

11/21+
DMV-11

DMV11 LINE UNIT DIAG2
CNDMDAO

COPYRIGHT (c) 1981-84
AH-T833A-MC
FICHE 01 OF 01

JUL 1984
digital
Made In USA



4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

.TITLE CNDMDAO DMV11 LINE UNIT DIAG2
.SBTTL PROGRAM DOCUMENT
.REM @

IDENTIFICATION

PRODUCT CODE: AC-T832A-MC
PRODUCT NAME: CNDMDAO DMV-11 LINE UNIT STATIC DIAGNOSTIC PART #2
PRODUCT DATE: APRIL 1984
MAINTAINER: ISS DIAGNOSTICS
AUTHORS: CHRIS BRIENEN
 DAVE HOFFMAN
 RAY MARSHALL
MODIFIED BY: JAKI BERG 9-APR-1984
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 PDP UNIBUS MASSBUS
DEC DECUS DECTAPE

PROGRAM DOCUMENT

47
48
49
50
51
52
53
54
55
56
57
58
59
60
61

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

REV A: ORIGINAL RELEASE BRIENEN, HOFFMAN, MARSHALL 14-JAN-81

REV B: INSTALLED OUTSTANDING PATCHES 11-JUL-83

CVDMD8 => CNDMDA JAKI BERG 9-APR-84

CHANGES WERE MADE TO CVDMD8 TO PRODUCE CNDMDA 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

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

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 XXDP.
 - 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

111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165

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 STATIC DIAGNOSTIC TESTING OF THE VIA, FIFO, AND USYRT (BCP/BOP MODES) ON THESE BOARDS. THE FOLLOWING FUNCTIONS WILL BE PERFORMED: VRC/CRC ERROR DETECTION AND ASSORTED BOP SPECIFIC FUNCTIONS (BIT STUFFING, ABORTS, FLAGS, SECONDARY STATION ADDRESSING, ETC).

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 (CNDMD) SHOULD BE THE FOURTH OF THE FIVE DMV-11 STATIC DIAGNOSTICS TO BE RUN (CNDMA/B/C SHOULD BE RUN FIRST). ERRORS FOUND IN THIS PROGRAM SHOULD BE CORRECTED BEFORE RUNNING THE FINAL LINE UNIT DIAGNOSTIC (CNDME).

PROGRAM DOCUMENT

167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223

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 35 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

224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280

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 XXDP+, 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
CNDMD-A-0
DMV-11 LINE UNIT TESTS - PART 2 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

281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337

6.3.1 START COMMAND

STA(RT)/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:

- HOE 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

338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394

THE FLAGS NAMED OR EQUATED TO 1 ARE SET, THOSE EQUATED TO 0 ARE CLEARED. A FLAG NOT SPECIFIED IS CLEARED. IF THE FLAGS SWITCH IS NOT GIVEN ALL FLAGS ARE CLEARED. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.4 END OF PASS SWITCH (/EOP:<INCR>)

<INCR> IS A DECIMAL NUMBER INDICATING HOW OFTEN (IN TERMS OF PASSES) IT IS DESIRED THAT THE END OF PASS MESSAGE BE PRINTED. THE DEFAULT IS AT THE END OF EVERY PASS. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.5 EFFECT OF START COMMAND

THE EFFECT OF THE START COMMAND IS TO INITIATE THE HARDWARE PARAMETER DIALOGUE, THE SOFTWARE PARAMETER DIALOGUE, AND THEN THE DIAGNOSTIC TESTS THEMSELVES.

THE HARDWARE PARAMETER DIALOGUE COMMENCES WITH THE QUESTION "# UNITS?" TO WHICH THE OPERATOR REPLIES WITH A DECIMAL NUMBER N FROM 1 TO 16. THE TERM "UNIT" REFERS TO THE DEVICE TO WHICH THIS SERIES OF DIAGNOSTICS IS DEDICATED. FOLLOWING THIS ARE THE QUESTIONS WHEREBY THE P-TABLES THEMSELVES WILL BE BUILT. EACH P-TABLE IS A CORE-RESIDENT TABLE CONTAINING ALL THE HARDWARE INFORMATION FOR ONE UNIT. THE OPERATOR MUST SUPPLY N (NUMBER OF UNITS) VALUES FOR EACH QUESTION. HE MAY DO THIS BY GIVING ONE ANSWER TO EACH QUESTION (IN WHICH CASE THE SERIES OF QUESTIONS WILL BE POSED N TIMES) OR BY GIVING N VALUES, SEPARATED BY COMMAS, TO EACH QUESTION (SERIES WILL BE POSED ONCE). EACH QUESTION IS FOLLOWED BY THE RESPONSE RADIX (D FOR DECIMAL, B FOR BINARY, O FOR OCTAL, L FOR YES/NO) IN PARENTHESES AND THE DEFAULT VALUE AFTER THE PARENTHESES.

FOLLOWING THE HARDWARE QUESTIONS ARE THE SOFTWARE QUESTIONS TO BUILD THE SOFTWARE TABLES, WHICH DEFINE THE MODE (QUICK VERIFY ETC.) THAT THE DIAGNOSTIC WILL EXECUTE IN.

WHEN THE QUESTION "# UNITS?" IS ANSWERED, MEMORY STORAGE IS ALLOCATED FOR THE P-TABLES, AND IF THERE IS NOT ENOUGH TO ACCOMMODATE THEM THE MESSAGE "TOO MANY UNITS" IS ISSUED. IN THIS CASE THE DIAGNOSTIC MUST BE EXECUTED MORE THAN ONCE TO TEST ALL UNITS.

EXAMPLE:

STA/TESTS:1:2-4:6:8-10/PASS:3/FLAGS:IER:HOE=1:UAM:LOE

THIS COMMAND WILL CAUSE THREE PASSES TO BE MADE, EACH PASS CONSISTING OF TESTS 1,2,3,4,6,8,9, AND 10 EXECUTED AGAINST ALL UNITS. THERE IS NO DIFFERENCE BETWEEN SAYING <FLAG> AND SAYING <FLAG=1>. THE NOTATION <FLAG=0> IS MEANINGFUL ONLY ON A COMMAND OTHER THAN START TO CLEAR A FLAG THAT WAS PREVIOUSLY SET. NOTE THAT ON ALL COMMANDS ONLY THE FIRST

PROGRAM DOCUMENT

395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451

THREE LETTERS ARE SCANNED.

6.3.2 RESTART COMMAND

```
*****  
RES(TART)/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

452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508

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

PRO(CCEED)/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

509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565

HARDWARE DIALOGUE. THIS COMMAND MUST BE FOLLOWED BY A RESTART OR CONTINUE. THE UNITS SWITCH MUST BE SPECIFIED. THE ADD COMMAND IS MEANINGFUL ONLY FOR UNITS THAT WERE PREVIOUSLY DROPPED.

6.3.6 DROP COMMAND

DRO(P)/UNITS:<UNIT-LIST>

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

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

6.3.6.2 EFFECT OF DROP COMMAND

THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.

6.3.7 PRINT COMMAND

PRI(NT)

6.3.7.1 EFFECT OF PRINT COMMAND

THE TOTAL NUMBER OF ERRORS FOR EACH UNIT SINCE THE LAST START OR RESTART COMMAND ARE PRINTED. THE ISR (INHIBIT STATISTICAL REPORTING) FLAG IS CLEARED.

6.3.8 DISPLAY COMMAND

DIS(PLAY)/UNITS:<UNIT-LIST>

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

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

6.3.8.2 EFFECT OF DISPLAY COMMAND

THE HARDWARE P-TABLES FOR ALL UNITS UNDER TEST ARE PRINTED OUT IN THE FORMAT IN WHICH THEY WERE ENTERED. ANY UNITS THAT WERE DROPPED BY THE OPERATOR "DROP" COMMAND ARE SO

PROGRAM DOCUMENT

566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622

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 SURPRESSED 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

623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679

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.

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.

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).

PROGRAM DOCUMENT

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

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 (D) ? 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 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 THRU 15.

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

PROGRAM DOCUMENT

727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783

7.0 TEST DESCRIPTIONS

```

*****
* TEST 1 <VRC PARITY GENERATION TEST>
*
* SUBTEST 1 - TEST OF CORRECT ODD VRC PARITY GENERATION :
* THE LINE UNIT IS PLACED IN CHAR MODE, WITH ODD VRC, AND 7-BIT CHARS SELECTED.
* THE DATA CHARS IN PATTERN Q ARE LOADED/TRANSMITTED/READ, AS THE 8TH BIT
* (PARITY BIT) OF EACH DATA CHAR IS SENT THE PROGRAM CHECKS TSO FOR THE PROPER
* STATE. FOR THE FIRST 4 CHARS IN PATTERN Q THE PARITY BIT SHOULD = 1, FOR THE
* LAST 4 CHARACTERS IT SHOULD = 0.
*
* SUBTEST 2 - TEST OF CORRECT EVEN VRC PARITY GENERATION :
* THE LINE UNIT IS PLACED IN CHAR MODE, WITH EVEN VRC AND 7-BIT CHARS SELECTED.
* THE DATA CHARS IN PATTERN Q ARE LOADED/TRANSMITTED/READ, AS THE 8TH BIT
* (PARITY BIT) OF EACH DATA CHAR IS SENT THE PROGRAM CHECKS TSO FOR THE PROPER
* STATE. FOR THE FIRST 4 CHARS IN PATTERN Q THE PARITY BIT SHOULD = 0, FOR THE
* LAST 4 CHARACTERS IT SHOULD = 1.
*
* DATA PATTERN Q = 000,003,014,060,001,007,037,177
*
* NOTE: SINCE THE ROUTINE "SERIAL" TREATS THE FIRST BIT RECEIVED FROM THE
* USYRT AS THE MSB, THE "EXPECTED BIT SEQUENCE" WILL HAVE A REVERSED
* BIT ORDER.
*****

