESS
4109
4110 016113      116      117      120  TXTMLO: .ASCIZ  /NOP/
4111 016117      122      105      101  TXTML1: .ASCIZ  /READ 1 BYTE/
4112 016133      127      122      111  TXTML2: .ASCIZ  /WRITE 1 BYTE/
4113 016150      116      120      122  TXTML3: .ASCIZ  /NPR-OUT 256 BYTES/
4114 016172      116      120      122  TXTML4: .ASCIZ  /NPR-IN 256 BYTES/
4115 016213      123      105      124  TXTML5: .ASCIZ  /SET MICROPROCESSOR'S PC/
4116 016243      125      116      104  TXTML6: .ASCIZ  /UNDEFINED/
4117 016255      101      114      114  TXTML7: .ASCIZ  /ALLOW U-PROCESSOR INTERRUPTS/
4118
4119 016312      126      111      101  TXTVR:  .ASCIZ  /VIA REGISTER /
4120 016330      117      122      102  TXTVR0: .ASCIZ  /ORB/
4121 016334      117      122      101  TXTVR1: .ASCIZ  /ORA/
4122 016340      104      104      122  TXTVR2: .ASCIZ  /ODRB/
4123 016345      104      104      122  TXTVR3: .ASCIZ  /ODRA/
4124 016352      124      061      103  TXTVR4: .ASCIZ  /T1CL/
4125 016357      124      061      103  TXTVR5: .ASCIZ  /T1CH/
4126 016364      124      061      114  TXTVR6: .ASCIZ  /T1LL/
4127 016371      124      061      114  TXTVR7: .ASCIZ  /T1LH/
4128 016376      124      062      103  TXTVR8: .ASCIZ  /T2CL/
4129 016403      124      062      103  TXTVR9: .ASCIZ  /T2CH/
4130 016410      123      122      000  TXTVRA: .ASCIZ  /SR/
4131 016413      101      103      122  TXTVRB: .ASCIZ  /ACR/
4132 016417      120      103      122  TXTVRC: .ASCIZ  /PCR/
4133 016423      111      106      122  TXTVRD: .ASCIZ  /IFR/
4134 016427      111      105      122  TXTVRE: .ASCIZ  /IER/
4135 016433      117      122      101  TXTVRF: .ASCIZ  /ORA/
4136
4137 016437      116      120      122  TXTNP:  .ASCIZ  /NPR /
4138 016444      103      117      116  TXTNP0: .ASCIZ  /CONTROL/
4139 016454      104      101      124  TXTNP1: .ASCIZ  /DATA HI/
4140 016464      104      101      124  TXTNP2: .ASCIZ  /DATA LO/
4141 016474      101      104      104  TXTNP3: .ASCIZ  /ADDR. OUT EX/
4142 016511      101      104      104  TXTNP4: .ASCIZ  /ADDR. OUT HI/
4143 016526      101      104      104  TXTNP5: .ASCIZ  /ADDR. OUT LO/
4144 016543      101      104      104  TXTNP6: .ASCIZ  /ADDR. IN EX/
4145 016557      101      104      104  TXTNP7: .ASCIZ  /ADDR. IN HI/
4146 016573      101      104      104  TXTNP8: .ASCIZ  /ADDR. IN LO/
4147
4148 016607      125      123      131  TXTUR:  .ASCIZ  /USYRT REG /
4149 016622      122      104      123  TXTUR0: .ASCIZ  /RDSRL/
4150 016630      122      104      123  TXTUR1: .ASCIZ  /RDSRH/
4151 016636      124      104      123  TXTUR2: .ASCIZ  /TDSRL/
4152 016644      124      104      123  TXTUR3: .ASCIZ  /TDSRH/
4153 016652      120      103      123  TXTUR4: .ASCIZ  /PCSARL/
4154 016661      120      103      123  TXTUR5: .ASCIZ  /PCSARH/
4155 016670      120      103      122  TXTUR6: .ASCIZ  /PCR/
4156 016674      125      123      124  TXTUR7: .ASCIZ  /USTAT/
4157
4158
4159
4160
4161
4162
4163

```

.SBTTLTEXT ADDRESS TABLES FOR ERROR HANDLERS "TXT..."
.....
----- TEXT ADDRESS TABLES USED BY ERROR HANDLERS -----

....TEXT ADDRESS TABLES FOR ERROR HANDLERS -- "TXT_T"

```

4164
4165
4166 016702 016113 016117 016133 XTMLT: .WORD TXTML0,TXTML1,TXTML2,TXTML3,TXTML4,TXTML5,TXTML6,TXTML7
      016710 016150 016172 016213
      016716 016243 016255
4167
4168 016722 016312
4169 016724 016330 016334 016340 TXTVRT: .WORD TXTVR0,TXTVR1,TXTVR2,TXTVR3,TXTVR4,TXTVR5,TXTVR6,TXTVR7
      016732 016345 016352 016357
      016740 016364 016371
4170 016744 016376 016403 016410 .WORD TXTVR8,TXTVR9,TXTVRA,TXTVRB,TXTVRC,TXTVRD,TXTVRE,TXTVRF
      016752 016413 016417 016423
      016760 016427 016433
4171
4172 016764 016437
4173 016766 016444 016454 016464 TXTNPT: .WORD TXTNP0,TXTNP1,TXTNP2,TXTNP3,TXTNP4,TXTNP5,TXTNP6,TXTNP7,TXTNP8
      016774 016474 016511 016526
      017002 016543 016557 016573
4174 017010 016622 016630 016636 TXTURT: .WORD TXTUR0,TXTUR1,TXTUR2,TXTUR3,TXTUR4,TXTUR5,TXTUR6,TXTUR7
      017016 016644 016652 016661
      017024 016670 016674
4175
4176

```

....TEXT ADDRESS TABLES FOR ERROR HANDLERS -- "TXT_T"

```

4178
4179
4180
4181 017030
017030
4182 017030 105037 002401
4183 017034 010145
4184 017036 013701 002400
4185 017042 022701 000017
4186 017046 002012
4187 017050
017050 010146
017052 012746 012122
017056 012746 000002
017062 010600
017064 104415
017066 062706 000006
4188 017072 000424
4189
4190 017074 001001
4191 017076 005001
4192 017100 022701 000007
4193 017104 002002
4194 017106 012701 000006
4195 017112 006301
4196 017114
017114 016146 016702
017120 013746 002400
017124 012746 012165
017130 012746 000003
017134 010600
017136 104415
017140 062706 000010
4197
4198 017144 012601
4199 017146 004737 020160
4200 017152
017152
017152 104423
4201
4202
4203
4204 017154
017154
4205 017154
017154 013746 002412
017160 012746 015074
017164 012746 011650
017170 012746 000003
017174 010600
017176 104414
017200 062706 000010
4206 017204 004737 020134
4207 017210
017210 013746 002404
017214 013746 002402
017220 013746 002400

```

```

-----
:SBTTL ....ERROR HANDLER -- ERR4 -- M-LOOP TIMEOUT ERROR HANDLING
-----
          BGNMSG  ERR4
          ERR4::
          ;MAKE SURE BIT 8 DOESN'T PRINT!
          ;SAVE THE WORKING REGISTER
          ;SAVE THIS FOR LATER
          ;WAS THIS AN M-LOOP REQUEST?
          ;YES, THEN REPORT THE FUNCTION CODE
          ;NO, THEN IT MUST BE A BSEL1 SETTING
          MOV      R1,-(SP)
          MOV      @FMT5,-(SP)
          MOV      @2,-(SP)
          MOV      SP,R0
          TRAP    C#PNTX
          ADD     @6,SP
          BR      20$
5$:      BNE     6$
          CLR    R1
          ;IF IT WAS A 17, THIS IS A "NOP" AND
          ; THE TEXT POINTER MUST SO REFLECT.
6$:      CMP    @7,R1
          ;IS FUNCTION CODE > 7?
          BGE    7$
          ;NO, THEN WE CAN HANDLE IT
          MOV    @6,R1
          ;YES, THAN IT'S UNDEFINED -- SAY 50
7$:      ASL    R1
          ;CONVERT TO A WORD OFFSET
          PRINTX @FMT5A,GDATA,TEXTMLT(R1) ;REPORT THE FAILING FUNCTION
          MOV    TEXTMLT(R1),-(SP)
          MOV    GDATA,-(SP)
          MOV    @FMT5A,-(SP)
          MOV    @3,-(SP)
          MOV    SP,R0
          TRAP  C#PNTX
          ADD   @10,SP
20$:     MOV    (SP)+,R1
          JSR   PC,ERR5$
          ;RESTORE THE WORKING REGISTER
          ;DUMP THE SELECT REGISTERS
          ENDMSG
          L10002:
          TRAP  C#MSG
-----
:SBTTL ....ERROR HANDLER -- ERR5 -- WORD SELECT REG. ERRORS
-----
          BGNMSG  ERR5
          ERR5::
          PRINTB @FMT2,@TX15,REGNUM
          MOV    REGNUM,-(SP)
          MOV    @TX15,-(SP)
          MOV    @FMT2,-(SP)
          MOV    @3,-(SP)
          MOV    SP,R0
          TRAP  C#PNTB
          ADD   @10,SP
          JSR   PC,XORGB
          PRINTB @FMT10,GDATA,BDATA,XDATA
          MOV    XDATA,-(SP)
          MOV    BDATA,-(SP)
          MOV    GDATA,-(SP)

```

...ERROR HANDLER -- ERR5 - WORD SELECT REG. ERRORS

```

017224 012746 012252                                MOV    #FMT10,-(SP)
017230 012746 000004                                MOV    #4,-(SP)
017234 010600                                        MOV    SP,RO
017236 104414                                        TRAP   C#PNTB
017240 062706 000012                                ADD    #12,SP
4208 017244 004737 020160                JSR    PC,ERR5$                ;DUMP THE SELECT REGISTERS
4209 017250                                ENDMSG
                                        L10003:
017250 104423                                TRAP   C#MSG
4210
4211 ;-----
4212 ;SBTTL ...ERROR HANDLER -- ERR7A -- USYRT REGISTER ERRORS
4213 ;-----
017252                                BGNMSG  ERR7A
017252                                ERR7A::
4214 017252 113701 002412                MOVB   REGNUM,R1
4215 017256 006301                                ASL   R1                ;AS PASSED, THIS WAS A BYTE OFFSET
4216 017260                                PRINTB #FMT15,#TXTUR,TXTURT(R1)
017260 016146 017010                                MOV    TXTURT(R1),-(SP)
017264 012746 016607                                MOV    #TXTUR,-(SP)
017270 012746 012366                                MOV    #FMT15,-(SP)
017274 012746 000003                                MOV    #3,-(SP)
017300 010600                                        MOV    SP,RO
017302 104414                                        TRAP   C#PNTB
017304 062706 000010                                ADD    #10,SP
4217 017310 004737 020134                JSR    PC,XORGB
4218 017314                                PRINTB #FMT3,GDATA,BDATA,XDATA
017314 013746 002404                                MOV    XDATA,-(SP)
017320 013746 002402                                MOV    BDATA,-(SP)
017324 013746 002400                                MOV    GDATA,-(SP)
017330 012746 011705                                MOV    #FMT3,-(SP)
017334 012746 000004                                MOV    #4,-(SP)
017340 010600                                        MOV    SP,RO
017342 104414                                        TRAP   C#PNTB
017344 062706 000012                                ADD    #12,SP
4219 017350                                PRINTB #ENDEMB
017350 012746 011640                                MOV    #ENDEMB,-(SP)
017354 012746 000001                                MOV    #1,-(SP)
017360 010600                                        MOV    SP,RO
017362 104414                                        TRAP   C#PNTB
017364 062706 000004                                ADD    #4,SP
4220 017370                                ENDMSG
                                        L10004:
017370 104423                                TRAP   C#MSG

```

```

4221 ;-----
4222 ;SBTTL ...ERROR HANDLER -- ERR10 -- USYRT REG ERROR (XOR, REG PRINTOUT)
4223 ;-----
4224
4225                                BGNMSG  ERR10
017372                                ERR10::
4226 017372                                PRINTB #FMT21,#TXT12,MPCSR
017372 013746 002472                                MOV    MPCSR,-(SP)
017376 012746 015541                                MOV    #TXT12,-(SP)
017402 012746 012523                                MOV    #FMT21,-(SP)
017406 012746 000003                                MOV    #3,-(SP)
017412 010600                                        MOV    SP,RO
017414 104414                                        TRAP   C#PNTB
017416 062706 000010                                ADD    #10,SP

```


....ERROR HANDLER -- ERR10 -- USYRT REG ERROR (XOR, REG PRINTO

```

4227 017422          PRINTB  #FMT22
      017422 012746 012533          MOV    #FMT22,-(SP)
      017426 012746 000001          MOV    #1,(SP)
      017432 010600          MOV    SP,RO
      017434 104414          TRAP   C#PNTB
      017436 062706 000004          ADD    #4,SP
4228 017442          MOV    REGNUM,R1
4229 017446 006301          ASL    R1          ;GET PTR TO USYRT REG ASCII
4230 017450          PRINTB  #FMT27,#TXTUR,TXTURT(R1)
      017450 016146 017010          MOV    TXTURT(R1),-(SP)
      017454 012746 016607          MOV    #TXTUR,-(SP)
      017460 012746 012732          MOV    #FMT27,-(SP)
      017464 012746 000003          MOV    #3,-(SP)
      017470 010600          MOV    SP,RO
      017472 104414          TRAP   C#PNTB
      017474 062706 000010          ADD    #10,SP
4231 017500          JSR    PC,XORGB          ;COMPUTE XOR OF GOOD AND BAD DATA
4232 017504          PRINTB  #FMT23,GDATA,BDATA,XDATA
      017504 013746 002404          MOV    XDATA,-(SP)
      017510 013746 002402          MOV    BDATA,-(SP)
      017514 013746 002400          MOV    GDATA,-(SP)
      017520 012746 012555          MOV    #FMT23,-(SP)
      017524 012746 000004          MOV    #4,-(SP)
      017530 010600          MOV    SP,RO
      017532 104414          TRAP   C#PNTB
      017534 062706 000012          ADD    #12,SP
4233 017540          JSR    PC,ERR12#          ;GET & PRINT USYRT REGISTERS
4234 017544          ENDMSG
      017544          L10005:
      017544 104423          TRAP   C#MSG

```

;SBTTLERROR HANDLER -- ERR11 -- VIA REG ERROR (XOR, REG PRINTOUT)

```

4240 017546          BGNMSG  ERR11
      017546          ERR11::
4241 017546          PRINTB  #FMT21,#TXT12,MPCSR
      017546 013746 002472          MOV    MPCSR,-(SP)
      017552 012746 015541          MOV    #TXT12,-(SP)
      017556 012746 012523          MOV    #FMT21,-(SP)
      017562 012746 000003          MOV    #3,-(SP)
      017566 010600          MOV    SP,RO
      017570 104414          TRAP   C#PNTB
      017572 062706 000010          ADD    #10,SP
4242 017576          PRINTB  #FMT22
      017576 012746 012533          MOV    #FMT22,-(SP)
      017602 012746 000001          MOV    #1,-(SP)
      017606 010600          MOV    SP,RO
      017610 104414          TRAP   C#PNTB
      017612 062706 000004          ADD    #4,SP
4243 017616          MOV    REGNUM,R1
4244 017622 006301          ASL    R1          ;GET PTR TO VIA REG ASCII
4245 017624          PRINTB  #FMT27,#TXTVR,TXTVRT(R1)
      017624 016146 016724          MOV    TXTVRT(R1),-(SP)
      017630 012746 016312          MOV    #TXTVR,-(SP)
      017634 012746 012732          MOV    #FMT27,-(SP)

```

...ERROR HANDLER -- ERR11 -- VIA REG ERROR (XOR, REG PRINTOUT)

```

017640 012746 000003                                MOV     #3,-(SP)
017644 010600                                MOV     SP,RO
017646 104414                                TRAP   C$PNTB
017650 062706 000010                                ADD    #10,SP
4246 017654 004737 020134          JSR     PC,XORGB          ;COMPUTE XOR OF GOOD AND BAD DATA
4247 017660          PRINTB  #FMT23,GDATA,BDATA,XDATA
017660 013746 002404                                MOV     XDATA,-(SP)
017664 013746 002402                                MOV     BDATA,-(SP)
017670 013746 002400                                MOV     GDATA,-(SP)
017674 012746 012555                                MOV     #FMT23,-(SP)
017700 012746 000004                                MOV     #4,-(SP)
017704 010600                                MOV     SP,RO
017706 104414                                TRAP   C$PNTB
017710 062706 000012                                ADD    #12,SP
4248 017714 004737 020356          JSR     PC,ERR11$        ;GET & PRINT VIA REGISTERS
4249 017720          ENDMMSG
017720          L10006:
017720 104423          TRAP   C$MSG
4250
4251
4252          ;-----
4253          ;SBTTL ...ERROR HANDLER -- ERR12 -- USYRT REG ERROR (USYRT PRINTOUT)
4254          ;-----
4255 017722          BGNMSG  ERR12
017722          PRINTB  #FMT21,#TXT12,MPCSR          ERR12::
4256 017722 013746 002472                                MOV     MPCSR,-(SP)
017726 012746 015541                                MOV     #TXT12,-(SP)
017732 012746 012523                                MOV     #FMT21,-(SP)
017736 012746 000003                                MOV     #3,-(SP)
017742 010600                                MOV     SP,RO
017744 104414                                TRAP   C$PNTB
017746 062706 000010                                ADD    #10,SP
4257 017752          PRINTB  #FMT22
017752 012746 012533                                MOV     #FMT22,-(SP)
017756 012746 000001                                MOV     #1,-(SP)
017762 010600                                MOV     SP,RO
017764 104414                                TRAP   C$PNTB
017766 062706 000004                                ADD    #4,SP
4258 017772 013701 002412          MOV     REGNUM,R1
4259 017776 006301          ASL    R1          ;GET PTR TO USYRT REG ASCII
4260 020000          PRINTB  #FMT27,#TXTUR,TXTURT(R1)
020000 016146 017010                                MOV     TXTURT(R1),-(SP)
020004 012746 016607                                MOV     #TXTUR,-(SP)
020010 012746 012732                                MOV     #FMT27,-(SP)
020014 012746 000003                                MOV     #3,-(SP)
020020 010600                                MOV     SP,RO
020022 104414                                TRAP   C$PNTB
020024 062706 000010                                ADD    #10,SP
4261 020030 004737 020710          JSR     PC,ERR12$        ;GET & PRINT USYRT REGISTERS
4262 020034          ENDMMSG
020034          L10007:
020034 104423          TRAP   C$MSG
4263
4264          ;-----
4265          ;SBTTL ...ERROR HANDLER -- ERR13 -- TRANSMISSION/ISO ERROR (XOR, REG PRINTOUT)
4266          ;-----

```

...ERROR HANDLER -- ERR13 -- TRANSMISSION/TSD ERROR (XOR, REG

```

4267
4268 020036          BGNMSG  ERR13
020036
4269 020036          PRINTB  #FMT21,#TXT12,MPCSR
020036 013746 002472
020042 012746 015541
020046 012746 012523
020052 012746 000003
020056 010600
020060 104414
020062 062706 000010
4270 020066 004737 020134      JSR      PC,XORGB          ;COMPUTE XOR OF GOOD AND BAD DATA
4271 020072          PRINTB  #FMT23,GDATA,BDATA,XDATA
020072 013746 002404
020076 013746 002402
020102 013746 002400
020106 012746 012555
020112 012746 000004
020116 010600
020120 104414
020122 062706 000012
4272 020126 004737 020710      JSR      PC,ERR12$        ;GET & PRINT USYRT REGISTERS
4273 020132          ENDMMSG
020132
020132 104423          L10010: TRAP  C$MSG
4274
4275
4276          .SBTTL  ....ERROR HANDLER SUBROUTINES
4277          ;-----
4278          ;----- SUBROUTINES USED ONLY BY ERROR HANDLERS -----
4279          ;-----
4280
4281          .SBTTL  ....ERROR HANDLER SUBROUTINE -- XORGB
4282
4283          ;      PERFORM EXCLUSIVE OR BETWEEN "GDATA" & "BDATA" PUTTING
4284          ;      THE RESULT IN "XDATA"
4285
4286 020134 010146          XORGB: MOV      R1,-(SP)          ;PRESERVE WORKING REGISTER
4287 020136 013701 002400      MOV      GDATA,R1          ;GET "GOOD" DATA
4288 020142 013737 002402 002404  MOV      BDATA,XDATA      ;AND "BAD" DATA
4289 020150 074177 002404      XOR      R1,XDATA          ;PERFORM EXCLUSIVE OR
4290 020154 012600          MOV      (SP)+,R1          ;RESTORE R1
4291 020156 000207          RTS      PC              ;RETURN
4292
4293
4294          .SBTTL  ....ERROR HANDLER SUBROUTINE -- ERR5$
4295          ;-----
4296          ;      COMMON ERROR SUBROUTINE TO PRINT SELECT REGISTERS
4297 020160          ERR5$:
4298 020160          PRINTX  #FMT4,#TXT6,#TXT4
020160 012746 014773
020164 012746 015076
020170 012746 011771
020174 012746 000003
020200 010600
020202 104415
020204 062706 000010

```

.....ERROR HANDLER SUBROUTINE -- ERR5\$

```

4299 020210          PRINTX  #FMT11,WSR0,WSR2,WSR4,WSR6 ;DUMP THE SELECT REGISTERS
      020210 013746 002264          MOV      WSR6,-(SP)
      020214 013746 002262          MOV      WSR4,-(SP)
      020220 013746 002260          MOV      WSR2,-(SP)
      020224 013746 002256          MOV      WSR0,-(SP)
      020230 012746 012347          MOV      #FMT11,-(SP)
      020234 012746 000005          MOV      #5,-(SP)
      020240 010600          MOV      SP,RO
      020242 104415          TRAP     C#PNTX
      020244 062706 000014          ADD      #14,SP
4300 020250          PRINTX  #FMT4B,#TXT4A
      020250 012746 015033          MOV      #TXT4A,-(SP)
      020254 012746 012062          MOV      #FMT4B,-(SP)
      020260 012746 000002          MOV      #2,-(SP)
      020264 010600          MOV      SP,RO
      020266 104415          TRAP     C#PNTX
      020270 062706 000006          ADD      #6,SP
4301 020274          PRINTX  #FMT11,WSR10,WSR12,WSR14,WSR16 ;DUMP THE SELECT REGISTERS
      020274 013746 002274          MOV      WSR16,-(SP)
      020300 013746 002272          MOV      WSR14,-(SP)
      020304 013746 002270          MOV      WSR12,-(SP)
      020310 013746 002266          MOV      WSR10,-(SP)
      020314 012746 012347          MOV      #FMT11,-(SP)
      020320 012746 000005          MOV      #5,-(SP)
      020324 010600          MOV      SP,RO
      020326 104415          TRAP     C#PNTX
      020330 062706 000014          ADD      #14,SP
4302 020334          PRINTB  #ENDEMB
      020334 012746 011640          MOV      #ENDEMB,-(SP)
      020340 012746 000001          MOV      #1,-(SP)
      020344 010600          MOV      SP,RO
      020346 104414          TRAP     C#PNTB
      020350 062706 000004          ADD      #4,SP
4303 020354          RTS      PC
4304
4305
4306          ;-----
4307          ;SBTTL .....ERROR HANDLER SUBROUTINE -- ERR11$
4308          ;-----
4309          ;      COMMON ERROR SUBROUTINE TO GET/PRINT VIA REGISTERS
4310 020356 004737 004502  ERR11$: JSR      PC,GETVRS          ;GET VIA REGS FOR PRINTOUT
4311 020362          PRINTX  #FMT24,#TXT16,#TXT17
      020362 012746 015717          MOV      #TXT17,-(SP)
      020366 012746 015704          MOV      #TXT16,-(SP)
      020372 012746 012634          MOV      #FMT24,-(SP)
      020376 012746 000003          MOV      #3,-(SP)
      020402 010600          MOV      SP,RO
      020404 104415          TRAP     C#PNTX
      020406 062706 000010          ADD      #10,SP
4312 020412          PRINTX  #FMT25,VREGS+0,VREGS+2,VREGS+4,VREGS+6
      020412 013746 002344          MOV      VREGS+6,-(SP)
      020416 013746 002342          MOV      VREGS+4,-(SP)
      020422 013746 002340          MOV      VREGS+2,-(SP)
      020426 013746 002336          MOV      VREGS+0,-(SP)
      020432 012746 012647          MOV      #FMT25,-(SP)
      020436 012746 000005          MOV      #5,-(SP)
      020442 010600          MOV      SP,RO

```

.....ERROR HANDLER SUBROUTINE -- ERR11\$

```

020444 104415
020446 062706 000014
4313 020452 PRINTX #FMT29 #TXT18
020452 012746 015754
020456 012746 012741
020462 012746 000002
020466 010600
020470 104415
020472 062706 000006
4314 020476 PRINTX #FMT26,VREGS+8.,VREGS+10.,VREGS+12.,VREGS+14.
020476 013746 002354
020502 013746 002352
020506 013746 002350
020512 013746 002346
020516 012746 012677
020522 012746 000005
020526 010600
020530 104415
020532 062706 000014
4315 020536 PRINTX #FMT29,#TXT19
020536 012746 016015
020542 012746 012741
020546 012746 000002
020552 010600
020554 104415
020556 062706 000006
4316 020562 PRINTX #FMT25,VREGS+16.,VREGS+18.,VREGS+20.,VREGS+22.
020562 013746 002364
020566 013746 002362
020572 013746 002360
020576 013746 002356
020602 012746 012647
020606 012746 000005
020612 010600
020614 104415
020616 062706 000014
4317 020622 PRINTX #FMT29,#TXT20
020622 012746 016051
020626 012746 012741
020632 012746 000002
020636 010600
020640 104415
020642 062706 000006
4318 020646 PRINTX #FMT26,VREGS+24.,VREGS+26.,VREGS+28.,VREGS+30.
020646 013746 002374
020652 013746 002372
020656 013746 002370
020662 013746 002366
020666 012746 012677
020672 012746 000005
020676 010600
020700 104415
020702 062706 000014
4319 020706 000207
4320
4321
4322
RTS PC
;.....
;SBITL .....ERROR HANDLER SUBROUTINE -- ERR12$

```

.....ERROR HANDLER SUBROUTINE -- ERR12\$

```

4323 ;-----
4324 ;      COMMON ERROR ROUTINE TO GET AND PRINTOUT USYRT REGISTERS
4325 ;
4326 020710 C04737 004402 ERR12$: JSR      PC,GETURS      ;GET USYRT REGS FOR PRINTOUT
4327 020714 PRINTX  #FMT24,#TXT13,#TXT14
                                MOV      #TXT14,-(SP)
                                MOV      #TXT13,-(SP)
                                MOV      #FMT24,-(SP)
                                MOV      #3,-(SP)
                                MOV      SP,R0
                                TRAP     C$PNTX
                                ADD      #10,SP
4328 020744 PRINTX  #FMT25,UREGS+0,UREGS+2,UREGS+4,UREGS+6
                                MOV      UREGS+6,-(SP)
                                MOV      UREGS+4,-(SP)
                                MOV      UREGS+2,-(SP)
                                MOV      UREGS+0,-(SP)
                                MOV      #FMT25,-(SP)
                                MOV      #5,-(SP)
                                MOV      SP,R0
                                TRAP     C$PNTX
                                ADD      #14,SP
                                MOV      #TXT15,-(SP)
                                MOV      #FMT29,-(SP)
                                MOV      #2,-(SP)
                                MOV      SP,R0
                                TRAP     C$PNTX
                                ADD      #6,SP
4329 021004 PRINTX  #FMT29,#TXT15
                                MOV      #TXT15,-(SP)
                                MOV      #FMT29,-(SP)
                                MOV      #2,-(SP)
                                MOV      SP,R0
                                TRAP     C$PNTX
                                ADD      #6,SP
4330 021030 PRINTX  #FMT26,UREGS+10,UREGS+12,UREGS+14,UREGS+16
                                MOV      UREGS+16,-(SP)
                                MOV      UREGS+14,-(SP)
                                MOV      UREGS+12,-(SP)
                                MOV      UREGS+10,-(SP)
                                MOV      #FMT26,-(SP)
                                MOV      #5,-(SP)
                                MOV      SP,R0
                                TRAP     C$PNTX
                                ADD      #14,SP
4331 021064 062706 000014
4332 021070 000207
4333 RTS      PC

```

LOAD DEVICE PROTECTION TABLE

4335
 4336
 4337
 4338
 4339
 4340
 4341
 4342 021072
 021072
 4343 021072 177777
 4344 021074 177777
 4345 021076 177777
 4346 021100

```

.SBTTL LOAD DEVICE PROTECTION TABLE
;////////////////////////////////////
;/ THIS TABLE IDENTIFIES THE LOAD DEVICE TO THE SUPERVISOR, SO THAT IT CAN BE
;/ PROTECTED FROM TESTING, IF DESIRED.
;////////////////////////////////////

      BGNPROT
      .WORD -1      ;DON'T CHK CSR ADRS
      .WORD -1      ;DON'T CHK MASSBUS UNIT NO.
      .WORD -1      ;DON'T CHK DRIVE NO.
      ENDPROT
      L$PROT::

```

INITIALIZE SECTION

```

4348          .SBTTL  INITIALIZE SECTION
4349
4350          ;////////////////////////////////////
4351          ;// THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
4352          ;// AT THE BEGINNING OF THE TEST SEQUENCE ON THE NEXT UNIT.
4353          ;////////////////////////////////////
4354
4355 021100          BGNINIT
4356          021100          L.$INIT::
4357 021100          SETVEC  #140,#170000,#340          ;ODT ROM ADDRESS          ;JB REV A-0
          021100 012746 000340          MOV          #340,-(SP)
          021104 012746 170000          MOV          #170000,-(SP)
          021110 012746 000140          MOV          #140,-(SP)
          021114 012746 000003          MOV          #3,-(SP)
          021120 104437          TRAP          C$SVEC
          021122 062706 000010          ADD          #10,SP
4358
4359
4360 021126 010637 002414          MOV          SP,PSTACK          ;SAVE BASE-LEVEL STACK POINTER
4361 021132 005037 002420          CLR          SUBRPC          ;CLEAR SUBR CALL PC
4362 021136 005037 002454          CLR          CHPTYP          ;CLEAR USYRT CHIP TYPE INDICATOR
4363 021142 005037 002452          CLR          ERROR1          ;CLEAR ERROR FLAG
4364 021146 005037 002456          CLR          SAVLEN          ;CLEAR CHAR LENGTH FROM SETUP
4365 021152 005737 002444          TST          FRSTIM          ;SEE IF FIRST TIME THROUGH AFTER LOAD
4366 021156 001007          BNE          6$          ;BR IF NOT
4367 021160 013737 000004 002446          MOV          @#4,SAVE4          ;SAVE ERROR TRAP VECTOR
4368 021166 013737 000006 002450          MOV          @#6,SAVE6
4369 021174 000406          BR          9$
4370
4371 021176 013737 002446 000004 6$:          MOV          SAVE4,@#4          ;RESTORE ERROR TRAP VECTOR
4372 021204 013737 002450 000006          MOV          SAVE6,@#6
4373
4374 021212 012737 000001 002444 9$:          MOV          #1,FRSTIM          ;MARK FLAG FOR NEXT TIME THROUGH
4375
4376          ;SEE IF PROGRAM JUST STARTED, BR IF YES
4377 021220          READEF  #EF.START
          021220 012700 000040          MOV          #EF.START,R0
          021224 104447          TRAP          C$REFG
4378 021226          BCOMPLETE          STARST          BCS          STARST
          021226 103415
4379
4380          ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
4381 021230          READEF  #EF.RESTART
          021230 012700 000037          MOV          #EF.RESTART,R0
          021234 104447          TRAP          C$REFG
4382 021236          BCOMPLETE          STARST          BCS          STARST
          021236 103411
4383
4384          ;SEE IF THIS IS A NEW PASS, BR IF YES
4385 021240          READEF  #EF.NEW
          021240 012700 000035          MOV          #EF.NEW,R0
          021244 104447          TRAP          C$REFG
4386 021246          BCOMPLETE          NEWST          BCS          NEWST
          021246 103411
4387
4388          ;SEE IF PROGRAM WAS JUST CONTINUED

```


INITIALIZE SECTION

```

4389 021250          READFF 0FF,CONTINUE
      021250 012700 000036
      021254 104447
4390 021256          BCOMPLETE      ENDIT
      021256 103473
4391 021260          BR      GETPRM
4392
4393 021262          STARST:
4394 021262 005037 002466          CLR      STARES          ;CLEAR FLAG TO SHOW JUST HAD STA OR RES
4395
4396          ;CLEAR DEVICE MAP
4397 021266 005037 002460          CLR      DEVMAP
4398 021272          NEWST:
4399 021272 012737 177777 002410          MOV      0-1,LOGDEV          ;RESET LOGICAL DEVICE TO -1
4400 021300 005237 002466          INC      STARES          ;INCREMENT NO. OF PASSES SINCE STA OR RES
4401 021304 012737 000001 002462          MOV      0BIT0,DEVPTR          ;INIT DEVICE MAP BIT POINTER
4402
4403          ; GET UNIBUS ADDRESS, VECTOR, PRIORITY LEVEL, SWITCH PACKS, TEST
4404          ; CONNECTOR INFORMATION FOR THIS LOGICAL DEVICE
4405 021312          GETPRM:
4406 021312 005237 002410          INC      LOGDEV          ;INCREMENT LOGICAL DEVICE NUMBER
4407 021316          GPHARD LOGDEV,R1          ;GET P-TABLE POINTER INTO R1
      021316 013700 002410          MOV      LOGDEV,R0
      021322 104442          TRAP   C1GPHRD
      021324 010001          MOV      R0,R1
4408 021326          BCOMPLETE      10#          ;BR IF DEVICE AVAILABLE
      021326 103403
4409 021330 006337 002462          ASL      DEVPTR          ;SHIFT DEVICE POINTER
4410 021334 000766          BR      GETPRM          ;SKIP THIS DEVICE
4411 021336 053737 002462 002460 10#:          BIS      DEVPTR,DEVMAP          ;SET BIT FOR THIS DEVICE
4412 021344 006337 002462          ASL      DEVPTR          ;SHIFT BIT POINTER
4413
4414 021350 012102          MOV      (R1),R2          ;R2=CSR ADDR VALUE
4415 021352 012703 002472          MOV      0MPCSR,R3          ;R3=POINTER TO CSR ADDR STORAGE AREA
4416
4417 021356 010223          11#:          MOV      R2,(R3)          ;PUT CSR ADDRESSES IN 'BSEL' AREA
4418 021360 005202          INC      R2          ;BUMP BSEL ADDR
4419 021362 022703 002532          CMP      0BSEL17+2,R3          ;ALL 16 ADDRESSES MOVED ?
4420 021366 001373          BNE     11#          ;NO: DO ANOTHER ADDRESS
4421          ;YES: CONTINUE
4422
4423 021370 011137 002532          MOV      (R1),MPIVEC          ;GET DMV11 INPUT INTRPT VECTOR
4424 021374 012137 002534          MOV      (R1),MPOVEC
4425 021400 062737 000004 002534          ADD      04,MPOVEC          ;GET DMV11 OUTPUT INTRPT VECTOR
4426 021406 012137 002536          MOV      (R1),MPRIOR          ;GET DMV11 DEVICE PRIORITY
4427 021412 012137 002540          MOV      (R1),LUSWI1          ;GET LU SWITCH PACK #1
4428 021416 012137 002542          MOV      (R1),LUSWI2          ;GET LU SWITCH PACK #2
4429 021422 012137 002544          MOV      (R1),BRDTYP          ;GET DMV-11 BOARD TYPE
4430 021426 012137 002546          MOV      (R1),TSTCON          ;GET TEST CONNECTOR INDICATOR
4431 021432 011137 002550          MOV      (R1),BDRATE          ;GET BAUD RATE FOR THIS DEVICE
4432          ;ISSUE LSI BUS RESET, TO INIT DMV11
4433 021436          BRESET
      021436 104433
4434 021440 005000          TRAP   C1RESET
4435 021442 000240          15#:          CLR      R0          ;0 TIME DELAY TO ALLOW COMPLETION
4436 021444 077002          NOP
4437 021446          SOB     R0,15#          ;0 OF DMV11 MICRODIAGNOSTICS.
          ENDIT:

```

(11)

INITIALIZE SECTION

4438 021446
021446
021446 104411

ENDINIT

L10912: TRAP C:INIT

AUTO DROP UNIT SECTION