*****
* TEST 2 <VRC ERROR DETECTION TEST>
*
* SUBTEST 1 - FORCING OF RERR USING ODD VRC
* THE USYRT IS PLACED IN CHAR MODE WITH ODD VRC AND BOTH TX AND RX CHAR
* LENGTH=7 BITS. THE RECEIVER AND TRANSMITTER ARE THEN SYNC'D. WHEN THE FIRST
* DATA CHARACTER IS LOADED INTO TXDB, THE RX CHAR LENGTH IS CHANGED TO 6 BITS.
* TWO 7 BIT CHARACTERS (+PARITY) ARE THEN TRANSMITTED, RESULTING IN A 16 BIT
* STREAM WHICH THE RECEIVER WILL READ AS TWO 6 BIT CHARS (+PARITY + 2 LEFT).
* THE FIRST "CHARACTER" READ WILL HAVE THE CORRECT PARITY; THE SECOND WILL
* NOT.
*
* SUBTEST 2 - FORCING OF RERR USING EVEN VRC
* THE USYRT IS PLACED IN CHAR MODE WITH EVEN VRC AND BOTH TX AND RX CHAR
* LENGTH=7 BITS. THE RECEIVER AND TRANSMITTER ARE THEN SYNC'D. WHEN THE FIRST
* DATA CHARACTER IS LOADED INTO TXDB, THE RX CHAR LENGTH IS CHANGED TO 6 BITS.
* TWO 7 BIT CHARACTERS (+PARITY) ARE THEN TRANSMITTED, RESULTING IN A 16 BIT
* STREAM WHICH THE RECEIVER WILL READ AS TWO 6 BIT CHARS (+PARITY + 2 LEFT).
* THE FIRST "CHARACTER" READ WILL HAVE THE CORRECT PARITY; THE SECOND WILL
* NOT.
*
*****

*****
* TEST 3 <BCP CRC GENERATION/DETECTION TEST>
*
* THIS TEST IS COMPOSED OF 2 SUBTESTS -- #1 EXPECTS GOOD CRC

```


PROGRAM DOCUMENT

784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840

: * GENERATION AND REPORT ERRORS -- #2 FORCES AN ERROR AND ONLY
: * REPORT WHEN THE CRC IS ACCEPTED AS GOOD. EACH IS
: * RUN AT THE CHARACTER LENGTHS OF 8 BITS FOR THE ENTIRITY
: * OF EACH MESSAGE. BOTH THE TRANSMITTER AND RECEIVER WILL BE SET TO
: * THE SAME CHARACTER LENGTH. ERROR LOOPING WILL BE ON THE FAILING
: * SUBTEST. TEXT STRINGS WILL BE LIMITED TO 5 CHARACTERS.

: * TEST 4 <BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST>
: *
: * THE USYRT IS INITIALIZED FOR BOP MODE WITH TTL LOOPBACK SELECTED.
: * "SECONDARY STATION ADDRESS" IS NOT USED AND NO CRC/VRC IS CALCULATED.
: * A PATTERN IS TRANSMITTED AND TERMINATED FOLLOWED BY A SECOND MESSAGE.
: * TERMINATION OF THE FIRST MESSAGE IS ACCOMPLISHED WITH A FLAG
: * CHARACTER BUT RXE IS NOT DROPPED SO THAT THE SECOND MESSAGE CAN BE
: * SENT WITHOUT RE-SYNCRONIZATION. SEVERAL FLAG'S ARE IDLED BETWEEN THE
: * TWO MESSAGES. DURING THE SECOND MESSAGE A RECEIVER OVERRUN CONDITION
: * IS FORCED. THROUGHOUT THIS TEST, BASIC RECEIVER OPERATION AND TIMING
: * IS CHECKED. TRANSMITTED INFORMATION IS VERIFIED BY CHECKING THE DATA
: * MADE AVAILABLE AT RXDB.
: *
: * TRANSMITTED PATTERN: FLAG FLAG 123 321 000 377 101 FLAG... FLAG
: * 321 123 377 000 276.
: *
: * RECEIVED PATTERN: 123 321 000 377 101 321 123.

: * TEST 5 <BOP RX SECONDARY STATION ADDRESSING>
: *
: * THE USYRT IS INITIALIZED FOR BOP MODE WITH TTL LEVEL LOOPBACK,
: * SAM = 1, APA=0, AND ECM = 7. USING SHORT MESSAGES, THE ADDRESSES
: * 000, 125, 252, 176, AND 177 ARE CHECKED TO SEE THAT THE RECEIVER
: * RECOGNIZES THEM CORRECTLY. IN EACH CASE (AT EACH ADDRESS), A SERIES OF
: * 20 DIFFERENT MESSAGES ARE SENT TO VERIFY THAT THE USYRT WILL ONLY
: * RESPOND TO THE SPECIFIED VALUE.
: *
: * TEST PATTERN: ADR 000 OCR ADR
: * WHERE ADR IS THE ADDRESS BEING TESTED AND OCA IS THE ONE'S
: * COMPLEMENT OF THAT ADDRESS.
: *

: * TEST 6 <BOP RX ALL PARTIES ADDRESS TEST>
: *
: * INITIALIZE THE USYRT FOR BOP MODE WITH TTL LEVEL LOOPBACK
: * SAM = 1, S/AR = 123(OCT.), APA = 1, AND ECM = 7.
: * A SERIES OF 256 DIFFERENT SHORT MESSAGES ARE SENT TO VERIFY THAT
: * THE USYRT WILL ONLY RESPOND TO THE SPECIFIED VALUE AND ALSO 377 (FF
: * HEX.).
: *
: * TEST PATTERN: ADR 000 OCA ADR

PROGRAM DOCUMENT

841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897

```

;* WHERE ADR IS THE ADDRESS BEING TESTED AND OCA IS THE ONE'S
;* COMPLEMENT OF THAT ADDRESS.
;*****
;*****
;* TEST 7 <BOP RX BIT STUFFING TEST>
;*
;* THE USYRT IS INITIALIZED AND THE FOLLOWING TEXT IS TRANSMITTED
;* (DELIMITED BY THE APPROPRIATE CONTROL CHARACTERS -- OF COURSE):
;*
;* 000, 017, 036, 074, 170, 360, 037, 076, 174, 370, 077, 176, 374,
;* 177, 376, 377.
;*
;* NOTE THAT THIS PATTERN CONSISTS OF CHARACTERS WHICH REQUIRE BIT
;* STUFFING BOTH INDIVIDUALLY AND IN COMBINATION WITH ADJACENT
;* CHARACTERS. THERE ARE ALSO CHARACTERS WHICH REQUIRE NO BIT STUFFING
;* AT ALL. ALL 16 CHARACTERS ARE READ BY THE RECEIVER AND COMPARED AS
;* THEY ARE MADE AVAILABLE AT RXDB.
;*****
;*****
;* TEST 8 <BOP RX UNDERRUN IDLE ABORTS/FLAGS>
;*
;* THE USYRT IS INITIALIZED AND A MESSAGE IS STARTED. THEN, A
;* TRANSMITTER UNDERRUN IS FORCED WITH IDLE = 0 -- CAUSING ABORT
;* CHARACTERS TO BE IDLED. THE RECEIVER SHOULD BE RESET BY THE ABORT
;* CHARACTER(S). VERIFY THAT RAB/GA BIT=1.
;* REPEAT THE ABOVE WITH IDLE=1.
;*****
;*****
;* TEST 9 <BOP RX LOST RXE TEST>
;*
;* THE USYRT IS INITIALIZED AND A MESSAGE IS STARTED. WHILE IN THE
;* MIDDLE OF TEXT, RXE IS DROPPED AND THE REACTION OF THE RECEIVER IS
;* MONITORED.
;*****
;*****
;* TEST 10 <BOP RX GA (GO-AHEAD) RECOGNITION>
;*
;* A SHORT MESSAGE IS TRANSMITTED FOLLOWED BY A GA CHARACTER (INSTEAD
;* OF A FLAG CHARACTER). THE RECEIVER IS OBSERVED FOR PROPER HANDLING
;* OF BOTH THE MESSAGE AND THE GA CHARACTER. THE RAB/GA STATUS BIT
;* SHOULD BE SET BY THE RECEIVER UPON RECOGNITION OF THE GA CHARACTER.
;*****
;*****
;* TEST 11 <BOP RX "ABC" TEST>
;*
;* THIS TEST IS COMPOSED OF 7 SUBTESTS -- EACH ONE CHECKING A DIFFERENT
;* EXPECTED VALUE IN ABC (THE 3 BIT "ASSEMBLED BIT COUNT" FIELD WITHIN

```

PROGRAM DOCUMENT

898
899
900
901
902
903
904
905
906

;* RDSR). IN EACH SUBTEST THE USYRT IS INITIALIZED AND A SMALL MESSAGE
;* IS STARTED. THE LAST CHARACTER IS SENT WITH ITS LENGTH BEING
;* SPECIFIED FIRST AS 1 BIT, THEN AS 2 BITS, THEN AS 3 BITS, ETC. IN THE
;* TRANSMITTER SIDE OF THE USYRT. IN ALL CASES THE RECEIVER IS LEFT SET
;* TO 8 BITS IN LENGTH AND WHEN THE FLAG CHARACTER IS DETECTED, ABC IS
;* CHECKED AND SHOULD MATCH TXCL. ERROR LOOPING WILL BE ON THE FAILING
;* SUBTEST.
;*
;*****

PROGRAM DOCUMENT

908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948

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

```

950          .SBTTL GENERAL EQUATES AND DS INVOCATION & SETUP
951
952
953          000000      HELP=0          ; CONTROL LISTING OF HELP INFORMATION
954                                     ;
955                                     ; HELP=0   NO LIST
956                                     ; HELP=1   LIST
957
963          002000      .=2000
964
965          .MCALL SVC
966 002000      SVC      ; INITIALIZE SUPERVISOR MACROS
967
968
969 002000      BGNMOD  LU1MOD
970
971
972          000001      $LSTIN= 1
973          000001      $LSTTAG= 1
974          000001      SVCINS= 1      ; LIST INSTRUCTIONS, SHIFTED RIGHT
975          000001      SVCTST= 1      ; LIST TEST TAGS, SHIFTED RIGHT
976          000001      SVCSUB= 1      ; LIST SUBTEST TAGS, SHIFTED RIGHT
977          000001      SVCGBL= 1      ; LIST GLOBAL TAGS, SHIFTED RIGHT
978          000001      SVCTAG= 1      ; LIST OTHER TAGS, SHIFTED RIGHT
979
980          ;          CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
981          ;          TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS.  CHANGE THE
982          ;          SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS.  YOU MAY
983          ;          CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.

```

PROGRAM HEADER

```

985
986
987
988
989
990
991 002000
992
1000
1001 002000
002000
002000      103
002001      116
002002      104
002003      115
002004      104
002005      000
002006      000
002007      000
002010
002010      101
002011
002011      060
002012
002012      000000
002014
002014      000036
002016
002016      034100
002020
002020      000000
002022
002022      002154
002024
002024      000000
002026
002026      034356
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,ERRTBL

          HEADER CNDMD,A,0,30.,0

```

```

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

L$REV::
          .ASCII /A/

L$DEPO::
          .ASCII /0/

L$UNIT::
          .WORD 0

L$TIML::
          .WORD 30.

L$HPCP::
          .WORD L$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$EXP1::
          .WORD 0

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

L$EF::

```

PROGRAM HEADER

002052 000000
 002054 000000
 002056
 002056 000000
 002060
 002060 003232
 002062
 002062 000000
 002064
 002064 000000
 002066
 002066 000000
 002070
 002070 024346
 002072
 002072 024342
 002074
 002074 000000
 002076
 002076 003252
 002100
 002100 104035
 002102
 002102 002176
 002104
 002104 023644
 002106
 002106 024340
 002110
 002110 024214
 002112
 002112 023636
 002114
 002114 000000
 002116
 002116 000000
 002120
 002120 000000

.WORD 0
 .WORD 0
 L\$SPC:: .WORD 0
 L\$DEVP:: .WORD 0
 L\$REPP:: .WORD L\$DVTYP
 .WORD 0
 L\$EXP4:: .WORD 0
 L\$EXP5:: .WORD 0
 L\$AUT:: .WORD L\$AU
 L\$DLT:: .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\$AUTO
 L\$PRT:: .WORD L\$PROT
 L\$TEST:: .WORD 0
 L\$DLY:: .WORD 0
 L\$HIME:: .WORD 0

1002
 1008
 1009

.EVEN

DISPATCH TABLE

1011
1012
1013 002122

1014
1015
1016 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
:;////////////////////

1017
1018 002122
002122 000013
002124
002124 024350
002126 025256
002130 026070
002132 026672
002134 027520
002136 030232
002140 030730
002142 032074
002144 032714
002146 033156
002150 033420

DISPATCH 11.

.WORD 11
L\$DISPATCH:;
.WORD T1
.WORD T2
.WORD T3
.WORD T4
.WORD T5
.WORD T6
.WORD T7
.WORD T8
.WORD T9
.WORD T10
.WORD T11

1019

DEFAULT HARDWARE P-TABLE

1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048

002152
002152 000010
002154
002154

002154 160020
002156 000300
002160 004000
002162 000000
002164 000000
002166 000000
002170 000000
002172 000001

002174
002174

.SBTTL DEFAULT HARDWARE P-TABLE
:////////////////////
:/ THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
:/ THE TEST-DEVICE PARAMETERS. THE STRUCTURE OF THIS TABLE
:/ IS IDENTICAL TO THE STRUCTURE OF THE RUN-TIME P-TABLE.
:////////////////////

BGNHW DFPTBL

160020
300
4000
000
000
0
0
1

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

;DMV11 CSR UNIBUS ADDRESS
;DMV11 INTERRUPT VECTOR
;DMV11 INTERRUPT PRIORITY LEVEL = 4
;SWITCH REG. #1 (BOOT ADDRESS)
;SWITCH REG. #2 (DDCMP ADDRESS)
;MODULE IS M8064
;H3254&H3255 USED
;BAUD RATE = 56 K
; 0 = 19.2 K
; 1 = 56 K

ENDHW

L10000:

M2

SOFTWARE P-TABLE

1050
1051
1052
1053
1054
1055
1056
1057
1058
1059

002174
002174 000000
002176
002176
002176
002176

.SBTTL SOFTWARE P-TABLE

:/ THE SOFTWARE P-TABLE CONTAINS THE VALUES OF THE PROGRAM
:/ PARAMETERS THAT CAN BE CHANGED BY THE OPERATOR.

BGNSW SFPTBL

ENDSW

.WORD L10001-L\$SW/2
L\$SW::
SFPTBL::
L10001:

GLOBAL EQUATES SECTION -- BASIC EQUATES

1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069 002176

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

```

1071      .SBTTL REGISTER DEFINITIONS -- MAINTENANCE REGISTERS -- SELN & BSELN
1072
1073      ;*****
1074      ;* MAINTENANCE REGISTER # 0 - BSEL0
1075      ;*****
1076      000020      IEO      = BIT4      ;"INTERRUPT ENABLE OUT"
1077      000001      IEI      = BIT0      ;"INTERRUPT ENABLE IN"
1078
1079      ; BIT 7 IS ALSO USED BY THE MICROCODE. ITS LABEL IS "RQI" WHICH STANDS FOR
1080      ; "REQUIST IN". IT'S PART OF THE HANDSHAKING FOR USING THE SEL & BSEL REG'S.
1081      ; HOWEVER, THE MAINT. LOOP DOES NOT MAKE USE OF THIS BIT AND IT IS THEREFORE
1082      ; UNNECESSARY TO DEFINE IT HERE.
1083
1084      ;*****
1085      ;* MAINTENANCE REGISTER # 1 - BSEL1
1086      ;*****
1087      000200      RUN      = BIT7      ;"RUN" & ALSO CONTROLS 6502 MICROPROCESSOR'S RDY STATE
1088      000100      MCLR     = BIT6      ;MASTER CLEAR
1089      000001      MREQ     = BIT0      ;M-LOOP ACCESS
1090      000301      STRTMLOP= RUN!MCLR!MREQ ;INITIATE M-LOOP
1091
1092      ;*****
1093      ;* MAINTENANCE REGISTER # 2 - BSEL2
1094      ;*****
1095      000200      MRDY     = BIT7      ;M-LOOP READY
1096
1097      ;*****
1098      ;* MAINTENANCE LOOP COMMAND DEFINITIONS
1099      ;*****
1100      000001      REDLOC   = 1      ;READ LOC. W/IN DMV-11 ---- (SEL4) ==> BSEL6
1101      000002      WRILOC   = 2      ;WRITE LOC. W/IN DMV-11 --- BSEL6 ==> (SEL4)
1102      000003      REDPAG   = 3      ;READ BLOCK W/IN DMV-11 --- (SEL6) ==> (SEL4)
1103      000004      WRIPAG   = 4      ;WRITE BLOCK W/IN DMV-11 -- (SEL4) ==> (SEL6)
1104      000005      EXECUT   = 5      ;SET 6502'S PC AND EXECUTE -- SEL6 ==> PC
1105      000007      DOTBMT   = 7      ;SET MAINTENANCE INTERRUPT DISABLE IN PROCESSOR
1106      ;STATUS --- [KB7] ==> BSEL3
1107

```

REGISTER DEFINITIONS -- USYRT

```

1109      .SBTTL REGISTER DEFINITIONS -- USYRT
1110
1111
1112      120400      USYRT = 120400      ;USYRT BASE ADDRESS = A100 (HEX)
1113
1114      ;*****
1115      ;* USYRT "RECEIVER DATA BUFFER" REGISTER -- READ ONLY
1116      ;*****
1117
1118      120400      RDSRL = 120400      ;ADDRESS OF THIS REG
1119
1120      ;*****
1121      ;* USYRT "RECEIVER STATUS" REGISTER -- READ ONLY
1122      ;*****
1123
1124      120401      RDSRH = 120401      ;ADDRESS OF THIS REG
1125
1126      ;BIT DEFINITIONS ON BYTE BASIS :
1127      000200      RERR = BIT7      ;ERROR CHECK
1128      000160      ABC = BIT6:BIT5:BIT4 ;ASSEMBLED BIT COUNT
1129      000010      ROR = BIT3      ;RECEIVER OVER RUN
1130      000004      RABGA = BIT2      ;RECEIVED ABORT/GA CHARACTER
1131      000002      REOM = BIT1      ;RECEIVED END-OF-MESSAGE
1132      000001      RSOM = BIT0      ;RECEIVED START-OF-MESSAGE
1133
1134      ;BIT DEFINITIONS ON WORD BASIS :
1135      100000      RXERR = BIT15      ;RECEIVED CRC/VRC ERROR
1136      004000      RXOR = BIT11      ;RECEIVER OVER RUN
1137      002000      RXABGA = BIT10      ;RECEIVED ABORT/GO AHEAD CHARACTER
1138      001000      RXEOM = BIT9      ;RECEIVED END-OF-MESSAGE
1139      000400      RXSOM = BIT8      ;RECEIVED START-OF-MESSAGE
1140
1141      000001      RERCHK = BIT0      ;FLAG TO INVOKE RERR CHK IN SUBROUTINE RXCHAR
1142
1143      ;*****
1144      ;* USYRT "TRANSMITTER DATA BUFFER" REGISTER
1145      ;*****
1146
1147      120402      TDSRL = 120402      ;ADDRESS OF THIS REG
1148
1149      ;*****
1150      ;* USYRT "TX STATUS AND CONTROL" REGISTER
1151      ;*****
1152
1153      120403      TDSRH = 120403      ;ADDRESS OF THIS REG
1154
1155      ;BIT DEFINITIONS ON BYTE BASIS :
1156      000200      TERR = BIT7      ;TRANSMITTER UNDERRUN ERROR
1157      000010      TGA = BIT3      ;TRANSMIT GO AHEAD
1158      000004      TAB = BIT2      ;TRANSMIT ABORT
1159      000002      TEOM = BIT1      ;TRANSMIT END-OF-MESSAGE
1160      000001      TSOM = BIT0      ;TRANSMIT START-OF-MESSAGE
1161
1162      ;BIT DEFINITIONS ON WORD BASIS :
1163      100000      TXERR = BIT15      ;TRANSMITTER UNDERRUN ERROR
1164      004000      TXGA = BIT11      ;TRANSMIT GO AHEAD
1165      002000      TXAB = BIT10      ;TRANSMIT ABORT

```

REGISTER DEFINITIONS -- USYRT