```

4440          .SBTTL AUTO DROP UNIT SECTION
4441
4442          ;////////////////////////////////////
4443          ;/ THE AUTO DROP CODING DETERMINES WHETHER OR NOT THE DEVICE WHOSE P-TABLE
4444          ;/ WAS JUST OBTAINED IS READY FOR TESTING, AND IT IS DROPPED IF NOT READY.
4445          ;////////////////////////////////////
4446
4447          ;*****
4448          ;
4449          ; THIS ALGORITHM IS THE SAME A CNOMA TEST # 1 EXCEPT THAT TEST
4450          ; WILL JUST REPORT THE FAILURE AND GO ON - THIS ROUTINE WILL CAUSE THE
4451          ; DEVICE TO BE DROPPED IF A BUS-TIMEOUT OCCURS WHEN ANY OF THE CSR'S
4452          ; ARE ACCESSED WITH EITHER A "TST" OR "TSTB" INSTRUCTION.
4453          ;
4454          ;-----*****
4455
4456          021450          BGNAUTO
4457
4458          021450          L$AUTO::
4459          021450          012746 000000          SETVEC  #4,#AD.HIT,#0 ;SETUP INVALID-ADDRESS TRAP VECTOR
4460          021454          012746 021566          MOV      #0,-(SP)
4461          021460          012746 000004          MOV      #AD.HIT,-(SP)
4462          021464          012746 000003          MOV      #4,-(SP)
4463          021470          104437          MOV      #3,-(SP)
4464          021472          062706 000010          TRAP    C$SVEC
4465          021476          005037 002622          ADD     #10,SP
4466          021502          012702 000001          CLR     TMP0 ;INITIALIZE TRAP FLAG REGISTER
4467          021506          013703 002472          MOV     #1,R2 ;FLAG BIT
4468
4469          021512          105723          MOV     BSELO,R3 ;INIT ADDRESS POINTER
4470          021514          006302          1$: TSTB  (R3)+ ;ACCESS THE CSR'S BY BYTES.
4471          021516          103375          ASL    R2
4472
4473          021520          013703 002472          BCC    1$
4474          021524          012702 000001          MOV     BSELO,R3 ;RE-INIT ADDRESS POINTER
4475          021530          005723          MOV     #1,R2 ;RE-INIT FLAG BIT
4476          021532          006302          2$: TST  (R3)+ ;ACCESS THE CSR'S BY WORDS.
4477          021534          006302          ASL    R2
4478          021536          103374          ASL    R2
4479          021540          012700 000004          BCC    2$
4480          021544          104436          CLRVEC #4 ;RESTORE THE VECTOR TO DS
4481          021546          005737 002622          MOV     #4,R0
4482          021552          001403          TRAP   C$CVEC
4483          021554          013700 002410          TST    TMP0 ;DID WE GET HIT WITH AN INVALID ADDRESS TRAP?
4484          021560          104451          BEQ    AD.OK ;NO, EXIT TEST
4485
4486          021562          000240          DODU   LOGDEV ;YES, DROP THIS LOGICAL DEV.
4487
4488          021564          104461          MOV     LOGDEV,R0
4489          021566          050237 002622          TRAP   C$DODU
4490
4491          AD.OK: NOP ;(FOR PATCHING IN A HALT IF NECESSARY)
4492
4493          ENDAUTO
4494
4495          L10013: TRAP C$AUTO
4496
4497          AD.HIT: BIS R2,TMP0 ;FLAG THE HIT IF WE GET IT!

```

E9

CNDMCAO DMV11 LINE UNIT DIAG1 MACRO M1200 22 FEB-84 15:31 PAGE 60 1

SEQ 0108

AUTO DROP UNIT SECTION

4484 021572 000002
4485

RTI

;RETURN

CLEANUP CODING SECTION

```

4487          .SBTTL  CLEANUP CODING SECTION
4488
4489          ;////////////////////////////////////
4490          ;// THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
4491          ;// AT THE END OF THE TEST SEQUENCE ON A PARTICULAR UNIT.
4492          ;////////////////////////////////////
4493
4494 021574          BGNCLN
4495          021574          L$CLEAN::
4496
4497 021574          ENDCLN
4498          021574          L10014:  TRAP  C$CLEAN
4499          021574 104412

```

DROP UNIT SECTION

```

4499          .SBTTL  DROP UNIT SECTION
4500
4501          ;////////////////////////////////////
4502          ;// THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
4503          ;// TO NO LONGER BE TESTED.
4504          ;////////////////////////////////////
4505
4506 021576          BGNDU
4507 021576          ;ISSUE UNIBUS RESET TO CLEAN UP
4508 021576 104433          BRESET
4509 021600          ENDDU
4509 021600          L10015:
4509 021600 104453          TRAP    C$RESET
4509 021600          TRAP    C$DU

```

ADD UNIT SECTION

4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521

021602
021602
021602
021602 104452

```
.SBTTL ADD UNIT SECTION
;////////////////////////////////////
; THE ADD UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
; TO BE (A) TESTED FOR THE FIRST TIME, OR (B) RESUMED IN TESTING. IF
; "EF.AUNIT" IS SET, THE UNIT WILL BE TESTED AS A NEW UNIT.
;////////////////////////////////////
```

BGNAU
ENDAU

L\$AU::
L10016: TRAP C\$AU

TEST 1 - TBMT MICROCODE INTERRUPT TEST

4531

.SBTTL TEST 1 - TBMT MICROCODE INTERRUPT TEST

```

*****
;
; TEST 1 -- TBMT MICROCODE INTERRUPT TEST
;
; THIS TEST CHECKS THE OPERATION OF THE TBMT (IRQ) INTERRUPT.
; THIS IS DONE BY ISSUING THE "SET MAINTENANCE INTERRUPT FLAG AND CLEAR
; INTERRUPT DISABLE IN PROCESSOR STATUS" COMMAND WHILE IN THE MAINTENANCE
; LOOP AND THEN CHECKING FOR BIT 7 OF BSEL3 TO BE SET (THE BIT IS SET
; BY THE MICROCODE WHEN THE TBMT INTERRUPT OCCURS).
;
;
;-----*****

```

```

;
; BGNTST
;
; T1::
4532 021604
4533 021604 004737 003374 JSR PC,MSTCLR ;PUT THE MICROPROCESSOR IN THE MAINTENANCE LOOP
4534 021610 103003 BCC .+8. ;IF NO ERROR, PROCEED
4535 021612 ERROR ;ELSE, REPORT IT AND
; TRAP C$ERROR
4536 021614 ESCAPE TST ; EXIT THIS TEST
; TRAP C$ESCAPE
; .WORD L10017-.
;
4537 021620 004537 005432 JSR R5,CKUSTS ;CHK USYRT STATUS FOR INIT STATE
4538 021624 000110 110 ;TBMT=1, TSO=1
4539 021626 103003 BCC .+8. ;IF ERROR, PRINT REPORT
4540 021630 ERROR
; TRAP C$ERROR
4541 021632 ESCAPE TST ; AND SKIP REMAINDER OF TEST
; TRAP C$ESCAPE
; .WORD L10017-.
;
4542 021634 000126
4543 021636 012777 000007 160632 MOV #DOTBMT,@SEL2 ;ISSUE "TBMT TEST" COMMAND
4544 021644 010346 MOV R3,-(SP)
4545 021646 012703 000050 MOV #40.,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
4546 021652 077301 SOB R3,1#
4547 021654 012603 MOV (SP)+,R3
4548 021656 132777 000200 160612 BITB #MRDY,@SEL2 ;"MRDY" SHOULD BE HIGH BY NOW.
4549 021664 001013 BNE 5# ;BR IF NO ERROR
4550 021666 004737 004210 JSR PC,GETWSR ;GET SELECT REGISTERS
4551 021672 012737 000007 002400 MOV #DOTBMT,GDATA ;IDENTIFY REQUESTED FUNCTION
4552 021700 104455 GEDF EM4,ERR4 ;REPORT "MRDY" TIMEOUT ERROR...
; "DEVICE FATAL" ERROR # 39
; TRAP C$ERDF
; .WORD 39
; .WORD EM4
; .WORD ERR4
4553 021702 000047
4554 021704 013235
4555 021706 017030
4556 021710 ESCAPE TST ;AND EXIT TEST
; TRAP C$ESCAPE
; .WORD L10017-.
4557 021712 000050
4558 021714 004737 004210 5# JSR PC,GETWSR ;GET CURRENT REGISTER CONTENTS

```


TEST 2 -- SWITCH SETTING TEST

4585

.SBTTL TEST 2 -- SWITCH SETTING TEST

```

;*****
;*
;*   TEST 2 -- SWITCH SETTING TEST
;*
;* SUBTEST #1:
;* THE TWO READABLE SWITCH PACKS WILL BE SAMPLED AND COMPARED AGAINST THE 2
;* VALUES IN THE P-TABLE. AN ERROR IS REPORTED ON A MISMATCH.
;*
;* SUBTEST #2:
;* THE SPEED SELECT SWITCH (SPDSEL) IS READ VIA THE VIAORA REGISTER (BIT PA4)
;* AND COMPARED AGAINST THE BAUD RATE VALUE IN THE P-TABLE. IF A MISMATCH
;* OCCURS IT WILL BE REPORTED. NOTE: THIS SUBROUTINE IS NOT RUN IF AN M8064
;* BOARD IS BEING TESTED (IT ONLY RUNS @56K... MAKING A SPEED SWITCH USELESS).
;*
;*****

```

```

;
;   BGNTST
;
4586 021764 004737 005420          JSR      PC,INIDMV          ;INIT DMV-11 (MAINT LOOP)          T2::
4587
4588 021770          BGNSUB
;-----
;
;   T2.1:
;
4589 021770 104402          JSR      R5,READI          ;GET "DDCMP ADDRESS"          TRAP      C$BSUB
4590 021772 004537 003610          .WORD   SWPDDCMP
4591 021776 121400          T3.SW1: .WORD   0          ; (IT WILL BE PUT HERE)
4592 022000 000000          JSR      R5,READI          ;GET "BOOT ADDRESS"
4593 022002 004537 003610          .WORD   SWPBOOT
4594 022006 121000          T3.SW2: .WORD   0          ; (IT WILL BE PUT HERE)
4595
4596 022012 123737 022000 002542    CMPB    T3.SW1,LUSWI2      ;# DOES "DDCMP ADDRESS" MATCH P-TABLE VALUE
4597 022020 001004          BNE     T3.ERR            ;NO, REPORT ERROR
4598
4599 022022 123737 022010 002540    CMPB    T3.SW2,LUSWI1      ;# DOES "BOOT ADDRESS" MATCH P-TABLE VALUE
4600 022030 001405          BEQ     T3.OK            ;NO, REPORT ERROR
4601
4602 022032          T3.ERR: GEDF    T3.EHD,T3.EM1      ;REPORT SWITCH SETTINGS DON'T MATCH P-TABLE
;           "DEVICE FATAL" ERROR # 41
;
;
;           TRAP      C$ERDF
;           .WORD    41
;           .WORD    T3.EHD
;           .WORD    T3.EM1
;
4603 022042          CKLOOP
;
;           TRAP      C$CLP1
;
4604
4605 022044 000240          T3.OK:  NOP
4606 022046          ENDSUB
;
;           L10021:
;           TRAP      C$ESUB
;-----
;
;   T2.2:
;
4608 022050          BGNSUB
;
4609 022052 104402 005737 002544    JSR      BRDTYP          ;IS THIS AN M8064 ?          TRAP      C$BSUB

```

TEST 2 -- SWITCH SETTING TEST

```

4610 022056 001433          BEQ      10$          ; IF YES: THEN SKIP THIS SUBROUTINE
4611 022060 004537 003610   JSR      R5,READI    ;GET VIAORA REGISTER
4612 022064 120017          VIAORA
4613 022066 000000          1$:      000          ;STATUS WORD GOES HERE
4614 022070 103003          BCC      ,+8.
4615 022072          ERROR
4616 022072 104460          TRAP      C$ERROR
4616 022074          ESCAPE  SUB
4616 022074 104410          TRAP      C$ESCAPE
4616 022076 000050          .WORD   L10022-.
4617 022100 142737 000357 022066   BICB    #357,1$     ;CLEAR ALL BUT SPEED SELECT BIT(PA4)
4618 022106 106237 022066   ASRB    1$          ;RIGHT JUSTIFY SPDSEL SWITCH BIT FOR
4619 022112 106237 022066   ASRB    1$          ; COMPARISON WITH OPERATOR'S REPLY
4620 022116 106237 022066   ASRB    1$
4621 022122 106237 022066   ASRB    1$
4622 022126 123737 022066 002550   CMPB    1$,BDRATE   ;IS THE SWITCH IN THE DESIRED POSITION?
4623 022134 001404          BEQ      10$          ; IF YES: END THE SUBROUTINE/TEST
4624
4625 022136          GEDF    T3.6,T3.EM2   ;REPORT ERROR
4625          ;          "DEVICE FATAL" ERROR # 42
4625          TRAP      C$ERDF
4625          .WORD   42
4625          .WORD   T3.6
4625          .WORD   T3.EM2
4626 022146          10$:   ENDSUB
4626 022146          L10022:  TRAP      C$ESUB
4627
4628 022150          ;-----
4628 022150          T3.END:  ENDTST
4628 022150          L10020:  TRAP      C$ETST
4629
4630 022152          BGNMSG  T3.EM1
4630 022152
4631 022152          PRINTB  #T3.1          ;PRINT ERROR MESSAGE
4631 022152 012746 022426          MOV      #T3.1,-(SP)
4631 022156 012746 000001          MOV      #1,-(SP)
4631 022162 010600          MOV      SP,R0
4631 022164 104414          TRAP    C$PNTB
4631 022166 062706 000004          ADD     #4,SP
4632 022172          PRINTX  #T3.2
4632 022172 012746 022513          MOV      #T3.2,-(SP)
4632 022176 012746 000001          MOV      #1,-(SP)
4632 022202 010600          MOV      SP,R0
4632 022204 104415          TRAP    C$PNTX
4632 022206 062706 000004          ADD     #4,SP
4633 022212          PRINTX  #T3.3
4633 022212 012746 022542          MOV      #T3.3,-(SP)
4633 022216 012746 000001          MOV      #1,-(SP)
4633 022222 010600          MOV      SP,R0
4633 022224 104415          TRAP    C$PNTX
4633 022226 062706 000004          ADD     #4,SP
4634 022232          PRINTX  #T3.4,LUSWI2,LUSWI1
4634 022232 013746 002540          MOV      LUSWI1,-(SP)
4634 022236 013746 002542          MOV      LUSWI2,-(SP)
4634 022242 012746 022573          MOV      #T3.4,-(SP)
4634 022246 012746 000003          MOV      #3,-(SP)

```

TEST 2 -- SWITCH SETTING TEST

```

022252 010600
022254 104415
022256 062706 000010
4635 022262 105037 022001 CLR B T3.SW1+1 ;MAKE SURE BITS 8 THROUGH 15 AREN'T REPORTED
4636 022266 105037 022011 CLR B T3.SW2+1 ; -- ESPECIALLY BIT 8!!
4637 022272 PRINTX #T3.5,T3.SW1,T3.SW2
022272 013746 022010 MOV T3.SW2,-(SP)
022276 013746 022000 MOV T3.SW1,-(SP)
022302 012746 022636 MOV #T3.5,-(SP)
022306 012746 000003 MOV #3,-(SP)
022312 010600 MOV SP,R0
022314 104415 TRAP C#PNTX
022316 062706 000010 ADD #10,SP
4638 022322 PRINTB #NEWLIN
022322 012746 011645 MOV #NEWLIN,-(SP)
022326 012746 000001 MOV #1,-(SP)
022332 010600 MOV SP,R0
022334 104414 TRAP C#PNTB
022336 062706 000004 ADD #4,SP
4639 022342 ENDMMSG
022342 L10023:
022342 104423 TRAP C#MSG
4640
4641 022344 BGNMSG T3.EM2
022344 T3.EM2::
4642 022344 PRINTB #FMT21,#TXT12,MPCSR
022344 013746 002472 MOV MPCSR,-(SP)
022350 012746 015541 MOV #TXT12,-(SP)
022354 012746 012523 MOV #FMT21,-(SP)
022360 012746 000003 MOV #3,-(SP)
022364 010600 MOV SP,R0
022366 104414 TRAP C#PNTB
022370 062706 000010 ADD #10,SP
4643 022374 004737 020356 JSR PC,ERR11$ ;GET AND PRINT VIA REGISTERS
4644 022400 ENDMMSG
022400 L10024:
022400 104423 TRAP C#MSG
4645
4646 .NLIST BEX
4647 022402 123 127 111 T3.END: .ASCIZ 'SWITCH SETTING TEST'
4648 022426 045 116 045 T3.1: .ASCIZ /#N#ASWITCH PACK SETTING DOES NOT MATCH P-TABLE ENTRY/
4649 022513 045 116 045 T3.2: .ASCIZ '#N#S22#A DDCMP BOOT'
4650 022542 045 116 045 T3.3: .ASCIZ '#N#S22#AADDRESS ADDRESS'
4651 022573 045 116 045 T3.4: .ASCIZ '#N#S6#AP-TABLE VALUES:#S1#03#S6#03'
4652 022636 045 116 045 T3.5: .ASCIZ '#N#S6#AREAD FROM DMV:#S2#03#S6#03'
4653 022700 123 120 104 T3.6: .ASCIZ /SPDSEL SWITCH DOESN'T MATCH P-TABLE BAUD RATE ENTRY/
4654 .LIST BEX
4655 .EVEN

```

TEST 3 -- USYRT MASTER CLEAR TEST

4673

.SBTTL TEST 3 -- USYRT MASTER CLEAR TEST

```

;*****
;*
;* TEST 3 -- USYRT MASTER CLEAR TEST
;*
;* ALL REGISTERS ARE LOADED WITH PATTERN E IN THE SAME SEQUENCE AS FOR
;* PATTERN F BELOW. THE USYRT IS THEN CLEARED BY A MASTER CLEAR
;* (BIT 6 OF BSEL 1). ALL REGISTERS ARE THEN CHECKED FOR
;* THE PROPER CONTENTS. THE INITIALIZED STATE OF THE REGISTERS IS CHECKED
;* AGAINST DATA PATTERN F.
;*
;* PATTERN E: 377, 377, 377, 377, 377, 377, 377, 366.
;* PATTERN F: 000, 000, 000, 000, 000, 000, 000, 110.
;*
;* SEQUENCE OF REGISTERS AS USED WITH PATTERNS E & F:
;*
;* RDSRL, RDSRH, TDSRL, TDSRH, PCSARL, PCSARH, PCR, USYRT STATUS REG
;*
;*****

```

```

;
; BGNTST
;
; T3::
4674 022764 004737 005420 JSR PC,INIDMV ;INIT DMV-11
4675 022770 005001 CLR R1 ;INIT USYRT REG ADRS POINTER
4676 022772 012702 002652 MOV #PATE,R2 ;INIT PATTERN E POINTER
4677 022776 016137 002552 023014 1$: MOV USYREG(R1),2$ ;GET USYRT REG ADRS
4678 023004 112237 023016 MOVB (R2)+,3$ ;GET A PATTERN BYTE
4679 023010 004537 003734 JSR R5,WRITEI ;WRITE A USYRT REG
4680 023014 000000 2$: .WORD 0 ;USYRT REG ADRS GOES HERE
4681 023016 000000 3$: .WORD 0 ;DATA BYTE GOES HERE
4682 023020 062701 000002 ADD #2,R1 ;INCR REG ADRS PTR
4683 023024 020127 000020 CMP R1,#16. ;SEE IF ALL REGS WRITTEN YET
4684 023030 002762 BLT 1$ ;BR IF NOT
4685 023032 004737 005420 JSR PC,INIDMV ;ISSUE MASTER CLEAR
4686 023036 005001 CLR R1 ;INIT USYRT REG ADRS PTR
4687 023040 012702 002662 MOV #PATF,R2 ;INIT DATA PATTERN POINTER
4688 023044 016137 002552 023056 6$: MOV USYREG(R1),7$ ;SET USYRT READ ADDRESS
4689 023052 004537 003610 JSR R5,READI ;READ A USYRT REG
4690 023056 000000 7$: .WORD 0 ;USYRT REG ADRS GOES HERE
4691 023060 000000 8$: .WORD 0 ;DATA READ IS RETURNED HERE
4692 023062 123722 023060 CMPB 8$,(R2)+ ;SEE IF REG CONTAINS EXPECTED DATA
4693 023066 001422 BEQ 9$ ;BR IF MATCH
4694 023070 010137 002412 MOV R1,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
4695 023074 006237 002412 ASR REGNUM ;GET WORD OFFSET
4696 023100 005037 002400 CLR GDATA ;GET EXPECTED DATA
4697 023104 116237 177777 002400 MOVB -1(R2),GDATA
4698 023112 013737 023060 002402 MOV 8$,BDATA ;GET ACTUAL DATA
4699 ;REPORT REG NOT CLEARED BY MASTER CLEAR
4700 023120 GEDF EM1,ERR10 ; "DEVICE FATAL" ERROR # 43
; TRAP C$ERDF
; .WORD 43
; .WORD EM1
; .WORD ERR10
023120 104455
023122 000053
023124 013103
023126 017372
4701 023130 ESCAPE TST

```

TEST 3 - USYRT MASTER CLEAR TEST

023130	104410									
023132	000014									
4702	023134	062701	000002	9:	ADD	02,R1			TRAP	C\$ESCAPE
4703	023140	020127	000020		CMP	R1,016.			.WORD	L10025-
4704	023144	002737			BLT	6:				
4705	023146			ENDTST						
	023146								L10025:	
	023146	104401							TRAP	C\$ETST

;INCR USYRT REG. ADRS PTR
;SEE IF ALL REGS READ YET
;BR IF NOT

TEST 4 - USYRT PROGRAM RESET TEST

4722

.SBTTL TEST 4 - USYRT PROGRAM RESET TEST

```

*****
;
; TEST 4 -- USYRT PROGRAM RESET TEST
;
; ALL REGISTERS ARE LOADED WITH PATTERN E IN THE SAME SEQUENCE AS FOR
; PATTERN F BELOW. THE USYRT IS THEN RESET BY ASSERTING PROGRAM RESET
; (BIT 0 @ A000) IN THE 6522 VIA. ALL REGISTERS ARE THEN CHECKED FOR
; THE PROPER CONTENTS. THE INITIALIZED STATE OF THE REGISTERS IS CHECKED
; AGAINST DATA PATTERN F.
;
; PATTERN E: 377, 377, 377, 377, 377, 377, 377, 366.
; PATTERN F: 000, 000, 000, 000, 000, 000, 000, 110.
;
; SEQUENCE OF REGISTERS AS USED WITH PATTERNS E & F:
;
; RDSRL, RDSRH, TDSRL, TDSRH, PCSARL, PCSARH, PCR, USYRT STATUS REG
;
*****

```

```

023150
4723 023150 004737 005420
4724 023154 005001
4725 023156 012702 002652
4726 023162 016137 002552 023200 1$:
4727 023170 112237 023202
4728 023174 004537 003734
4729 023200 000000 2$:
4730 023202 000000 3$:
4731 023204 062701 000002
4732 023210 020127 000020
4733 023214 002762
4734
4735 023216 004537 005126
4736 023222 103003
4737 023224
023224 104460
4738 023226
023226 104410
023230 000002
4739
4740
4741 023232
023232
023232 104401

```

```

;
; BGNTST
;
; JSR PC,INIDMV ;INIT DMV-11 T4:;
; CLR R1 ;INIT USYRT REG ADRS POINTER
; MOV #PATE,R2 ;INIT PATTERN E POINTER
; MOV USYREG(R1),2$ ;GET USYRT REG ADRS
; MOVB (R2)+,3$ ;GET A PATTERN BYTE
; JSR R5,WRITEI ;WRITE A USYRT REG
; .WORD 0 ;USYRT REG ADRS GOES HERE
; .WORD 0 ;DATA BYTE GOES HERE
; ADD #2,R1 ;INCR REG ADRS PTR
; CMP R1,#16. ;SEE IF ALL REGS WRITTEN YET
; BLT 1$ ;BR IF NOT
;
; JSR R5,RSTCHK ;RESET USYRT/VERIFY SAME
; BCC ,+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
;
; ESCAPE TST ;SKIP TO END OF TEST
;
; TRAP C$ERROR
; TRAP C$ESCAPE
; .WORD L10026-.
;
; ENDTST
;
; L10026:
; TRAP C$ETST

```

TEST 5 -- USYRT REGISTER ADDRESSING TEST

4756

.SBITL TEST 5 -- USYRT REGISTER ADDRESSING TEST

```

*****
;*
;* TEST 5 -- USYRT REGISTER ADDRESSING TEST
;*
;* FIRST, A MASTER CLEAR IS ISSUED, TO INITIALIZE THE USYRT REGS TO
;* PATTERN F. THEN, EACH REGISTER IS WRITTEN WITH A BYTE OF PATTERN J,
;* AND AFTER EACH IS WRITTEN, ALL ARE READ AND COMPARED TO THE CURRENT
;* EXPECTED VALUES. THIS IS PERFORMED FOR ALL REGISTERS -- INCLUDING THE
;* READ ONLY REGS -- IN ORDER TO MAKE SURE THAT EACH REGISTER ONLY RESPONDS
;* TO ITS OWN ADDRESS.
;* PATTERN F: 000, 000, 000, 000, 000, 000, 000, 110
;* PATTERN J: 000, 000, 001, 002, 004, 020, 040, 010
;*
;* SEQUENCE OF REGISTERS AS USED WITH PATTERNS F & J:
;* RDSRL, RDSRH, TDSRL, TDSRH, PCSARL, PCSARH, PCR, USYRT STATUS REG
;*
*****

```

```

;
; BGNTST
;
; T5:
4757 023234 004737 005420 JSR PC,INIDMV ;ISSUE MASTER CLEAR TO INIT DMV
4758 023240 012702 002662 MOV #PATF,R2 ;INIT PATTERN I POINTER
4759 023244 012703 002572 MOV #REDDAT,R3 ;INIT PTR TO EXPECTED DATA AREA
4760 023250 112223 112223 1$: MOVVB (R2)+,(R3)+ ;MOV PATTERN F INTO REDDAT TABLE
4761 023252 020227 002672 CMP R2,#PATG
4762 023256 103774 BLO 1$
4763 023260 005001 CLR R1 ;INIT USYRT REG ADRS PTR FOR WRITING
4764 023262 005002 CLR R2 ;INIT INDEX FOR WRITING
4765 023264 016137 002552 023330 2$: MOV USYREG(R1),3$ ;SET USYRT REG ADRS
4766 023272 116237 003013 023332 MOVVB PATJ(R2),4$ ;SET DATA FOR WRITE
4767 023300 113762 023332 002572 MOVVB 4$,REDDAT(R2) ;SET EXPECTED DATA BYTE
4768 023306 023727 023330 120402 CMP 3$,#TDSRL ;SEE IF WRITING TDSRL
4769 023314 001003 BNE 10$ ;BR IF NOT
4770 023316 142737 000100 002601 BICB #100,REDDAT+7 ;FIX EXPECTED USTAT VALUE
4771 023324 004537 003734 10$: JSR R5,WRITEI ;WRITE BYTE INTO A USYRT REG
4772 023330 000000 3$: .WORD 0 ;REG ADRS GOES HERE
4773 023332 000000 4$: .WORD 0 ;DATA BYTE GOES HERE
4774 023334 005003 CLR R3 ;INIT USYRT REG ADRS PTR FOR READING
4775 023336 005004 CLR R4 ;INIT INDEX FOR READING
4776 023340 016337 002552 023352 5$: MOV USYREG(R3),6$ ;SET USYRT REG ADRS
4777 023346 004537 003610 JSR R5,READI ;READ A USYRT REG
4778 023352 000000 6$: .WORD 0 ;REG ADRS GOES HERE
4779 023354 000000 7$: .WORD 0 ;DATA BYTE READ IS RETURNED HERE
4780 023356 123764 023354 002572 CMPB 7$,REDDAT(R4) ;SEE IF BYTE READ MATCHES EXPECTED BYTE
4781 023364 001420 BEQ 8$ ;BR IF MATCH
4782 023366 010437 002412 MOV R4,REGNUM ;SET FAILING REG NO. FOR ERROR REPORT
4783 023372 005037 002400 CLR GDATA ;GET EXPECTED DATA
4784 023376 116437 002572 002400 MOVVB REDDAT(R4),GDATA
4785 023404 013737 023354 002402 MOV 7$,BDATA ;GET ACTUAL DATA
4786 ;REPORT REGISTER MISCOMPARE ERROR
4787 023412 GEDF EM66,ERR10 ;
; "DEVICE FATAL" ERROR # 44
; TRAP C$ERDF
; .WORD 44
; .WORD EM66

```


TEST 5 -- USYRT REGISTER ADDRESSING TEST

4788	023420	017372							.WORD	ERR10
	023422			ESCAPE	TST					
	023422	104410							TRAP	C\$ESCAPE
	023424	000032							.WORD	L10027-
4789	023426	062703	000002	8\$:	ADD	#2,R3				
4790	023432	005204			INC	R4				
4791	023434	020427	000010		CMP	R4,#10				
4792	023440	002737			BLT	5\$				
4793	023442	062701	000002		ADD	#2,R1				
4794	023446	005202			INC	R2				
4795	023450	020227	000010		CMP	R2,#10				
4796	023454	002703			BLT	2\$				
4797	023456			ENDTST						
	023456									
	023456	104401							L10027:	
									TRAP	C\$ETST

```

; INCR USYRT REG ADRS PTR FOR READING
; INCR READ INDEX
; SEE IF ALL REGS READ YET
; BR IF NOT ALL READ YET
; INCR USYRT REG ADRS PTR FOR WRITING
; INCR WRITE INDEX
; SEE IF ALL REGS WRITTEN YET
; BR IF NOT ALL WRITTEN YET

```

71.0

TEST 6 -- R/W BIT TEST OF PCSAR HIGH BYTE

4806

.SBTTL TEST 6 -- R/W BIT TEST OF PCSAR HIGH BYTE

```

;*****
;
; TEST 6 -- R/W BIT TEST OF PCSAR HIGH BYTE
;
; WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN G.
;
; PATTERN G: 000, 001, 003, 004, 005, 007, 100, 101, 103, 104, 105,
; 107, 000, 017, 027, 041, 200, 277, 103, 144, 115, 157, 000.
;
;*****

```

```

;
; BGNTST
;
; T6::
4807 023460 004737 005420 JSR PC,INIDMV ;INIT DMV/VIA & START UP MAINT. LOOP
4808 023460 103003 BCC 30$ ;IF AN ERROR OCCURED,
4809 023466 104460 ERROR ;REPORT IT &
; TRAP C$ERROR
4810 023470 ESCAPE TST ; EXIT
; TRAP C$ESCAPE
; .WORD L10030-.
4811
4812 023474 012701 002672 30$: MOV #PATG,R1 ;POINT TO PATTERN TABLE
4813 023500 012703 000025 MOV #<PATH-PATG-2>,R3 ;GET # OF ENTRIES IN TABLE
4814 023504 012737 000005 002412 MOV #5,REGNUM ;ERROR INDEX FOR PCSARH
4815
4816 023512 T7.LP: BGNSUB ;THE SUBTEST ONLY TESTS THE ONE PATTERN
; T6.1:
; TRAP C$BSUB
4817 023512 104402
4818 023514 111137 002376 MOVB (R1),TDATA ;SETUP TEST DATA BYTE FOR "STUREG"
4819 023520 112137 002400 MOVB (R1)+,GDATA ;SETUP EXPECTED DATA BYTE FOR "STUREG"
4820 023524 012700 120405 MOV #PCSARH,R0 ;SPECIFY THE REGISTER BEING TESTED
4821 023530 004737 004272 JSR PC,STUREG ;PERFORM STATIC TEST OF THE SPECIFIED REGISTER
4822 023534 103003 BCC 10$ ;WAS AN ERROR FOUND?
4823 023536 104460 ERROR ;YES, REPORT IT AND
; TRAP C$ERROR
4824 023540 ESCAPE TST ; EXIT FROM THE TEST. "CKLOOP" IS IMPLIED
; TRAP C$ESCAPE
; .WORD L10030-.
4825
4826 023544 104403 10$: ENDSUB
; L10031:
; TRAP C$ESUB
4827
4828 023546 077317 SOB R3,T7.LP ;IF THERE IS IN FACT MORE DATA, LOOP BACK TO
4829 ;TEST IT. ELSE, FALL OUT OF LOOP AND TEST
4830 023550 ENDTST
; L10030:
; TRAP C$ETST
4831 023550 104401

```

TEST 7 -- R/W BIT TEST OF S/AR REGISTER

4839

.SBTTL TEST 7 -- R/W BIT TEST OF S/AR REGISTER

```

;*****
;
; TEST 7 -- R/W BIT TEST OF S/AR REGISTER
;
; WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN H.
;
; PATTERN H: 125, 252, 000, 377, 000, 001, 002, 004, 010, 020, 040, 100,
;            200, 000, 377, 376, 375, 373, 367, 357, 337, 277, 177, 377,
;            000
;*****

```

```

;
; BGNTST
;
4840 023552 004737 005420          T7:;
4841 023556 103003              JSR    PC,INIDMV      ;INIT DMV & START UP THE MAINT. LOOP
4842 023560 104460              BCC    30$          ;IF AN ERROR OCCURED,
                                ERROR      ;REPORT IT &
                                TRAP      C$ERROR
4843 023562 104410              ESCAPE TST           ; EXIT
                                TRAP      C$ESCAPE
                                023564 000056          .WORD  L10032-.
4844
4845 023566 012701 002721      30$: MOV    @PATH,R1      ;POINT TO PATTERN TABLE
4846 023572 012703 000027      MOV    @<PATI-PATH-2>,R3 ;GET # OF ENTRIES IN TABLE
4847 023576 012737 000004 002412 MOV    @4,REGNUM      ;ERROR INDEX FOR S/AR BYTE
4848
4849 023604 023604 104402      T8.LP: BGNSUB       ;THE SUBTEST ONLY TESTS THE ONE PATTERN
                                T7.1:
                                TRAP      C$BSUB
4850
4851 023606 111137 002376      MOVB   (R1),TDATA     ;SETUP TEST DATA BYTE FOR "STUREG"
4852 023612 112137 002400      MOVB   (R1)+,GDATA   ;SETUP EXPECTED DATA BYTE FOR "STUREG"
4853 023616 012700 120404      MOV    @PCARL,R0     ;SPECIFY THE REGISTER BEING TESTED
4854 023622 004737 004272      JSR    PC,STUREG     ;PERFORM STATIC TEST OF THE SPECIFIED REGISTER
4855 023626 103003              BCC    10$          ;WAS AN ERROR FOUND?
4856 023630 104460              ERROR      ;YES, REPORT IT AND
                                TRAP      C$ERROR
4857 023632 104410              ESCAPE TST           ; EXIT FROM THE TEST. "CKLOOP" IS IMPLIED
                                TRAP      C$ESCAPE
                                023634 000006          .WORD  L10032-.
4858
4859 023636 104403              10$: ENDSUB
                                L10033:
                                TRAP      C$ESUB
4860
4861 023640 077317              SOB    R3,T8.LP     ;IF THERE IS IN FACT MORE DATA, LOOP BACK TO
4862
4863 023642 104401              ENDTST             ;TEST IT. ELSE, FALL OUT OF LOOP AND TEST
                                L10032:
                                TRAP      C$ETST

```

TEST 8 -- R/W BIT TEST OF PCR REGISTER

4874

.SBTTL TEST 8 -- R/W BIT TEST OF PCR REGISTER