```

1166      001000      TXEOM   = BIT9           ; TRANSMIT END-OF-MESSAGE
1167      000400      TXSOM   = BIT8           ; TRANSMIT START-OF-MESSAGE
1168
1169      ;:*****
1170      ;* USYRT "SYNC/SECONDARY ADDRESS" REGISTER
1171      ;:*****
1172
1173      120404      PCSARL  = 120404        ; ADDRESS OF THIS REG
1174      000226      SYNCH   = 226         ; STANDARD SYNCH CHARACTER
1175
1176      ;:*****
1177      ;* USYRT "MODE CONTROL"
1178      ;:*****
1179
1180      120405      PCSARH  = 120405        ; ADDRESS OF THIS REG
1181
1182      ;BIT DEFINITIONS ON BYTE BASIS:
1183
1184      000200      APA     = BIT7         ; "ALL PARTIES ADDRESS" ENABLE
1185      000100      PROTO  = BIT6         ; SPECIFIES BOP/CCP PROTOCOL -- 0 = BOP
1186      000040      STRIP  = BIT5         ; STRIP EXTRA SYNC'S IN CCP MODE, SEE GA CHARS IN BOP
1187      000020      SECAD  = BIT4         ; SECONDARY ADDRESS MODE -- BOP MODE ONLY
1188      000010      IDLE   = BIT3         ; IDLE & SYNC CHAR. TRANSMISSION CONTROL
1189      000007      XYZ    = BIT2!BIT1!BIT0 ; CRC/PARITY SELECTION CONTROL
1190
1191      ;BIT DEFINITIONS ON WORD BASIS:
1192
1193      100000      APAD    = BIT15        ; "ALL PARTIES ADDRESS" ENABLE
1194      040000      DDCMP   = BIT14        ; CODE FOR DDCMP MODE
1195      020000      STRIPS  = BIT13        ; STRIP EXTRA SYNC'S IN CCP MODE, SEE GA CHARS IN BOP
1196      010000      SECADR  = BIT12        ; SECONDARY ADDRESS MODE -- BOP MODE ONLY
1197      004000      IDLES   = BIT11        ; IDLE & SYNC CHAR. TRANSMISSION CONTROL
1198      000400      CRCOS   = BIT8         ; CODE FOR CRC-CCITT-0 SELECTION
1199      001400      CRC16   = BIT9!BIT8    ; CODE FOR CRC-16 SELECTION
1200      003400      NOCHK  = BIT10!BIT9!BIT8 ; CODE FOR NO ERROR CHECKING
1201      002400      EVRC   = BIT10!BIT8    ; CODE FOR VRC EVEN CHECK
1202      002000      OVRC   = BIT10        ; CODE FOR VRC ODD CHECK
1203
1204      ;:*****
1205      ;* USYRT "DATA LENGTH SELECT" REGISTER
1206      ;:*****
1207
1208      120407      PCR     = 120407        ; ADDRESS OF THIS REG
1209
1210      ;BIT DEFINITIONS:
1211
1212      000340      TXDL    = BIT7!BIT6!BIT5 ; TRANSMIT DATA LENGTH SELECTION
1213      000020      EXADD   = BIT4         ; EXTENDED ADDRESS FIELD -- NOT USED OR TESTED
1214      000010      EXCON  = BIT3         ; EXTENDED CONTROL FIELD -- NOT USED OR TESTED
1215      000007      RXDL   = BIT2!BIT1!BIT0 ; RECEIVER DATA LENGTH SELECTION
1216
1217      ;:*****
1218      ;* USYRT STATUS REGISTER (ADDR. A400)
1219      ;:*****
1220      122000      USTATR  = 122000        ; USYRT STATUS REGISTER ADDRESS = A400 (HEX)
1221
1222      ;BIT DEFINITIONS:

```

REGISTER DEFINITIONS -- USYRT

1223				
1224	000200	RDA	= BIT7	;RECEIVER DATA AVAILABLE
1225	000100	TBMT	= BIT6	;TRANSMITTER BUFFER EMPTY
1226	000040	RXACT	= BIT5	;RECEIVER ACTIVE
1227	000020	RSA	= BIT4	;RECEIVER STATUS AVAILABLE
1228	000010	TSO	= BIT3	;TRANSMITTER SERIAL OUTPUT
1229	000004	TXACT	= BIT2	;TRANSMITTER ACTIVE
1230	000002	TXU	= BIT1	;TRANSMITTER UNDERRUN
1231	000001	SFR	= BIT0	;SYNC/FLAG RECEIVED

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1233 .SBTTL REGISTER DEFINITIONS -- 6522 VIA CHIP
1234
1235 120000 VIA = 120000 ;VIA BASE ADDRESS = A000 (HEX)
1236
1237 ;:*****
1238 ;* MODEM & MAINTENANCE CONTROL -- "ORB" 8 BIT PORT B -- WRITE ONLY
1239 ;:*****
1240
1241 120000 VIAORB = 120000 ;ADDRESS OF THIS REGISTER -- HEX = A0X0
1242
1243 000200 NULCLK = BIT7 ;"NULL CLK L" -- NULL CLOCK
1244 000100 RXEN = BIT6 ;"RXENL" -- USYRT RECEIVER ENABLE
1245 000040 TXEN = BIT5 ;"TXENL" -- USYRT TRANSMITTER ENABLE
1246 000020 DTR = BIT4 ;"DTR" -- DATA TERMINAL READY
1247 000010 RTSND = BIT3 ;"RTSND" -- REQUEST TO SEND
1248 000004 HDX = BIT2 ;"HDX" -- HALF DUPLEX
1249 000002 TTLOOP = BIT1 ;"SELECT TTL LEVEL LOOPBACK"
1250 000001 PRESET = BIT0 ;"PRESET H" --
1251 000000 DTRL = 0 ;DTR IS ASSERTED LOW
1252
1253 ;:*****
1254 ;* MODEM STATUS REGISTER -- "ORA" 8 BIT PORT A -- READ ONLY
1255 ;:*****
1256
1257 120001 VIAMS = 120001 ;ADDRESS OF THIS REGISTER -- HEX = A0X1
1258
1259 000200 RING = BIT7 ;"RING H" --
1260 000100 CARIER = BIT6 ;"CARRIER H" --
1261 000040 MDMRDY = BIT5 ;"MODEM RDY H" --
1262 000020 SPEED = BIT4 ;"BAUD RATE SWITCH -- (19.2K/56K)
1263 000010 CTS = BIT3 ;"CTS H -- CLEAR TO SEND
1264 000004 TM = BIT2 ;"TEST MODE H" --
1265 000002 RCVDAT = BIT1 ;"RCV DATA H" --
1266 000001 UMAINT = BIT0 ; SELECT USYRT INT LOOPBACK **SELECT BIT**
1267
1268
1269 ;:*****
1270 ;* DATA DIRECTION FOR PORT B -- "DDRB" -- READ/WRITE
1271 ;:*****
1272
1273 120002 VIADPB = 120002 ;ADDRESS OF THIS REGISTER -- HEX = A0X2
1274
1275 ; ALL BITS ARE DEFINED THE SAME:
1276 ; THE BIT SETTING DEFINED THE DIRECTION OF ITS RELATED BIT IN BIT PORT B
1277
1278 ; INITIALIZED TO 377 (HEX = FF) -- PORT B IS READ/WRITE
1279
1280
1281 ;:*****
1282 ;* DATA DIRECTION FOR PORT A -- "DDRA" -- READ/WRITE
1283 ;:*****
1284
1285 120003 VIADPA = 120003 ;ADDRESS OF THIS REGISTER -- HEX = A0X3
1286
1287 ; ALL BITS ARE DEFINED THE SAME:
1288 ; THE BIT SETTING DEFINED THE DIRECTION OF ITS RELATED BIT IN BIT PORT A
1289

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1290      ;      INITIALIZED TO 001 (HEX = 01) -- PORT A IS READ ONLY (EXCEPT FOR
1291      ;      BIT0 WHICH ENABLES USYRT INTERNAL LOOPBACK).
1292
1293
1294
1295      ;*****
1296      ;* TIMER 1 LOW ORDER (LATCH & COUNTER) -- "T1L-L" & "T1C-L" -- WRITE & READ
1297      ;*****
1298
1299      120004      VIAT1A = 120004      ;ADDRESS OF THIS REGISTER -- HEX = A0X4
1300
1301      ;      WHEN WRITING, LOW ORDER LATCH IS LOADED.
1302      ;      WHEN READING, LOW ORDER COUNTER IS READ.
1303
1304
1305
1306      ;*****
1307      ;* TIMER 1 HIGH ORDER COUNTER & TRIGGER -- "T1L-H AND TRIGGER" & "T1C-H"
1308      ;*      -- WRITE & READ