```

;*****
;*
;* TEST 8 -- R/W BIT TEST OF PCR REGISTER
;*
;* PATTERN I IS LOADED INTO PCR (HIGH) AND THE DATA READ BACK AND
;* CHECKED.
;*
;* PATTERN I: 000, 041, 102, 143, 204, 245, 306, 347, 000, 001, 002,
;* 004, 040, 100, 200, 000, 346, 345, 343, 307, 247, 147, 347, 242,
;* 105, 347, 010, 020, 367, 357, 030, 027, 377.
;*
;*****

```

```

;
; BGNTST
;
4875 023644 004737 005420 JSR PC,INIDMV ;INIT DMV/VIA & START UP MAINT. LOOP
4876 023650 103003 BCC 30$ ;IF AN ERROR OCCURED.
4877 023652 104460 ERROR ;REPORT IT &
; TRAP C$ERROR
4878 023654 104410 ESCAPE TST ; EXIT
; TRAP C$ESCAPE
023656 000056 .WORD L10034-.
;
4879
4880 023660 012701 002752 30$: MOV #PATI,R1 ;POINT TO PATTERN TABLE
4881 023664 012703 000037 MOV #<PATJ-PATI-2>,R3 ;GET # OF ENTRIES IN TABLE
4882 023670 012737 000006 002412 MOV #6,REGNUM ;ERROR INDEX FOR PCR REGISTER
4883
4884 023676 104402 T9.LP: BGNSUB ;THE SUBTEST ONLY TESTS THE ONE PATTERN
; TRAP C$BSUB
023676 104402 T8.1:
023676 104402 ;
4885
4886 023700 111137 002376 MOV# (R1),TDATA ;SETUP TEST DATA BYTE FOR "STUREG"
4887 023704 112137 002400 MOV# (R1)+,GDATA ;SETUP EXPECTED DATA BYTE FOR "STUREG"
4888 023710 012700 120407 MOV #PCR,R0 ;SPECIFY THE REGISTER BEING TESTED
4889 023714 004737 004272 JSR PC,STUREG ;PERFORM STATIC TEST OF THE SPECIFIED REGISTER
4890 023720 103003 BCC 10$ ;WAS AN ERROR FOUND?
4891 023722 104460 ERROR ;YES, REPORT IT AND
; TRAP C$ERROR
4892 023724 104410 ESCAPE TST ; EXIT FROM THE TEST. "CKLOOP" IS IMPLIED
; TRAP C$ESCAPE
023726 000006 .WORD L10034-.
4893
4894 023730 104403 10$: ENDSUB
; TRAP C$ESUB
023730 104403 L10035:
023730 104403 ;
4895
4896 023732 077317 SOB R3,T9.LP ;IF THERE IS IN FACT MORE DATA, LOOP BACK TO
4897 ;TEST IT. ELSE, FALL OUT OF LOOP AND TEST
4898 023734 104401 ENDTST
; TRAP C$ETST
023734 104401 L10034:

```

TEST 9 -- R/W BIT TEST OF TDSR REGISTER'S HIGH BYTE

4915

.SBTTL TEST 9 -- R/W BIT TEST OF TDSR REGISTER'S HIGH BYTE

```

;*****
;
; TEST 9 -- R/W BIT TEST OF TDSR REGISTER'S HIGH BYTE
;
; PATTERN K IS LOADED INTO TDSR (HIGH) AND THE DATA READ BACK IS
; COMPARED AGAINST PATTERN L. (UNPREDICTABLE BITS ARE MASKED OFF TO 0
; WHEN READING FOR COMPARISON.)
;
; PATTERN K: 000, 377, 376, 375, 373, 376, 177, 377, 000, 001, 002,
; 004, 010, 200, 125, 252, 000.
;
; PATTERN L: 000, 017, 016, 015, 013, 016, 017, 017, 000, 001, 002,
; 004, 010, 000, 005, 012, 000.
;
; NOTE THAT THE UNDEFINED BITS (12, 13, & 14) ARE MASKED OFF TO 0'S
; FOR THE COMPARISON. ALSO THAT BIT 15 IS A READ/ONLY BIT AND CAN'T BE
; SET -- THEREFORE SHOULD ALWAYS BE READ AS A 0 BY THIS TEST.
;
;*****

```

```

;
; BGNTST
;
; T9::
4916 023736 004737 005420 JSR PC,INIDMV ;INIT DMV/VIA & START UP MAINT. LOOP
4917 023742 103003 BCC 30$ ;IF AN ERROR OCCURED,
4918 023744 104460 ERROR ;REPORT IT &
; TRAP C$ERROR
4919 023746 104410 ESCAPE TST ; EXIT
; TRAP C$ESCAPE
; .WORD L10036-.
; 023750 000114
4920
4921 023752 012701 003023 30$: MOV #PATK,R1 ;POINT TO PATTERN TABLE
4922 023756 012703 000017 MOV #<PATL-PATK-2>,R3 ;GET # OF ENTRIES IN TABLE
4923 023762 012702 003044 MOV #PATL,R2 ;POINT TO "EXPECTED" DATA PATTERN TABLE
4924
4925 023766 T10.LP: BGNSUB ;THE SUBTEST ONLY TESTS THE ONE PATTERN
; T9.1:
; TRAP C$BSUB
; 023766 104402
4926
4927 023770 112137 002376 MOVB (R1)+,TDATA ;SETUP TEST DATA BYTE
4928 023774 112237 002400 MOVB (R2)+,GDATA ;SETUP EXPECTED DATA BYTE
4929
4930 024000 004537 003722 JSR R5,WRITE ;WRITE TO DMV-11
; .WORD TDSRH ; DMV-11 ADDRESS WRITTEN TO
; .WORD TDATA ; LOCATION OF DATA WRITTEN
4931 024004 120403
4932 024006 002376
4933
4934 024010 004537 003476 JSR R5,READ ;READ FROM DMV-11
; .WORD TDSRH ; DMV-11 ADDRESS READ FROM
; .WORD BDATA ; LOCATION WHERE READ DATA IS PUT
4935 024014 120403
4936 024016 002402
4937
4938 024020 042737 177760 002402 BIC #177760,BDATA ;MASK OUT "DON'T CARE" BITS
4939 024026 023737 002400 002402 CMP GDATA,BDATA ;READ DATA = EXPECTED DATA ?
4940 024034 001411 BEQ 10$ ;WAS AN ERROR FOUND?
4941
4942 024036 012737 000003 002412 MOV #3,REGNUM ;YES: SET UP REGISTER NUMBER
4943 024044 GDF EM25,ERR7A ;REPORT IT AND

```

TEST 9 -- R/W BIT TEST OF TDSR REGISTER'S HIGH BYTE

```

024044 104455
024046 000055
024050 013366
024052 017252
4944 024054          ESCAPE TST
024054 104410
024056 000006
4945
4946 024060          10$:  ENDSUB
024060
024060 104403
4947
4948 024062 077337
4949
4950 024064          ENDTST
024064
024064 104401

```

```

; "DEVICE FATAL" ERROR # 45
TRAP C$ERDF
.WORD 45
.WORD EM25
.WORD ERR7A
; EXIT FROM THE TEST. "CKLOOP" IS IMPLIED
TRAP C$ESCAPE
.WORD L10036-.

L10037:
TRAP C$ESUB

;IF THERE IS IN FACT MORE DATA, LOOP BACK TO
;TEST IT. ELSE, FALL OUT OF LOOP AND TEST

L10036:
TRAP C$ETST

```

TEST 10 -- R/W BIT TEST OF TXDB REGISTER

4958

.SBTTL TEST 10 -- R/W BIT TEST OF TXDB REGISTER

```

;*****
;*
;* TEST 10 -- R/W BIT TEST OF TXDB REGISTER
;*
;* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN H.
;*
;* PATTERN H: 000, 001, 002, 004, 010, 020, 040, 100, 200, 000, 377,
;* 376, 375, 373, 367, 357, 337, 277, 177, 377, 000
;*
;*****

```

```

;
; BGNTST
;
4959 024066 004737 005420          JSR    PC,INIDMV      ;INIT DMV/VIA & START UP MAINT. LOOP
4960 024072 103003                BCC    30$           ;IF AN ERROR OCCURED,
4961 024074 104460                ERROR                   ;REPORT IT &
;                                     TRAP    C$ERROR
4962 024076 104410                ESCAPE TST           ; EXIT
;                                     TRAP    C$ESCAPE
;                                     .WORD  L10040-.
4963
4964 024102 012701 002721          30$:  MOV    @PATH,R1      ;POINT TO PATTERN TABLE
4965 024106 012703 000027          MOV    @<PATI-PATH-2>,R3 ;GET # OF ENTRIES IN TABLE
4966 024112 012737 000002 002412  MOV    @2,REGNUM      ;ERROR INDEX FOR TXDB REGISTER
4967
4968 024120                T11.LP: BGNSUB      ;THE SUBTEST ONLY TESTS THE ONE PATTERN
;                                     T10.1:
;                                     TRAP    C$BSUB
4969
4970 024122 111137 002376          MOVB   (R1),TDATA    ;SETUP TEST DATA BYTE FOR "STUREG"
4971 024126 112137 002400          MOVB   (R1)+,GDATA   ;SETUP EXPECTED DATA BYTE FOR "STUREG"
4972 024132 012700 120402          MOV    @TDSRL,R0     ;SPECIFY THE REGISTER BEING TESTED
4973 024136 004737 004272          JSR    PC,STUREG     ;PERFORM STATIC TEST OF THE SPECIFIED REGISTER
4974 024142 103003                BCC    10$           ;WAS AN ERROR FOUND?
4975 024144 104460                ERROR                   ;YES, REPORT IT AND
;                                     TRAP    C$ERROR
4976 024146 104410                ESCAPE TST           ; EXIT FROM THE TEST. "CKLOOP" IS IMPLIED
;                                     TRAP    C$ESCAPE
;                                     .WORD  L10040-.
4977
4978 024152                10$:  ENDSUB
;                                     L10041:
;                                     TRAP    C$ESUB
4979
4980 024154 077317                SOB    R3,T11.LP     ;IF THERE IS IN FACT MORE DATA, LOOP BACK TO
4981
4982 024156                ENDTST              ;TEST IT. ELSE, FALL OUT OF LOOP AND TEST
;                                     L10040:
;                                     TRAP    C$ETST

```


TEST 12 -- PSEUDO R/W BIT TEST OF RDSR'S HIGH BYTE

5021

.SBTTL TEST 12 -- PSEUDO R/W BIT TEST OF RDSR'S HIGH BYTE

```

;*****
;*
;* TEST 12 -- PSEUDO R/W BIT TEST OF RDSR'S HIGH BYTE
;*
;* WRITE, READ (BUT NO COMPARE) OF EACH WORD IN DATA PATTERN H. THIS IS
;* PRIMARILY TO PROVIDE A SCOPE LOOP FUNCTION ON THIS REGISTER.
;*
;* PATTERN H: 000, 001, 002, 004, 010, 020, 040, 100, 200, 000, 377,
;* 376, 375, 373, 367, 357, 337, 277, 177, 377, 000
;*
;--*****

```

```

024234
5022 024234 004737 005420
5023 024240 103003
5024 024242
      024242 104460
5025 024244
      024244 104410
      024246 000040
5026
5027 024250 012701 002721
5028 024254 012703 000027
5029
5030 024260 012137 024272
5031
5032 024264 004537 003734
5033 024270 120401
5034 024272 000000
5035
5036 024274 004537 003610
5037 024300 120401
5038 024302 000000
5039
5040 024304 077313
5041
5042 024306
      024306
      024306 104401

```

```

;
;      BGNTST
;
;      T12::
;      JSR    PC,INIDMV      ;INIT DMV/VIA & START UP MAINT. LOOP
;      BCC    30$           ;IF AN ERROR OCCURED,
;      ERROR  ;REPORT IT &
;
;      ESCAPE TST           ; EXIT
;
;      TRAP   C$ERROR
;      TRAP   C$ESCAPE
;      .WORD  L10043-
;
;      30$:  MOV    #PATH,R1      ;POINT TO PATTERN TABLE
;      MOV    #<PATI-PATH-2>,R3 ;GET # OF ENTRIES IN TABLE
;
;      20$:  MOV    (R1)+,2$
;
;      JSR    R5,WRITEI      ;WRITE TO DMV-11
;      .WORD  RDSRH          ; DMV-11 ADDRESS WRITTEN TO
;      2$:   .WORD  0         ; ACTUAL DATA WRITTEN
;
;      JSR    R5,READI       ;READ FROM DMV-11
;      .WORD  RDSRH          ; DMV-11 ADDRESS READ FROM
;      .WORD  0             ; READ DATA IS PUT HERE
;
;      SOB   R3,20$         ;IF MORE DATA, LOOP BACK TO WRITE/READ IT,
;                          ; ELSE, FALL OUT OF LOOP AND TEST
;
;      ENDTST
;
;      L10043:
;      TRAP   C$ETST

```

TEST 13 -- NULL CLOCK TEST

5062

.SBTTL TEST 13 -- NULL CLOCK TEST

```

;*****
;*
;* TEST 13 -- NULL CLOCK TEST
;*
;* FIRST, A MASTER CLEAR IS DONE TO INIT THE DMV. THEN, THE T1 TIMER ON THE
;* VIA CHIP IS PROGRAMMED FOR SQUARE WAVE CLOCK GENERATION ON PB7 (BIT 7
;* OF VIA OUTPUT REG B), WITH A BAUD RATE = 56 KBAUD. THIS IS THE MODE OF
;* VIA OPERATION WHICH IS USED TO GENERATE THE NULL CLOCK. THEN, THE PROGRAM
;* SCANS ORB REPEATEDLY TO MONITOR THE NULL CLOCK BIT, IN THE FOLLOWING
;* SEQUENCE :
;* - THE PROGRAM REPEATEDLY CHECKS THE NULL CLOCK BIT FOR THE 1 STATE, AND
;* IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MICRO-SEC (A GROSS TIMEOUT
;* INTERVAL), AN ERROR IS REPORTED. (AT 56 KBAUD, THE CLOCK SHOULD
;* HAVE A PERIOD OF ABOUT 18 MICRO-SEC.)
;* - THE PROGRAM NEXT REPEATEDLY CHECKS THE NULL CLOCK BIT FOR THE 0 STATE,
;* AND IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MICRO-SEC, AN ERROR IS
;* REPORTED.
;* - THE PROGRAM NEXT REPEATEDLY CHECKS THE NULL CLOCK BIT FOR THE 1 STATE
;* AGAIN, AND IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MICRO-SEC,
;* AN ERROR IS REPORTED.
;*
;*****

```

```

024310
5063 024310 004737 005420
5064 024314 004537 003734
5065 024320 120004
5066 024322 000017
5067 024324 004537 003734
5068 024330 120005
5069 024332 000000
5070

```

```

;
; BGNTST
;
; T13::
; JSR PC,INIDMV ;INIT DMV11
; JSR R5,WRITEI ;LOAD T1C-L, T1L-L FOR 56K BAUD
; VIAT1A
; 15.
; JSR R5,WRITEI ;LOAD T1C-H, T1L-H, START CLOCK
; VIAT1B
; 000
;-----
; WAIT FOR NULCLK BIT TO BE SET TO 1
;-----
5073 024334 012737 170000 002602
5074 024342 004537 003610
5075 024346 120000
5076 024350 000000
5077 024352 132737 000200 024350
5078 024360 001025
5079 024362 005237 002602
5080 024366 001365
5081
5082 024370 012737 000000 002412
5083 024376 012737 000200 002400
5084 024404 013737 024350 002402
5085 024412 012737 000177 002402
5086
5087 024420
; REPORT NUL. CLK BIT AT 0
; GEDF ERR11
;
; "DEVICE FATAL" ERROR # 46
; TRAP C$ERDF
; .WORD 46
; .WORD EMS
; .WORD ERR11

```

```

J24420 10455
024422 000056
024424 013252
024426 017546

```


TEST 14 BCP TX RESET W/IDLE = 0

5149

.SBTTL TEST 14 BCP TX RESET W/IDLE = 0

```

*****
;
; TEST 14 -- BCP TX REJET W/IDLE = 0
;
; THE USYRT IS INITIALIZED FOR "BYTE-CONTROL PROTOCOL" (BCP) WITH IDLE
; SET TO ZERO AND A 125 SYNC CHARACTER IS LOADED INTO S/AR. A 226 SYNC
; CHARACTER IS LOADED INTO TXDB SO THAT THE SOURCE OF SYNC CHARACTERS
; CAN BE LATER DETERMINED, THE VALID STATE OF THE USYRT REGISTERS IS
; READ AND CHECKED, TXE IS ASSERTED TO ENABLE THE TRANSMITTER LOGIC.
; THEN, TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE OBSERVING
; TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
; (TXBE SHOULD GO HIGH, AT THIS TIME THE S/AR'S SYNC CHARACTER SHOULD
; BE LOADED INTO TXSO AND TSOM IS AGAIN SET -- DRIVING TXBE LOW.)
; THREE SYNC CHARACTERS ARE SENT/RECEIVED; THE FIRST TWO SYNCHRONIZE
; THE RECEIVER, THE THIRD IS DIRECTLY READ (STRIP SYNC IS OFF) AND
; COMPARED AGAINST 125 (THE S/AR SYNC CHARACTER).
; IF VALUE READ IS 226, THEN TXDB PROVIDED THE SYNC (IE: ERROR).
; THE USYRT IS THEN RESET AND REGISTERS ARE AGAIN READ AND CHECKED.
; THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY
; ONE MARK AND THREE SYNC CHARACTERS (FROM THE S/AR) IS TRANSMITTED.
; ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN THE
; SEQUENCE.
;
*****

```

```

;
; BGNTST
;
; T14::
;
5150 024632
5151 024632 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
5152
5153 024636 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5154 024642 043525 DDCMP!NOCHK!125 ;SET DDCMP,NO CHECK,S/AR(SYNC)=125
5155 024644 000000 0 ;USE 8 BIT CHARS
5156 024646 103003 BCC ,+8. ;BR IF NO ERROR
5157 024650 ERROR ;REPORT STACKED ERROR
5158 024650 104460 TRAP C!ERROR
5158 024652 ESCAPE TST ;SKIP TO END OF TEST
5158 024652 104410 TRAP C!ESCAPE
5158 024654 000106 .WORD L10045-.
5159
5160 024656 004537 010010 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTER
5161 024662 000001 TSOM ;AND KNOCK DOWN TBMT
5162 024664 000007 7.
5163
5164 024666 004537 010010 JSR R5,TXCTRL ;OUTPUT 2ND SYNC CHARACTER
5165 024672 000001 TSOM ;AND KNOCK DOWN TBMT
5166 024674 000010 8.
5167
5168 024676 004537 010010 JSR R5,TXCTRL ;OUTPUT 3RD SYNC CHARACTER (125)
5169 024702 000001 TSOM ;AND KNOCK DOWN TBMT
5170 024704 000010 8.
5171
5172 024706 004537 011364 JSR R5,RCV1ST ;CLOCK AND RCV TSOM
5173 024712 000000 0
5174 024714 103003 BCC ,+8. ;BR IF NO ERROR

```

TEST 14 - BCP TX RESET W IDLE = 0

```

5175 024716          ERROR          ;REPORT STACKED ERROR
      024716 104460
5176 024720          ESCAPE TST     ;SKIP TO END OF TEST
      024720 104410
      024722 000040
5177
5178 024724 004537 010110          JSR      R5,RXCHAR      ;READ AND CHECK FOR 5/AR 125 SYNC
5179 024730 000125                    125          ; AND CLOCK IN NEXT ONE.
5180 024732 000000                    0
5181 024734 000010                    8.
5182 024736 103003                    BCC     .+8.          ;BR IF NO ERROR
5183 024740          ERROR          ;REPORT STACKED ERROR
      024740 104460
5184 024742          ESCAPE TST     ;SKIP TO END OF TEST
      024742 104410
      024744 000016
5185
5186 024746 004537 005126          JSR      R5,RSTCHK     ;RESET USYRT/VERIFY SAME
5187 024752 103003                    BCC     .+8.          ;BR IF NO ERROR
5188 024754          ERROR          ;REPORT STACKED ERROR
      024754 104460
5189 024756          ESCAPE TST
      024756 104410
      024760 000002
5190
5191 024762          ENDTST
      024762
      024762 104401

```

L10045: TRAP C#ETST

TEST 15 -- BCP TX RESET W/IDLE * 1

5215

.SBTTL TEST 15 -- BCP TX RESET W/IDLE * 1

```

;*****
;*
;* TEST 15 -- BCP TX RESET W/IDLE * 1
;*
;* THE USYRT IS INITIALIZED FOR "BYTE-CONTROL PROTOCOL" (BCP) WITH IDLE
;* SET TO ONE AND A 226 SYNC CHARACTERS LOADED INTO S/AR AND TXDB.
;* THE VALID STATE OF THE USYRT REGISTERS IS READ AND CHECKED. TXE IS
;* ASSERTED TO ENABLE THE TRANSMITTER LOGIC.
;* THEN, TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE OBSERVING
;* TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
;* (TXBE SHOULD GO HIGH; AT THIS TIME THE TXDB SYNC CHARACTER SHOULD
;* BE LOADED INTO TXSO AND TSOM IS AGAIN SET -- DRIVING TXBE LOW.)
;* AFTER THE RECEIVER IS SYNCHRONIZED (TWO SYNC CHARACTERS), TXDB IS
;* LOADED WITH A 125 AND, WITH TSOM STILL = 1 (SYNC SOURCE = TXDB), THE
;* USYRT IS AGAIN CLOCKED.
;* AT THIS POINT, IF THE IDLE BIT WORKED, THE VALUE 125 WILL BE READ
;* BY THE RECEIVER. OTHERWISE A 226 WILL BE READ, INDICATING TXDB WASN'T
;* PROVIDING THE SYNC CHARACTERS.
;* WHEN TXBE GOES HIGH AGAIN, THE USYRT IS RESET.
;* ALL REGISTERS ARE AGAIN READ AND CHECKED.
;* THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY
;* ONE MARK AND THREE SYNC'S (226,226,125 FROM TXDB) ARE TRANSMITTED.
;* ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN THE
;* SEQUENCE.
;*
;*****

```

```

;
; BGNTST
;
; T15::
5216 024764
5217 024764 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
5218
5219 024770 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5220 024774 047626 DDCMP!IDLES!NOCHK!SYNCH ;SET DDCMP,STRIP,IDLE,NO CHECK,SYNCH=226
5221 024776 000000 0 ;USE 8 BIT CHARS
5222 025000 103003 BCC .+8. ;BR IF NO ERROR
5223 025002 ERROR ;REPORT STACKED ERROR
5224 025002 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
5225 025004 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5226 025004 000136 .WORD L10046-.
5225
5226 025010 004537 007676 JSR R5,TXCHAR ;LOAD 2ND SYNCH, TX 1ST SYNCH
5227 025014 000226 SYNCH
5228 025016 000007 7.
5229 025020 103003 BCC .+8. ;BR IF NO ERROR
5230 025022 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
5231 025022 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
5232 025024 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5233 025024 000116 .WORD L10046-.
5232
5233 025030 004537 007676 JSR R5,TXCHAR ;LOAD 125, TX 2ND SYNCH
5234 025034 000125 125
5235 025036 000010 8.

```


TEST 16 -- BCP TX UNDERRUN W/T SOM TERMINATION

5284

.SBTTL TEST 16 -- BCP TX UNDERRUN W/T SOM TERMINATION

```

;*****
;*
;* TEST 16 -- BCP TX UNDERRUN W/T SOM TERMINATION
;*
;* THE USYRT IS INITIALIZED FOR BCP WITH IDLE = 1 AND TXC IS MANUALLY
;* CONTROLLED UNTIL TWO SYNC CHARACTERS AND ONE DATA CHARACTER (000)
;* HAVE BEEN TRANSMITTED. AT THIS TIME WHEN TXBE IS ASSERTED BY THE
;* USYRT, NO DATA IS LOADED INTO TXDB -- FORCING AN UNDER RUN
;* CONDITION. TXU AND TERR ARE CHECKED BOTH BEFORE AND AFTER THEIR
;* EXPECTED ASSERTIONS. AFTER THE FIRST NON-DATA CHARACTER (WHICH
;* SHOULD BE THE MARK CHARACTER) HAS BEEN STARTED, IDLE IS SET TO 0.
;* THIS SHOULD FORCE THE NEXT NON-DATA CHARACTER TO BE A SYNC CHARACTER
;* FROM S/AR. WHILE THIS SYNC CHARACTER IS BEING TRANSMITTED, TSOM IS
;* ASSERTED (CLEARING TXU AND TERR) -- IDLE IS LEFT AT 0. TXBE IS THEN
;* CYCLED THROUGH AT LEAST ONE MORE SYNC CHARACTER AND THE TEST IS
;* ABORTED. ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS
;* WITHIN THE SEQUENCE.
;* NOTE: BITS SHIFT OUT OF TX LSB FIRST.
;*
;*****

```

```

;
; BGNTST
;
5285 025146 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T16::
5286
5287 025152 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5288 025156 047626 DDCMP!IDLES!NOCHK!SYNCH ;SET DDCMP,STRIP,IDLE,NO CHECK,SYNCH=226
5289 025160 040000 NORXEN ;USE 8 BIT CHARS, RECEIVER DISABLED
5290 025162 103003 BCC .+8. ;BR IF NO ERROR
5291 025164 ERROR ;REPORT STACKED ERROR
025164 104460 TRAP C$ERROR
5292 025166 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
025166 104410 .WORD L10047-.
025170 000462
5293
5294 025172 004537 007116 JSR R5,CHKTSO ;CHECK 1ST BIT OF EXPECTED "SYNCH"
5295 025176 000000 0 ; CHARACTER (SHOULD BE 0)
5296 025200 103003 BCC .+8. ;BR IF NO ERROR
5297 025202 ERROR ;REPORT STACKED ERROR
025202 104460 TRAP C$ERROR
5298 025204 ESCAPE TST ;AND EXIT TEST TRAP C$ESCAPE
025204 104410 .WORD L10047-.
025206 000444
5299
5300 025210 004537 007256 JSR R5,SERIAL ;READ REMAINING 7 BITS OF "SYNCH" CHARACTER
5301 025214 000007 7. ; (OFF OF TSO BIT)
5302 025216 000151 151 ; EXPECTED BIT SEQUENCE (0010110)
5303 025220 103003 BCC .+8. ;BR IF NO ERROR
5304 025222 ERROR ;REPORT STACKED ERROR
025222 104460 TRAP C$ERROR
5305 025224 ESCAPE TST ;AND EXIT TEST TRAP C$ESCAPE
025224 104410 .WORD L10047-.
025226 000424
5306
5307 025230 004537 003734 JSR R5,WRITEI ;LOAD 1ST DATA CHARACTER (000)

```


TEST 16 -- BCP TX UNDERRUN WITH TERMINATION

```

5308 025234 120402          TDSRL
5309 025236 000000          000
5310
5311 025240 004537 010010    JSR      R5,TXCTRL      ;CLEAR TSOM
5312 025244 000000          000
5313 025246 000000          0
5314
5315 025250 004537 007256    JSR      R5,SERIAL      ;READ 2ND SYNCH CHARACTER VIA TSO
5316 025254 000010          8.          ; 8 BIT CHAR/CLOCK TICKS
5317 025256 000151          151         ; EXPECTED BIT SEQUENCE (010010110)
5318 025260 103003          BCC      .+8.          ;BR IF NO ERROR
5319 025262          ERROR          ;REPORT STACKED ERROR
5320 025264          104460          ESCAPE   TST          ;AND EXIT TEST
5320 025264          104410          TRAP    C$ERROR
5320 025266          000364          .WORD   L10047-.
5321
5322          ;-----
5322          ; VERIFY THAT TXU AND TERR BITS ARE ZERO AT THIS POINT
5322          ;-----
5323
5324 025270 004537 005432    JSR      R5,CKUSTS      ;CHECK USYRT STATUS
5325 025274 000114          TXACT!TBMT!TSO        ; FOR TRANSMITTER ACTIVE (NO TXU YET!)
5326 025276 103003          BCC      .+8.          ;BR IF NO ERROR
5327 025300          ERROR          ;REPORT STACKED ERROR
5328 025302          104460          ESCAPE   TST          ;SKIP TO END OF TEST
5328 025302          104410          TRAP    C$ERROR
5328 025304          000346          .WORD   L10047-.
5329
5330 025306 004537 003610    JSR      R5,READI      ;GET TX ERROR BIT (TO SEE IF SET)
5331 025312 120403          TDSRH        ; TERR BIT IS IN TDSRH
5332 025314 000000          000         ; TDSRH STORED HERE
5333 025316 133727 025314 000200 10$: BITB   10$,#BIT7    ;CHECK TERR BIT
5334 025324 001406          BEQ      15$          ; BR IF TERR NOT SET
5335 025326          GEDF   EM98,ERR12 ;REPORT ERROR: "TERR NOT CLEAR"
5335          ; "DEVICE FATAL" ERROR # 49
5336 025326          104455          TRAP    C$ERDF
5336 025330          000061          .WORD   49
5336 025332          014357          .WORD   EM98
5336 025334          017722          .WORD   ERR12
5336 025336          ESCAPE   TST          ;AND EXIT TEST
5336 025336          104410          TRAP    C$ESCAPE
5336 025340          000312          .WORD   L10047-.
5337
5338          ;-----
5338          ; NOW READ 1ST DATA CHARACTER (AND CHECK TERR=1,TXU=1)
5338          ;-----
5339
5340 025342 004537 007256    15$: JSR      R5,SERIAL      ;READ DATA CHARACTER (000) VIA TSO
5341 025346 000010          8.          ; 8 BIT CHAR/CLOCK TICKS
5342 025350 000000          000         ; EXPECTED BIT SEQUENCE (00000000)
5343 025352 103003          BCC      .+8.          ;BR IF NO ERROR
5344 025354          ERROR          ;REPORT STACKED ERROR
5345 025354          104460          ESCAPE   TST          ;AND EXIT TEST
5345 025354          104410          TRAP    C$ERROR
5345 025356          000272          .WORD   L10047-.
5346
5347 025362 004537 011614    JSR      R5,STEPLU     ;GENERATE 1 TICK TO UNDERRUN TX
5348 025366 000001          1

```

TEST 16 -- BCP TX UNDERRUN W/T SOM TERMINATION

```

5349
5350 025370 004537 005432      JSR      R5,CKUSTS      ;CHECK USYRT STATUS
5351 025374 000116      TBMT!TSO!TXACT!TXU    ; FOR TBMT AND TX UNDERRUN
5352 025376 103003      BCC      .+8.          ;BR IF NO ERROR
5353 025400      ERROR              ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5354 025402      ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD    L10047-.
5355
5356 025406 004537 003610      JSR      R5,READI      ;GET TX ERROR BIT (TO SEE IF SET)
5357 025412 120403      TDSRH          ; TERR BIT IS IN TDSRH
5358 025414 000000      000          ; TDSRH STORED HERE
5359 025416 133727 025414 000200 20$:    BITB      20$,#BIT7    ;CHECK TERR BIT
5360 025424 001006      BNE      25$          ; BR IF TERR IS SET
5361 025426      GEDF      EM99,ERR12 ;REPORT ERROR: "TERR NOT SET"
                                ; "DEVICE FATAL" ERROR # 50
                                TRAP      C$ERDF
                                .WORD    50
                                .WORD    EM99
                                .WORD    ERR12
5362 025436      ESCAPE TST          ;AND EXIT TEST
                                TRAP      C$ESCAPE
                                .WORD    L10047-.
5363
5364 ; -----
5365 ; NOW CHECK MARK CHARACTER (RESULT OF UNDERRUNNING TRANSMITTER)
5366 025442 004537 007116 25$:    JSR      R5,CHKTSO    ;CHECK 1ST BIT OF EXPECTED "MARK"
5367 025446 000001      1          ; CHARACTER (BIT SHOULD=1)
5368 025450 103003      BCC      .+8.          ;BR IF NO ERROR
5369 025452      ERROR              ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5370 025454      ESCAPE TST          ;AND EXIT TEST
                                TRAP      C$ESCAPE
                                .WORD    L10047-.
5371
5372 025460 004537 007256      JSR      R5,SERIAL    ;READ MARK CHARACTER VIA TSO
5373 025464 000007      7          ; 7 BIT CHAR/CLOCK TICKS
5374 025466 000177      177         ; EXPECTED BIT SEQUENCE (1111111)
5375 025470 103003      BCC      .+8.          ;BR IF NO ERROR
5376 025472      ERROR              ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5377 025474      ESCAPE TST          ;AND EXIT TEST
                                TRAP      C$ESCAPE
                                .WORD    L10047-.
5378
5379 ; -----
5380 ; SET IDLE=0, VERIFY SYNCH CHARACTER IS OUTPUT
5381 025500 004537 003734      JSR      R5,WRITEI    ;SET IDLE BIT=0
5382 025504 120405      PCSARH          ;
5383 025506 000107      PROTO!XYZ
5384
5385 025510 004537 007256      JSR      R5,SERIAL    ;READ MARK CHARACTER VIA TSO
5386 025514 000010      8.          ; 8 BIT CHAR/CLOCK TICKS
5387 025516 000377      377         ; EXPECTED BIT SEQUENCE (11111111)
5388 025520 103003      BCC      .+8.          ;BR IF NO ERROR
5389 025522      ERROR              ;REPORT STACKED ERROR

```

TEST 16 -- BCP TX UNDERRUN W/T SOM TERMINATION

```

5390 025522 104460          ESCAPE TST          ;AND EXIT TEST          TRAP C$ERROR
      025524          ;AND EXIT TEST          TRAP C$ESCAPE
      025524 104410          ;AND EXIT TEST          .WORD L10047-.
      025526 000124          ;AND EXIT TEST          ;AND EXIT TEST

5391 025530 004537 007256 JSR R5,SERIAL ;READ SYNCH CHARACTER VIA TSO
5392 025534 000010      8. ; 8 BIT CHAR/CLOCK TICKS
5393 025536 000151      151 ; EXPECTED BIT SEQUENCE (010010110)
5394 025540 103003      BCC .+8. ;BR IF NO ERROR
5395 025542          ERROR ;REPORT STACKED ERROR
      025542 104460          ;AND EXIT TEST          TRAP C$ERROR
5396 025544          ESCAPE TST          ;AND EXIT TEST          TRAP C$ESCAPE
      025544 104410          ;AND EXIT TEST          .WORD L10047-.
      025546 000104          ;AND EXIT TEST          ;AND EXIT TEST

5398 025550 004537 010010 JSR R5, TXCTRL ; * ASSERT TSOM (SHOULD CLEAR TXU, TERR)
5399 025554 000001      TSOM ; *
5400 025556 000000      0 ; *
5401 025556 000000          ; *
5402          ;-----
5403          ; VERIFY THAT TXU AND TERR BITS ARE ZERO AT THIS POINT
5404          ;-----
5405 025560 004537 005432 JSR R5, CKUSTS ;CHECK USYRT STATUS
5406 025564 000014      TXACT!TSO ; FOR TRANSMITTER ACTIVE (NO TXU YET!)
5407 025566 103003      BCC .+8. ;BR IF NO ERROR
5408 025570          ERROR ;REPORT STACKED ERROR
      025570 104460          ;AND EXIT TEST          TRAP C$ERROR
5409 025572          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ESCAPE
      025572 104410          ;SKIP TO END OF TEST          .WORD L10047-.
      025574 000056          ;SKIP TO END OF TEST          ;SKIP TO END OF TEST