1309      ;*****
1310
1311      120005      VIAT1B = 120005      ;ADDRESS OF THIS REGISTER -- HEX = A0X5
1312
1313      ;      WHEN WRITING; HIGH ORDER LATCH IS LOADED, BOTH LOW & HIGH ORDER LATCHES
1314      ;      ARE LOADED INTO THE COUNTER, AND THE COUNTER IS STARTED.
1315
1316      ;      WHEN READING, THE HIGH ORDER COUNTER IS READ.
1317
1318
1319
1320      ;*****
1321      ;* TIMER 1 LOW ORDER LATCH -- "T1L-L" -- READ/WRITE
1322      ;*****
1323
1324      120006      VIAT1C = 120006      ;ADDRESS OF THIS REGISTER -- HEX = A0X6
1325
1326      ;      THE LOW ORDER LATCH IS READ OR LOADED. THIS LATCH IS USED TO LOAD THE
1327      ;      COUNTER WHEN T1MODE (IN VIAACR) = 3
1328
1329
1330
1331      ;*****
1332      ;* TIMER 1 HIGH ORDER LATCH -- "T1L-H" -- READ/WRITE
1333      ;*****
1334
1335      120007      VIAT1D = 120007      ;ADDRESS OF THIS REGISTER -- HEX = A0X7
1336
1337      ;      THE HIGH ORDER LATCH IS READ OR LOADED. THIS LATCH IS USED TO LOAD THE
1338      ;      COUNTER WHEN T1MODE (IN VIAACR) = 3
1339
1340
1341
1342      ;*****
1343      ;* TIMER 2 LOW ORDER (LATCH & COUNTER) -- "T2L-L" & "T2C-L" -- WRITE & READ
1344      ;*****
1345
1346      120010      VIAT2A = 120010      ;ADDRESS OF THIS REGISTER -- HEX = A0X8

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1347
1348 ; WHEN WRITING, LOW ORDER LATCH IS LOADED.
1349 ; WHEN READING, LOW ORDER COUNTER IS READ.
1350
1351
1352
1353 ;*****
1354 ;* TIMER 2 HIGH ORDER COUNTER & TRIGGER -- "T2L-H AND TRIGGER" & "T2C-H"
1355 ;* -- WRITE & READ
1356 ;*****
1357
1358 120011 VIAT2B = 120011 ;ADDRESS OF THIS REGISTER -- HEX = A0X9
1359
1360 ; WHEN WRITING; HIGH ORDER LATCH IS LOADED, BOTH LOW & HIGH ORDER LATCHES
1361 ; ARE LOADED INTO THE COUNTER, AND THE COUNTER IS STARTED.
1362
1363 ; WHEN READING, THE HIGH ORDER COUNTER IS READ.
1364
1365 ;*****
1366 ;* SHIFT REGISTER -- "SR" -- READ/WRITE
1367 ;*****
1368
1369 120012 VIASR = 120012 ;ADDRESS OF THIS REGISTER -- HEX = A0XA
1370
1371 ; SHIFTING IS CONTROLLED BY THE SETTING OF VIASRC (ACR2 ---> ACR4) IN VIAACR
1372
1373
1374
1375 ;*****
1376 ;* AUXILIARY CONTROL REGISTER -- "ACR" -- READ/WRITE
1377 ;*****
1378
1379 120013 VIAACR = 120013 ;ADDRESS OF THIS REGISTER -- HEX = A0XB
1380
1381 000300 T1MODE = BIT7!BIT6 ;CONTROL THE MODE OF TIMER # 1
1382
1383 ;BIT 7:
1384 ; 0 PB7 DISABLED -- ONLY T1TO IN VIAIFR REFLECTS TIMEOUT
1385 ; 1 PB7 & T1TO REFLECT TIMEOUT
1386
1387 ;BIT 6:
1388 ; 0 TIMER 1 IN ONE-SHOT MODE
1389 ; 1 TIMER 1 IN CONTINUOUS SQUARE WAVE MODE
1390
1391 000040 T2MODE = BITS ;CONTROLS THE MODE OF TIMER # 1
1392
1393 ; 0 PULSE COUNTING MODE
1394 ; 1 INTERVAL TIMER MODE
1395
1396 000034 SRMODE = BIT4!BIT3!BIT2 ;CONTROLS THE MODE OF THE SHIFT REGISTER
1397
1398 ; 0 SR DISABLED
1399 ; 1 SHIFT IN UNDER CONTROL OF T2, SHFT PULSES GEN'D ON CB1
1400 ; 2 SHIFT IN AT SYS. CLOCK RATE, SHFT PULSES GEN'D ON CB1
1401 ; 3 SHIFT IN UNDER CONTROL OF EXTERNAL INPUT PULSES
1402 ; 4 SHIFT OUT -- FREE RUNNING -- RATE CONTROLLED BY T2
1403 ; 5 SHIFT OUT -- RATE CONTROLLED BY T2 -- PULSES ON CB1

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1404                                     ; 6   SHIFT OUT -- SYS. CLOCK RATE -- PULSES ON CB1
1405                                     ; 7   SHIFT OUT -- UNDER CONTROL OF PULSES APPLIED TO CB1
1406
1407      000002      PBLENB = BIT1          ;PB LATCH CONTROL -- 1 ENABLES LATCH
1408      000001      PALENB = BIT0         ;PA LATCH CONTROL -- 1 ENABLES LATCH
1409
1410
1411
1412
1413      ;:*****
1414      ;* PERIPHERAL CONTROL REGISTER -- "PCR" -- READ/WRITE
1415      ;:*****
1416
1417      120014      VIAPCR = 120014         ;ADDRESS OF THIS REGISTER -- HEX = A0XC
1418
1419      000340      CB2CTL = BIT7!BIT6!BIT5 ;CB2 MODE SELECT
1420      00C020      CB1CTL = BIT4         ;CB1 MODE SELECT
1421      000016      CA2CTL = BIT3!BIT2!BIT1 ;CA2 MODE SELECT
1422      000001      CA1CTL = BIT0         ;CA1 MODE SELECT
1423
1424
1425
1426      ;:*****
1427      ;* INTERRUPT FLAG REGISTER -- "IFR" -- READ ONLY
1428      ;:*****
1429
1430      120015      VIAIFR = 120015        ;ADDRESS OF THIS REGISTER -- HEX = A0XD
1431
1432      000200      FLGIRQ = BIT7          ;SET WHEN A FLAG IN THIS REG. GOES HIGH AND
1433                                          ;ITS CORRESPONDING BIT IN VIAIER IS SET.
1434                                          ;(I.E. VIAIER IS THE ENABLE REGISTER FOR THE
1435                                          ;FOR THE SETTING OF IRQ AND THE ISSUANCE OF
1436                                          ;AN INTERRUPT TO THE 6502 WHEN IRQ IS SET.)
1437
1438      000100      FLGT1 = BIT6           ;TIMEOUT OF TIMER 1
1439      000040      FLGT2 = BIT5           ;TIMEOUT OF TIMER 2
1440      000020      FLGCB1 = BIT4         ;ACTIVE TRANSITION OF PIN 18 (CB1)
1441      000010      FLGCB2 = BIT3         ;ACTIVE TRANSITION OF PIN 19 (CB2)
1442      000004      FLGSR = BIT2          ;COMPLETION OF 8 SHIFTS
1443      000002      FLGCA1 = BIT1         ;ACTIVE TRANSITION OF PIN 40 (CA1)
1444      000001      FLGCA2 = BIT0         ;ACTIVE TRANSITION OF PIN 39 (CA2)
1445
1446
1447
1448      ;:*****
1449      ;* INTERRUPT ENABLE REGISTER -- "IER" -- READ/WRITE
1450      ;:*****
1451
1452      120016      VIAIER = 120016        ;ADDRESS OF THIS REGISTER -- HEX = A0XE
1453
1454      000200      INTSC = BIT7           ;CONTROLS THE SETTING OR CLEARING OF BITS IN
1455                                          ;THE REST OF IER. IF = 0 THE OTHER BITS IN
1456                                          ;THIS REG., IF SET, WILL CLEAR THEIR RESPECTIVE
1457                                          ;BITS IN THE INT. ENAB. REG.. IF = 1, THE
1458                                          ;RESPECTIVE BITS WILL BE SET.
1459
1460      ; WHEN WRITING THIS REG., THE COMMENT ABOVE HOLDS.

```

K3

REGISTER DEFINITIONS -- 6522 VIA CHIP

1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481

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

```

1483      .SBTTL REGISTER DEFINITIONS -- MISC
1484
1485      ;;*****
1486      ;* SWITCH PACKS
1487      ;;*****
1488
1489      121000      SWPBOT = 121000      ;"BOOT ADDRESS" SWITCH PACK [A200]
1490      121400      SWPDDCMP = 121400      ;"DDCMP ADDRESS" SWITCH PACK [A300]
1491
1492      ;MISCELLANEOUS EQUATES
1493
1494      100000      TCCHEK = BIT15      ;FLAG TO REQUEST H3254,5 CHECK
1495      001000      RAMADR = 001000      ;STARTING ADRS OF RAM PAGE 2 (ADRS 0200 HEX)
1496
1497      000002      EIAV35 = BIT1      ;SELECT V.35 OR EIA 423/232C
1498      000001      INTGRL = BIT0      ;SELECT INTEGRAL MODEM
1499
1500      040000      NCRXEN = BIT14      ;KILL RXEN DURING "INITRN"
1501      001000      NOLOOP = BIT9      ;KILL TTLOOP DURING "INITRN"
1502
1503      000200      NCTBMT = BIT7      ;DISABLE INITIAL TBMT=0 CHECK IN TXCHAR
1504
1505      100000      NOCRDA = BIT15      ;DISABLE INITIAL RDA=0 CHECK IN RXCHAR
1506      040000      NFCRDA = BIT14      ;DISABLE FINAL RDA=1 CHECK IN RXCHAR
1507      020000      NCRACT = BIT13      ;DISABLE RXACT=1 CHECK AFTER CLOCKING (RXCHAR)
1508

```

GLOBAL DATA SECTION

1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779 002176
 002176
 002176 000000
 002200 000000
 002202 000000
 002204 000000
 1780
 1781
 1782
 1783
 1784 002206
 1785 002206 000000
 1786 002210
 1787 002210 000000
 1788 002212
 1789 002212 000000
 1790 002214
 1791 002214 000000
 1792 002216
 1793 002216 000000
 1794 002220
 1795 002220 000000
 1796 002222
 1797 002222 000000
 1798 002224
 1799 002224 000000
 1800 002226 000000
 1801 002230 000000
 1802 002232 000000
 1803 002234 000000
 1804 002236 000000
 1805 002240 000000
 1806 002242 000000
 1807 002244 000000
 1808
 1809 002246
 1810
 1811
 1812 002266

```

.SBTTL GLOBAL DATA SECTION

;//////////////////////////////////////
; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
; IN MORE THAN ONE TEST.
;//////////////////////////////////////

;*****
; CONTROL BLOCK FOR STACKED ERROR MESSAGES
;-----

                ERRRTBL                                L$ERRRTBL::

ERRRTYP::      .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:
BSR1:         .WORD  0
WSR4:
BSR2:         .WORD  0
WSR6:
BSR3:         .WORD  0
WSR10:
BSR4:         .WORD  0
WSR12:
BSR5:         .WORD  0
WSR14:
BSR6:         .WORD  0
WSR16:
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

```

1814
1815
1816
1817 002326 000000
1818 002330 000000
1819 002332 000000
1820 002334 000000
1821 002336 000000
1822 002340 000000
1823 002342 000000
1824 002344 000000
1825 002346 000000
1826 002350 000000
1827 002352 000000
1828
1829 002354 000000
1830 002356 000000
1831 002360 000000
1832 002362 000000
1833 002364 000000
1834 002366 000000
1835 002370 000000
1836 002372 000000
1837 002374 000000
1838 002376 000000
1839 002400 000000
1840 002402 000000
1841 002404 000000
1842 002406 000000
1843 002410 000000
1844 002412 000000
1845 002414 000000
1846 002416 000000
1847 002420 000000
1848

```

```

;*****
;* MISCELLANEOUS STORAGE
;*****
TDATA: .WORD 0 ; TEST DATA
GDATA: .WORD 0 ; GOOD DATA
BDATA: .WORD 0 ; BAD DATA
XDATA: .WORD 0 ; EXCLUSIVE-OR BETWEEN GOOD AND BAD DATA
SCRACH: .WORD 0 ; GEN'L PURPOSE SCRATCH WORD
LOGDEV: .WORD 0 ; LOGICAL DEVICE NUMBER
REGNUM: .WORD 0 ; CONTAINS A DEVICE REGISTER NUMBER
PSTACK: .WORD 0 ; CONTAINS BASE LEVEL PROGRAM STACK POINTER
PRIOR: .WORD 0 ; CPU PRIORITY FOR PRINTOUT
SUBRPC: .WORD 0 ; PC OF SUBR CALL FOR ERROR REPORTS
INTFLG: .WORD 0 ; INTERRUPT RECEIVED FLAGS
; BIT 0 FOR TX, BIT 1 FOR RCV
ERRFLG: .WORD 0 ; SUBROUTINE ERROR FLAG
TIMFLG: .WORD 0 ; EVENT TIME-OUT FLAG
RETADR: .WORD 0 ; SUBR ERROR RETURN ADDRESS
REDBYT: .WORD 0 ; LO BYTE CONTAINS BYTE READ FROM LU REG
WRIBYT: .WORD 0 ; LO BYTE CONTAINS BYTE TO LOAD INTO LU REG
LOADAT: .WORD 0 ; CONTAINS TEST DATA LOADED INTO REG
GOODAT: .WORD 0 ; STORAGE FOR EXPECTED DATA
BADDAT: .WORD 0 ; STORAGE FOR ACTUAL DATA
FRSTIM: .WORD 0 ; FLAG=0 IF PROGRAM JUST LOADED
SAVE4: .WORD 0 ; SAVE LOC 4 HERE (ERROR TRAP VECTOR)
SAVE6: .WORD 0 ; SAVE LOC 6 HERE (ERROR TRAP VECTOR)
ERROR1: .WORD 0 ; SUBR ERR. BIT FLAGS (DEF'D IN GLOBAL EQUATES)
CHPTYP: .WORD 0 ; USYRT CHIP TYPE, =0 FOR SMC, ELSE =1
SAVLEN: .WORD 0 ; SAVED TX AND RCV CHAR LENGTHS
DEVMAP: .WORD 0 ; BIT MAP OF ACTIVE DEVICES
DEVPTR: .WORD 0 ; DEVICE MAP BIT POINTER
UNIT: .WORD 0 ; CONTAINS UNIT NO. (1 TO N)
STARES: .WORD 0 ; FLAG TO SHOW NO. OF PASSES SINCE STA OR RES
TSTNUM: .WORD 0 ; NO. OF CURRENT TEST (FOR SOME TESTS)

```


GLOBAL DATA SECTION

```

1850          ;:***** CURRENT DEVICE PARAMETERS *****
1851 002422   BSEL0:
1852 002422   SEL0:
1853 002422   160020   MPCSR: .WORD      160020           ;POINTER TO DMV11 CSR'S
1854 002424   160021   BSEL1: .WORD      160021           ;POINTER TO BSEL1
1855 002426   BSEL2:
1856 002426   160022   SEL2: .WORD      160022           ;POINTER TO SEL2
1857 002430   160023   BSEL3: .WORD      160023           ;POINTER TO BSEL3
1858 002432   BSEL4:
1859 002432   160024   SEL4: .WORD      160024           ;POINTER TO SEL4
1860 002434   160025   BSEL5: .WORD      160025           ;POINTER TO BSEL5
1861 002436   BSEL6:
1862 002436   160026   SEL6: .WORD      160026           ;POINTER TO SEL6
1863 002440   160027   BSEL7: .WORD      160027           ;POINTER TO BSEL7
1864 002442   BSEL10:
1865 002442   160030   SEL10: .WORD     160030           ;POINTER TO SEL10
1866 002444   160031   BSEL11: .WORD     160031           ;POINTER TO BSEL11
1867 002446   BSEL12:
1868 002446   160032   SEL12: .WORD     160032           ;POINTER TO SEL12
1869 002450   160033   BSEL13: .WORD     160033           ;POINTER TO BSEL13
1870 002452   BSEL14:
1871 002452   160034   SEL14: .WORD     160034           ;POINTER TO SEL14
1872 002454   160035   BSEL15: .WORD     160035           ;POINTER TO BSEL15
1873 002456   BSEL16:
1874 002456   160036   SEL16: .WORD     160036           ;POINTER TO SEL16
1875 002460   160037   BSEL17: .WORD     160037           ;POINTER TO BSEL17
1876
1877 002462   000300   MPIVEC: .WORD     300           ;DMV11 INPUT INTERRUPT VECTOR
1878 002464   000304   MPOVEC: .WORD     304           ;DMV11 OUTPUT INTERRUPT VECTOR
1879 002466   000240   MPRIOR: .WORD     240           ;DMV11 DEVICE PRIORITY
1880 002470   000000   LUSWI1: .WORD     0            ;LINE UNIT SWITCH PACK #1
1881 002472   000000   LUSWI2: .WORD     0            ;LINE UNIT SWITCH PACK #2
1882 002474   000000   BRDTYP: .WORD     0            ;0=M8064, 1=M8053/V.35,2=M8053/EIA
1883 002476   000000   TSTCON: .WORD     0            ;TEST CONNECTOR INDICATOR
1884 002500   000001   BDRATE: .WORD     1            ;BAUD RATE = 56 K
1885          :                0 = 19.2 K
1886          :                1 = 56 K

```

GLOBAL DATA SECTION

```

1888
1889 002502 120400
1890 002504 120401
1891 002506 120402
1892 002510 120403
1893 002512 120404
1894 002514 120405
1895 002516 120407
1896 002520 122000
1897
1898
1899 002522
1900
1901
1902 002532 000000
1903 002534 000000
1904 002536 000000
1905 002540 000000
1906 002542 000000
1907 002544 000000
1908 002546 000000
1909 002550 000000
1910
1911
1912 002552 000000
1913 002554 000000
1914 002556 000000
1915 002560 000000
1916 002562 000000
1917 002564 000000
1918 002566 000000
1919 002570 000000
1920
1921
1922 002572
1923 002572 377
1924 002573 000
1925 002574 000
1926 002575 360
1927 002576 000
1928 002577 000
1929 002600 347
1930
1931 002601 200

```

```

;TABLE OF USYRT REGISTER ADDRESSES
USYREG: .WORD 120400 ;ADDRESS OF RDSRL
        .WORD 120401 ;ADDRESS OF RDSRH
        .WORD 120402 ;ADDRESS OF TDSRL
        .WORD 120403 ;ADDRESS OF TDSRH
        .WORD 120404 ;ADDRESS OF PCSARL
        .WORD 120405 ;ADDRESS OF PCSARH
        .WORD 120407 ;ADDRESS OF PCR
        .WORD 122000 ;ADDRESS OF USYRT STATUS REG

;***** STORAGE FOR DATA READ IN ADDRESS TESTS *****
REDDAT: .BLKB 8.