5410 025576 004537 003610 JSR R5, READI ;GET TX ERROR BIT (TO SEE IF SET)
5411 025602 120403      TDSRH ; TERR BIT IS IN TDSRH
5412 025604 000000      000 ; TDSRH STORED HERE
5413 025606 133727 025604 000200 17$: BITB 17$, #BIT7 ;CHECK TERR BIT
5414 025614 001406      BEQ 18$ ; BR IF TERR NOT SET
5415 025616          GEQF EM98,ERR12 ;REPORT ERROR: "TERR NOT CLEAR"
      025616 104455          ; "DEVICE FATAL" ERROR # 51
      025620 000063          ;AND EXIT TEST          TRAP C$ERDF
      025622 014357          ;AND EXIT TEST          .WORD 51
      025624 017722          ;AND EXIT TEST          .WORD EM98
      025626          ;AND EXIT TEST          .WORD ERR12
5417 025626          ESCAPE TST          ;AND EXIT TEST          TRAP C$ESCAPE
      025626 104410          ;AND EXIT TEST          .WORD L10047-.
      025630 000022          ;AND EXIT TEST          ;AND EXIT TEST

5418          ;-----
5419          ; READ/CHECK FOR SYNCH CHARACTER
5420          ;-----
5421 025632 004537 007256 18$: JSR R5,SERIAL ;READ !SYNCH! CHARACTER VIA TSO
5422 025636 000010      8. ; 8 BIT CHAR/CLOCK TICKS
5423 025640 000151      151 ; EXPECTED BIT SEQUENCE (10010110)
5424 025642 103003      BCC .+8. ;BR IF NO ERROR
5425 025644          ERROR ;REPORT STACKED ERROR
      025644 104460          ;AND EXIT TEST          TRAP C$ERROR
5426 025646          ESCAPE TST          ;AND EXIT TEST          TRAP C$ESCAPE
      025646 104410          ;AND EXIT TEST          .WORD L10047-.
      025650 000002          ;AND EXIT TEST          ;AND EXIT TEST
5427 025652          ENDTST

```

K11

CNDMCAO DMV11 LINE UNIT DIAG1 MACRO M1200 22-FEB-84 15:31 PAGE 81-4

SEQ 0140

TEST 16 -- BCP TX UNDERRUN W/T SOM TERMINATION

025652
025652 104401

L10047: TRAP C\$ETST

TEST 17 -- BCP TX UNDERRUN W/RESET TERMINATION

5443

.SBTTL TEST 17 -- BCP TX UNDERRUN W/RESET TERMINATION

```

; +*****
; *
; * TEST 17 -- BCP TX UNDERRUN W/RESET TERMINATION
; *
; * THE USYRT IS INITIALIZED FOR BCP WITH IDLE = 1 AND TXC IS MANUALLY
; * CONTROLLED UNTIL TWO SYNC CHARACTERS AND ONE DATA CHARACTER HAVE
; * BEEN TRANSMITTED. AT THIS TIME WHEN TXBE IS ASSERTED BY THE USYRT,
; * NO DATA IS LOADED INTO TXDB -- FORCING AN UNDER RUN CONDITION, TXU
; * AND TERR ARE CHECKED BOTH BEFORE AND AFTER THEIR EXPECTED
; * ASSERTIONS. AFTER THE FIRST NON-DATA CHARACTER (WHICH SHOULD BE THE
; * MARK CHARACTER) HAS BEEN STARTED, IDLE IS SET TO 0. THIS SHOULD
; * FORCE THE NEXT NON-DATA CHARACTER TO BE A SYNC CHARACTER.
; * IMMEDIATELY AFTER THIS SYNC CHARACTER HAS BEING TRANSMITTED, A
; * PROGRAM RESET IS ISSUED AND ALL REGISTERS ARE CHECKED. ERROR
; * LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN THE
; * SEQUENCE.
; *
; -*****

```

```

;
; BGNTST
;
5444 025654 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T17::
5445
5446 025660 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5447 025664 047626 DDCMP!IDLES!NOCHK!SYNCH ;SET DDCMP,STRIP,IDLE,NO CHECK,SYNCH=226
5448 025666 040000 NORXEN ;USE 8 BIT CHARS, RECEIVER DISABLED
5449 025670 103003 BCC .+8. ;BR IF NO ERROR
5450 025672 ERROR ;REPORT STACKED ERROR
5451 025674 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
025674 104410 ; TRAP C$ESCAPE
025676 000354 .WORD L10050-.
5452
5453 025700 004537 007116 JSR R5,CHKTSO ;CHECK 1ST BIT OF EXPECTED "SYNCH"
5454 025704 000000 0 ; CHARACTER (SHOULD BE 0)
5455 025706 103003 BCC .+8. ;BR IF NO ERROR
5456 025710 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
025710 104460 ; TRAP C$ERROR
5457 025712 ESCAPE TST ;AND EXIT TEST TRAP C$ESCAPE
025712 104410 ; TRAP C$ESCAPE
025714 000336 .WORD L10050-.
5458
5459 025716 004537 007256 JSR R5,SERIAL ;READ REMAINING 7 BITS OF "SYNCH" CHARACTER
5460 025722 000007 7. ; (OFF OF TSO BIT)
5461 025724 000151 ; EXPECTED BIT SEQUENCE (0010110)
5462 025726 103003 BCC .+8. ;BR IF NO ERROR
5463 025730 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
025730 104460 ; TRAP C$ERROR
5464 025732 ESCAPE TST ;AND EXIT TEST TRAP C$ESCAPE
025732 104410 ; TRAP C$ESCAPE
025734 000316 .WORD L10050-.
5465
5466 025736 004537 003734 JSR R5,WRITEI ;LOAD 1ST DATA CHARACTER (000)
5467 025742 120402 TDSRL
5468 025744 000000 000

```

TEST 17 -- BCP TX UNDERRUN W/RESET TERMINATION

```

5469
5470 025746 004537 010010      JSR      R5,TXCTRL      ;CLEAR TSOM
5471 025752 000000              000
5472 025754 000000              0
5473
5474 025756 004537 007256      JSR      R5,SERIAL      ;READ 2ND SYNCH CHARACTER VIA TSO
5475 025762 000010              8.          ; 8 BIT CHAR/CLOCK TICKS
5476 025764 000151              151        ; EXPECTED BIT SEQUENCE (010010110)
5477 025766 103003              BCC      .+8.          ;BR IF NO ERROR
5478 025770              ERROR          ;REPORT STACKED ERROR
                    104460
5479 025772              ESCAPE TST          ;AND EXIT TEST
                    025772 104410
                    025774 000256              TRAP      C$ERROR
                    .WORD      L10050-.

5480
5481      ;-----
5482      ; VERIFY THAT TXU AND TERR BITS ARE ZERO AT THIS POINT
5483      ;-----
5483 025776 004537 005432      JSR      R5,CKUSTS      ;CHECK USYRT STATUS
5484 026002 000114      TXACT!TBMT!TSO          ; FOR TRANSMITTER ACTIVE (NO TXU YET!)
5485 026004 103003              BCC      .+8.          ;BR IF NO ERROR
5486 026006              ERROR          ;REPORT STACKED ERROR
                    104460
5487 026010              ESCAPE TST          ;SKIP TO END OF TEST
                    026010 104410
                    026012 000240              TRAP      C$ERROR
                    .WORD      L10050-.

5488
5489 026014 004537 003610      JSR      R5,READI      ;GET TX ERROR BIT (TO SEE IF SET)
5490 026020 120403      TDSRH          ; TERR BIT IS IN TDSRH
5491 026022 000000              000          ; TDSRH STORED HERE
5492 026024 133727 026022 000200 10$: BITB      10$,0BIT7 ;CHECK TERR BIT
5493 026032 001406              BEQ      15$          ; BR IF TERR NOT SET
5494 026034              GEDF      EM98,ERR12 ;REPORT ERROR: "TERR NOT CLEAR"
                    ; "DEVICE FATAL" ERROR # 52
                    104455
                    026036 000064              TRAP      C$ERDF
                    026040 014357              .WORD      52
                    026042 017722              .WORD      EM98
                    .WORD      ERR12
5495 026044              ESCAPE TST          ;AND EXIT TEST
                    026044 104410
                    026046 000204              TRAP      C$ESCAPE
                    .WORD      L10050-.

5496
5497      ;-----
5498      ; NOW READ 1ST DATA CHARACTER (AND CHECK TERR=1,TXU=1)
5499      ;-----
5499 026050 004537 007256 15$: JSR      R5,SERIAL      ;READ DATA CHARACTER (000) VIA TSO
5500 026054 000010              8.          ; 8 BIT CHAR/CLOCK TICKS
5501 026056 000000              000          ; EXPECTED BIT SEQUENCE (00000000)
5502 026060 103003              BCC      .+8.          ;BR IF NO ERROR
5503 026062              ERROR          ;REPORT STACKED ERROR
                    104460
5504 026064              ESCAPE TST          ;AND EXIT TEST
                    026064 104410
                    026066 000164              TRAP      C$ESCAPE
                    .WORD      L10050-.

5505
5506 026070 004537 011614      JSR      R5,STEPLU      ;GENERATE 1 TICK TO UNDERRUN TX
5507 026074 000001              1
5508
5509 026076 004537 005432      JSR      R5,CKUSTS      ;CHECK USYRT STATUS

```

TEST 17 -- BCP TX UNDERRUN W/RESET TERMINATION

```

5510 026102 000116          TBMT!TSO!TXACT!TXU      ; FOR TBMT AND TX UNDERRUN
5511 026104 103003          BCC      ,+8.           ;BR IF NO ERROR
5512 026106 104460          ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5513 026110 104410          ESCAPE TST           ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD    L10050-.
5514 026112 000140
5515 026114 004537 003610    JSR      R5,ERR1       ;GET TX ERROR BIT (TO SEE IF SET)
5516 026120 120403          TDSRH          ; TERR BIT IS IN TDSRH
5517 026122 000000          000             ; TDSRH STORED HERE
5518 026124 133727 026122 000200 20$: BITB      20$,#BIT7    ;CHECK TERR BIT
5519 026132 001006          BNE      25$       ; BR IF TERR IS SET
5520 026134 104455          GEDF      EI199,ERR12 ;REPORT ERROR: "TERR NOT SET"
                                ; "DEVICE FATAL" ERROR # 53
                                TRAP      C$ERDF
                                .WORD    53
                                .WORD    EM99
                                .WORD    ERR12
5521 026144 104410          ESCAPE TST           ;AND EXIT TEST
                                TRAP      C$ESCAPE
                                .WORD    L10050-.
5522 026146 000104
5523
5524
5525 026150 004537 003734    25$: JSR      R5,WRITEI   ;CLEAR IDLE BIT
5526 026154 120405          PCSARH
5527 026156 000107          BIT6!BIT2!BIT1!BIT0 ;DDCMP FORMAT, NO ERROR CHECKING
5528
5529 026160 004537 007116    JSR      R5,CHKTSO   ;CHECK 1ST BIT OF EXPECTED "MARK"
5530 026164 000001          1               ; CHARACTER (BIT SHOULD=1)
5531 026166 103003          BCC      ,+8.           ;BR IF NO ERROR
5532 026170 104460          ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5533 026172 104410          ESCAPE TST           ;AND EXIT TEST
                                TRAP      C$ESCAPE
                                .WORD    L10050-.
5534 026174 000056
5535 026176 004537 007256    JSR      R5,SERIAL   ;READ MARK CHARACTER VIA TSO
5536 026202 000007          7               ; 7 BIT CHAR/CLOCK TICKS
5537 026204 000177          177             ; EXPECTED BIT SEQUENCE (1111111)
5538 026206 103003          BCC      ,+8.           ;BR IF NO ERROR
5539 026210 104460          ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5540 026212 104410          ESCAPE TST           ;AND EXIT TEST
                                TRAP      C$ESCAPE
                                .WORD    L10050-.
5541 026214 000036
5542
5543
5544
5545 026216 004537 007256    JSR      R5,SERIAL   ;READ SYNCH CHARACTER VIA TSO
5546 026222 000010          8.              ; 8 BIT CHAR/CLOCK TICKS
5547 026224 000151          151             ; EXPECTED BIT SEQUENCE (0000000)
5548 026226 103003          BCC      ,+8.           ;BR IF NO ERROR
5549 026230 104460          ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR

```

TEST 17 RCP TX UNDERRUN W/RESET TERMINATION

5550	026232			ESCAPE TST		AND EXIT TEST		
	026232	104410					TRAP	C ESCAPE
	026234	000016					.WORD	L10050-
5551								
5552	026236	004537	005126	JSR	R5,RSTCHK	RESET USRT/VERIFY SAME		
5553	026242	103003		BCC	.+8.	BR IF NO ERROR		
5554	026244			ERROR		REPORT STACKED ERROR		
	026244	104460					TRAP	C ERROR
5555	026246			ESCAPE TST		SKIP TO END OF TEST		
	026246	104410					TRAP	C ESCAPE
	026250	000002					.WORD	L10050 .
5556	026252			ENDTST				
	026252						L10050:	
	026252	104401					TRAP	C ETST

TEST 18 BCP TX DISABLE TEST

5568

.SBTTL TEST 18 -- BCP TX DISABLE TEST

```

*****
|*
|* TEST 18 -- BCP TX DISABLE TEST
|*
|* THE USYRT IS INITIALIZED FOR BCP AND A MESSAGE IS STARTED. ONCE THE
|* SECOND DATA CHARACTER IS LOADED INTO TXDB TXE IS DROPPED. TXSD IS
|* WATCHED TO ASSURE THAT THE CHARACTER BEING TRANSMITTED IS COMPLETED.
|* WHEN IT IS, THE USYRT SHOULD DROP TXA AND STOP TRANSMITTING -- THE
|* LAST CHARACTER LOADED INTO TXDB SHOULD BE LOST.
|*
|* CHARACTERS LOADED: 125 252
|* CHARACTERS TRANSMITTED: 125
|*
*****

```

```

|*
|* BGNTST
|*
|* T18::
|*
5569 026254
5570 026254 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
5571
5572 026260 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5573 026264 043626 DDCMP!NOCHK!SYNCH ;SET DDCMP,NO CHECK,SYNCH=226
5574 026266 000000 0 ;USE 8 BIT CHARS
5575 026270 103003 BCC ,+8. ;BR IF NO ERROR
5576 026272 ERROR ;REPORT STACKED ERROR
|* TRAP C#ERROR
|*
5577 026274 ESCAPE TST ;SKIP TO END OF TEST
|* TRAP C#ESCAPE
|* .WORD L10051-.
|*
5578
5579 026300 004537 010010 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTER
5580 026304 000001 TSOM ;AND KNOCK DOWN TBMT
5581 026306 000007 7.
5582
5583 026310 004537 010010 JSR R5,TXCTRL. ;CLEAR TSOM (GET READY TO SEND DATA)
5584 026314 000000 000
5585 026316 000000 0
5586
5587 026320 004537 007676 JSR R5,TXCHAR ;LOAD 125, TX 2ND SYNCH
5588 026324 000125 125
5589 026326 000010 8.
5590 026330 103003 BCC ,+8. ;BR IF NO ERROR
5591 026332 ERROR ;REPORT STACKED ERROR
|* TRAP C#ERROR
|*
5592 026334 ESCAPE TST ;SKIP TO END OF TEST
|* TRAP C#ESCAPE
|* .WORD L10051-.
|*
5593
5594 026340 004537 007676 JSR R5,TXCHAR ;LOAD 252
5595 026344 000252 252
5596 026346 000000 0
5597 026350 103003 BCC ,+8. ;BR IF NO ERROR
5598 026352 ERROR ;REPORT STACKED ERROR
|* TRAP C#ERROR
|*
5599 026354 ESCAPE TST ;SKIP TO END OF TEST

```

TEST 18 -- BCP TX DISABLE TEST

```

026354 104410
026356 000102
5600
5601 026360 004537 003734 JSR R5,WRITEI ;DROP TRANSMIT ENABLE (TXE)
5602 026364 120000 VIAORB
5603 026366 000102 RXEN!TTLOOP
5604
5605 026370 004537 011614 JSR R5,STEPLU ;CLOCK IN 125
5606 026374 000007 7. ;***
5607
5608 026376 004537 011364 JSR R5,RCV1ST ;CLOCK AND RCV 125
5609 026402 000000 0
5610 026404 103003 BCC .+8. ;BR IF NO ERROR
5611 026406 ERROR ;REPORT STACKED ERROR
026408 104460
5612 026410 ESCAPE TST ;SKIP TO END OF TEST
026410 104410 TRAP C!ESCAPE
026412 000046 .WORD L10051-.
5613
5614 026414 004537 010110 JSR R5,RXCHAR ;READ AND CHECK FOR 125
5615 026420 000125 125
5616 026422 000000 0
5617 026424 000010 8. ;AND CLOCK IN NEXT CHAR.
5618 026426 103003 BCC .+8. ;BR IF NO ERROR
5619 026430 ERROR ;REPORT STACKED ERROR
026430 104460 TRAP C!ERROR
5620 026432 ESCAPE TST ;SKIP TO END OF TEST
026432 104410 TRAP C!ESCAPE
026434 000024 .WORD L10051-.
5621
5622 026436 004537 010110 JSR R5,RXCHAR ;READ AND CHECK FOR 377
5623 026442 000377 377 ;** IF 252 IS READ, THEN DROPPING
5624 026444 000000 0 ;** TXEN DIDN'T WORK !!!
5625 026446 000010 8.
5626 026450 103003 BCC .+8. ;BR IF NO ERROR ****
5627 026452 ERROR ;REPORT STACKED ERROR ****
026452 104460 TRAP C!ERROR
5628 026454 ESCAPE TST ;SKIP TO END OF TEST ****
026454 104410 TRAP C!ESCAPE
026456 000002 .WORD L10051-.
5629 026460 ENDTST
026460 L10051: TRAP C!ETST
026460 104401

```

TEST 19 -- FIFO STACKING CHARACTERS TEST

5644

.SBTTL TEST 19 -- FIFO STACKING CHARACTERS TEST

```

;*****
;*
;* TEST 19 -- FIFO STACKING CHARACTERS TEST
;*
;* THE USYRT IS SETUP FOR BCP MODE WITH NO ERROR DETECTION.
;* THIS TEST BEGINS BY SYNCHRONIZING THE RECEIVER AND THEN PROCEEDS
;* TO FILL THE 8 CHARACTER RECEIVER FIFO WITH THE CHARACTERS:
;* 1/2(SYNCH),000,377,125,252,347,030,303,1/2(074).
;* THESE CHARACTERS ARE THEN READ OFF OF THE FIFO AND CHECKED. NOTE
;* THAT NO CLOCKS ARE PROVIDED WHEN RECEIVING THE CHARACTERS SINCE THEY
;* ARE SUPPLIED BY THE FIFO SUPPORT LOGIC IN GROUPS OF 4 TICKS (WHEN
;* RDA = 0).
;* ALSO NOTE THAT DUE TO FIFO TIMING, TWO 'HALF CHARACTERS' ARE LOADED
;* INTO THE FIFO (THE 1ST AND LAST CHARACTERS).
;*
;*****

```

```

;
; BGNTST
;
5645 026462 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T19::
5646
5647 026466 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5648 026472 043626 DDCMP!NOCHK!SYNCH ;SET DDCMP,NO CHECK,SYNCH=226
5649 026474 000000 0 ;USE 8 BIT CHARS
5650 026476 103003 BCC .+8. ;BR IF NO ERROR
5651 026500 ERROR ;REPORT STACKED ERROR
5652 026500 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
5652 026502 104410 TRAP C$ESCAPE
5652 026504 000216 .WORD L10052-.
5653
5654 026506 004537 010010 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTERS
5655 026512 000001 TSOM ;
5656 026514 000007 7. ;
5657 026516 004537 010010 JSR R5,TXCTRL ;CLEAR TSOM
5658 026522 000000 000
5659 026524 000000 0
5660
;-----
; FILL THE FIFO WITH CHARACTERS (DATA1 - DATA8)
;-----
5663 026526 012702 026724 MOV #TXIBL3,R2 ;SET UP TABLE POINTER
5664 026532 112237 026542 5$: MOVB (R2)+,10$ ;SETUP TRANSMIT CHARACTER
5665
5666 026536 004537 007676 JSR R5,TXCHAR ;TRANSMIT A CHARACTER
5667 026542 000000 000 ;** HOLE FOR NEXT TX CHARACTER
5668 026544 100010 NCTBMT+256.!8. ;NO CHECK OF INITIAL TBMT=0
5669 026546 103003 BCC .+8. ;BR IF NO ERROR
5670 026550 ERROR ;REPORT STACKED ERROR
5671 026550 104460 TRAP C$ERROR
5671 026552 ESCAPE TST ;SKIP TO END OF TEST
5671 026552 104410 TRAP C$ESCAPE
5671 026554 000146 .WORD L10052-.
5672
5673 026556 022702 026736 CMP #TXEND3,R2

```

TEST 19 -- FIFO STACKING CHARACTERS TEST

```

5674 026562 001363          BNE      5$
5675
5676 026564 004537 011614    JSR      R5,STEPLU      ;ADD A FEW TX TICKS TO COMPLETELY
5677 026570 000005          5          ; FILL UP FIFO.
5678
5679          ;-----
5680          ; THE FIFO IS NOW FULL OF CHARACTERS.
5681          ; NOW READ/VERIFY THEM.
5682 026572 012702 026724    MOV      #TXTBL3,R2     ;SET UP TABLE POINTER
5683 026576 112237 026606    15$:    MOVB    (R2)+,20$     ;SETUP EXPECTED CHARACTER
5684
5685 026602 004537 010110    JSR      R5,RXCHAR     ;READ & CHK CHARACTER
5686 026606 000000          20$:    000          ;** HOLE FOR EXPECTED RECEIVE CHAR.
5687 026610 000000          0          ;
5688 026612 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
5689 026614 103003          BCC     .+8.         ;BR IF NO ERROR
5690 026616 104460          ERROR          ;REPORT STACKED ERROR
5691 026620 104460          ESCAPE  TST          ;SKIP TO END OF TEST          TRAP    C$ERROR
5692 026620 104410          .WORD          TRAP    C$ESCAPE
5693 026622 000100          .WORD          L10052-.
5694
5695
5696 026624 022702 026734    CMP      #TXEND3-2,R2
5697 026630 001362          BNE     15$
5698
5699 026632 004537 011614    JSR      R5,STEPLU     ;CLOCK IN LAST FEW CHARACTERS OFF
5700 026636 000015          15          ; OF FIFO.
5701
5702 026640 004537 010110    JSR      R5,RXCHAR     ;READ & CHK CHARACTER
5703 026644 000303          303
5704 026646 000000          0
5705 026650 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
5706 026652 103003          BCC     .+8.         ;BR IF NO ERROR
5707 026654 104460          ERROR          ;REPORT STACKED ERROR          TRAP    C$ERROR
5708 026656 104460          ESCAPE  TST          ;SKIP TO END OF TEST          TRAP    C$ESCAPE
5709 026658 104410          .WORD          TRAP    C$ESCAPE
5710 026660 000042          .WORD          L10052-.
5711
5712
5713 026662 004537 010110    JSR      R5,RXCHAR     ;READ & CHK CHARACTER
5714 026666 000074          074
5715 026670 000000          0
5716 026672 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
5717 026674 103003          BCC     .+8.         ;BR IF NO ERROR
5718 026676 104460          ERROR          ;REPORT STACKED ERROR          TRAP    C$ERROR
5719 026678 104460          ESCAPE  TST          ;SKIP TO END OF TEST          TRAP    C$ESCAPE
5720 026680 104410          .WORD          TRAP    C$ESCAPE
5721 026682 000020          .WORD          L10052-.
5722
5723
5724 026704 004537 011532    JSR      R5,ENDTRN     ;SHUT DOWN TRANSMITTER, RECEIVER
5725 026710 000010          8.
5726 026712 103003          BCC     .+8.         ;BR IF NO ERROR
5727 026714 104460          ERROR          ;REPORT STACKED ERROR          TRAP    C$ERROR
5728 026716 104460          ESCAPE  TST          ;SKIP TO END OF TEST          TRAP    C$ESCAPE
5729 026718 104410          .WORD          TRAP    C$ESCAPE

```

TEST 19 - FIFO STACKING CHARACTERS TEST

```

026720 000002
5720 026722
      026722
      026722 104401
5721
5722 026724      226
5723 026725      226
5724 026726      000
5725 026727      377
5726 026730      125
5727 026731      252
5728 026732      347
5729 026733      030
5730 026734      303
5731 026735      074
5732 026736      000
5733
5734

```

ENDTST

```

      .WORD L10052
L10052: TRAP C$ETST

```

```

;-----;
TXTBL3: .BYTE 226 ;SYNCH
        .BYTE 226 ;SYNCH
        .BYTE 000
        .BYTE 377
        .BYTE 125
        .BYTE 252
        .BYTE 347
        .BYTE 030
        .BYTE 303
        .BYTE 074
TXEND3: .BYTE 000
        .EVEN
;-----;

```

TEST 20 -- BCP CHARACTER LENGTH TEST

5744

.SBTTL TEST 20 -- BCP CHARACTER LENGTH TEST

```

;*****
;*
;* TEST 20 -- BCP CHARACTER LENGTH TEST
;*
;* THE USYRT IS INITIALIZED FOR BCP WITH NO ERROR CHECKING. TXC IS MANUALLY
;* CONTROLLED UNTIL TWO SYNC CHARACTERS HAVE BEEN TRANSMITTED. THEN 3
;* SUBTESTS FOLLOW, EACH ONE USING A DIFFERENT TRANSMIT CHARACTER LENGTH
;* STARTING AT FIVE (5) AND ENDING WITH SEVEN (7).
;*
;* TEST PATTERN: 111 222 333 044 155 266 377
;*
;*****

```

5745 026740 004737 005420

```

;
; BGNTST
;
; JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
;
;-----
; SUBROUTINE # 1: 5 BIT CHARACTERS
;-----

```

5746
5747
5748

5749 026744 104402 007400
026744 004537
5750 026746 004537
5751 026752 043626
5752 026754 000245
5753 026756 103003
5754 026760 104460
5755 026762 104410
026764 000126

```

;
; BGNSUB
;
; JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
; DDCMP!NOCHK!SYNCH ;SET CHAR MODE,NO ERROR CHECKING,S/AR=226
; BIT7!BIT5!BIT2!BIT0 ;TXCL=RXCL=5 BITS
; BCC .+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
;
; ESCAPE SUB ;SKIP TO END OF TEST
;
; TRAP C$ESCAPE
; .WORD L10054-.
;
; TRAP C$B SUB
;
; T20.1:

```

5756 026766 004537 010010
5758 026772 000001
5759 026774 000004
5760 026776 004537 010010
5761 027002 000000
5762 027004 000000
5763

```

;
; JSR R5,TXCTRL ;LOAD 2ND SYNCH,TX 1ST SYNCH
; TSOM
; 4.
; JSR R5,TXCTRL ;CLEAR TSOM
; 000
; 0
;-----

```

5764 027006 012703 027436
5765 027012 112337 027022
5766

```

;
; 10$: MOV #T24TBL,R3 ;SET UP DATA TABLE POINTER
; MOVB (R3)+,1$ ;INSTALL NEXT TX CHARACTER
;-----

```

5767 027016 004537 007676
5768 027022 000000
5769 027024 000005
5770 027026 103003
5771 027030 104460
5772 027032 104410
027034 000056

```

;
; 1$: JSR R5,TXCHAR ;TRANSMIT CHARACTER ( **> RX/FIFO )
; 000 ;** HOLE FOR NEXT CHARACTER **
; 5.
; BCC .+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
;
; ESCAPE SUB ;SKIP TO END OF TEST
;
; TRAP C$ERROR
;
; TRAP C$ESCAPE
; .WORD L10054-.
;

```

5773 027036 022703 027447
5775 027042 001363
5776

```

;
; CMP #T24TBL+9.,R3 ;ALL CHARACTERS TRANSMITTED ?
; BNE 10$ ; IF NOT, TX ANOTHER ONE
;-----

```

TEST 20 -- BCP CHARACTER LENGTH TEST

```

5777 027044 012703 027436      MOV     #T24TBL,R3      ;SET UP DATA TABLE POINTER
5778 027050 112337 027066      MOVB   (R3)+,4$       ;INSTALL NEXT EXPECTED RX CHARACTER
5779 027054 142737 000340 027066  BICB   #340,4$       ;MASK OUT UNTRANSMITTED BITS
5780
5781 027062 004537 010110      JSR    R5,RXCHAR     ;READ/CHECK NEXT CHARACTER
5782 027066 000000      4$:    000           ;** HOLE FOR NEXT EXPECTED CHARACTER
5783 027070 000000      0
5784 027072 100000      NCCRDA           ;NO INITIAL CHECK OF RDA=0
5785 027074 103003      BCC    .+8.        ;BR IF NO ERROR
5786 027076      ERROR           ;REPORT STACKED ERROR
5787 027100      ESCAPE  SUB      ;SKIP TO END OF TEST                TRAP    C$ERROR
5788 027100 104410      .WORD           TRAP    C$ESCAPE
5789 027104 022703 027445      CMP    #T24TBL+7,R3 ;ALL CHARACTERS CHECKED ?
5790 027110 001357      BNE    40$         ; IF NOT, CHECK ANOTHER ONE
5791
5792 027112      ENDSUB
5793 027112      L10054:         TRAP    C$ESUB
5794 027112 104403
5795
5796 027114      ;-----
5797 027114      ; SUBROUTINE # 2: 6 BIT CHARACTERS
5798 027114      ;-----
5799 027114      BGNSUB
5800 027116 004537 007400      JSR    R5,INITRN    ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5801 027122 043626      DDCMP!NOCHK!SYNCH  ;SET CHAR MODE,NO ERROR CHECKING,S/AR=226
5802 027124 000306      BIT7!BIT6!BIT2!BIT1 ;TXCL=RXCL=6 BITS
5803 027126 103003      BCC    .+8.        ;BR IF NO ERROR
5804 027130      ERROR           ;REPORT STACKED ERROR                TRAP    C$ERROR
5805 027132      ESCAPE  SUB      ;SKIP TO END OF TEST                TRAP    C$ESCAPE
5806 027134 104410      .WORD           TRAP    C$ESCAPE
5807 027136 004537 010010      JSR    R5,TXCTRL    ;LOAD 2ND SYNCH,TX 1ST SYNCH
5808 027142 000001      TSOM
5809 027144 000005      5.
5810 027146 004537 010010      JSR    R5,TXCTRL    ;CLEAR TSOM
5811 027152 000000      000
5812 027154 000000      0
5813
5814 027156 012703 027436      MOV     #T24TBL,R3      ;SET UP DATA TABLE POINTER
5815 027162 112337 027172      MOVB   (R3)+,2$       ;INSTALL NEXT TX CHARACTER
5816
5817 027166 004537 007676      JSR    R5,TXCHAR     ;TRANSMIT CHARACTER ( ==> RX/FIFO )
5818 027172 000000      2$:    000           ;** HOLE FOR NEXT CHARACTER **
5819 027174 000006      6.
5820 027176 103003      BCC    .+8.        ;BR IF NO ERROR
5821 027200      ERROR           ;REPORT STACKED ERROR                TRAP    C$ERROR
5822 027202      ESCAPE  SUB      ;SKIP TO END OF TEST                TRAP    C$ESCAPE
5823 027204 104410      .WORD           TRAP    C$ESCAPE
5824 027206 000056

```

TEST 20 -- BCF CHARACTER LENGTH TEST

```

5821 027206 022703 027450          CMP      #T24TBL+10.,R3 ;ALL CHARACTERS TRANSMITTED ?
5822 027212 001363          BNE      20$           ; IF NOT, TX ANOTHER ONE
5823                               ;-----
5824 027214 012703 027436          MOV      #T24TBL,R3   ;SET UP DATA TABLE POINTER
5825 027220 112337 027236          50$:    MOVB   (R3)+,5$  ;INSTALL NEXT EXPECTED RX CHARACTER
5826 027224 142737 000300 027236  BICB   #300,5$      ;MASK OUT UNTRANSMITTED BITS
5827
5828 027232 004537 010110          JSR      R5,RXCHAR   ;READ/CHECK NEXT CHARACTER
5829 027236 000000          5$:    000           ;** HOLE FOR NEXT EXPECTED CHARACTER
5830 027240 000000          0
5831 027242 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
5832 027244 103003          BCC     .+8.        ;BR IF NO ERROR
5833 027246          ERROR          ;REPORT STACKED ERROR
5834 027250          ESCAPE  SUB          ;SKIP TO END OF TEST                TRAP    C$ERROR
5835 027252 000010          .WORD   L10055-.
5836 027254 022703 027445          CMP      #T24TBL+7,R3 ;ALL CHARACTERS CHECKED ?
5837 027260 001357          BNE      50$           ; IF NOT, CHECK ANOTHER ONE
5838
5839 027262          ENDSUB
5840 027262 104403          L10055:          TRAP    C$ESUB
5841
5842          ;-----
5843          ; SUBROUTINE # 3: 7 BIT CHARACTERS
5844          ;-----
5845          BGNSUB
5846          T20.3:          TRAP    C$BSUB
5847          JSR      R5,INITRN   ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5848          DDCMP!NOCHK!SYNCH ;SET CHAR MODE,NO ERROR CHECKING,S/AR=226
5849          TXDL!RXDL          ;TXCL=RXCL=7 BITS
5850          BCC     .+8.        ;BR IF NO ERROR
5851          ER.ROR          ;REPORT STACKED ERROR                TRAP    C$ERROR
5852          ESCAPE  SUB          ;SKIP TO END OF TEST                TRAP    C$ESCAPE
5853          .WORD   L10056-.
5854
5855          JSR      R5,TXCTRL   ;LOAD 2ND SYNCH,TX 1ST SYNCH
5856          TSOM
5857          6.
5858          JSR      R5,TXCTRL   ;CLEAR TSOM
5859          000
5860          0
5861          ;-----
5862          30$:    MOV      #T24TBL,R3   ;SET UP DATA TABLE POINTER
5863          MOVB   (R3)+,3$  ;INSTALL NEXT TX CHARACTER
5864
5865          3$:    JSR      R5,TXCHAR   ;TRANSMIT CHARACTER ( ==> RX/FIFO )
5866          000           ;** HOLE FOR NEXT CHARACTER **
5867          7.
5868          BCC     .+8.        ;BR IF NO ERROR
5869          ERROR          ;REPORT STACKED ERROR                TRAP    C$ERROR
5870          ESCAPE  SUB          ;SKIP TO END OF TEST

```


TEST 20 -- BCP CHARACTER LENGTH TEST