;:***** GEN'L PURPOSE SCRATCH STORAGE *****
REG0: .WORD 0
REG1: .WORD 0
REG2: .WORD 0
REG3: .WORD 0
REG4: .WORD 0
REG5: .WORD 0
REG6: .WORD 0
REG7: .WORD 0

;:***** SCRATCH STORAGE FOR MESSAGE REPORTING *****
TMP0: .WORD 0
TMP1: .WORD 0
TMP2: .WORD 0
TMP3: .WORD 0
TMP4: .WORD 0
TMP5: .WORD 0
TMP6: .WORD 0
TMP7: .WORD 0

;***** INBUS LU REG BIT MASKS FOR UNPREDICTABLE BITS *****
UPBITS: .BYTE 377 ;MASK FOR RDBR
        .BYTE 000 ;MASK FOR RDSR
        .BYTE 000 ;MASK FOR TDBR
        .BYTE 360 ;MASK FOR TDSR
        .BYTE 000 ;MASK FOR SSAR
        .BYTE 000 ;MASK FOR PCSAR
        .BYTE 347 ;MASK FOR PCR

TDSRNRW: .BYTE 200 ;TDSR NON-R/W BITS

```

DATA TEST PATTERNS

1933			.SBTTL DATA TEST PATTERNS
1934			;***** DATA PATTERN E *****
1935	002602		PATE:
1936	002602	377	.BYTE 377
1937	002603	377	.BYTE 377
1938	002604	377	.BYTE 377
1939	002605	377	.BYTE 377
1940	002606	377	.BYTE 377
1941	002607	377	.BYTE 377
1942	002610	377	.BYTE 377
1943	002611	366	.BYTE 366
1944			
1945			;***** DATA PATTERN F *****
1946	002612		PATF:
1947	002612	000	.BYTE 000
1948	002613	000	.BYTE 000
1949	002614	000	.BYTE 000
1950	002615	000	.BYTE 000
1951	002616	000	.BYTE 000
1952	002617	000	.BYTE 000
1953	002620	000	.BYTE 000
1954	002621	110	.BYTE 110
1955			
1956			;***** DATA PATTERN G *****
1957	002622		PATG:
1958	002622	000	.BYTE 000
1959	002623	001	.BYTE 001
1960	002624	003	.BYTE 003
1961	002625	004	.BYTE 004
1962	002626	005	.BYTE 005
1963	002627	007	.BYTE 007
1964	002630	100	.BYTE 100
1965	002631	101	.BYTE 101
1966	002632	103	.BYTE 103
1967	002633	104	.BYTE 104
1968	002634	105	.BYTE 105
1969	002635	107	.BYTE 107
1970	002636	000	.BYTE 000
1971	002637	017	.BYTE 017
1972	002640	027	.BYTE 027
1973	002641	041	.BYTE 041
1974	002642	200	.BYTE 200
1975	002643	277	.BYTE 277
1976	002644	103	.BYTE 103
1977	002645	144	.BYTE 144
1978	002646	115	.BYTE 115
1979	002647	157	.BYTE 157
1980	002650	000	.BYTE 000
1981			
1982			;***** DATA PATTERN X1 *****
1983	002651		PATX1:
1984	002651	125	.BYTE 125
1985	002652	252	.BYTE 252
1986	002653	000	.BYTE 000
1987	002654	377	.BYTE 377
1988	002655	001	.BYTE 001
1989	002656	002	.BYTE 002

DATA TEST PATTERNS

1990	002657	004	.BYTE	004
1991	002660	010	.BYTE	010
1992	002661	020	.BYTE	020
1993	002662	040	.BYTE	040
1994	002663	100	.BYTE	100
1995	002664	200	.BYTE	200
1996	002665	376	.BYTE	376
1997	002666	375	.BYTE	375
1998	002667	373	.BYTE	373
1999	002670	367	.BYTE	367
2000	002671	357	.BYTE	357
2001	002672	337	.BYTE	337
2002	002673	277	.BYTE	277
2003	002674	177	.BYTE	177
2004	002675	176	.BYTE	176

***** DATA PATTERN I *****
PATI:

2007	002676		.BYTE	000
2008	002676	000	.BYTE	041
2009	002677	041	.BYTE	102
2010	002700	102	.BYTE	143
2011	002701	143	.BYTE	204
2012	002702	204	.BYTE	245
2013	002703	245	.BYTE	306
2014	002704	306	.BYTE	347
2015	002705	347	.BYTE	000
2016	002706	000	.BYTE	001
2017	002707	001	.BYTE	002
2018	002710	002	.BYTE	004
2019	002711	004	.BYTE	040
2020	002712	040	.BYTE	100
2021	002713	100	.BYTE	200
2022	002714	200	.BYTE	000
2023	002715	000	.BYTE	346
2024	002716	346	.BYTE	345
2025	002717	345	.BYTE	343
2026	002720	343	.BYTE	307
2027	002721	307	.BYTE	247
2028	002722	247	.BYTE	147
2029	002723	147	.BYTE	347
2030	002724	347	.BYTE	242
2031	002725	242	.BYTE	105
2032	002726	105	.BYTE	347
2033	002727	347	.BYTE	010
2034	002730	010	.BYTE	020
2035	002731	020	.BYTE	367
2036	002732	367	.BYTE	357
2037	002733	357	.BYTE	030
2038	002734	030	.BYTE	027
2039	002735	027	.BYTE	377
2040	002736	377	.BYTE	

***** DATA PATTERN J *****
PATJ:

2041				
2042				
2043	002737		.BYTE	000
2044	002737	000	.BYTE	000
2045	002740	000	.BYTE	001
2046	002741	001		

DATA TEST PATTERNS

2047	002742	002	.BYTE	002
2048	002743	004	.BYTE	004
2049	002744	020	.BYTE	020
2050	002745	040	.BYTE	040
2051	002746	010	.BYTE	010

2052
2053:***** DATA PATTERN K *****
PATK:

2054	002747		.BYTE	000
2055	002747	000	.BYTE	000
2056	002750	377	.BYTE	377
2057	002751	376	.BYTE	376
2058	002752	375	.BYTE	375
2059	002753	373	.BYTE	373
2060	002754	376	.BYTE	376
2061	002755	177	.BYTE	177
2062	002756	377	.BYTE	377
2063	002757	000	.BYTE	000
2064	002760	001	.BYTE	001
2065	002761	002	.BYTE	002
2066	002762	004	.BYTE	004
2067	002763	010	.BYTE	010
2068	002764	200	.BYTE	200
2069	002765	125	.BYTE	125
2070	002766	252	.BYTE	252
2071	002767	000	.BYTE	000

2072
2073:***** DATA PATTERN L *****
PATL:

2074	002770		.BYTE	000
2075	002770	000	.BYTE	000
2076	002771	017	.BYTE	017
2077	002772	016	.BYTE	016
2078	002773	015	.BYTE	015
2079	002774	013	.BYTE	013
2080	002775	016	.BYTE	016
2081	002776	017	.BYTE	017
2082	002777	017	.BYTE	017
2083	003000	000	.BYTE	000
2084	003001	001	.BYTE	001
2085	003002	002	.BYTE	002
2086	003003	004	.BYTE	004
2087	003004	010	.BYTE	010
2088	003005	000	.BYTE	000
2089	003006	005	.BYTE	005
2090	003007	012	.BYTE	012
2091	003010	000	.BYTE	000

DATA TEST PATTERNS

```

2093
2094          ;***** DATA PATTERN Q *****
2095 003011    000          PATQ:  .BYTE  000
2096 003012    003          .BYTE  003
2097 003013    014          .BYTE  014
2098 003014    060          .BYTE  060
2099 003015    001          .BYTE  001
2100 003016    007          .BYTE  007
2101 003017    037          .BYTE  037
2102 003020    177          .BYTE  177
2103
2104          ;***** DATA PATTERN INVERTED Q *****
2105 003021    000          PATQB:  .BYTE  000          ;INVERTED 000 (7 BIT)
2106 003022    140          .BYTE  140          ;INVERTED 003 (7 BIT)
2107 003023    030          .BYTE  030          ;INVERTED 014 (7 BIT)
2108 003024    006          .BYTE  006          ;INVERTED 060 (7 BIT)
2109 003025    100          .BYTE  100          ;INVERTED 001 (7 BIT)
2110 003026    160          .BYTE  160          ;INVERTED 007 (7 BIT)
2111 003027    174          .BYTE  174          ;INVERTED 037 (7 BIT)
2112 003030    177          .BYTE  177          ;INVERTED 177 (7 BIT)
2113
2114 003031          ENDPAT:
2115          .EVEN
2116
2117          ;*** RECEIVED DATA BUFFER (64. WORDS) ***
2118 003032          RCVBUF: .BLKW  64.
2119
2120
2121
2122

```

GLOBAL TEXT SECTION

2124
 2125
 2126
 2127
 2128
 2129
 2130
 2131
 2132
 2133
 2134
 2135 003232
 003232
 003232 115 070 060
 003235 065 063 040
 003240 117 122 040
 003243 115 070 060
 003246 066 064 000

.SBTTL GLOBAL TEXT SECTION

```

*****
;# THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
;# MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
;# MORE THAN ONE TEST.
*****

```