```

027352 104410
027354 000056
5867
5868 027356 022703 027447          CMP      #T24TBL+9.,R3      ;ALL CHARACTERS TRANSMITTED ?
5869 027362 001363          BNE      30$              ; IF NOT, TX ANOTHER ONE
5870          ;-----
5871 027364 012703 027436          MOV      #T24TBL,R3      ;SET UP DATA TABLE POINTER
5872 027370 112337 027406          MOVB    (R3)+,6$        ;INSTALL NEXT EXPECTED RX CHARACTER
5873 027374 142737 000200 027406    BICB    #200,6$        ;MASK OUT UNTRANSMITTED BITS
5874
5875 027402 004537 010110          JSR      R5,RXCHAR      ;READ/CHECK NEXT CHARACTER
5876 027406 000000          6$:    000              ;** HOLE FOR NEXT EXPECTED CHARACTER
5877 027410 000000          0
5878 027412 100000          NOCRDA                    ;NO INITIAL CHECK OF RDA=0
5879 027414 103003          BCC     .+8.             ;BR IF NO ERROR
5880 027416          ERROR                    ;REPORT STACKED ERROR
5881 027420          ESCAPE SUB              ;SKIP TO END OF TEST
5882 027420 104410          TRAP   C$ESCAPE
5883 027422 000010          .WORD  L10056-.
5884 027424 022703 027445          CMP      #T24TBL+7,R3   ;ALL CHARACTERS CHECKED ?
5885 027430 001357          BNE      60$           ; IF NOT, CHECK ANOTHER ONE
5886 027432          ENDSUB
5887 027432 104403          L10056: TRAP   C$ESUB
5888 027434          ENDTST
5889 027434 104401          L10053: TRAP   C$ETST
5889 027436          111
5890 027437          222
5891 027440          333
5892 027441          044
5893 027442          155
5894 027443          266
5895 027444          377
5896 027445          000
5897 027446          000
5898 027447          000
5899
5900
;-----
T24TBL: .BYTE 111 ;D1
        .BYTE 222 ;D2
        .BYTE 333 ;D3
        .BYTE 044 ;D4
        .BYTE 155 ;D5
        .BYTE 266 ;D6
        .BYTE 377 ;D7
        .BYTE 000 ;FILLER 1
        .BYTE 000 ;FILLER 2
        .BYTE 000 ;FILLER 3
        .EVEN
;-----

```

TEST 21 -- BOP TX TABORT/(IDLE = 0) TEST

5915

.SBTTL TEST 21 -- BOP TX TABORT/(IDLE = 0) TEST

```

*****
;*
;*      TEST 21 -- BOP TX TABORT/(IDLE = 0) TEST
;*
;* THE USYRT IS INITIALIZED FOR "BIT-ORIENTED PROTOCOL" (BOP) WITH IDLE
;* SET TO ZERO. TXE AND TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE
;* OBSERVING TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
;* NEXT, TXBE SHOULD GO HIGH; AT THIS TIME AN ALL ZEROS CHARACTER WILL BE
;* LOADED INTO TXOB DRIVING TXBE LOW. THE TRANSMITTER IS CLOCKED THROUGH
;* ONE CHARACTER. WHEN TXBE GOES HIGH AGAIN, TABORT IS ASSERTED CAUSING
;* ABORT TO BE TRANSMITTED. ALL CHARACTERS ARE CHECKED AT TXSO.
;* THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY TWO
;* FLAGS, ONE ZERO CHARACTER, AND ONE ABORT CHARACTER IS SENT (INTO THE BIT
;* BUCKET). ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN
;* THE SEQUENCE.
;*
*****

```

```

;
;      BGNTST
;
5916 027450 004737 005420      JSR      PC,INIDMV      ;INIT VIA      T21::
5917
5918 027454 004537 007400      JSR      R5,INITRN     ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5919 027460 003400              NOCHK                ;BOP MODE, NO ERROR CHECK
5920 027462 040000              NORXEN               ;NO RECEIVER ENABLE,USE 8 BIT CHARS
5921 027464 103003              BCC      .+8.         ;BR IF NO ERROR
5922 027466 104460              ERROR                ;REPORT STACKED ERROR
5923 027470 104410              ESCAPE  TST          ;SKIP TO END OF TEST      TRAP      C$ERROR
5924 027472 000150              .WORD      L10057-.   TRAP      C$ESCAPE
5925 027474 004537 007116      JSR      R5,CHKTSO     ;CHECK 1ST BIT OF EXPECTED "FLAG"
5926 027500 000000              0                    ; CHARACTER (SHOULD BE 0)
5927 027502 103003              BCC      .+8.         ;BR IF NO ERROR
5928 027504 104460              ERROR                ;REPORT STACKED ERROR      TRAP      C$ERROR
5929 027506 104410              ESCAPE  TST          ;AND EXIT TEST          TRAP      C$ESCAPE
5930 027510 000132              .WORD      L10057-.   .WORD
5931 027512 004537 007256      JSR      R5,SERIAL     ;READ REMAINING 7 BITS OF "FLAG" CHARACTER
5932 027516 000007              7.                   ; (OFF OF TSO BIT)
5933 027520 000176              176                  ; EXPECTED BIT SEQUENCE (1111110)
5934 027522 103003              BCC      .+8.         ;BR IF NO ERROR
5935 027524 104460              ERROR                ;REPORT STACKED ERROR      TRAP      C$ERROR
5936 027526 104410              ESCAPE  TST          ;AND EXIT TEST          TRAP      C$ESCAPE
5937 027530 000112              .WORD      L10057-.   .WORD
5938 027532 004537 010010      JSR      R5,TXCTRL     ;CLEAR TSOM
5939 027536 000000              000
5940 027540 000000              0
5941

```

TEST 21 -- BOP TX TABORT/(IDLE = 0) TEST

5942	027542	004537	003734	JSR	R5,WRITEI	;LOAD 000 CHARACTER		
5943	027546	120402		TDSRL				
5944	027550	000000		000				
5945								
5946	027552	004537	007256	JSR	R5,SERIAL	;READ FLAG CHARACTER VIA TSO		
5947	027556	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
5948	027560	000176		176		; EXPECTED BIT SEQUENCE (01111110)		
5949	027562	103003		BCC	+.8.	;BR IF NO ERROR		
5950	027564			ERROR		;REPORT STACKED ERROR		
	027564	104460					TRAP	C\$ERROR
5951	027566			ESCAPE	TST	;AND EXIT TEST		
	027566	104410					TRAP	C\$ESCAPE
	027570	000052					.WORD	L10057-
5952								
5953	027572	004537	010010	JSR	R5,TXCTRL	;SET TXABT BIT		
5954	027576	000004		TAB				
5955	027600	000000		0				
5956								
5957	027602	004537	007256	JSR	R5,SERIAL	;READ 000 CHARACTER VIA TSO		
5958	027606	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
5959	027610	000000		000		; EXPECTED BIT SEQUENCE (00000000)		
5960	027612	103003		BCC	+.8.	;BR IF NO ERROR		
5961	027614			ERROR		;REPORT STACKED ERROR		
	027614	104460					TRAP	C\$ERROR
5962	027616			ESCAPE	TST	;AND EXIT TEST		
	027616	104410					TRAP	C\$ESCAPE
	027620	000022					.WORD	L10057-
5963								
5964	027622	004537	007256	JSR	R5,SERIAL	;READ ABORT CHARACTER VIA TSO		
5965	027626	000007		7		; 7 BIT CHAR/CLOCK TICKS		
5966	027630	000177		177		; EXPECTED BIT SEQUENCE (1111111)		
5967	027632	103003		BCC	+.8.	;BR IF NO ERROR		
5968	027634			ERROR		;REPORT STACKED ERROR		
	027634	104460					TRAP	C\$ERROR
5969	027636			ESCAPE	TST	;AND EXIT TEST		
	027636	104410					TRAP	C\$ESCAPE
	027640	000002					.WORD	L10057-
5970	027642			ENDTST				
	027642						L10057:	
	027642	104401					TRAP	C\$ETST

TEST 22 -- BOP TX TABORT/(IDLE = 1) TEST

5985

.SBTTL TEST 22 -- BOP TX TABORT/(IDLE = 1) TEST

```

;*****
;
; TEST 22 -- BOP TX TABORT/(IDLE = 1) TEST
;
; THE USYRT IS INITIALIZED FOR "BIT-ORIENTED PROTOCOL" (BOP) WITH IDLE
; SET TO ONE. TXE AND TSOM IS ASSERTED AND TXC IS MANUALLY STEPPED WHILE
; OBSERVING TXA -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
; NEXT, TXBE SHOULD GO HIGH; AT THIS TIME AN ALL ZEROS CHARACTER WILL BE
; LOADED INTO TXDB DRIVING TXBE LOW. THE TRANSMITTER IS CLOCKED THROUGH
; ONE CHARACTER. WHEN TXBE GOES HIGH AGAIN, TABORT IS ASSERTED CAUSING
; ABORT TO BE TRANSMITTED. ALL CHARACTERS ARE CHECKED AT TXSO.
; THIS TEST WILL GO NO FURTHER INTO THE TRANSMIT SEQUENCE SO THAT ONLY TWO
; FLAGS, ONE ZERO CHARACTER, AND ONE FLAG CHARACTER IS SENT (INTO THE BIT
; BUCKET). ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN
; THE SEQUENCE.
;*****

```

```

;
; BGNTST
;
; T22::
5986 027644 004737 005420 JSR PC,INIDMV ;INIT VIA
5987
5988 027650 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5989 027654 007400 NOCHK!IDLES ;BOP MODE, NO ERROR CHECK, IDLE=1
5990 027656 040000 NORXEN ;NO RECEIVER ENABLE,USE 8 BIT CHARS
5991 027660 103003 BCC .+8. ;BR IF NO ERROR
5992 027662 ERROR ;REPORT STACKED ERROR
027662 104460
5993 027664 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
027664 104410 TRAP C$ESCAPE
027666 000150 .WORD L10060-.
5994
5995 027670 004537 007116 JSR R5,CHKTSO ;CHECK 1ST BIT OF EXPECTED "FLAG"
5996 027674 000000 0 CHARACTER (SHOULD BE 0)
5997 027676 103003 BCC .+8. ;BR IF NO ERROR
5998 027700 ERROR ;REPORT STACKED ERROR
027700 104460 TRAP C$ERROR
5999 027702 ESCAPE TST ;AND EXIT TEST TRAP C$ESCAPE
027702 104410 TRAP C$ESCAPE
027704 000132 .WORD L10060-.
6000
6001 027706 004537 007256 JSR R5,SERIAL ;READ REMAINING 7 BITS OF "FLAG" CHARACTER
6002 027712 000007 7. ; (OFF OF TSO BIT)
6003 027714 000176 176 ; EXPECTED BIT SEQUENCE (1111110)
6004 027716 103003 BCC .+8. ;BR IF NO ERROR
6005 027720 ERROR ;REPORT STACKED ERROR
027720 104460 TRAP C$ERROR
6006 027722 ESCAPE TST ;AND EXIT TEST TRAP C$ESCAPE
027722 104410 TRAP C$ESCAPE
027724 000112 .WORD L10060-.
6007
6008 027726 004537 010010 JSR R5,TXCTRL ;CLEAR TSOM
6009 027732 000000 000
6010 027734 000000 0
6011

```

TEST 22 - ROP TX TABORT/(IDLE - 1) TEST

6012	027736	004537	003734	JSR	R5,WRITEI	;LOAD 000 CHARACTER		
6013	027742	120402		TDSRL				
6014	027744	000000		000				
6015								
6016	027746	004537	007256	JSR	R5,SERIAL	;READ FLAG CHARACTER VIA TSO		
6017	027752	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
6018	027754	000176		176		; EXPECTED BIT SEQUENCE (01111110)		
6019	027756	103003		BCC	.+8.	;BR IF NO ERROR		
6020	027760			ERROR		;REPORT STACKED ERROR		
		104460					TRAP	C!ERROR
6021	027762			ESCAPE	TST	;AND EXIT TEST		
	027762	104410					TRAP	C!ESCAPE
	027764	000052					.WORD	L10060-
6022								
6023	027766	004537	010010	JSR	R5,TXCTRL	;SET TXABT BIT		
6024	027772	000004		TAB				
6025	027774	000000		0				
6026								
6027	027776	004537	007256	JSR	R5,SERIAL	;READ 000 CHARACTER VIA TSO		
6028	030002	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
6029	030004	000000		000		; EXPECTED BIT SEQUENCE (00000000)		
6030	030006	103003		BCC	.+8.	;BR IF NO ERROR		
6031	030010			ERROR		;REPORT STACKED ERROR		
		104460					TRAP	C!ERROR
6032	030012			ESCAPE	TST	;AND EXIT TEST		
	030012	104410					TRAP	C!ESCAPE
	030014	000022					.WORD	L10060-
6033								
6034	030016	004537	007256	JSR	R5,SERIAL	;READ ABORT (FLAG) CHARACTER VIA TSO		
6035	030022	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
6036	030024	000176		176		; EXPECTED BIT SEQUENCE (01111110)		
6037	030026	103003		BCC	.+8.	;BR IF NO ERROR		
6038	030030			ERROR		;REPORT STACKED ERROR		
		104460					TRAP	C!ERROR
6039	030032			ESCAPE	TST	;AND EXIT TEST		
	030032	104410					TRAP	C!ESCAPE
	030034	000002					.WORD	L10060-
6040	030036							
	030036							
	030036	104401					L10060;	TRAP
								C!TST

ENDTST

TEST 23 - BOP TX TXGA (TRANSMIT GO AHEAD) TEST

6055

.SBTTL TEST 23 - BOP TX TXGA (TRANSMIT GO AHEAD) TEST

```

;*****
;*
;* TEST 23 - BOP TX TXGA (TRANSMIT GO AHEAD) TEST
;*
;* THE USYRT IS INITIALIZED FOR BOP AND TXE IS ASSERTED. TSOM IS ASSERTED
;* AND TXA OBSERVED -- IT SHOULD BE ASSERTED WITHIN TWO (2) CLOCK CYCLES.
;* NEXT, TXBE SHOULD GO HIGH; AT THIS TIME AN ALL ZEROS CHARACTER WILL BE
;* LOADED INTO TXDB DRIVING TXBE LOW. WHEN TXBE GOES HIGH AGAIN, TXGA
;* IS ASSERTED CAUSING GA TO BE TRANSMITTED. THE SEQUENCE OF EVENTS IS
;* CONTINUALLY MONITORED WHILE TXC IS MANUALLY CONTROLLED AND ALL
;* CHARACTERS ARE CHECKED AT TXSO. THIS TEST WILL GO NO FURTHER INTO
;* THE TRANSMIT SEQUENCE SO THAT ONLY TWO FLAGS, ONE ZERO CHARACTER
;* AND ONE GA CHARACTER IS SENT (INTO THE BIT BUCKET).
;* ERROR LOOPING WILL DEPEND ON WHERE THE FIRST ERROR OCCURS WITHIN
;* THE SEQUENCE.
;*
;*****

```

```

;
;          BGNST
;
6056 030040 004737 005420      JSR      PC,INIDMV      ;INIT VIA          T23::
6057
6058 030044 004537 007400      JSR      R5,INITRN     ;LOAD 1 SUM, CLK TX UNTIL ACTIVE
6059 030050 003400              NOCHK                ;BOP MODE, NO ERROR CHECK
6060 030052 040000              NCRXEN              ;NO RECEIVER ENABLE, USE 8 BIT CHARS
6061 030054 103003              BCC      ,+8.        ;BR IF NO ERROR
6062 030056 104460              ERROR             ;REPORT STACKED ERROR
6063 030060 104410              ESCAPE TST          ;SKIP TO END OF TEST          TRAP      C$ERROR
6064 030062 000150              .WORD      L10061-.
6065 030064 004537 007116      JSR      R5,CHKTSO     ;CHECK 1ST BIT OF EXPECTED "FLAG"
6066 030070 000000              0                    ; CHARACTER (SHOULD BE 0)
6067 030072 103003              BCC      ,+8.        ;BR IF NO ERROR
6068 030074 104460              ERROR             ;REPORT STACKED ERROR          TRAP      C$ERROR
6069 030076 104410              ESCAPE TST          ;AND EXIT TEST                TRAP      C$ESCAPE
6070 030100 000132              .WORD      L10061-.
6071 030102 004537 007256      JSR      R5,SERIAL     ;READ REMAINING 7 BITS OF "FLAG" CHARACTER
6072 030106 000007              7.                    ; (OFF OF TSO BIT)
6073 030110 000176              176                   ; EXPECTED BIT SEQUENCE (1111110)
6074 030112 103003              BCC      ,+8.        ;BR IF NO ERROR
6075 030114 104460              ERROR             ;REPORT STACKED ERROR          TRAP      C$ERROR
6076 030116 104410              ESCAPE TST          ;AND EXIT TEST                TRAP      C$ESCAPE
6077 030120 000112              .WORD      L10061-.
6078 030122 004537 010010      JSR      R5,TXCTRL    ;CLEAR TSOM
6079 030126 000000              000
6080 030130 000000              0
6081

```

TEST 23 -- BOP TX TXGA (TRANSMIT GO AHEAD) TEST

```

6082 030132 004537 003734 JSR R5,WRITEI ;LOAD 000 CHARACTER
6083 030136 120402 TDSRL
6084 030140 000000 000
6085
6086 030142 004537 007256 JSR R5,SERIAL ;READ "FLAG" CHARACTER VIA TSO
6087 030146 000010 8. ; 8 BIT CHAR/CLOCK TICKS
6088 030150 000176 176 ; EXPECTED BIT SEQUENCE (01111110)
6089 030152 103003 BCC ,+8. ;BR IF NO ERROR
6090 030154 104460 ERROR ;REPORT STACKED ERROR
6091 030156 ESCAPE TST ;AND EXIT TEST TRAP C$ERROR
6091 030156 104410 TRAP C$ESCAPE
6091 030160 000052 .WORD L10061-.
6092
6093 030162 004537 010010 JSR R5,TXCTRL ;SET TX GO-AHEAD AND TEOM
6094 030166 000012 TGA!TEOM
6095 030170 000000 0
6096
6097 030172 004537 007256 JSR R5,SERIAL ;READ 000 CHARACTER VIA TSO
6098 030176 000010 8. ; 8 BIT CHAR/CLOCK TICKS
6099 030200 000000 000 ; EXPECTED BIT SEQUENCE (00000000)
6100 030202 103003 BCC ,+8. ;BR IF NO ERROR
6101 030204 104460 ERROR ;REPORT STACKED ERROR
6102 030206 ESCAPE TST ;AND EXIT TEST TRAP C$ERROR
6102 030206 104410 TRAP C$ESCAPE
6102 030210 000022 .WORD L10061-.
6103
6104 030212 004537 007256 JSR R5,SERIAL ;READ GO-AHEAD CHARACTER VIA TSO
6105 030216 000010 8. ; 8 BIT CHAR/CLOCK TICKS
6106 030220 000177 177 ; EXPECTED BIT SEQUENCE (01111111)
6107 030222 103003 BCC ,+8. ;BR IF NO ERROR
6108 030224 104460 ERROR ;REPORT STACKED ERROR
6109 030226 ESCAPE TST ;AND EXIT TEST TRAP C$ERROR
6109 030226 104410 TRAP C$ESCAPE
6109 030230 000002 .WORD L10061-.
6110 030232 ENDTST
6110 030232 104401 L10061: TRAP C$ETST

```

TEST 24 -- BOP TX MESSAGE WITHOUT CRC

6122

.SBTTL TEST 24 -- BOP TX MESSAGE WITHOUT CRC

```

*****
;*
;* TEST 24 -- BOP TX MESSAGE WITHOUT CRC
;*
;* THE USYRT IS INITIALIZED FOR BOP MODE WITH NO ERROR DETECTION. TXC IS THEN
;* MANUALLY CONTROLLED UNTIL TWO FLAG CHARACTERS HAVE BEEN TRANSMITTED. THEN
;* A 5 CHARACTER MESSAGE IS TRANSMITTED, RECEIVED, CHECKED, AND TERMINATED
;* (WITH TEOM). A CHECK IS MADE TO ASCERTAIN THAT NO CRC OR VRC IS GENERATED
;* -- FLAG CHARACTERS SHOULD FOLLOW THE DATA.
;* (NOTE: NO BIT STUFFING OCCURS IN THIS TEST)
;*
;* TEST MESSAGE: FLAG FLAG 000 307 125 252 201 FLAG
;*
*****

```

```

;
; BGNTST
;
; T24::
6123 030234 004737 005420 JSR PC,INIDMV ;INIT DMV-11. ENTER M-LOOP
6124 030234 004737 005420
6125 030240 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
6126 030244 003400 NOCHK ;SET BOP MODE,NO CHECK
6127 030246 000000 0 ;USE 8 BIT CHARS
6128 030250 103003 BCC .+8. ;BR IF NO ERROR
6129 030252 104460 ERROR ;REPORT STACKED ERROR
6130 030254 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
6131 030254 104410
6132 030256 000332 TRAP C$ESCAPE
; .WORD L10062-.
6131
6132 030260 004537 010010 JSR R5,TXCTRL ;LOAD 2ND FLAG, TX 1ST FLAG
6133 030264 000001 TSOM
6134 030266 000007 7.
6135
6136 030270 004537 010010 JSR R5,TXCTRL ;CLEAR TSOM
6137 030274 000000 000
6138 030276 000000 0
6139
6140 030300 004537 007676 JSR R5,TXCHAR ;LOAD 000, TX 2ND FLAG
6141 030304 000000 000
6142 030306 000010 8.
6143 030310 103003 BCC .+8. ;BR IF NO ERROR
6144 030312 104460 ERROR ;REPORT STACKED ERROR
6145 030314 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
6146 030314 104410 TRAP C$ESCAPE
6147 030316 000272 .WORD L10062-.
6146
6147 030320 004537 007676 JSR R5,TXCHAR ;LOAD 307, TX 000
6148 030324 000307 307
6149 030326 000010 8.
6150 030330 103003 BCC .+8. ;BR IF NO ERROR
6151 030332 104460 ERROR ;REPORT STACKED ERROR
6152 030334 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
; TRAP C$ESCAPE

```


TEST 24 -- BOP TX MESSAGE WITHOUT CRC

6153	030336	000252					.WORD	L10062-.
6154	030340	004537	007676	JSR	R5, TXCHAR	;LOAD 125; TX 307		
6155	030344	000125						
6156	030346	000010						
6157	030350	103003						
6158	030352	104460						
6159	030354	104410		ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030354	104410					TRAP	C\$ESCAPE
	030356	000232					.WORD	L10062-.
6160								
6161	030360	004537	007676	JSR	R5, TXCHAR	;LOAD 252		
6162	030364	000252						
6163	030366	000000						
6164	030370	103003						
6165	030372	104460						
6166	030374	104410		ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030374	104410					TRAP	C\$ESCAPE
	030376	000212					.WORD	L10062-.
6167								
6168	030400	004537	011364	JSR	R5, RCV1ST	;CLOCK AND RCV 000		
6169	030404	000000						
6170	030406	103003						
6171	030410	104460						
6172	030412	104410		ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030412	104410					TRAP	C\$ESCAPE
	030414	000174					.WORD	L10062-.
6173								
6174	030416	004537	010110	JSR	R5, RXCHAR	;READ & CHK 000, RCV 307		
6175	030422	000400				; & CHECK RSOM=1		
6176	030424	000000						
6177	030426	000010						
6178	030430	103003						
6179	030432	104460						
6180	030434	104410		ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030434	104410					TRAP	C\$ESCAPE
	030436	000152					.WORD	L10062-.
6181								
6182	030440	004537	007676	JSR	R5, TXCHAR	;LOAD 201		
6183	030444	000201						
6184	030446	000000						
6185	030450	103003						
6186	030452	104460						
6187	030454	104410		ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030454	104410					TRAP	C\$ESCAPE
	030456	000132					.WORD	L10062-.
6188								
6189	030460	004537	010110	JSR	R5, RXCHAR	;READ & CHK 307, RCV 125		
6190	030464	000307						
6191	030466	000000						
6192	030470	000010						
6193	030472	103003		BCC	.+8.	;BR IF NO ERROR		

TEST 25 -- BOP RX CHARACTER LENGTH TEST

6249

.SBTTL TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

*****
;*
;* TEST 25 -- BOP RX CHARACTER LENGTH TEST
;*
;* THE USYRT IS INITIALIZED FOR BOP WITH CRC-CCITT PRESET TO 1'S. TXC
;* IS MANUALLY CONTROLLED UNTIL TWO FLAG CHARACTERS HAVE BEEN
;* TRANSMITTED. THEN 6 SUBTESTS FOLLOW, EACH ONE USING A DIFFERENT
;* TRANSMIT CHARACTER LENGTH STARTING AT TWO (2) AND ENDING WITH SEVEN
;* (7). IN EACH SUBTEST, TWO 8 BIT CHARACTERS WILL BE TRANSMITTED
;* BEFORE TXCL IS CHANGED TO THE CHARACTER LENGTH BEING TESTED. THIS
;* CORRESPONDS TO NORMAL USAGE WHERE EITHER:
;*
;* 1 -- A MESSAGE OF CHARACTERS WHICH ARE LESS THEN 8 BITS IS
;* SENT AS A STREAM OF 8 BIT CHARACTERS AND THE REMAINING
;* BITS ARE SENT AS A CHARACTER OF LESS THEN 8 BITS OR
;*
;* 2 -- A HEADER OF TWO 8 BIT CHARACTERS IS SENT FOLLOWED BY A
;* DATA STREAM OF DATA CHARACTERS WHICH MAY BE LESS THEN 8
;* BITS IN LENGTH (I.E. 2, 3, 4, 5, 6, OR 7 BIT
;* CHARACTERS).
;*
;* THE TEST PATTERN IS: 123 321 111 222 333 044 155 266 377
;*
```

```

030612
6250 030612 004737 005420
6251
6252
6253
6254 030616
030616
030616 104402
6255 030620 004537 007400
6256 030624 004226
6257 030626 000002
6258 030630 103003
6259 030632
030632 104460
6260 030634
030634 104410
030636 000344
6261
6262 030640 004537 010010
6263 030644 000001
6264 030646 000007
6265 030650 004537 010010
6266 030654 000000
6267 030656 000000
6268 030660 004537 007676
6269 030664 000123
6270 030666 000010
6271 030670 103003
6272 030672
```

```

;
; BGNTST
;
; JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T25:
;-----
; SUBROUTINE # 1: 2 BIT CHARACTERS
;-----
; BGNSUB
;
; JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE TRAP C$BSUB
; IDLES!SYNCH ;SET BOP MODE,CRC-CCITT-1,S/AR*226
; BIT1 ;INITIALLY: TXCL*8 BITS / RXCL*2 BITS
; BCC .+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
;
; ESCAPE SUB ;SKIP TO END OF TEST TRAP C$ERROR
;
; JSR R5,TXCTRL ;LOAD 2ND FLAG,TX 1ST FLAG TRAP C$ESCAPE
; TSOM ;.WORD L10064-.
; 7.
; JSR R5,TXCTRL ;CLEAR TSOM
; 000
; 0
; JSR R5,TXCHAR ;LOAD 123(HEADER1), TX 2ND FLAG
; 123
; 8.
; BCC .+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
```


TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6317 031042          ESCAPE SUB          ;SKIP TO END OF TEST
      031042 104410
      031044 000136          TRAP C$ESCAPE
                                .WORD L10064-.
6318
6319 031046 004537 010110 JSR R5,RXCHAR ;READ/CHECK 321(HEADER2),RCV 001(DATA1)
6320 031052 000321      321
6321 031054 000000      0
6322 031056 100000      NOCRDA          ;NO INITIAL CHECK OF RDA=0
6323 031060 103003      BCC .+8.        ;BR IF NO ERROR
6324 031062          ERROR          ;REPORT STACKED ERROR
      031062 104460          TRAP C$ERROR
6325 031064          ESCAPE SUB          ;SKIP TO END OF TEST
      031064 104410          TRAP C$ESCAPE
      031066 000114          .WORD L10064-.
6326 ;-----
6327 031070 012703 033524 ;-----
6328 031074 112337 031112 20$: MOV #T30TBL,R3 ;SET UP DATA TABLE POINTER
6329 031100 142737 000374 031112 MOVB (R3)+,2$ ;INSTALL NEXT EXPECTED RX CHARACTER
      0330 BICB #374,2$ ;MASK OUT UNTRANSMITTED BITS
6331 031106 004537 010110 2$: JSP R5,RXCHAR ;READ/CHECK NEXT CHARACTER
6332 031112 000000      000 ;** HOLE FOR NEXT EXPECTED CHARACTER
6333 031114 000000      0
6334 031116 100000      NOCRDA          ;NO INITIAL CHECK OF RDA=0
6335 031120 103003      BCC .+8.        ;BR IF NO ERROR
6336 031122          ERROR          ;REPORT STACKED ERROR
      031122 1 4460          TRAP C$ERROR
6337 031124          ESCAPE SUB          ;SKIP TO END OF TEST
      031124 104410          TRAP C$ESCAPE
      031126 000054          .WORD L10064-.
6338
6339 031130 022703 033531      CMP #T30TBL+5,R3 ;ALL CHARACTERS CHECKED ?
6340 031134 001357      BNE 20$ ; IF NOT, CHECK ANOTHER ONE
6341 ;-----
6342 031136 004537 010110 JSR R5,RXCHAR ;READ/CHK DATA6(002), RCV DATA7
6343 031142 000002      002
6344 031144 000000      0
6345 031146 120000      NOCRDA!NOCRDA ;NO INITIAL CHECK OF RDA=0
6346 031150 103003      BCC .+8.        ;DON'T CHECK FOR FINAL RXACT=1
6347 031152          ERROR          ;BR IF NO ERROR
      031152 104460          ;REPORT STACKED ERROR
6348 031154          ESCAPE SUB          ;SKIP TO END OF TEST
      031154 104410          TRAP C$ERROR
      031156 000024          .WORD L10064-.
6349
6350 031160 004537 010110 JSR R5,RXCHAR ;READ/CHK DATA7(003), RCV FIRST FLAG
6351 031164 001003      RXEOM!003 ;& CHECK RERR BIT=0 (GOOD CRC) & REOM=1
6352 031166 000001      RERCHK ;NO INITIAL CHECK OF RDA=0
6353 031170 160000      NOCRDA!NOCRDA!NOCRDA ;DON'T CHECK FOR FINAL RDA=RXACT=1
6354 031172 103003      BCC .+8.        ;BR IF NO ERROR
6355 031174          ERROR          ;REPORT STACKED ERROR
      031174 104460          TRAP C$ERROR
6356 031176          ESCAPE SUB          ;SKIP TO END OF TEST
      031176 104410          TRAP C$ESCAPE
      031200 000002          .WORD L10064-.
6357 031202          ENDSUB
      031202          L10064: TRAP C$ESUB
      031202 104403

```

TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6358
6359
6360
6361 031204
      031204
      031204 104402
6362 031206 004537 007400
6363 031212 000226
6364 031214 000003
6365 031216 103003
6366 031220
      031220 104460
6367 031222
      031222 104410
      031224 000344
6368
6369 031226 004537 010010
6370 031232 000001
6371 031234 000007
6372 031236 004537 010010
6373 031242 000000
6374 031244 000000
6375 031246 004537 007676
6376 031252 000123
6377 031254 000010
6378 031256 103003
6379 031260
      031260 104460
6380 031262
      031262 104410
      031264 000304
6381
6382 031266 004537 007676
6383 031272 000321
6384 031274 000010
6385 031276 103003
6386 031300
      031300 104460
6387 031302
      031302 104410
      031304 000264
6388
6389 031306 004537 003734
6390 031312 120407
6391 031314 000143
6392 031316 004537 007676
6393 031322 000111
6394 031324 000010
6395 031326 103003
6396 031330
      031330 104460
6397 031332
      031332 104410
      031334 000234
6398
6399 031336 012703 033525
6400 031342 112337 031352

```

```

-----
; SUBROUTINE # 2: 3 BIT CHARACTERS
-----
      BGNSUB
      T25.2:
      TRAP C$BSUB
      JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
      SYNCH ;SET BOP MODE,CRC-CCITT-1,S/AR=226
      BIT1!BIT0 ;INITIALLY: TXCL=8 BITS / RXCL=3 BITS
      BCC .+8. ;BR IF NO ERROR
      ERROR ;REPORT STACKED ERROR
      TRAP C$ERROR
      ESCAPE SUB ;SKIP TO END OF TEST
      TRAP C$ESCAPE
      .WORD L10065-.
      JSR R5,TXCTRL ;LOAD 2ND FLAG, TX 1ST FLAG
      TSOM
      7.
      JSR R5,TXCTRL ;CLEAR TSOM
      000
      0
      JSR R5,TXCHAR ;LOAD 123(HEADER1), TX 2ND FLAG
      123
      8.
      BCC .+8. ;BR IF NO ERROR
      ERROR ;REPORT STACKED ERROR
      TRAP C$ERROR
      ESCAPE SUB ;SKIP TO END OF TEST
      TRAP C$ESCAPE
      .WORD L10065-.
      JSR R5,TXCHAR ;LOAD 321(HEADER2), TX 123(HEADER1)
      321
      8.
      BCC .+8. ;BR IF NO ERROR
      ERROR ;REPORT STACKED ERROR
      TRAP C$ERROR
      ESCAPE SUB ;SKIP TO END OF TEST
      TRAP C$ESCAPE
      .WORD L10065-.
      JSR R5,WRITEI ;NOW CHANGE TXCL TO 3 BITS TOO
      PCR
      BIT6!BITS!BIT1!BIT0
      JSR R5,TXCHAR ;LOAD 111(DATA1), TX 321(HEADER2)
      111
      8.
      BCC .+8. ;BR IF NO ERROR
      ERROR ;REPORT STACKED ERROR
      TRAP C$ERROR
      ESCAPE SUB ;SKIP TO END OF TEST
      TRAP C$ESCAPE
      .WORD L10065-.
-----
10$: MOV #T30TBL+1,R3 ;SET UP DATA TABLE POINTER
      MOVB (R3)+,1$ ;INSTALL NEXT TX CHARACTER

```

TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6401
6402 031346 004537 007676          JSR      R5,TXCHAR      ;TRANSMIT CHARACTER ( ==> RX/FIFO )
6403 031352 000000          1$:     000            ;** HOLE FOR NEXT CHARACTER **
6404 031354 000003          3.
6405 031356 103003          BCC      .+8.          ;BR IF NO ERROR
6406 031360          ERROR          ;REPORT STACKED ERROR
        031360 104460          TRAP     C$ERROR
6407 031362          ESCAPE  SUB          ;SKIP TO END OF TEST
        031362 104410          TRAP     C$ESCAPE
        031364 000204          .WORD   L10065-.
6408
6409 031366 022703 033533          CMP      #T30TBL+7,R3  ;ALL CHARACTERS TRANSMITTED ?
6410 031372 001363          BNE     10$            ; IF NOT, TX ANOTHER ONE
6411          ;-----
6412 031374 004537 010010          JSR      R5,TXCTRL     ;SET TEOM
6413 031400 000002          TEOM
6414 031402 000000          0
6415 031404 004537 011614          JSR      R5,STEPLU     ; AND TX DATA7 + SOME FLAGS
6416 031410 000044          36.
6417
6418 031412 004537 010110          JSR      R5,RXCHAR     ;READ & CHK 123(HEADER1), RCV 321(HEADER2)
6419 031416 000523          RXSOM!123            ; & CHECK FOR RSOM*1
6420 031420 000000          0
6421 031422 100000          NOCRDA              ;NO INITIAL CHECK OF RDA=0
6422 031424 103003          BCC      .+8.          ;BR IF NO ERROR
6423 031426          ERROR          ;REPORT STACKED ERROR
        031426 104460          TRAP     C$ERROR
6424 031430          ESCAPE  SUB          ;SKIP TO END OF TEST
        031430 104410          TRAP     C$ESCAPE
        031432 000136          .WORD   L10065-.
6425
6426 031434 004537 010110          JSR      R5,RXCHAR     ;READ/CHECK 321(HEADER2),RCV 001(DATA1)
6427 031440 000321          321
6428 031442 000000          0
6429 031444 100000          NOCRDA              ;NO INITIAL CHECK OF RDA=0
6430 031446 103003          BCC      .+8.          ;BR IF NO ERROR
6431 031450          ERROR          ;REPORT STACKED ERROR
        031450 104460          TRAP     C$ERROR
6432 031452          ESCAPE  SUB          ;SKIP TO END OF TEST
        031452 104410          TRAP     C$ESCAPE
        031454 000114          .WORD   L10065-.
6433          ;-----
6434 031456 012703 033524          MOV      #T30TBL,R3   ;SET UP DATA TABLE POINTER
6435 031462 112337 031500          40$:   MOVB     (R3)+,4$   ;INSTALL NEXT EXPECTED RX CHARACTER
6436 031466 142737 000370 031500          BICB     #370,4$      ;MASK OUT UNTRANSMITTED BITS
6437
6438 031474 004537 010110          JSR      R5,RXCHAR     ;READ/CHECK NEXT CHARACTER
6439 031500 000000          4$:     000            ;** HOLE FOR NEXT EXPECTED CHARACTER
6440 031502 000000          0
6441 031504 100000          NOCRDA              ;NO INITIAL CHECK OF RDA=0
6442 031506 103003          BCC      .+8.          ;BR IF NO ERROR
6443 031510          ERROR          ;REPORT STACKED ERROR
        031510 104460          TRAP     C$ERROR
6444 031512          ESCAPE  SUB          ;SKIP TO END OF TEST
        031512 104410          TRAP     C$ESCAPE
        031514 000054          .WORD   L10065-.
6445

```

TEST 25 .. BOP RX CHARACTER LENGTH TEST

```

6446 031516 022703 033531      CMP      #T30TBL+5,R3      ;ALL CHARACTERS CHECKED ?
6447 031522 001357              BNE      40$              ; IF NOT, CHECK ANOTHER ONE
6448                               ;-----
6449 031524 004537 010110      JSR      R5,RXCHAR        ;READ/CHK DATA6(006), RCV DATA7
6450 031530 000006              006
6451 031532 000000              0
6452 031534 020000              NCRACT                    ;DON'T CHECK FOR FINAL RXACT=1
6453 031536 103003              BCC      ,+8.             ;BR IF NO ERROR
6454 031540 104460              ERROR                    ;REPORT STACKED ERROR
6455 031542 104410              ESCAPE  TST              ;SKIP TO END OF TEST          TRAP  C$ERROR
6456 031544 001756              .WORD                    ;SKIP TO END OF TEST          TRAP  C$ESCAPE
6457 031546 004537 010110      JSR      R5,RXCHAR        ;READ/CHK DATA7(007), RCV FIRST FLAG
6458 031552 001007              RXEOM!007                ;AND CHECK FOR REOM=1
6459 031554 000001              RERCHK                    ; & CHECK RERR BIT=0 (GOOD CRC)
6460 031556 060000              NCRDA!NCRACT              ;DON'T CHECK FOR FINAL RDA=RXACT=1
6461 031560 103003              BCC      ,+8.             ;BR IF NO ERROR
6462 031562 104460              ERROR                    ;REPORT STACKED ERROR
6463 031564 104410              ESCAPE  TST              ;SKIP TO END OF TEST          TRAP  C$ERROR
6464 031566 001734              .WORD                    ;SKIP TO END OF TEST          TRAP  C$ESCAPE
6464 031570 104403              ENDSUB                    ;SKIP TO END OF TEST          TRAP  C$ESUB
6465 031570 104403              ENDSUB                    ;SKIP TO END OF TEST          TRAP  C$ESUB
6466                               ;-----
6467                               ; SUBROUTINE # 3: 4 BIT CHARACTERS
6468                               ;-----
6468 031572 104402              BCNSUB                    ;SKIP TO END OF TEST          TRAP  C$BSUB
6469 031574 004537 007400      JSR      R5,INITRN        ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
6470 031600 000226              SYNCH                      ;SET BOP MODE,CRC-CCITT-1,S/AR=226
6471 031602 000004              BIT2                       ;INITIALLY: TXCL=8 BITS / RXCL=4 BITS
6472 031604 103003              BCC      ,+8.             ;BR IF NO ERROR
6473 031606 104460              ERROR                    ;REPORT STACKED ERROR
6474 031610 104410              ESCAPE  SUB              ;SKIP TO END OF TEST          TRAP  C$ERROR
6475 031612 000344              .WORD                    ;SKIP TO END OF TEST          TRAP  C$ESCAPE
6476 031614 004537 010010      JSR      R5,TXCTRL        ;LOAD 2ND FLAG, TX 1ST FLAG
6477 031620 000001              TSOM
6478 031622 000007              7.
6479 031624 004537 010010      JSR      R5,TXCTRL        ;CLEAR TSOM
6480 031630 000000              000
6481 031632 000000              0
6482 031634 004537 007676      JSR      R5,TXCHAR        ;LOAD 123(HEADER1), TX 2ND FLAG
6483 031640 000123              123
6484 031642 000010              8.
6485 031644 103003              BCC      ,+8.             ;BR IF NO ERROR
6486 031646 104460              ERROR                    ;REPORT STACKED ERROR
6487 031650 104410              ESCAPE  SUB              ;SKIP TO END OF TEST          TRAP  C$ERROR
6488 031650 104410              .WORD                    ;SKIP TO END OF TEST          TRAP  C$ESCAPE

```


TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

031652 000304                                     .WORD  L10066-.
6488
6489 031654 004537 007676      JSR      R5,TXCHAR      ;LOAD 321(HEADER2), TX 123(HEADER1)
6490 031660 000321              321
6491 031662 000010              8.
6492 031664 103003      BCC      .+8.          ;BR IF NO ERROR
6493 031666              ERROR          ;REPORT STACKED ERROR
031666 104460                                     TRAP   C$ERROR
6494 031670              ESCAPE  SUB          ;SKIP TO END OF TEST
031670 104410                                     TRAP   C$ESCAPE
031672 000264                                     .WORD  L10066-.
6495
6496 031674 004537 003734      JSR      R5,WRITEI     ;NOW CHANGE TXCL TO 4 BITS TOO
6497 031700 120407      PCR
6498 031702 000204      BIT7!BIT2
6499 031704 004537 007676      JSR      R5,TXCHAR      ;LOAD 111(DATA1), TX 321(HEADER2)
6500 031710 000111              111
6501 031712 000010              8.
6502 031714 103003      BCC      .+8.          ;BR IF NO ERROR
6503 031716              ERROR          ;REPORT STACKED ERROR
031716 104460                                     TRAP   C$ERROR
6504 031720              ESCAPE  SUB          ;SKIP TO END OF TEST
031720 104410                                     TRAP   C$ESCAPE
031722 000234                                     .WORD  L10066-.
6505
-----
6506 031724 012703 033525      MOV      @T30TBL+1,R3  ;SET UP DATA TABLE POINTER
6507 031730 112337 031740      10$:    MOVB      (R3)+,1$  ;INSTALL NEXT TX CHARACTER
6508
6509 031734 004537 007676      JSR      R5,TXCHAR      ;TRANSMIT CHARACTER ( --> RX/FIFO )
6510 031740 000000      1$:    000          ;** HOLE FOR NEXT CHARACTER **
6511 031742 000004              4.
6512 031744 103003      BCC      .+8.          ;BR IF NO ERROR
6513 031746              ERROR          ;REPORT STACKED ERROR
031746 104460                                     TRAP   C$ERROR
6514 031750              ESCAPE  SUB          ;SKIP TO END OF TEST
031750 104410                                     TRAP   C$ESCAPE
031752 000204                                     .WORD  L10066-.
6515
6516 031754 022703 033533      CMP      @T30TBL+7,R3  ;ALL CHARACTERS TRANSMITTED ?
6517 031760 001363      10$:    BNE      10$          ; IF NOT, TX ANOTHER ONE
6518
-----
6519 031762 004537 010010      JSR      R5,TXCTRL     ;SET TEOM
6520 031766 000002      TEOM
6521 031770 000000              0
6522 031772 004537 011614      JSR      R5,STEPLU     ; AND TX DATA7 + SOME FLAGS
6523 031776 000044              36.
6524
6525 032000 004537 010110      JSR      R5,RXCHAR     ;READ & CHK 123(HEADER1), RCV 321(HEADER2)
6526 032004 000523      RXSOM!123          ; & CHECK FOR RSOM=1
6527 032006 000000              0
6528 032010 100000      NCCRDA          ;NO INITIAL CHECK OF RDA=0
6529 032012 103003      BCC      .+8.          ;BR IF NO ERROR
6530 032014              ERROR          ;REPORT STACKED ERROR
032014 104460                                     TRAP   C$ERROR
6531 032016              ESCAPE  SUB          ;SKIP TO END OF TEST
032016 104410                                     TRAP   C$ESCAPE
032020 000136                                     .WORD  L10066-.

```

TEST 25 BOP RX CHARACTER LENGTH TEST

```

6532
6533 032022 004537 010110      JSR      R5,RXCHAR      ;READ/CHECK 521(HEADER2),RCV 011(DATA1)
6534 032026 000321              321
6535 032030 000000              0
6536 032032 100000      NOCRDA      ;NO INITIAL CHECK OF RDA=0
6537 032034 103003      BCC      .+8.      ;BR IF NO ERROR
6538 032036 104460      ERROR      ;REPORT STACKED ERROR
6539 032040 104460      ESCAPE SUB      ;SKIP TO END OF TEST      TRAP      C1ERROR
        032040 104410
        032042 000114      TRAP      C1ESCAPE
        .WORD      L10066-.
6540
6541 032044 012703 033524      MOV      #T30TBL,R3      ;SET UP DATA TABLE POINTER
6542 032050 112337 032066      401:  MOVB      (R3),41      ;INSTALL NEXT EXPECTED RX CHARACTER
6543 032054 142737 000360 032066      BICB      #360,41      ;MASK OUT UNTRANSMITTED BITS
6544
6545 032062 004537 010110      JSR      R5,RXCHAR      ;READ/CHECK NEXT CHARACTER
6546 032066 000000      41:  000      ;** HOLE FOR NEXT EXPECTED CHARACTER
6547 032070 000000              0
6548 032072 100000      NOCRDA      ;NO INITIAL CHECK OF RDA=0
6549 032074 103003      BCC      .+8.      ;BR IF NO ERROR
6550 032076 104460      ERROR      ;REPORT STACKED ERROR
6551 032100 104460      ESCAPE SUB      ;SKIP TO END OF TEST      TRAP      C1ERROR
        032100 104410      TRAP      C1ESCAPE
        032102 000054      .WORD      L10066-.
6552
6553 032104 022703 033531      CMP      #T30TBL+5,R3      ;ALL CHARACTERS CHECKED ?
6554 032110 001357      401:  BNE      401      ; IF NOT, CHECK ANOTHER ONE
6555
6556 032112 004537 010110      JSR      R5,RXCHAR      ;READ/CHK DATA6(006), RCV DATA7
6557 032116 000006      006
6558 032120 000000              0
6559 032122 020000      NCRACT      ;DON'T CHECK FOR FINAL RXACT=1
6560 032124 103003      BCC      .+8.      ;BR IF NO ERROR
6561 032126 104460      ERROR      ;REPORT STACKED ERROR
6562 032130 104460      ESCAPE TST      ;SKIP TO END OF TEST      TRAP      C1ERROR
        032130 104410      TRAP      C1ESCAPE
        032132 001370      .WORD      L10063-.
6563
6564 032134 004537 010110      JSR      R5,RXCHAR      ;READ/CHK DATA7(017), RCV FIRST FLAG
6565 032140 001017      RXEOM!017      ; AND CHECK FOR REOM=1
6566 032142 000001      RERCHK      ; & CHECK RERR BIT=0 (GOOD CRC)
6567 032144 060000      NOCRDA!NCRACT      ;DON'T CHECK FOR FINAL RDA=RXACT=1
6568 032146 103003      BCC      .+8.      ;BR IF NO ERROR
6569 032150 104460      ERROR      ;REPORT STACKED ERROR
6570 032152 104460      ESCAPE TST      ;SKIP TO END OF TEST      TRAP      C1ERROR
        032152 104410      TRAP      C1ESCAPE
        032154 001346      .WORD      L10063-.
6571 032156 104403      ENDSUB      ;SKIP TO END OF TEST      TRAP      C1ESUB
        032156 104403      L10066:
6572
6573
6574
SUBROUTINE # 4: 5 BIT CHARACTERS

```


TEST 25 - BOP RX CHARACTER LENGTH TEST

```

6618 032330 000005          5.
6619 032332 103003          BCC      .+8.      ;BR IF NO ERROR
6620 032334          ERROR          ;REPORT STACKED ERROR
        032334 104460
6621 032336          ESCAPE SUB      ;SKIP TO END OF TEST          TRAP      C1ERROR
        032336 104410          TRAP      C1ESCAPE
        032340 000204          .WORD      L10067-.
6622
6623 032342 022703 033533    CMP      #T30TBL+7,R3    ;ALL CHARACTERS TRANSMITTED ?
6624 032346 001363          BNE      10#            ; IF NOT, TX ANOTHER ONE
6625          ;-----
6626 032350 004537 010010    JSR      R5,TXCTRL      ;SET TEOM
6627 032354 000002          TEOM
6628 032356 000000          O
6629 032360 004537 011614    JSR      R5,STEPLU      ; AND TX DATA? + SOME FLAGS
6630 032364 000044          36.
6631
6632 032366 004537 010110    JSR      R5,RXCHAR      ;READ & CHK 123(HEADER1), RCV 321(HEADER2)
6633 032372 000523          RXSOM:123             ; & CHECK FOR RSOM+1
6634 032374 000000          O
6635 032376 100000          NOCRDA             ;NO INITIAL CHECK OF RDA=0
6636 032400 103003          BCC      .+8.      ;BR IF NO ERROR
6637 032402          ERROR          ;REPORT STACKED ERROR          TRAP      C1ERROR
        032402 104460          TRAP      C1ESCAPE
6638 032404          ESCAPE SUB      ;SKIP TO END OF TEST          TRAP      C1ESCAPE
        032404 104410          .WORD      L10067-.
        032406 000136
6639
6640 032410 004537 010110    JSR      R5,RXCHAR      ;READ/CHECK 321(HEADER2),RCV 011(DATA1)
6641 032414 000321          321
6642 032416 000000          O
6643 032420 100000          NOCRDA             ;NO INITIAL CHECK OF RDA=0
6644 032422 103003          BCC      .+8.      ;BR IF NO ERROR
6645 032424          ERROR          ;REPORT STACKED ERROR          TRAP      C1ERROR
        032424 104460          TRAP      C1ESCAPE
6646 032426          ESCAPE SUB      ;SKIP TO END OF TEST          TRAP      C1ESCAPE
        032426 104410          .WORD      L10067-.
        032430 000114
6647          ;-----
6648 032432 012703 033524    MOV      #T30TBL,R3    ;SET UP DATA TABLE POINTER
6649 032436 112337 032454    40# :   MOVB      (R3)+,4#    ;INSTALL NEXT EXPECTED RX CHARACTER
6650 032442 142737 000340 032454    BICB      #340,4#      ;MASK OUT UNTRANSMITTED BITS
6651
6652 032450 004537 010110    JSR      R5,RXCHAR      ;READ/CHECK NEXT CHARACTER
6653 032454 000000          4# :   OOO             ;** HOLE FOR NEXT EXPECTED CHARACTER
6654 032456 000000          O
6655 032460 100000          NOCRDA             ;NO INITIAL CHECK OF RDA=0
6656 032462 103003          BCC      .+8.      ;BR IF NO ERROR
6657 032464          ERROR          ;REPORT STACKED ERROR          TRAP      C1ERROR
        032464 104460          TRAP      C1ESCAPE
6658 032466          ESCAPE SUB      ;SKIP TO END OF TEST          TRAP      C1ESCAPE
        032466 104410          .WORD      L10067-.
        032470 000054
6659
6660 032472 022703 033531    CMP      #T30TBL+5,R3    ;ALL CHARACTERS CHECKED ?
6661 032476 001357          BNE      40#            ; IF NOT, CHECK ANOTHER ONE
6662          ;-----

```

TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6663 032500 004537 010110      JSR      R5,RXCHAR      ;READ/CHK DATA6(026), RCV DATA7
6664 032504 000026              026
6665 032506 000001      RERCHK      ; & CHECK RERR BIT=0 (GOOD CRC)
6666 032510 020000      NCRACT      ;DON'T CHECK FOR INITIAL RDA=0
6667 032512 103003      BCC      .+8.      ;BR IF NO ERROR
6668 032514 104460      ERROR      ;REPORT STACKED ERROR
6669 032516 104410      ESCAPE TST      ;SKIP TO END OF TEST
6669 032516 104410              TRAP      C#ERROR
6669 032520 001002              .WORD      TRAP      C#ESCAPE
6670 032520 001002              .WORD      L10063-.
6670
6671 032522 004537 010110      JSR      R5,RXCHAR      ;READ/CHK DATA7(037), RCV FIRST FLAG
6672 032526 001037      RXEOM!037      ; & CHECK FOR REOM=1
6673 032530 000000      0
6674 032532 060000      N#CRDA!NCRACT      ;DON'T CHECK FOR FINAL RDA=RXACT=1
6675 032534 103003      BCC      .+8.      ;BR IF NO ERROR
6676 032536 104460      ERROR      ;REPORT STACKED ERROR
6677 032540 104410      ESCAPE TST      ;SKIP TO END OF TEST
6677 032540 104410              TRAP      C#ERROR
6677 032542 000760              .WORD      TRAP      C#ESCAPE
6678 032544 104403      ENDSUB              .WORD      L10063-.
6678 032544 104403              L10067:      TRAP      C#ESUB
6679
6680 ;-----
6681 ; SUBROUTINE # 5: 6 BIT CHARACTERS
6682 ;-----
6682 032546 104402      BGN#SUB              125.5:      TRAP      C#BSUB
6683 032550 004537 007400      JSR      R5,TNITRN      ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
6684 032554 000226      SYNCH      ;SET BOP MODE,CRC-COITI-1,S/AR=226
6685 032556 000006      BIT2!BIT1      ;INITIALLY: TACL=8 BITS / RXCL=6 BITS
6686 032560 103003      BCC      .+8.      ;BR IF NO ERROR
6687 032562 104460      ERROR      ;REPORT STACKED ERROR
6688 032564 104410      ESCAPE SUB      ;SKIP TO END OF TEST
6688 032564 104410              TRAP      C#ERROR
6688 032566 000344              .WORD      TRAP      C#ESCAPE
6689 032570 004537 010010      JSR      R5,TXCTRL      ;LOAD 2ND FLAG, TX 1ST FLAG
6690 032574 000001      TSOM
6691 032576 000007      7.
6692 032576 000007      7.
6693 032600 004537 010010      JSR      R5,TXCTRL      ;CLEAR TSOM
6694 032604 000000      000
6695 032606 000000      0
6696 032610 004537 007676      JSR      R5,TXCHAR      ;LOAD 123(HEADER1), TX 2ND FLAG
6697 032614 000123      123
6698 032616 000010      8.
6699 032620 103003      BCC      .+8.      ;BR IF NO ERROR
6700 032622 104460      ERROR      ;REPORT STACKED ERROR
6701 032624 104410      ESCAPE SUB      ;SKIP TO END OF TEST
6701 032624 104410              TRAP      C#ERROR
6701 032626 000304              .WORD      TRAP      C#ESCAPE
6702 032626 000304              .WORD      L10070-.
6703 032630 004537 007676      JSR      R5,TXCHAR      ;LOAD 321(HEADER2), TX 123(HEADER1)

```

TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6704 032634 000321          321
6705 032636 000010          8.
6706 032640 103003          BCC      .+8.          ;BR IF NO ERROR
6707 032642          ERROR          ;REPORT STACKED ERROR
        032642 104460          TRAP      C$ERROR
6708 032644          ESCAPE SUB          ;SKIP TO END OF TEST
        032644 104410          TRAP      C$ESCAPE
        032646 000264          .WORD   L10070-.
6709
6710 032650 004537 003734          JSR      R5,WRITEI          ;NOW CHANGE TXCL TO 6 BITS TOO
6711 032654 120407          PCR
6712 032656 000306          BIT7!BIT6!BIT2!BIT1
6713 032660 004537 007676          JSR      R5,TXCHAR          ;LOAD 111(DATA1), TX 321(HEADER2)
6714 032664 000111          111
6715 032666 000010          8.
6716 032670 103003          BCC      .+8.          ;BR IF NO ERROR
6717 032672          ERROR          ;REPORT STACKED ERROR
        032672 104460          TRAP      C$ERROR
6718 032674          ESCAPE SUB          ;SKIP TO END OF TEST
        032674 104410          TRAP      C$ESCAPE
        032676 000234          .WORD   L10070-.
6719
6720 032700 012703 033525          ;-----
10$:  MOV      #T30TBL+1,R3          ;SET UP DATA TABLE POINTER
        MOVB    (R3)+,1$          ;INSTALL NEXT TX CHARACTER
6722
6723 032710 004537 007676          JSR      R5,TXCHAR          ;TRANSMIT CHARACTER ( **> RX/FIFO )
1$:  000          ;** HOLE FOR NEXT CHARACTER **
        6.
6725 032716 000006          BCC      .+8.          ;BR IF NO ERROR
6726 032720 103003          ERROR          ;REPORT STACKED ERROR
        032722 104460          TRAP      C$ERROR
6727 032722          ESCAPE SUB          ;SKIP TO END OF TEST
        032724 104410          TRAP      C$ESCAPE
        032726 000204          .WORD   L10070-.
6728
6729
6730 032730 022703 033533          CMP      #T30TBL+7,R3          ;ALL CHARACTERS TRANSMITTED ?
6731 032734 001363          ENE     10$          ; IF NOT, TX ANOTHER ONE
6732
6733 032736 004537 010010          ;-----
        JSR      R5,TXCTRL          ;SET TEOM
6734 032742 000002          TEOM
6735 032744 000000          0
6736 032746 004537 011614          JSR      R5,STEPLU          ; AND TX DATA7 + SOME FLAGS
6737 032752 000044          36.
6738
6739 032754 004537 010110          JSR      R5,RXCHAR          ;READ & CHK 123(HEADER1), RCV 321(HEADER2)
6740 032760 000523          RXSOM!123          ; & CHECK FOR RSOM=1
6741 032762 000000          0
6742 032764 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
6743 032766 103003          BCC      .+8.          ;BR IF NO ERROR
6744 032770          ERROR          ;REPORT STACKED ERROR
        032770 104460          TRAP      C$ERROR
6745 032772          ESCAPE SUB          ;SKIP TO END OF TEST
        032772 104410          TRAP      C$ESCAPE
        032774 000136          .WORD   L10070-.
6746
6747 032776 004537 010110          JSR      R5,RXCHAR          ;READ/CHECK 321(HEADER2),RCV 011(DATA1)
6748 033002 000321          321
    
```

TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6749 033004 000000          0
6750 033006 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
6751 033010 103003          BCC          .+8. ;BR IF NO ERROR
6752 033012 104460          ERROR          ;REPORT STACKED ERROR
6753 033014          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP C$ERROR
        033014 104410          ;                                TRAP C$ESCAPE
        033016 000114          ;                                .WORD L10070-.
6754 ;-----;
6755 033020 012703 033524      MOV      #T30TBL,R3      ;SET UP DATA TABLE POINTER
6756 033024 112337 033042      40$: MOVB   (R3)+,4$      ;INSTALL NEXT EXPECTED RX CHARACTER
6757 033030 142737 000300 033042 BICB   #300,4$          ;MASK OUT UNTRANSMITTED BITS
6758
6759 033036 004537 010110      JSR     R5,RXCHAR        ;READ/CHECK NEXT CHARACTER
6760 033042 000000      4$: 000          ;** HOLE FOR NEXT EXPECTED CHARACTER
6761 033044 000000          0
6762 033046 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
6763 033050 103003          BCC          .+8. ;BR IF NO ERROR
6764 033052          ERROR          ;REPORT STACKED ERROR
        033052 104460          ;                                TRAP C$ERROR
6765 033054          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP C$ESCAPE
        033054 104410          ;                                TRAP C$ESCAPE
        033056 000054          ;                                .WORD L10070-.
6766
6767 033060 022703 033531      CMP     #T30TBL+5,R3    ;ALL CHARACTERS CHECKED ?
6768 033064 001357          BNE     40$            ; IF NOT, CHECK ANOTHER ONE
6769 ;-----;
6770 033066 004537 010110      JSR     R5,RXCHAR        ;READ/CHK DATA6(066), RCV DATA7
6771 033072 000066          066
6772 033074 000001          RERCHK          ; & CHECK RERR BIT=0 (GOOD CRC)
6773 033076 020000          NCRACT          ;DON'T CHECK FOR INITIAL RDA=0
6774 033100 103003          BCC          .+8. ;BR IF NO ERROR
6775 033102          ERROR          ;REPORT STACKED ERROR
        033102 104460          ;                                TRAP C$ERROR
6776 033104          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ESCAPE
        033104 104410          ;                                TRAP C$ESCAPE
        033106 000414          ;                                .WORD L10063-.
6777
6778 033110 004537 010110      JSR     R5,RXCHAR        ;READ/CHK DATA7(077), RCV FIRST FLAG
6779 033114 001077          RXEOM!077        ; & CHECK FOR REOM=1
6780 033116 000000          0
6781 033120 060000          NCRDA!NCRACT      ;DON'T CHECK FOR FINAL RDA=RXACT=1
6782 033122 103003          BCC          .+8. ;BR IF NO ERROR
6783 033124          ERROR          ;REPORT STACKED ERROR
        033124 104460          ;                                TRAP C$ERROR
6784 033126          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ESCAPE
        033126 104410          ;                                TRAP C$ESCAPE
        033130 000372          ;                                .WORD L10063-.
6785 033132          ENDSUB          ;                                L10070: TRAP C$ESUB
        033132 104403          ;
6786 ;-----;
6787 ; SUBROUTINE # 6: 7 BIT CHARACTERS
6788 ;-----;
6789 033134          BGNSUB          ;                                T25.6: TRAP C$BSUB
        033134 104402          ;

```

TEST 25 -- BOP RX CHARACTER LENGTH TEST

```

6790 033136 004537 007400      JSR      R5,INITRN      ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
6791 033142 000226              SYNCH                    ;SET BOP MODE,CRC-CCITT-1,S/AR=226
6792 033144 000007              RXDL                     ;INITIALLY; TXCL=8 BITS / RXCL=7 BITS
6793 033146 103003              BCC      .+8.            ;BR IF NO ERROR
6794 033150                      ERROR                     ;REPORT STACKED ERROR
                                TRAP      C#ERROR
6795 033152                      ESCAPE  SUB              ;SKIP TO END OF TEST
                                TRAP      C#ESCAPE
                                .WORD    L10071-.
                                033152 104410
                                033154 000344

6796
6797 033156 004537 010010      JSR      R5,TXCTRL      ;LOAD 2ND FLAG,TX 1ST FLAG
6798 033162 000001              TSOM
6799 033164 000007              7.
6800 033166 004537 010010      JSR      R5,TXCTRL      ;CLEAR TSOM
6801 033172 000000              000
6802 033174 000000              0
6803 033176 004537 007676      JSR      R5,TXCHAR      ;LOAD 123(HEADER1), TX 2ND FLAG
6804 033202 000123              123
6805 033204 000010              8.
6806 033206 103003              BCC      .+8.            ;BR IF NO ERROR
6807 033210                      ERROR                     ;REPORT STACKED ERROR
                                TRAP      C#ERROR
6808 033212                      ESCAPE  SUB              ;SKIP TO END OF TEST
                                TRAP      C#ESCAPE
                                .WORD    L10071-.
                                033212 104410
                                033214 000304

6809
6810 033216 004537 007676      JSR      R5,TXCHAR      ;LOAD 321(HEADER2), TX 123(HEADER1)
6811 033222 000321              321
6812 033224 000010              8.
6813 033226 103003              BCC      .+8.            ;BR IF NO ERROR
6814 033230                      ERROR                     ;REPORT STACKED ERROR
                                TRAP      C#ERROR
6815 033232                      ESCAPE  SUB              ;SKIP TO END OF TEST
                                TRAP      C#ESCAPE
                                .WORD    L10071-.
                                033232 104410
                                033234 000264

6816
6817 033236 004537 003734      JSR      R5,WRITEI      ;NOW CHANGE TXCL TO 6 BITS TOO
6818 033242 120407              PCR
6819 033244 000347              TXDL!RXDL
6820 033246 004537 007676      JSR      R5,TXCHAR      ;LOAD 111(DATA1), TX 321(HEADER2)
6821 033252 000111              111
6822 033254 000010              8.
6823 033256 103003              BCC      .+8.            ;BR IF NO ERROR
6824 033260                      ERROR                     ;REPORT STACKED ERROR
                                TRAP      C#ERROR
6825 033262                      ESCAPE  SUB              ;SKIP TO END OF TEST
                                TRAP      C#ESCAPE
                                .WORD    L10071-.
                                033262 104410
                                033264 000234

6826
6827 033266 012703 033525      ;-----
10$: MOV      @T30TBL+1,R3      ;SET UP DATA TABLE POINTER
6828 033272 112337 033302      MOVB     (R3)+,1$        ;INSTALL NEXT TX CHARACTER
6829
6830 033276 004537 007676      JSR      R5,TXCHAR      ;TRANSMIT CHARACTER ( **> RX/FIFO )
6831 033302 000000      1$: 000                  ;** HOLE FOR NEXT CHARACTER **
6832 033304 000007              7.
6833 033306 103003              BCC      .+8.            ;BR IF NO ERROR
6834 033310                      ERROR                     ;REPORT STACKED ERROR

```


TEST 25 -- BOP RX CHARACTER LENGTH TEST

6835	033310	104460					TRAP	C\$ERROR
	033312			ESCAPE	SUB	;SKIP TO END OF TEST		
	033312	104410					TRAP	C\$ESCAPE
	033314	000204					.WORD	L10071-
6836								
6837	033316	022703	033533					
6838	033322	001363						
6839								
6840	033324	004537	010010					
6841	033330	000002						
6842	033332	000000						
6843	033334	004537	011614					
6844	033340	000044						
6845								
6846	033342	004537	010110					
6847	033346	000523						
6848	033350	000000						
6849	033352	100000						
6850	033354	103003						
6851	033356							
	033356	104460						
6852	033360							
	033360	104410						
	033362	000136						
6853								
6854	033364	004537	010110					
6855	033370	000321						
6856	033372	000000						
6857	033374	100000						
6858	033376	103003						
6859	033400							
	033400	104460						
6860	033402							
	033402	104410						
	033404	000114						
6861								
6862	033406	012703	033524					
6863	033412	112337	033430					
6864	033416	142737	000200	033430				
6865								
6866	033424	004537	010110					
6867	033430	000000						
6868	033432	000000						
6869	033434	100000						
6870	033436	103003						
6871	033440							
	033440	104460						
6872	033442							
	033442	104410						
	033444	000054						
6873								
6874	033446	022703	033531					
6875	033452	001357						
6876								
6877	033454	004537	010110					
6878	033460	000066						
6879	033462	000001						

TEST 25 - BOP RX CHARACTER LENGTH TEST

```

6880 033464 020000          NCRACT          ;DON'T CHECK FOR INITIAL RDA=0
6881 033466 103003          BCC          .+8.  ;BR IF NO ERROR
6882 033470          ERROR          ;REPORT STACKED ERROR
        033470 104460          ;
6883 033472          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
        033472 104410          ;
        033474 000026          ;
6884          ;
6885 033476 004537 010110    JSR          R5,RXCHAR          ;READ/CHK DATA7(177), RCV FIRST FLAG
6886 033502 001177          RXEOM!177          ; & CHECK FOR REOM=1
6887 033504 000000          0
6888 033506 060000          NRCRDA!NCRACT          ;DON'T CHECK FOR FINAL RDA=RXACT=1
6889 033510 103003          BCC          .+8.  ;BR IF NO ERROR
6890 033512          ERROR          ;REPORT STACKED ERROR
        033512 104460          ;
6891 033514          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
        033514 104410          ;
        033516 000004          ;
6892 033520          ENDSUB          ;
        033520          ;
        033520 104403          ;
6893 033522          ENDTST          ;
        033522          ;
        033522 104401          ;
6894          ;
6895 033524          111          T30TBL: .BYTE 111 ;D1
6896 033525          222          .BYTE 222 ;D2
6897 033526          333          .BYTE 333 ;D3
6898 033527          044          .BYTE 044 ;D4
6899 033530          155          .BYTE 155 ;D5
6900 033531          266          .BYTE 266 ;D6
6901 033532          377          .BYTE 377 ;D7
6902          ;
6903          ;

```

```

L10071: TRAP C$ESUB
L10063: TRAP C$ETST

```

TEST 26 -- TX "SPACING SEQUENCE"

6911

.SBTTL TEST 26 -- TX "SPACING SEQUENCE"

```

*****
;*
;* TEST 26 -- TX "SPACING SEQUENCE"
;*
;* THE TRANSMITTER IS INITIALIZED AND THE "SPACING SEQUENCE" IS FORCED
;* BY ASSERTING BOTH TSOM & TEOM AT THE SAME TIME -- CHECK THE BIT
;* STREAM FOR ACCURACY (SPACES) AND COMPLETENESS (16 OF THEM). WHEN TXBE
;* GOES HIGH (= 1) A SMALL MESSAGE IS SENT.
;*
*****

```

```

6912 033534 004737 005420
6913
6914
6915
6916 033540 004537 003734
6917 033544 120013
6918 033546 000200
6919 033550 004537 003734
6920 033554 120006
6921 033556 000300
6922 033560 004537 003734
6923 033564 120007
6924 033566 000000
6925
6926 033570 004537 003734
6927 033574 120000
6928 033576 000031
6929 033600 004537 003734
6930 033604 120000
6931 033606 000030
6932
6933 033610 004537 003734
6934 033614 120404
6935 033616 000000
6936 033620 004537 003734
6937 033624 120405
6938 033626 000007
6939
6940 033630 004537 003734
6941 033634 120000
6942 033636 000172
6943
6944 033640 004537 006036
6945 033644 000001
6946 033646 103003
6947 033650
033650 104460
6948 033652
033652 104410
033654 000234
6949
6950

```

```

;
; BGNTST
;
;-----T26:-----
; JSR PC,INIDMV ;INIT VIA
;-----
; SET UP USYRT AND VIA REGISTERS
;-----
; JSR R5,WRITEI ;SET ACR FOR T1 ONE-SHOT MODE
; VIAACR
; 200
; JSR R5,WRITEI ;LOAD VIA T1L-L
; VIAT1C
; 300
; JSR R5,WRITEI ;LOAD VIA T1L-H
; VIAT1D
; 000
;
; JSR R5,WRITEI ;RESET THE USYRT
; VIAORB
; RTSND!DTR!PRESET
; JSR R5,WRITEI ;CLEAR USYRT RESET BIT
; VIAORB
; RTSND!DTR
;
; JSR R5,WRITEI ;LOAD USYRT PCSARL
; PCSARL
; 000
; JSR R5,WRITEI ;LOAD USYRT PCSARH
; PCSARH
; XYZ ;(NO ERROR CHECKING)
;
; JSR R5,WRITEI ;SET UP USYRT
; VIAORB
; RTSND!TXEN!RXEN!DTR!TTLOOP
;
; JSR R5,CKTBMT ;CHK FOR TBMT = 1
; 1
; BCC .+8. ;IF NO ERROR, PROCEED
; ERROR ;ELSE, REPORT IT AND
;
; ESCAPE TST ; EXIT THIS TEST TRAP C$ERROR
;
; TRAP C$ESCAPE
; .WORD L10072-
;-----
; INIT SPACING SEQUENCE BY SETTING TSOM AND TEOM

```

TEST 26 -- TX "SPACING SEQUENCE"

```

6951
6952 033656 004537 010010      ;-----
6953 033662 000003      6$: JSR      R5, TXCTRL      ;SET TSOM AND TEOM
6954 033664 000000      TSOM:TEOM      ; ("SPACING SEQUENCE")
6955
6956 033666 004537 011614      JSR      R5, STEPLU      ;1 TICK TO START SPACE SEQUENCE
6957 033672 000001      1
6958
6959 033674 004537 007256      JSR      R5, SERIAL      ;READ 000 CHARACTER VIA TSO
6960 033700 000010      8.      ; 8 BIT CHAR/CLOCK TICKS
6961 033702 000000      000      ; EXPECTED BIT SEQUENCE (0000000)
6962 033704 103003      BCC      .+8.      ;BR IF NO ERROR
6963 033706 104460      ERROR      ;REPORT STACKED ERROR
6964 033710 104410      ESCAPE TST      ;AND EXIT TEST
6965 033712 000176      TRAP      C$ERROR
6966 033714 004537 006036      JSR      R5, CKTBMT      ;CHK FOR TBMT = 1
6967 033720 000001      1
6968 033722 103003      BCC      .+8.      ;IF NO ERROR, PROCEED
6969 033724 104460      ERROR      ;ELSE, REPORT IT AND
6970 033726 104410      ESCAPE TST      ; EXIT THIS TEST
6971 033730 000160      TRAP      C$ERROR
6972 033732 004537 010010      ;-----
6973 033736 000001      JSR      R5, TXCTRL      ;CLEAR TEOM
6974 033740 000000      0
6975
6976 033742 004537 007256      JSR      R5, SERIAL      ;READ 000 CHARACTER VIA TSO
6977 033746 000010      8.      ; 8 BIT CHAR/CLOCK TICKS
6978 033750 000000      000      ; EXPECTED BIT SEQUENCE (00000000)
6979 033752 103003      BCC      .+8.      ;BR IF NO ERROR
6980 033754 104460      ERROR      ;REPORT STACKED ERROR
6981 033756 104410      ESCAPE TST      ;AND EXIT TEST
6982 033760 000130      TRAP      C$ERROR
6983 033762 004537 006036      JSR      R5, CKTBMT      ;CHK FOR TBMT = 1
6984 033766 000001      1
6985 033770 103003      BCC      .+8.      ;IF NO ERROR, PROCEED
6986 033772 104460      ERROR      ;ELSE, REPORT IT AND
6987 033774 104410      ESCAPE TST      ; EXIT THIS TEST
6988 033776 000112      TRAP      C$ERROR
6989
6990      ; SEND "SHORT MESSAGE" (125,252)
6991 034000 004537 003734      ;-----
6992 034004 120402      JSR      R5, WRITEI      ;LOAD 125 CHARACTER
6993 034006 000125      TDSRL
6994 034010 004537 010010      JSR      R5, TXCTRL      ;CLEAR TSOM
6995 034014 000000      000
    
```

TEST 26 -- TX "SPACING SEQUENCE"

6996	034016	000000		0				
6997								
6998	034020	004537	007256	JSR	R5,SERIAL	;READ FLAG CHARACTER (176) VIA TSO		
6999	034024	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
7000	034026	000176		176		; EXPECTED BIT SEQUENCE (01111110)		
7001	034030	103003		BCC	+.8.	;BR IF NO ERROR		
7002	034032			ERROR		;REPORT STACKED ERROR		
	034032	104460					TRAP	C\$ERROR
7003	034034			ESCAPE	TST	;AND EXIT TEST		
	034034	104410					TRAP	C\$ESCAPE
	034036	000052					.WORD	L10072-
7004								
7005	034040	004537	003734	JSR	R5,WRITEI	;LOAD 252 CHARACTER		
7006	034044	120402		TDSRL				
7007	034046	000252		252				
7008								
7009	034050	004537	007256	JSR	R5,SERIAL	;READ 1ST DATA CHARACTER (125) VIA TSO		
7010	034054	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
7011	034056	000252		252		; EXPECTED BIT SEQUENCE (01010110)		
7012	034060	103003		BCC	+.8.	;BR IF NO ERROR		
7013	034062			ERROR		;REPORT STACKED ERROR		
	034062	104460					TRAP	C\$ERROR
7014	034064			ESCAPE	TST	;AND EXIT TEST		
	034064	104410					TRAP	C\$ESCAPE
	034066	000022					.WORD	L10072-
7015								
7016	034070	004537	007256	JSR	R5,SERIAL	;READ 2ND DATA CHARACTER (252) VIA TSO		
7017	034074	000010		8.		; 8 BIT CHAR/CLOCK TICKS		
7018	034076	000125		125		; EXPECTED BIT SEQUENCE (10101010)		
7019	034100	103003		BCC	+.8.	;BR IF NO ERROR		
7020	034102			ERROR		;REPORT STACKED ERROR		
	034102	104460					TRAP	C\$ERROR
7021	034104			ESCAPE	TST	;AND EXIT TEST		
	034104	104410					TRAP	C\$ESCAPE
	034106	000002					.WORD	L10072-
7022	034110			ENDTST				
	034110						L10072:	
	034110	104401					TRAP	C\$ETS1

TEST 27 -- FIFO OVERRUN INTEGRITY TEST

7037

.SBTTL TEST 27 -- FIFO OVERRUN INTEGRITY TEST

```

;+*****
;*
;* TEST 27 -- FIFO OVERRUN INTEGRITY TEST
;*
;* THIS TEST BEGINS BY SYNCHRONIZING THE RECEIVER AND THEN PROCEEDS TO FILL
;* THE 8 CHARACTER RECEIVER FIFO UNTIL RXOR WITH THE CHARACTERS:
;* (SYNCH),000,377,125,252,347,030,303,074,125.
;* THESE CHARACTERS ARE THEN READ OFF OF THE FIFO AND CHECKED. OF IMPORTANCE
;* IS THE INTEGRITY OF THE LAST OVERRUN-CAUSING FIFO CHARACTER (IT SHOULD
;* REMAIN INTACT).
;* NOTE THAT NO CLOCKS ARE PROVIDED WHEN RECEIVING THE CHARACTERS SINCE THEY
;* ARE SUPPLIED BY THE FIFO SUPPORT LOGIC IN GROUPS OF 4 TICKS (WHEN
;* RDA * 0).
;*
;+*****

```

```

; BGNTST
;
; T27::
7038 034112 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
7039
7040 034116 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
7041 034122 043626 DDCMP!NOCHK!SYNCH ;SET UDCMP,NO CHECK,SYNCH=226
7042 034124 000000 0 ;USE 8 BIT CHARS
7043 034126 103003 BCC .+8. ;BR IF NO ERROR
7044 034130 ERROR ;REPORT STACKED ERROR
7045 034132 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
034130 104460
034132 104410 TRAP C$ESCAPE
034134 000324 .WORD L10073-.
7046 034136 004537 010010 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTERS
7047 034142 000001 TSOM ;
7048 034144 000007 7.
7049
7050 034146 004537 010010 JSR R5,TXCTRL ;CLEAR TSOM
7051 034152 000000 000
7052 034154 000000 0
7053
7054 ; FILL THE FIFO WITH CHARACTERS AND FORCE OVERRUN
7055
7056 034156 012702 034462 MOV #TXB4,R2 ;SET UP TABLE POINTER
7057 034162 112237 034172 5$: MOVB (R2)+,10$ ;SETUP TRANSMIT CHARACTER
7058
7059 034166 004537 007676 JSR R5,TXCHAR ;TRANSMIT A CHARACTER
7060 034172 000000 000 ;** HOLE FOR NEXT TX CHARACTER
7061 034174 100010 NCTBMT+256.!8. ;NO CHECK OF INITIAL TBMT=0
7062 034176 103003 BCC .+8. ;BR IF NO ERROR
7063 034200 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
034200 104460
7064 034202 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
034202 104410 .WORD L10073-.
034204 000254
7065
7066 034206 022702 034475 CMP #TXEND4,R2 ;CONTINUE TRANSMITTING UNTIL

```

TEST 27 FIFO OVERRUN INTEGRITY TEST

```

7067 034212 001363          BNE      54          ;OVERRUN.
7068
7069 034214 004537 006476    JSR      R5,CKROR    ;VERIFY RECEIVED OVERRUN OCCURED
7070 034220 000001          1
7071 034222 103003          BCC      .+8.        ;BR IF OK
7072 034224          ERROR          ;REPORT STACKED ERROR
7073 034224 104460          ESCAPE   T5T         ;SKIP TO END OF TEST          TRAP      C!ERROR
7074 034226 104410          .WORD          TRAP      C!ESCAPE
7075 034230 000230          .WORD          L10073-.
7076
7077
; THE RECEIVER SHOULD HAVE OVERRUN. VERIFY THIS AND READ/VERIFY
; THE REMAINING FIFO CHARACTERS (ESPECIALLY THE LAST ONE).
7078 034232 004537 010110    JSR      R5,RXCHAR    ;READ & CHK CHARACTER
7079 034236 000226          SYNCH          ; (THIS CHARACTER ISN'T AFFECTED)
7080 034240 000000          0
7081 034242 100000          NOCRDA        ;NO INITIAL CHECK OF RDA=0
7082 034244 103003          BCC      .+8.        ;BR IF NO ERROR
7083 034246          ERROR          ;REPORT STACKED ERROR          TRAP      C!ERROR
7084 034246 104460          ESCAPE   T5T         ;SKIP TO END OF TEST          TRAP      C!ESCAPE
7085 034250 104410          .WORD          .WORD          L10073-.
7086 034252 000206
7086 034254 004537 010110    JSR      R5,RXCHAR    ;READ & CHK CHARACTER
7087 034260 000000          000          ; (THIS IS THE OVERRITTEN CHARACTER)
7088 034262 000000          0            ; ... WAS A "SYNCH" BEFORE OVERRITTEN
7089 034264 100000          NOCRDA        ;NO INITIAL CHECK OF RDA=0
7090 034266 103003          BCC      .+8.        ;BR IF NO ERROR
7091 034270          ERROR          ;REPORT STACKED ERROR          TRAP      C!ERROR
7092 034270 104460          ESCAPE   T5T         ;SKIP TO END OF TEST          TRAP      C!ESCAPE
7093 034272 104410          .WORD          .WORD          L10073
7094 034274 000164
7094 034276 012702 034465    MOV      #TXIBL4-3,R2 ;SET UP TABLE POINTER
7095 034302 112237 034312    154:    MOVB      (R2)+,201 ;SETUP EXPECTED CHARACTER
7096
7097 034306 004537 010110    JSR      R5,RXCHAR    ;READ & CHK CHARACTER
7098 034312 000000          201:    000          ;** HOLE FOR EXPECTED RECEIVE CHAR.
7099 034314 000000          0
7100 034316 100000          NOCRDA        ;NO INITIAL CHECK OF RDA=0
7101 034320 103003          BCC      .+8.        ;BR IF NO ERROR
7102 034322          ERROR          ;REPORT STACKED ERROR          TRAP      C!ERROR
7103 034322 104460          ESCAPE   T5T         ;SKIP TO END OF TEST          TRAP      C!ESCAPE
7104 034324 104410          .WORD          .WORD          L10073-.
7105 034326 000132
7104 034330 022702 034472    CMP      #TXEND4-3,R2
7105 034334 001362          BNE      154
7106
7107 034336 012702 000004    MOV      #4,R2        ;TRANSMIT 4 EXTRA CHARACTERS TO AVOID
7108 034342 004537 007676    301:    JSR      R5, TXCHAR    ;TRANSMITTER UNDERRUN.
7109 034346 000333          333          ;FILLER CHARACTER
7110 034350 100010          NCTBMT+256.!8.      ;NO CHECK OF INITIAL TBMT=0
7111 034352 077205          SOB      R2,301

```

TEST 27 FIFO OVERRUN INTEGRITY TEST

```

7110
7113 034354 004537 010110 JSR R5,RXCHAR ;READ & CHK CHARACTER
7114 034360 000303 303
7115 034362 000000 0
7116 034364 100000 NOCRDA ;NO INITIAL CHECK OF RDA=0
7117 034366 103003 BCC ;+8. ;BR IF NO ERROR
7118 034370 104460 ERROR ;REPORT STACKED ERROR TRAP C1ERROR
034370 104460
7119 034372 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C1ESCAPE
034372 104410 .WORD L10073
034374 000064
7120
7121 034376 004537 010110 JSR R5,RXCHAR ;READ & CHK CHARACTER
7122 034402 000074 074
7123 034404 000000 0
7124 034406 100000 NOCRDA ;NO INITIAL CHECK OF RDA=0
7125 034410 103003 BCC ;+8. ;BR IF NO ERROR
7126 034412 104460 ERROR ;REPORT STACKED ERROR TRAP C1ERROR
034412 104460
7127 034414 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C1ESCAPE
034414 104410 .WORD L10073
034416 000042
7128
7129 034420 004537 010110 JSR R5,RXCHAR ;* READ & CHK FINAL CHARACTER
7130 034424 000125 125 ;*
7131 034426 000000 0 ;*
7132 034430 100000 NOCRDA ;* NO INITIAL CHECK OF RDA=0
7133 034432 103003 BCC ;+8. ;* BR IF NO ERROR
7134 034434 104460 ERROR ;* REPORT STACKED ERROR TRAP C1ERROR
034434 104460
7135 034436 104410 ESCAPE TST ;* SKIP TO END OF TEST TRAP C1ESCAPE
034436 104410 .WORD L10073
034440 000020
7136
7137 034442 004537 011532 JSR R5,ENDTRN ;SHUT DOWN TRANSMITTER, RECEIVER
7138 034446 000011 11
7139 034450 103003 BCC ;+8. ;BR IF NO ERROR
7140 034452 104460 ERROR ;REPORT STACKED ERROR TRAP C1ERROR
034452 104460
7141 034454 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C1ESCAPE
034454 104410 .WORD L10073
034456 000002
7142 034460 104401 ENDTST TRAP C1ETST
034460 104401 L10073:
034460 104401
7143
7144 034462 226 TXIBLA: .BYTE 226
7145 034463 226 .BYTE 226
7146 034464 000 .BYTE 000
7147 034465 377 .BYTE 377
7148 034466 125 .BYTE 125
7149 034467 252 .BYTE 252
7150 034470 347 .BYTE 347
7151 034471 030 .BYTE 030
7152 034472 303 .BYTE 303
7153 034473 074 .BYTE 074
7154 034474 125 .BYTE 125
    
```


015

TEST 27 FIFO OVERRUN INTEGRITY TEST

7155 034475 000
7156
7157

TXEND4: .B711 000
.EVEN
;-

[Faint, mostly illegible data or test results, possibly including hex values and status indicators]

TEST 28 BCP RX OVERRUN SET AND CLEAR TEST

7177

.SBTTL TEST 28 - BCP RX OVERRUN SET AND CLEAR TEST

```

*****
*
* TEST 28 -- BCP RX OVERRUN SET AND CLEAR TEST
*
* THE USYRT IS INITIALIZED AND THREE SUBTESTS ARE PERFORMED.
*
* 1 -- AN OVERRUN CONDITION IS FORCED, RECEIVER STATUS REGISTER IS
* READ TWICE: ONCE TO VERIFY ROR BIT = 1, AND AGAIN TO VERIFY
* THAT THE FIRST READ CLEARED ROR .
*
* 2 -- AN OVERRUN CONDITION IS FORCED (BY THE SAME TECHNIQUE USED IN
* (2), THE USYRT IS RESET AND THE PROPER STATE OF ALL REGISTERS
* IS VERIFIED.
*
* 3 -- AN OVERRUN CONDITION IS FORCED (AS ABOVE). RXE IS THEN DROPPED
* AND A DELAY IS PROVIDED TO ALLOW TIME FOR THE FIFO TO FLUSH
* (CAUSED BY RDA GOING LOW). RXE IS THEN RE-INITIALIZED AND ROR
* IS CHECKED = 0. THE RECEIVER IS THEN RE-SYNCHED AND THE TEST IS
* TERMINATED.
*
*****

```

```

;
; RGNTST
;
;***** SUBTEST 01 *****
; BGNSUB
;
; T28.1:
; TRAP C:BSUB
;
; JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
;
; JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
; DDCMP:NOCHK:SYNCH ;SET DDCMP, NO ERR CHECKING, SYNCH=226
; 0 ;USE 8 BIT CHARS
; BCC ,+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
; TRAP C:ERROR
;
; ESCAPE SUB ;SKIP TO END OF TEST
; TRAP C:ESCAPE
; .WORD L10075-.
;
; JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTERS
; TSOM
; 7.
; JSR R5,TXCTRL ;OUTPUT 2ND SYNC CHARACTER
; TSOM
; 8.
; JSR R5,TXCTRL ;CLEAR TSOM
; 000
; 0
;
; JSR R5,CKROR ;CHECK FOR RXOR = 0
; 0
; BCC ,+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR

```

```

7178 034476
7179 034476 034476
034476 104402
7180 034500 004737 005420
7181
7182 034504 004537 007400
7183 034510 043626
7184 034512 000000
7185 034514 103003
7186 034516 104460
7187 034520 034520 104410
034522 000112
7188
7189 034524 004537 010010
7190 034530 000001
7191 034532 000007
7192 034534 004537 010010
7193 034540 000001
7194 034542 000010
7195 034544 004537 010010
7196 034550 000000
7197 034552 000000
7198
7199 034554 004537 006476
7200 034560 000000
7201 034562 103003
7202 034564

```

TEST 28 -- BCP RX OVERRUN SET AND CLEAR TEST

```

034564 104460
7203 034565          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP C$ERROR
      034566 104410
      034570 000044          TRAP C$ESCAPE
7204          .WORD L10075
7205 034572 004537 011614 JSR R5,STEPLU ;FORCE RECEIVER OVERRUN
7206 034576 000116 78.
7207
7208 034600 004537 006476 JSR R5,CKROR ;CHECK FOR RXOP = 1
7209 034604 000001 1
7210 034606 103003 BCC .+8. ;BR IF NO ERROR
7211 034610          ERROR ;REPORT STACKED ERROR
      034610 104460          TRAP C$ERROR
7212 034612          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ESCAPE
      034612 104410          .WORD L10074-.
      034614 000262
7213
7214 034616 004537 006476 JSR R5,CKROR ;CHECK FOR RXOR = 0
7215 034622 000000 0
7216 034624 103003 BCC .+8. ;BR IF NO ERROR
7217 034626          ERROR ;REPORT STACKED ERROR          TRAP C$ERROR
      034626 104460
7218 034630          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP C$ESCAPE
      034630 104410          .WORD L10075-.
      034632 000002
7219 034634          ENDSUB
      034634          L10075: TRAP C$ESUB
      034634 104403
7220          ;***** SUBTEST 02 *****
7221 034636          BCNSUB          T28.2: TRAP C$BSUB
      034636
      034636 104402
7222 034640 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
7223
7224 034644 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
7225 034650 043626 DDCMP:NOCHK!SYNCH ;SET DDCMP, NO ERR CHECKING, SYNCH=226
7226 034652 000000 0 ;USE 8 BIT CHARS
7227 034654 103003 BCC .+8. ;BR IF NO ERROR
7228 034656          ERROR ;REPORT STACKED ERROR          TRAP C$ERROR
      034656 104460
7229 034660          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP C$ESCAPE
      034660 104410          .WORD L10076-.
      034662 000054
7230
7231 034664 004537 010010 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTERS
7232 034670 000001 TSOM ;
7233 034672 000007 7.
7234 034674 004537 010010 JSR R5,TXCTRL ;OUTPUT 2ND SYNC CHARACTER
7235 034700 000001 TSOM ;
7236 034702 000010 8.
7237 034704 004537 010010 JSR R5,TXCTRL ;CLEAR TSOM
7238 034710 000000 000
7239 034712 000000 0
7240 034714 004537 011614 JSR R5,STEPLU ;FORCE RECEIVER OVERRUN
7241 034720 000116 78.
7242
7243 034722 004537 005126 JSR R5,RSTCHK ;RESET USYRT/VERIFY SAME

```


TEST 28 - BCP RX OVERRUN SET AND CLEAR TEST

7288	035074		ENDSUB
	035074		
	035074	104403	
7289	035076		ENDTST
	035076		
	035076	104401	

L10077:	TRAP	C\$E\$UB
L10074:	TRAP	C\$E\$TST

TEST 29 -- BCP RX SYNC CHARACTER RECOGNITION

7302

.SBTTL TEST 29 -- BCP RX SYNC CHARACTER RECOGNITION

```

*****
:
: * TEST 29 -- BCP RX SYNC CHARACTER RECOGNITION
: *
: * THE FOLLOWING MESSAGE IS INITIATED WITHOUT ASSERTING RXE AND ONCE
: * THE DATA IS BEING TRANSMITTED, RXE IS ASSERTED (IE: *):
: *
: * SYNC * SYNC DATA DATA DATA SYNC SYNC SYNC SYNC DATA DATA DATA
: * SYNC SYNC DATA DATA DATA SYNC SYNC
: *
: * THE RECEIVER SHOULD IGNORE THE FIRST STRING OF DATA CHARACTERS, USE
: * THE NEXT TWO SYNC CHARACTERS FOR SYNCHRONIZATION, THEN PASS THE REST
: * OF THE MESSAGE (7 SYNC AND 6 DATA CHARACTERS) THROUGH RXDB REGISTER.
: *
: *****

```

```

:
: BGNTST
:
: T29::
7303 035100
7304 035100 004737 005420 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
7305
7306 035104 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
7307 035110 043626 DDCMP!NOCHK!SYNCH ;SET DDCMP,NO CHECK,SYNCH=226
7308 035112 040000 NORXEN ;USE 8 BIT CHARS; LEAVE RXEN=0
7309 035114 103003 BCC .+8. ;BR IF NO ERROR
7310 035116 ERROR ;REPORT STACKED ERROR
7311 035116 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
7311 035120 104410 TRAP C$ESCAPE
7311 035122 000414 .WORD L10100-.
7312
7313 035124 004537 010010 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTER
7314 035130 000001 TSOM ;(IGNORED BY RECEIVER)
7315 035132 000007 7.
7316 035134 004537 010010 JSR R5,TXCTRL ;CLEAR TSOM
7317 035140 000000 000
7318 035142 000000 0
7319 035144 004537 003730 JSR R5,WRITEI ;ENABLE RECEIVER (RXEN => 1)
7320 035150 120000 VIAORB
7321 035152 000142 RXEN!TXEN!TTLOOP
7322
7323 ; THE RECEIVER SHOULD IGNORE THE NEXT STRING OF CHARACTERS
7324
7325 035154 004537 007676 JSR R5,TXCHAR ;LOAD 000, TX 2ND SYNCH
7326 035160 000000 000
7327 035162 000010 8.
7328 035164 103003 BCC .+8. ;BR IF NO ERROR
7329 035166 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
7330 035166 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
7330 035170 104410 TRAP C$ESCAPE
7330 035172 000344 .WORD L10100-.
7331
7332 035174 004537 007676 JSR R5,TXCHAR ;LOAD 377, TX 000
7333 035200 000377 377

```

TEST 29 -- BCP RX SYNC CHARACTER RECOGNITION

7334	035202	000010		8.				
7335	035204	103003		BCC	.,+8.	;BR IF NO ERROR		
7336	035206			ERROR		;REPORT STACKED ERROR		
	035206	104460					TRAP	C\$ERROR
7337	035210			ESCAPE	TST	;SKIP TO END OF TEST		
	035210	104410					TRAP	C\$ESCAPE
	035212	000324					.WORD	L10100-.
7338								
7339	035214	004537	007676	JSR	R5, TXCHAR	;LOAD 125, TX 377		
7340	035220	000125		125				
7341	035222	000010		8.				
7342	035224	103003		BCC	.,+8.	;BR IF NO ERROR		
7343	035226			ERROR		;REPORT STACKED ERROR		
	035226	104460					TRAP	C\$ERROR
7344	035230			ESCAPE	TST	;SKIP TO END OF TEST		
	035230	104410					TRAP	C\$ESCAPE
	035232	000304					.WORD	L10100-.
7345								
7346	035234	004537	007676	JSR	R5, TXCHAR	;TX 125, LOAD SYNCH		
7347	035240	000226		SYNCH				
7348	035242	000010		8.				
7349	035244	103003		BCC	.,+8.	;BR IF NO ERROR		
7350	035246			ERROR		;REPORT STACKED ERROR		
	035246	104460					TRAP	C\$ERROR
7351	035250			ESCAPE	TST	;SKIP TO END OF TEST		
	035250	104410					TRAP	C\$ESCAPE
	035252	000264					.WORD	L10100-.
7352								
7353	035254	004537	006176	JSR	R5, CKRDA	;CHECK RECEIVE DATA AVAILABLE		
7354	035260	000000		0		; (NO DATA EXPECTED)		
7355	035262	103003		BCC	.,+8.	;BR IF NO ERROR		
7356	035264			ERROR		;REPORT STACKED ERROR		
	035264	104460					TRAP	C\$ERROR
7357	035266			ESCAPE	TST	;SKIP TO END OF TEST		
	035266	104410					TRAP	C\$ESCAPE
	035270	000246					.WORD	L10100-.
7358								
7359								
7360								
7361								
7362	035272	004537	007676	JSR	R5, TXCHAR	;LOAD 2ND SYNCH, TX 1ST SYNCH		
7363	035276	000226		SYNCH				
7364	035300	000010		8.				
7365	035302	103003		BCC	.,+8.	;BR IF NO ERROR		
7366	035304			ERROR		;REPORT STACKED ERROR		
	035304	104460					TRAP	C\$ERROR
7367	035306			ESCAPE	TST	;SKIP TO END OF TEST		
	035306	104410					TRAP	C\$ESCAPE
	035310	000226					.WORD	L10100-.
7368								
7369	035312	004537	007676	JSR	R5, TXCHAR	;LOAD 3ND SYNCH, TX 2ND SYNCH		
7370	035316	000226		SYNCH				
7371	035320	000010		8.				
7372	035322	103003		BCC	.,+8.	;BR IF NO ERROR		
7373	035324			ERROR		;REPORT STACKED ERROR		
	035324	104460					TRAP	C\$ERROR
7374	035326			ESCAPE	TST	;SKIP TO END OF TEST		

; THE RECEIVER SHOULD SYNCHRONIZE ON THE NEXT TWO SYNC CHARACTERS
; AND THEN READ THE REMAINING ONES.

TEST 29 -- BCP RX SYNC CHARACTER RECOGNITION

```

7415 035470 004537 010110      JSR      R5,RXCHAR      ;CLOCK/RECEIVE/CHECK PREVIOUS CHARACTER
7416 035474 000000      20$:    000            ;** HOLE FOR EXPECTED CHARACTER
7417 035476 000000      0
7418 035500 000010      8.
7419 035502 103003      BCC     .+8.          ;BR IF NO ERROR
7420 035504      ERROR            ;REPORT STACKED ERROR
       035504 104460
7421 035506      ESCAPE  TST          ;SKIP TO END OF TEST              TRAP   C$ERROR
       035506 104410      .WORD   L10100-.
       035510 000026
7422
7423 035512 022702 035555      CMP     $TXEND1,R2
7424 035516 001347      BNE     5$
7425      ;-----;
7426 035520 004537 011532      JSR      R5,ENDTRN     ;SHUT DOWN TRANSMITTER, RECEIVER
7427 035524 000010      8.
7428 035526 103003      BCC     .+8.          ;BR IF NO ERROR
7429 035530      ERROR            ;REPORT STACKED ERROR
       035530 104460      TRAP   C$ERROR
7430 035532      ESCAPE  TST          ;SKIP TO END OF TEST              TRAP   C$ESCAPE
       035532 104410      .WORD   L10100-.
       035534 000002
7431
7432 035536      ENDTST
       035536      L10100:
       035536 104401      TRAP   C$ETST
7433      ;-----;
7434 035540      226      TXTBL1: .BYTE 226      ;SYNCH
7435 035541      226      .BYTE 226      ;SYNCH
7436 035542      000      .BYTE 000
7437 035543      377      .BYTE 377
7438 035544      125      .BYTE 125
7439 035545      226      .BYTE 226      ;SYNCH
7440 035546      226      .BYTE 226      ;SYNCH
7441 035547      252      .BYTE 252
7442 035550      101      .BYTE 101
7443 035551      202      .BYTE 202
7444 035552      226      .BYTE 226      ;SYNCH
7445 035553      226      .BYTE 226      ;SYNCH
7446 035554      000      .BYTE 000
7447 035555      000      TXEND1: .BYTE 000
7448      .EVEN
7449      ;-----;

```

TEST 30 -- BCP RX STRIP-SYNC TEST

7464

.SBTTL TEST 30 -- BCP RX STRIP-SYNC TEST

```

;*****
;*
;* TEST 30 -- BCP RX STRIP-SYNC TEST
;*
;* THE USYRT IS INITIALIZED WITH THE STRIP-SYNC CONTROL BIT ASSERTED.
;* THE FOLLOWING MESSAGE IS THEN INITIATED WITHOUT ASSERTING RXE AND
;* ONCE THE DATA IS BEING TRANSMITTED, RXE IS ASSERTED (IE: *):
;*
;* SYNC * SYNC DATA DATA DATA SYNC SYNC SYNC SYNC SYNC DATA DATA DATA
;* SYNC SYNC DATA DATA DATA SYNC SYNC
;*
;* THE RECEIVER SHOULD IGNORE THE FIRST STRING OF DATA CHARACTERS, USE
;* THE NEXT TWO SYNC CHARACTERS FOR SYNCHRONIZATION, IGNORE THE NEXT
;* THREE SYNC CHARACTERS, AND PASS THE REST OF THE MESSAGE (4 SYNC AND
;* 6 DATA CHARACTERS) THROUGH RXDB REGISTER.
;*
;*****

```

```

7465 035556 004737 005420
7466
7467 035562 004537 007400
7468 035566 063626
7469 035570 040000
7470 035572 103003
7471 035574
035574 104460
7472 035576
035576 104410
035600 000474
7473
7474 035602 004537 010010
7475 035606 000001
7476 035610 000007
7477 035612 004537 010010
7478 035616 000000
7479 035620 000000
7480 035622 004537 003734
7481 035626 120000
7482 035630 000142
7483
7484
7485
7486 035632 004537 007676
7487 035636 000000
7488 035640 000010
7489 035642 103003
7490 035644
035644 104460
7491 035646
035646 104410
035650 000424
7492
7493 035652 004537 007676

```

```

;
; BGNTST
;
; JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T30::
;
; JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
DDCMP!STRIPS!NOCHK!SYNCH ;SET DDCMP,NO CHECK,SYNCH=226
NORXEN ;USE 8 BIT CHARS; LEAVE RXEN=0
BCC ,+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR TRAP C$ERROR
;
; ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
; .WORD L10101-.
;
; JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTER
TSOM ; (IGNORED BY RECEIVER)
7.
; JSR R5,TXCTRL ;CLEAR TSOM
000
0
; JSR R5,WRITEI ;ENABLE RECEIVER (RXEN => 1)
VIAORB
RXEN!TXEN!TTLOOP
;-----
; THE RECEIVER SHOULD IGNORE THE NEXT STRING OF CHARACTERS
;-----
; JSR R5,TXCHAR ;LOAD 000, TX 2ND SYNCH
000
8.
BCC ,+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR TRAP C$ERROR
;
; ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
; .WORD L10101-.
;
; JSR R5,TXCHAR ;LOAD 377, TX 000

```

TEST 30 - BCP RX STRIP-SYNC TEST

7494	035656	000377		377				
7495	035660	000010		8.				
7496	035662	103003		BCC	+.8.		;BR IF NO ERROR	
7497	035664			ERROR			;REPORT STACKED ERROR	
	035664	104460						TRAP C\$ERROR
7498	035666			ESCAPE	TST		;SKIP TO END OF TEST	TRAP C\$ESCAPE
	035666	104410						.WORD L10101-
	035670	000404						
7499								
7500	035672	004537	007676	JSR	R5, TXCHAR		;LOAD 125, TX 377	
7501	035676	000125		125				
7502	035700	000010		8.				
7503	035702	103003		BCC	+.8.		;BR IF NO ERROR	
7504	035704			ERROR			;REPORT STACKED ERROR	
	035704	104460						TRAP C\$ERROR
7505	035706			ESCAPE	TST		;SKIP TO END OF TEST	TRAP C\$ESCAPE
	035706	104410						.WORD L10101-
	035710	000364						
7506								
7507	035712	004537	007676	JSR	R5, TXCHAR		;LOAD SYNCH, TX 125	
7508	035716	000226		SYNCH				
7509	035720	000010		8.				
7510	035722	103003		BCC	+.8.		;BR IF NO ERROR	
7511	035724			ERROR			;REPORT STACKED ERROR	
	035724	104460						TRAP C\$ERROR
7512	035726			ESCAPE	TST		;SKIP TO END OF TEST	TRAP C\$ESCAPE
	035726	104410						.WORD L10101-
	035730	000344						
7513								
7514	035732	004537	006176	JSR	R5, CKRDA		;CHECK RECEIVE DATA AVAILABLE	
7515	035736	000000		0			; (NO DATA EXPECTED)	
7516	035740	103003		BCC	+.8.		;BR IF NO ERROR	
7517	035742			ERROR			;REPORT STACKED ERROR	
	035742	104460						TRAP C\$ERROR
7518	035744			ESCAPE	TST		;SKIP TO END OF TEST	TRAP C\$ESCAPE
	035744	104410						.WORD L10101-
	035746	000326						
7519								
7520								
7521								
7522								
7523	035750	004537	007676	JSR	R5, TXCHAR		;LOAD 2ND SYNCH, TX 1ST SYNCH	
7524	035754	000226		SYNCH				
7525	035756	000010		8.				
7526	035760	103003		BCC	+.8.		;BR IF NO ERROR	
7527	035762			ERROR			;REPORT STACKED ERROR	
	035762	104460						TRAP C\$ERROR
7528	035764			ESCAPE	TST		;SKIP TO END OF TEST	TRAP C\$ESCAPE
	035764	104410						.WORD L10101-
	035766	000306						
7529								
7530	035770	004537	007676	JSR	R5, TXCHAR		;LOAD 3ND SYNCH, TX 2ND SYNCH	
7531	035774	000226		SYNCH				
7532	035776	000010		8.				
7533	036000	103003		BCC	+.8.		;BR IF NO ERROR	
7534	036002			ERROR			;REPORT STACKED ERROR	
	036002	104460						TRAP C\$ERROR

: THE RECEIVER SHOULD SYNCHRONIZE ON THE NEXT TWO SYNC CHARACTERS,
: STRIP THE NEXT THREE, AND THEN READ THE REMAINING ONES.

TEST 30 BCP RX STRIP SYNC TEST

7535	036004			ESCAPE	TST	;SKIP TO END OF TEST		
	036004	104410					TRAP	C#ESCAPE
	036006	000266					.WORD	L10101-
7536								
7537	036010	004537	007676	JSR	R5, TXCHAR	;LOAD 4TH SYNCH, TX 3RD SYNCH		
7538	036014	000226		SYNCH				
7539	036016	000010		8.				
7540	036020	103003		BCC	.+8.	;BR IF NO ERROR		
7541	036022			ERROR		;REPORT STACKED ERROR		
	036022	104460					TRAP	C#ERROR
7542	036024			ESCAPE	TST	;SKIP TO END OF TEST		
	036024	104410					TRAP	C#ESCAPE
	036026	000246					.WORD	L10101-
7543								
7544	036030	004537	007676	JSR	R5, TXCHAR	;LOAD 5TH SYNCH, TX 4TH SYNCH		
7545	036034	000226		SYNCH				
7546	036036	000010		8.				
7547	036040	103003		BCC	.+8.	;BR IF NO ERROR		
7548	036042			ERROR		;REPORT STACKED ERROR		
	036042	104460					TRAP	C#ERROR
7549	036044			ESCAPE	TST	;SKIP TO END OF TEST		
	036044	104410					TRAP	C#ESCAPE
	036046	000226					.WORD	L10101-
7550								
7551	036050	004537	007676	JSR	R5, TXCHAR	;LOAD DATA1 ,TX 5TH SYNCH		
7552	036054	000252		252				
7553	036056	000010		8.				
7554	036060	103003		BCC	.+8.	;BR IF NO ERROR		
7555	036062			ERROR		;REPORT STACKED ERROR		
	036062	104460					TRAP	C#ERROR
7556	036064			ESCAPE	TST	;SKIP TO END OF TEST		
	036064	104410					TRAP	C#ESCAPE
	036066	000206					.WORD	L10101-
7557								
7558	036070	004537	007676	JSR	R5, TXCHAR	;LOAD DATA2 ,TX DATA1		
7559	036074	000347		347				
7560	036076	000010		8.				
7561	036100	103003		BCC	.+8.	;BR IF NO ERROR		
7562	036102			ERROR		;REPORT STACKED ERROR		
	036102	104460					TRAP	C#ERROR
7563	036104			ESCAPE	TST	;SKIP TO END OF TEST		
	036104	104410					TRAP	C#ESCAPE
	036106	000166					.WORD	L10101-
7564								
7565	036110	004537	007676	JSR	R5, TXCHAR	;LOAD DATA3		
7566	036114	000030		030				
7567	036116	000000		0				
7568	036120	103003		BCC	.+8.	;BR IF NO ERROR		
7569	036122			ERROR		;REPORT STACKED ERROR		
	036122	104460					TRAP	C#ERROR
7570	036124			ESCAPE	TST	;SKIP TO END OF TEST		
	036124	104410					TRAP	C#ESCAPE
	036126	000146					.WORD	L10101-
7571								
7572	036130	004537	011364	JSR	R5, RCV151	;CLOCK AND RECEIVE DATA1		
7573	036134	000000		0				
7574	036136	103003		BCC	.+8.	;BR IF NO ERROR		

TEST 30 BCF RX STRIP SYNC TEST

```

7575 036140          ERROR          ;REPORT STACKED ERROR
      036140 104460
7576 036142          ESCAPE TST      ;SKIP TO END OF TEST
      036142 104410          TRAP     C$ERROR
      036144 000130          .WORD   L10101-.
7577
7578 036146 004557 010110          JSR     R5,RXCHAR      ;RECEIVE/CHECK DATA1 (252)
7579 036152 000252          252
7580 036154 000000          0
7581 036156 000010          8.
7582 036160 103003          BCC    .+8.          ;BR IF NO ERROR
7583 036162          ERROR          ;REPORT STACKED ERROR
      036162 104460          TRAP     C$ERROR
7584 036164          ESCAPE TST      ;SKIP TO END OF TEST
      036164 104410          TRAP     C$ESCAPE
      036166 000106          .WORD   L10101-.
7585
7586 036170 012702 036276          ;-----
7587 036174 112237 036232          5$:   MOV     @TXIBL2,R2    ;SET UP TABLE POINTER
7588 036200 116237 000001 036212  MOVB   (R2)+,20$     ;SETUP EXPECTED CHARACTER
7589                                MOVB   1(R2),10$     ;SETUP TRANSMIT CHARACTER
7590 036206 004537 007676          JSR     R5, TXCHAR    ;LOAD A CHARACTER
7591 036212 000000          10$:  000
7592 036214 000000          0
7593 036216 103003          BCC    .+8.          ;BR IF NO ERROR
7594 036220          ERROR          ;REPORT STACKED ERROR
      036220 104460          TRAP     C$ERROR
7595 036222          ESCAPE TST      ;SKIP TO END OF TEST
      036222 104410          TRAP     C$ESCAPE
      036224 000050          .WORD   L10101-.
7596
7597 036226 004537 010110          JSR     R5,RXCHAR    ;CLOCK/RECEIVE/CHECK PREVIOUS CHARACTER
7598 036232 000000          20$:  000
7599 036234 000000          0
7600 036236 000010          8.
7601 036240 103003          BCC    .+8.          ;BR IF NO ERROR
7602 036242          ERROR          ;REPORT STACKED ERROR
      036242 104460          TRAP     C$ERROR
7603 036244          ESCAPE TST      ;SKIP TO END OF TEST
      036244 104410          TRAP     C$ESCAPE
      036246 000026          .WORD   L10101-.
7604
7605 036250 022702 036310          CMP     @TXEND2,R2
7606 036254 001347          BNE    5$
7607
7608 036256 004537 011532          ;-----
7609 036262 000010          JSR     R5,ENDTRN    ;SHUT DOWN TRANSMITTER, RECEIVER
7610 036264 103003          8.
7611 036266          BCC    .+8.          ;BR IF NO ERROR
      036266 104460          ERROR          ;REPORT STACKED ERROR
      036266 104460          TRAP     C$ERROR
7612 036270          ESCAPE TST      ;SKIP TO END OF TEST
      036270 104410          TRAP     C$ESCAPE
      036272 000002          .WORD   L10101-.
7613 036274          ENDTST
      036274          L10101:
7614          ;-----
      104401          TRAP     C$ETST

```

TEST 30 BCP RX STRIP-SYNC TEST

7615	036276	347	TXBL2:	.BYTE	347		
7616	036277	030		.BYTE	030		
7617	036300	226		.BYTE	226	!SYNCH	
7618	036301	226		.BYTE	226	!SYNCH	
7619	036302	252		.BYTE	252		
7620	036303	101		.BYTE	101		
7621	036304	202		.BYTE	202		
7622	036305	226		.BYTE	226	!SYNCH	
7623	036306	226		.BYTE	226	!SYNCH	
7624	036307	000		.BYTE	000		
7625	036310	000	TXEND2:	.BYTE	000		
7626				.EVEN			
7627			;-----				

TEST 31 BCP RX LOST RXE TEST

7635

..SBTTL TEST 31 -- BCP RX LOST RXE TEST

```

:*****
:
: TEST 31 -- BCP RX LOST RXE TEST
:
: THE USYRT IS INITIALIZED (CRC16,STRIPS,BCP MODE) AND A MESSAGE IS STARTED.
: WHILE IN THE MIDDLE OF TEXT, RXE IS DROPPED AND THE REACTION OF THE
: RECEIVER IS MONITORED.
:
:*****

```

```

:
: BGNTST
:
: T31::
7636 036312 004737 005420 JSR PC,INIDMV ;INIT DMV-11. ENTER M-LOOP
7637
7638 036316 004537 007400 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
7639 036322 065626 DDCMP!STRIPS!IDLES!CRC16!SYNCH ;SET DDCMP, STRIP, IDLE, CRC-16, SYNCH=226
7640 036324 000000 0 ;USE 3 BIT CHARS
7641 036326 103003 BCC .+8. ;BR IF NO ERROR
7642 036330 ERROR ;REPORT STACKED ERROR
7643 036332 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
7644 036332 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
7645 036334 000334 .WORD L10102-.
7646
7647 036336 004537 007676 JSR R5, TXCHAR ;LOAD 2ND SYNCH, TX 1ST SYNCH
7648 036342 000226 SYNCH
7649 036344 000007 7.
7650 036346 103003 BCC .+8. ;BR IF NO ERROR
7651 036350 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
7652 036352 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
7653 036354 000314 .WORD L10102-.
7654
7655 036356 004537 010010 JSR R5, TXCTRL ;CLEAR TSOM
7656 036362 000000 000
7657 036364 000000 0
7658 036366 004537 007676 JSR R5, TXCHAR ;LOAD 000, TX 2ND SYNCH
7659 036372 000000 000
7660 036374 000010 8.
7661 036376 103003 BCC .+8. ;BR IF NO ERROR
7662 036400 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
7663 036402 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
7664 036402 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
7665 036404 000264 .WORD L10102-.
7666
7667 036406 004537 007676 JSR R5, TXCHAR ;LOAD 125, TX 000
7668 036412 000125 125
7669 036414 000010 8.
7670 036416 103003 BCC .+8. ;BR IF NO ERROR
7671 036420 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
7672 036422 104460 ESCAPE TST ;SKIP TO END OF TEST

```

TEST 31 BCP RX LOST RXE TEST

	036422	104410					TRAP	C\$ESCAPE
	036424	000244					.WORD	L10102-.
7668								
7669	036426	004537	007676	JSR	R5, TXCHAR	;LOAD 252, TX 125		
7670	036432	000252			252			
7671	036434	000010			8.			
7672	036436	103003		BCC	.+8.	;BR IF NO ERROR		
7673	036440			ERROR		;REPORT STACKED ERROR		
	036440	104460					TRAP	C\$ERROR
7674	036442			ESCAPE	TST	;SKIP TO END OF TEST		
	036442	104410					TRAP	C\$ESCAPE
	036444	000224					.WORD	L10102-.
7675								
7676	036446	004537	007676	JSR	R5, TXCHAR	;LOAD 377, TX 252		
7677	036452	000377			377			
7678	036454	000010			8.			
7679	036456	103003		BCC	.+8.	;BR IF NO ERROR		
7680	036460			ERROR		;REPORT STACKED ERROR		
	036460	104460					TRAP	C\$ERROR
7681	036462			ESCAPE	TST	;SKIP TO END OF TEST		
	036462	104410					TRAP	C\$ESCAPE
	036464	000204					.WORD	L10102-.
7682								
7683	036466	004537	007676	JSR	R5, TXCHAR	;LOAD 000		
7684	036472	000000			000			
7685	036474	000000			0			
7686	036476	103003		BCC	.+8.	;BR IF NO ERROR		
7687	036500			ERROR		;REPORT STACKED ERROR		
	036500	104460					TRAP	C\$ERROR
7688	036502			ESCAPE	TST	;SKIP TO END OF TEST		
	036502	104410					TRAP	C\$ESCAPE
	036504	000164					.WORD	L10102-.
7689								
7690	036506	004537	011364	JSR	R5, RCV1ST	;CLOCK AND RCV 000		
7691	036512	000000			0			
7692	036514	103003		BCC	.+8.	;BR IF NO ERROR		
7693	036516			ERROR		;REPORT STACKED ERROR		
	036516	104460					TRAP	C\$ERROR
7694	036520			ESCAPE	TST	;SKIP TO END OF TEST		
	036520	104410					TRAP	C\$ESCAPE
	036522	000146					.WORD	L10102-.
7695								
7696	036524	004537	010110	JSR	R5, RXCHAR	;READ & CHK 000, RCV 125		
7697	036530	000000			000			
7698	036532	000000			0			
7699	036534	000010			8.			
7700	036536	103003		BCC	.+8.	;BR IF NO ERROR		
7701	036540			ERROR		;REPORT STACKED ERROR		
	036540	104460					TRAP	C\$ERROR
7702	036542			ESCAPE	TST	;SKIP TO END OF TEST		
	036542	104410					TRAP	C\$ESCAPE
	036544	000124					.WORD	L10102-.
7703								
7704								
7705								
7706								
7707	036546	004537	003476	JSR	R5, READ	;READ VALUE OF VIAORB REGISTER		

; TX AND RX NOW SYNC'D. ONE CHARACTER HAS BEEN READ/VERIFIED....
; NOW CLEAR RXEN (RECEIVER ENABLE).

(15)

TEST 31 -- BCP RX LOST RXE TEST

```

7708 036552 120000 VIAORB ; AND PUT IT IN LOCATION PROVIDED
7709 036554 036602 CGORB ; BELOW (CGORB).
7710 036556 103003 BCC .+8. ;BR IF NO ERROR
7711 036560 ERROR ;REPORT ERROR
;
; TRAP C$ERROR
7712 036562 ESCAPE TST
; TRAP C$ESCAPE
; 036562 104410 .WORD L10102-.
; 036564 000104
7713
7714 036566 042737 000100 036602 BIC @RXEN,CGORB ;CLEAR RXEN BIT OF ORB STATUS WORD
7715
7716 036574 004537 003734 JSR R5,WRITEI ;WRITE NEW STATUS WORD (W/RXEN=0)
7717 036600 120000 VIAORB ;BACK INTO VIAORB
7718 036602 000000 CGORB: .WORD 0 ;ACTUAL DATA WRITTEN TO VIAORB
7719 036604 103003 BCC .+8. ;BR IF NO ERROR
7720 036606 ERROR ;REPORT ERROR
; TRAP C$ERROR
; 036606 104460
7721 036610 ESCAPE TST
; TRAP C$ESCAPE
; 036610 104410 .WORD L10102-.
; 036612 000056
7722
7723 ;-----
7724 ; RXEN NOW CLEARED CHECK USYRT STATUS, TX ANOTHER CHARACTER, AND
7725 ; CHECK USYRT STATUS AGAIN.
;-----
7726 036614 004537 005432 JSR R5,CKUSTS ;CHK USYRT STATUS FOR PROPER STATE
7727 036620 000104 TXACT!TBMT ;TXACT = 1, TBMT = 1
7728 036622 103003 BCC .+8. ;BR IF NO ERROR
7729 036624 ERROR
; TRAP C$ERROR
; 036624 104460
7730 036626 ESCAPE TST
; TRAP C$ESCAPE
; 036626 104410 .WORD L10102-.
; 036630 000040
7731
7732 036632 004537 007676 JSR R5,TXCHAR ;LOAD/TRANSMIT 303
7733 036636 000303 303
7734 036640 100010 NCTBMT+256.!8.
7735 036642 103003 BCC .+8. ;BR IF NO ERROR
7736 036644 ERROR ;REPORT STACKED ERROR
; TRAP C$ERROR
; 036644 104460
7737 036646 ESCAPE TST ;SKIP TO END OF TEST
; TRAP C$ESCAPE
; 036646 104410 .WORD L10102-.
; 036650 000020
7738
7739 036652 004537 005432 JSR R5,CKUSTS ;CHK USYRT STATUS FOR PROPER STATE
7740 036656 000114 TXACT!TSO!TBMT ;TXACT = 1
7741 036660 103003 BCC .+8. ;BR IF NO ERROR
7742 036662 ERROR
; TRAP C$ERROR
; 036662 104460
7743 036664 ESCAPE TST
; TRAP C$ESCAPE
; 036664 104410 .WORD L10102-.
; 036666 000002
7744 036670 ENDTST
; L10102: TRAP C$ETST
; 036670 104401

```

HARDWARE PARAMETER CODING SECTION

.SBTTL HARDWARE PARAMETER CODING SECTION

```

7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759 036672          BGNHRD
      036672 000041
      036674
                                     .WORD L10103-L$HARD/2
                                     L$HARD::
7760
7761 036674          GPRMA  ADDRES,0,0,160020,177776,YES
      036674 000031
      036676 036776
      036700 160020
      036702 177776
                                     .WORD  T$CODE
                                     .WORD  ADDRES
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM
7762 036704          GPRMA  VECTOR,2,0,0,674,YES
      036704 001031
      036706 037024
      036710 000000
      036712 000674
                                     .WORD  T$CODE
                                     .WORD  VECTOR
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM
7763 036714          GPRMD  PRIRTY,4,0,7000,4,7,YES
      036714 002032
      036716 037055
      036720 007000
      036722 000004
      036724 000007
                                     .WORD  T$CODE
                                     .WORD  PRIRTY
                                     .WORD  7000
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM
7764 036726          GPRMD  SW1.M,6,0,377,0,377,YES
      036726 003032
      036730 037106
      036732 000377
      036734 000000
      036736 000377
                                     .WORD  T$CODE
                                     .WORD  SW1.M
                                     .WORD  377
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM
7765 036740          GPRMD  SW2.M,10,0,377,0,377,YES
      036740 004032
      036742 037147
      036744 000377
      036746 000000
      036750 000377
                                     .WORD  T$CODE
                                     .WORD  SW2.M
                                     .WORD  377
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM
7766 036752          GPRMD  BDTY.M,12,0,7,0,2,YES
      036752 005032
      036754 037211
      036756 000007
      036760 000000
      036762 000002
                                     .WORD  T$CODE
                                     .WORD  BDTY.M
                                     .WORD  7
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM
7767 036764          GPRMD  BR.M,16,0,17,0,1,YES
      036764 007032
      036766 037274
      036770 000017
      036772 000000
      036774 000001
                                     .WORD  T$CODE
                                     .WORD  BR.M
                                     .WORD  17
                                     .WORD  T$L0LIM
                                     .WORD  T$HILIM

```

HARDWARE PARAMETER CODING SECTION

```

7768
7769 036776          ENDHRD
                                .EVEN
                                L10103:
7770
7771
7772 036776          104      105      126  .NLIST  BEX
7773 037024          104      105      126  ADDRES: .ASCIZ /DEVICE CSR ADDRESS : /
7774 037055          104      105      126  VECTOR: .ASCIZ /DEVICE VECTOR ADDRESS : /
7775 037106          123      127      111  PRIRTY: .ASCIZ /DEVICE PRIORITY LEVEL : /
7776 037147          123      127      111  SW1.M:  .ASCIZ /SWITCH PACK # 1 (BOOT ADDRESS): /
7777 037211          102      117      101  SW2.M:  .ASCIZ /SWITCH PACK # 2 (DDCMP ADDRESS): /
7778 037274          102      101      125  BDTY.M: .ASCIZ /BOARD TYPE (0=M8064, 1=M8053-V.35, 2=M8053-EIA) : /
7779
7780          .LIST  BEX
          .EVEN
    
```

116

SOFTWARE PARAMETER CODING SECTION

7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794 037346
037346 000000
037350
7795
7796 037350
037350

.SBTTL SOFTWARE PARAMETER CODING SECTION

```

;////////////////////////////////////
; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
; WITH THE OPERATOR.
;////////////////////////////////////

```

BGNSFT

```

.WORD L10104-L$SOFT/2
L$SOFT::

```

ENDSFT

```

.EVEN
L10104:

```

16

***** PATCH AREA FOR DEBUG *****

7798
7799
7800 037350
7801 037450
7802 037451 000240
7803 037452 000240
7804 037454 000240
7805
7806
7807
7808
7809 037456
7810
7811
7812 037456

 037456 000000
 037460 000000
 037462
7813
7814 000001

.SBTTL ***** PATCH AREA FOR DEBUG *****
PATCH:
 , , +100
 NOP
 NOP
 NOP
;*****
.SBTTL "ENDMOD" STATEMENT
 ENDMOD
.SBTTL "LASTAD" STATEMENT & END OF PROGRAM
 LASTAD

L\$LAST::
.END

.EVEN
.WORD 0
.WORD 0

SYMBOL TABLE

ABC	=	000160	BSR0	002256	C\$ESUB	=	000003	EM3	013211	FLGCA2	=	000001		
ADDRES	=	036776	BSR1	002260	C\$ETST	=	000001	EM30	013535	FLGCB1	=	000020		
ADR	=	000020	BSR10	002276	C\$EXIT	=	000032	EM7	013556	FLGCB2	=	000010		
AD.HIT	=	021566	BSR11	002300	C\$GETB	=	000026	EM	013573	FLGIRQ	=	000200		
AD.OK	=	021562	BSR12	002302	C\$GETW	=	000027	EM	013621	FLGSR	=	000004		
APA	=	000200	BSR13	002304	C\$GMAN	=	000043	EM36	013642	FLGT1	=	000100		
APAD	=	100000	BSR14	002306	C\$GPHR	=	000042	EM39	013657	FLGT2	=	000040		
ASSEMB	=	000010	BSR15	002310	C\$GPL0	=	000030	EM4	013235	FMT10	=	012252		
BADDAT	=	002442	BSR16	002312	C\$GPRI	=	000040	EM40	013701	FMT10A	=	012326		
BDATA	=	002402	BSR17	002314	C\$INIT	=	000011	EM5	013252	FMT11	=	012347		
BDRATE	=	002550	BSR2	002262	C\$INLP	=	000020	EM54	013717	FMT15	=	012366		
BDTY.M	=	037211	BSR3	002264	C\$MANI	=	000050	EM6	013302	FMT15A	=	012420		
BIT0	=	000001	BSR4	002266	C\$MEM	=	000031	EM66	013741	FMT19	=	012472		
BIT00	=	000001	BSR5	002270	C\$MSG	=	000023	EM68	013760	FMT2	=	011650		
BIT01	=	000002	BSR6	002272	C\$OPEN	=	000034	EM69	014007	FMT21	=	012523		
BIT02	=	000004	BSR7	002274	C\$PNTB	=	000014	EM70	014025	FMT22	=	012533		
BIT03	=	000010	CARRIER	=	000100	C\$PNTF	=	000017	EM71	014047	FMT23	=	012555	
BIT04	=	000020	CA1CTL	=	000001	C\$PNTS	=	000016	EM72	014065	FMT24	=	012634	
BIT05	=	000040	CA2CTL	=	000016	C\$PNTX	=	000015	EM73	014107	FMT25	=	012647	
BIT06	=	000100	CB1CTL	=	000020	C\$QIO	=	000377	EM74	014124	FMT26	=	012677	
BIT07	=	000200	CB2CTL	=	000340	C\$ROBU	=	000007	EM75	014145	FMT27	=	012732	
BIT08	=	000400	CGORB	=	036602	C\$REFG	=	000047	EM76	014161	FMT29	=	012741	
BIT09	=	001000	CHKTSO	=	007116	C\$RESE	=	000033	EM77	014201	FMT3	=	011705	
BIT1	=	000002	CHPTYP	=	002454	C\$REVI	=	000003	EM78	014215	FMT30	=	012746	
BIT10	=	002000	CKRACT	=	005676	C\$RFLA	=	000021	EM90	014235	FMT31	=	013003	
BIT11	=	004000	CKRDA	=	006176	C\$RPT	=	000025	EM91	014266	FMT32	=	013051	
BIT12	=	010000	CKROR	=	006476	C\$SEFG	=	000046	EM92	014323	FMT4	=	011771	
BIT13	=	020000	CKRSA	=	006336	C\$SPRI	=	000041	EM98	014357	FMT4A	=	012027	
BIT14	=	040000	CKSEOM	=	006636	C\$SVEC	=	000037	EM99	014404	FMT4B	=	012062	
BIT15	=	100000	CKTACT	=	005536	C\$TPRI	=	000013	ENDEMB	=	011640	FMT4C	=	012067
BIT2	=	000004	CKTBMT	=	006036	DDCMP	=	040000	ENDIT	=	021446	FMT5	=	012122
BIT3	=	000010	CKUSTS	=	005432	DEVMAP	=	002460	ENDPAT	=	003105	FMT5A	=	012165
BIT4	=	000020	CRC16	=	001400	DEVPTR	=	002462	ENDTRN	=	011532	FRSTIM	=	002444
BIT5	=	000040	CTS	=	000010	DFPTBL	=	002224	ERRBLK	=	002254	F\$AU	=	000015
BIT6	=	000100	C\$AU	=	000052	DIAGMC	=	000000	ERRFLC	=	002424	F\$AUTO	=	000020
BIT7	=	000200	C\$AUTO	=	000061	DOTBMT	=	000007	ERRMSG	=	002252	F\$BGN	=	000040
BIT8	=	000400	C\$BRK	=	000022	DTR	=	000020	ERRNBR	=	002250	F\$CLEA	=	000007
BIT9	=	001000	C\$BSEG	=	000004	DTRL	=	000000	ERROR1	=	002452	F\$DU	=	000016
BOE	=	000400	C\$BSUB	=	000002	D.BUG	=	000000	ERRTYP	=	002246	F\$END	=	000041
BRDTYP	=	002544	C\$CEFG	=	000045	EF.CON	=	000036	ERR10	=	017372	F\$HARD	=	000004
BR.M	=	037274	C\$CLCK	=	000062	EF.NEW	=	000035	ERR11	=	017546	F\$HW	=	000013
BSELO	=	002472	C\$CLEA	=	000012	EF.PWR	=	000034	ERR11\$	=	020356	F\$INIT	=	000006
BSEL1	=	002474	C\$CLOS	=	000035	EF.RES	=	000037	ERR12	=	017722	F\$JMP	=	000050
BSEL10	=	002512	C\$CLP1	=	000006	EF.STA	=	000040	ERR12\$	=	020710	F\$MOD	=	000000
BSEL11	=	002514	C\$CVEC	=	000036	EIAV35	=	000002	ERR13	=	020036	F\$MSG	=	000011
BSEL12	=	002516	C\$DCLN	=	000044	EM1	=	013103	ERR4	=	017030	F\$PROT	=	000021
BSEL13	=	002520	C\$DODU	=	000051	EM100	=	014425	ERR5	=	017154	F\$PWR	=	000017
BSEL14	=	002522	C\$DRPT	=	000024	EM101	=	014445	ERR5\$	=	020160	F\$RPT	=	000012
BSEL15	=	002524	C\$DU	=	000053	EM102	=	014471	ERR7A	=	017252	F\$SEG	=	000003
BSEL16	=	002526	C\$EDIT	=	000003	EM14	=	013332	EVL	=	000004	F\$SOFT	=	000005
BSEL17	=	002530	C\$ERDF	=	000055	EM16	=	013346	EVRC	=	002400	F\$SRV	=	000010
BSEL2	=	002476	C\$ERHR	=	000056	EM2	=	013142	EXADD	=	000020	F\$SUB	=	000002
BSEL3	=	002500	C\$ERRO	=	000060	EM25	=	013366	EXCON	=	000010	F\$SW	=	000014
BSEL4	=	002502	C\$ERSF	=	000054	EM26	=	013414	EXECUT	=	000005	F\$TEST	=	000001
BSEL5	=	002504	C\$ERSO	=	000057	EM27	=	013446	E\$END	=	002100	GDATA	=	002400
BSEL6	=	002506	C\$ESCA	=	000010	EM28	=	013477	E\$LOAD	=	000035	GETBSR	=	004046
BSEL7	=	002510	C\$ESEG	=	000005	EM29	=	013520	FLGCA1	=	000002	GETPRM	=	021312

SYMBOL TABLE

GETURS	004402	I\$SRV	=	000041	L\$SPCP	002020	G	L10063	033522	PATL	003044			
GETVRS	004502	I\$SUR	=	000041	L\$SPTP	002024	G	L10064	031202	PATQ	003065			
GETWSR	004210	I\$TST	=	000041	L\$STA	002030	G	L10065	031570	PATQB	003075			
GOODAT	002440	J\$JMP	=	000167	L\$SW	002246	G	L10066	032156	PBLENB	=	000002		
G\$CNT0	=	LOADAT		002436	L\$TEST	002114	G	L10067	032544	PCR	=	120407		
G\$DELM	=	LOE	=	040000	L\$TIML	002014	G	L10070	033132	PCSARH	=	120405		
G\$DISP	=	LOGDEV		002410	L\$UNIT	002012	G	L10071	033520	PCSARL	=	120404		
G\$EXCP	=	LOT	=	000010	L10000	002244		L10072	034110	PNT	=	001000	G	
G\$HILI	=	LUSWI1		002540	L10001	002246		L10073	034460	PRESET	=	000001		
G\$LOLI	=	LUSWI2		002542	L10002	017152		L10074	035076	PRI	=	002000	G	
G\$NO	=	LU1MOD		002000	L10003	017250		L10075	034634	PRIOR		002416		
G\$OFFS	=	L\$ACP		002110	L10004	017370		L10076	034736	PRIPTY		037055		
G\$OFST	=	L\$APT		002036	L10005	017544		L10077	035074	PRI00	=	000000	G	
G\$PRMA	=	L\$AU		021602	L10006	017720		L10100	035536	PRI01	=	000040	G	
G\$PRMD	=	L\$AUT		002070	L10007	020034		L10101	036274	PRI02	=	000100	G	
G\$PRML	=	L\$AUTO		021450	L10010	020132		L10102	036670	PRI03	=	000140	G	
G\$RADA	=	L\$CCP		002106	L10012	021446		L10103	036776	PRI04	=	000200	G	
G\$RADB	=	L\$CLEA		021574	L10013	021564		L10104	037350	PRI05	=	000240	G	
G\$RADD	=	L\$CO		002032	L10014	021574		MCLR	=	000100	PRI06	=	000300	G
G\$RADL	=	L\$DEPO		002011	L10015	021600		MDMRDY	=	000040	PRI07	=	000340	G
G\$RADO	=	L\$DESC		003326	L10016	021602		MLWRI		003744	PROTO	=	000100	
G\$XFER	=	L\$DESP		002076	L10017	021762		MPCSR		002472	PSTACK		002414	
G\$YES	=	L\$DEVP		002060	L10020	022150		MPIVEC		002532	RABGA	=	000004	
HDX	=	L\$DISP		002124	L10021	022046		MPOVEC		002534	RAMADR	=	001000	
HELP	=	L\$DLY		002116	L10022	022146		MPRIOR		002536	RCVBUF		003106	
HOE	=	L\$DTP		002040	L10023	022342		MRDY	=	000200	RCVDAT	=	000002	
IBE	=	L\$DTYP		002034	L10024	022400		MREQ	=	000001	RCV1ST		011364	
IDLE	=	L\$DU		021576	L10025	023146		MSTCLR		003374	RDA	=	000200	
IDLES	=	L\$DUT		002072	L10026	023232		NCRACT	=	020000	RDSRH	=	120401	
IDU	=	L\$DVTY		003306	L10027	023456		NCTBMT	=	000200	RDSRL	=	120400	
IEI	=	L\$EF		002052	L10030	023550		NEWLIN		011645	READ		003476	
IEO	=	L\$ENVI		002044	L10031	023544		NEWST		021272	READI		003610	
IER	=	L\$ERRT		002246	L10032	023642		NFCRDA	=	040000	REDBYT		002432	
INIDMV	005420	L\$ETP		002102	L10033	023636		NOCHK	=	003400	REDDAT		002572	
INITRN	007400	L\$EXP1		002046	L10034	023734		NOCRDA	=	100000	REDLOC	=	000001	
INITT1	004556	L\$EXP4		002064	L10035	023730		NOLOOP	=	001000	REDPAG	=	000003	
INITT2	004756	L\$EXP5		002066	L10036	024064		NORXEN	=	040000	REGNUM		002412	
INTFLG	002422	L\$HARD		036674	L10037	024060		NULCLK	=	000200	REG0		002602	
INTGRL	=	L\$HIME		002120	L10040	024156		OVRC	=	002000	REG1		002604	
INTSC	=	L\$HPCP		002016	L10041	024152		O\$APTS	=	000000	REG2		002606	
ISR	=	L\$HPTP		002022	L10042	024232		O\$AU	=	000001	REG3		002610	
IXE	=	L\$HW		002224	L10043	024306		O\$BGNR	=	000000	REG4		002612	
I\$AU	=	L\$ICP		002104	L10044	024630		O\$BGNS	=	000000	REG5		002614	
I\$AUTO	=	L\$INIT		021100	L10045	024762		O\$DU	=	000001	REG6		002616	
I\$CLN	=	L\$LADP		002026	L10046	025144		O\$ERRT	=	000001	REG7		002610	
I\$DU	=	L\$LAST		037462	L10047	025652		O\$GNSW	=	000000	REOM	=	000002	
I\$HRD	=	L\$LOAD		002100	L10050	026252		O\$POIN	=	000001	RERCHK	=	000001	
I\$INIT	=	L\$LUN		002074	L10051	026460		O\$SETU	=	000000	RERR	=	000200	
I\$MOD	=	L\$MREV		002050	L10052	026722		PALENB	=	000001	RETADR		002430	
I\$MSG	=	L\$NAME		002000	L10053	027434		PATCH		037350	RIN;	=	000200	
I\$PROT	=	L\$PRIO		002042	L10054	027112		PATE		002652	ROR	=	000010	
I\$PTAB	=	L\$PROT		021072	L10055	027262		PATF		002662	RSA	=	000020	
I\$PWR	=	L\$PRT		002112	L10056	027432		PATG		002672	RSOM	=	000001	
I\$RPT	=	L\$REPP		002062	L10057	027642		PATH		002721	RSTCHK		005126	
I\$SEG	=	L\$REV		002010	L10060	030036		PATI		002752	RTSND	=	000010	
I\$SETU	=	L\$SOFT		037350	L10061	030232		PATJ		003013	RUN	=	000200	
I\$SFT	=	L\$SPC		002056	L10062	030610		PATK		003023	RXABGA	=	002000	

CNDMCAO DMV11 LINE U....B1
.....C1
.....D1
.....E1
.....F1
.....G1
.....H1
.....I1
.....J1
.....K1
.....L1
.....M1
.....N1

.....B5
.....C5
.....D5
.....E5
.....F5
.....G5
.....H5
.....I5
.....J5
.....K5
.....L5
.....M5
.....N5

.....B9
.....C9
.....D9
.....E9
.....F9
.....G9
.....H9
.....I9
.....J9
.....K9
.....L9
.....M9
.....N9

.....B13
.....C13
.....D13
.....E13
.....F13
.....G13
.....H13
.....I13
.....J13
.....K13
.....L13
.....M13
.....N13

.....B2
.....C2
.....D2
.....E2
.....F2
.....G2
.....H2
.....I2
.....J2
.....K2
.....L2
.....M2
.....N2

.....B6
.....C6
.....D6
.....E6
.....F6
.....G6
.....H6
.....I6
.....J6
.....K6
.....L6
.....M6
.....N6

.....B10
.....C10
.....D10
.....E10
.....F10
.....G10
.....H10
.....I10
.....J10
.....K10
.....L10
.....M10
.....N10

.....B14
.....C14
.....D14
.....E14
.....F14
.....G14
.....H14
.....I14
.....J14
.....K14
.....L14
.....M14
.....N14

.....B3
.....C3
.....D3
.....E3
.....F3
.....G3
.....H3
.....I3
.....J3
.....K3
.....L3
.....M3
.....N3

.....B7
.....C7
.....D7
.....E7
.....F7
.....G7
.....H7
.....I7
.....J7
.....K7
.....L7
.....M7
.....N7

.....B11
.....C11
.....D11
.....E11
.....F11
.....G11
.....H11
.....I11
.....J11
.....K11
.....L11
.....M11
.....N11

.....B15
.....C15
.....D15
.....E15
.....F15
.....G15
.....H15
.....I15
.....J15
.....K15
.....L15
.....M15
.....N15

.....B4
.....C4
.....D4
.....E4
.....F4
.....G4
.....H4
.....I4
.....J4
.....K4
.....L4
.....M4
.....N4

.....B8
.....C8
.....D8
.....E8
.....F8
.....G8
.....H8
.....I8
.....J8
.....K8
.....L8
.....M8
.....N8

.....B12
.....C12
.....D12
.....E12
.....F12
.....G12
.....H12
.....I12
.....J12
.....K12
.....L12
.....M12
.....N12

.....B16
.....C16
.....D16
.....E16
.....F16
.....G16
.....H16
.....I16
.....J16
.....K16
.....L16
.....M16
.....N16

SYMBOL TABLE

UPBITS 002642	VIAIER 120016	VIAT1D 120007	WSR0 002256	XYZ 000007
UREGS 002316	VIAIFR 120015	VIAT2A 120010	WSR10 002266	X\$AI WA 000000
USTATR 122000	VIAMS 120001	VIAT2B 120011	WSR12 002270	X\$AL5 000040
USYREG 002552	VIAORA 120017	VREGS 002336	WSR14 002272	X\$OFFS 000400
USYRT 120400	VIAORB 120000	WAIT50 005312	WSR16 002274	X\$TRUE 000020
VECTOR 037024	VIAPCR 120014	WRIBYT 002434	WSR2 002260	\$ 000065
VIA 120000	VIASR 120012	WRILOC 000002	WSR4 002262	\$LSTIN 000001
VIAACR 120013	VIAT1A 120004	WRIPAG 000004	WSR6 002264	\$LSTTA 000001
VIADPA 120003	VIAT1B 120005	WRITE 003722	XDATA 002404	\$T 000037
VIADPB 120002	VIAT1C 120006	WRITEI 003734	XORGB 020134	

. ABS. 037462 000
 000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 31208 WORDS (122 PAGES)

DYNAMIC MEMORY: 19748 WORDS (75 PAGES)

ELAPSED TIME: 00:06:36

CNDMCA.BIC,CNDMCA.SEQ/CR/-SP=SVC34.MLB/ML,CNDMCA.P11