```

*****
;* NAMES OF DEVICES SUPPORTED BY PROGRAM
*****
DEV TYP <M8053 OR M8064>

```

```

L$DVTYP::
.ASCIZ M8053 OR M8064

```

.EVEN

2136
 2137
 2138
 2139
 2140
 2141 000012
 2142 003252
 003252
 003252 104 115 126
 STS - PART 2 OF 3/
 003255 055 061 061
 003260 040 114 111
 003263 116 105 040
 003266 125 116 111
 003271 124 040 124
 003274 105 123 124
 003277 123 040 055
 003302 040 120 101
 003305 122 124 040
 003310 062 040 117
 003313 106 040 063
 003316 000

```

*****
;* TITLE OF PROGRAM
*****

```

```

.RADIX 10.
DESCRIPT <DMV-11 LINE UNIT TESTS - PART 2 OF 3>

```

```

L$DESC::
.ASCIZ /DMV-11 LINE UNIT TE

```

.EVEN

2143 000010
 2144
 2145

.RADIX 8.

GLOBAL SUBROUTINE SECTION

2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185

003362
003370
003376
003404
2186
2187
2188
2189
2190
2191
2192
2193

.SBTTL GLOBAL SUBROUTINE SECTION

.SBTTLM-LOOP -- MSTCLR -- MASTER CLEAR AND ENTER M-LOOP
:*****
: MSTCLR -- MASTER CLEAR & ENTER M-LOOP
:
: CALLING SEQUENCE:
:
: JSR PC,MSTCLR
: 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>
:
:--*****

MSTCLR: MOVB #RUN!MCLR!MREQ,@BSEL1 ;INITIATE M-LOOP

1\$: MOV R3,-(SP)
MOV #24.,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
SOB R3,1\$
MOV (SP)+,R3

BITB #MRDY,@BSEL2 ;DID THE M-LOOP FINISH
BNE 5\$;YES. GOOD. RETURN
JSR PC,GETWSR ;GET BYTE SELECT REGISTERS
MOV #RUN!MCLR!MREQ,GDATA ;IDENTIFY REQUESTED FUNCTION
GDF EM3,ERR4 ;"MRDY" TIMEOUT
; QUEUE "DEVICE FATAL" ERROR # 1
MOV #T.EDF,ERRTYP
MOV #1,ERRNBR
MOV #EM3,ERRMSG
MOV #ERR4,ERRBLK

5\$: SEC ;SET CARRY TO INDICATE ERROR
BR 9\$;EXIT WITH THE "ERROR" FLAG (CARRY BIT) SET
9\$: CLC ;CLEAR C BIT FOR NO ERRORS
RTS PC ;RETURN

....M-LOOP -- READ

```

2195 .SBTTL ....M-LOOP -- READ
2196 ;*****
2197 ; READ - READ THE SPECIFIED ADDRESS WITHIN THE DMV-11 (M8053)
2198 ;
2199 ; CALLING SEQUENCE:
2200 ;
2201 ; JSR R5,READ
2202 ; .WORD <ADDRESS OF REGISTER WITHIN DMV-11>
2203 ; .WORD <DESTINATION ADDRESS WITHIN LSI-11>
2204 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2205 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2206 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2207 ;
2208 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2209 ;
2210 ;-----*****
2211
2212 003422 012577 177004 READ: MOV (R5)+,@SEL4 ;SETUP SOURCE POINTER
2213 003426 112777 000001 176772 MOVB @REDLOC,@SEL2 ;TELL M-LOOP TO GIVE US THE REQUESTED DATA
2214
2215 003434 010346 MOV R3,-(SP)
2216 003436 012703 000050 MOV #40.,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2217 003442 077301 1$: SOB R3,1$
2218 003444 012603 MOV (SP)+,R3
2219
2220 003446 132777 000200 176752 BITB @MRDY,@SEL2 ;DID THE M-LOOP FINISH
2221 003454 001023 BNE 5$ ;YES, GOOD. RETURN
2222
2223 003456 004737 004134 JSR PC,GETWSR ;GET BYTE SELECT REGISTERS
2224 003462 012737 000001 002330 MOV @REDLOC,GDATA ;IDENTIFY REQUESTED FUNCTION
2225 003470 GTDF EM4,ERR4 ;"MRDY" TIMEOUT
; QUEUE "DEVICE FATAL" ERROR # 2
; MOV #T.EDF,ERRTYP
; MOV #2,ERRNBR
; MOV #EM4,ERRMSG
; MOV #ERR4,ERRBLK
003470 012737 000001 002176
003476 012737 000002 002200
003504 012737 014141 002202
003512 012737 021274 002204
2226 003520 000261 SEC ;INDICATE AN ERROR HAS BEEN STACKED
2227 003522 000401 BR 6$ ;RETURN WITH THAT INDICATION
2228
2229 003524 000241 5$: CLC ;INDICATE "NO ERROR"
2230 003526 117735 176704 6$: MOVB @SEL6,@(R5)+ ;PUT DATA WHERE CALLER WANTS IT
2231 003532 000205 RTS R5 ;RETURN
2232
2233
2234
2235

```

....M-LOOP -- READ IMMEDIATE

2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268

003602
003610
003616
003624
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278

```
.SBTTL ....M-LOOP -- READ IMMEDIATE
:*****
: READI - READ IMMEDIATE THE SPECIFIED ADDRESS WITHIN THE DMV-11 (M8053)
:
: CALLING SEQUENCE:
:
:   JSR   R5,READI
:   .WORD <ADDRESS OF REGISTER WITHIN DMV-11>
:   .WORD <DESTINATION -- CONTENTS OF REG. IS PUT HERE>
:   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>
:
:-----*****
```

```
READI:
MOV      (R5)+,@SEL4      ;SETUP SOURCE POINTER
MOV      #REDLOC,@BSEL2   ;TELL M-LOOP TO GIVE US THE REQUESTED DATA

MOV      R3,-(SP)
MOV      #40,R3           ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
1$:      SOB      R3,1$
         MOV      (SP)+,R3

BITB     #MRDY,@BSEL2     ;DID THE M-LOOP FINISH
BNE      5$              ;YES, GOOD. RETURN

JSR      PC,GETWSR        ;GET BYTE SELECT REGISTERS
MOV      #REDLOC,GDATA   ;IDENTIFY REQUESTED FUNCTION
GTDF     EM4,ERR4        ;"MRDY" TIMEOUT
:          QUEUE "DEVICE FATAL" ERROR # 3
:
:          MOV     #T.EDF,ERRTYP
:          MOV     #3,ERRNBR
:          MOV     #EM4,ERRMSG
:          MOV     #ERR4,ERRBLK

SEC      ;INDICATE AN ERROR HAS BEEN STACKED
BR       6$              ;RETURN WITH THAT INDICATION

5$:      CLC
6$:      MOV      @SEL6,(R5)+
         RTS      R5      ;INDICATE "NO ERROR"
:          ;PUT DATA WHERE CALLER WANTS IT
:          ;RETURN
```

....M-LOOP -- WRITE

2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297 003646 012577 176560
2298 003652 113577 176560
2299 003656 000404
2300
2301
2302
2303

```

.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)+,@SEL4      ;SETUP SOURCE POINTER
        MOVB   @R5+,@SEL6      ;MAKE DATA AVAILABLE TO M-LOOP
        BR     MLWRI           ;THE REST OF THIS ROUTINE IS THE SAME AS "WRITEI"

```

....M-LOOP -- WRITE IMMEDIATE

```

2305 .SBTTL ....M-LOOP -- WRITE IMMEDIATE
2306 ;*****
2307 ; WRITEI - WRITE IMMEDIATE THE SPECIFIED DATA INTO THE SPECIFIED DMV-11 ADDRESS
2308 ;
2309 ; CALLING SEQUENCE:
2310 ;
2311 ; JSR R5,WRITEI
2312 ; .WORD - <ADDRESS OF REGISTER WITHIN DMV-11>
2313 ; .WORD <DATA FIELD -- DATA TO BE WRITTEN IN DMV-11>
2314 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2315 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2316 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2317 ;
2318 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2319 ;
2320 ;-----
2321
2322 003660 WRITEI:
2323 003660 012577 176546 MOV (R5)+,@SEL4 ;SETUP SOURCE POINTER
2324 003664 012577 176546 MOV (R5)+,@SEL6 ;MAKE DATA AVAILABLE TO M-LOOP
2325 003670 112777 000002 176530 MLWRI: MOVB @WRILOC,@BSEL2 ;TELL M-LOOP TO WRITE THE DATA
2326
2327 003676 010346 MOV R3,-(SP)
2328 003700 012703 000050 MOV #40,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2329 003704 077301 1$: SOB R3,1$
2330 003706 012603 MOV (SP)+,R3
2331
2332 003710 132777 000200 176510 BITB @MRDY,@BSEL2 ;DID THE M-LOOP FINISH
2333 003716 001023 BNE 5$ ;YES, GOOD. RETURN
2334 003720 004737 004134 JSR PC,GETWSR ;GET BYTE SELECT REGISTERS
2335 003724 012737 000002 002330 MOV @WRILOC,GDATA ;IDENTIFY REQUESTED FUNCTION
2336 003732 GTDF EM4,ERR4 ;"MRDY" TIMEOUT
; QUEUE "DEVICE FATAL" ERROR # 4
; MOV #T.EDF,ERRTYP
; MOV #4,ERRNBR
; MOV #EM4,ERRMSG
; MOV #ERR4,ERRBLK
003732 012737 000001 002176
003740 012737 000004 002200
003746 012737 014141 002202
003754 012737 021274 002204
2337 003762 000261 SEC ;INDICATE AN ERROR HAS BEEN STACKED
2338 003764 000401 BR 6$ ;RETURN WITH THAT INDICATION
2339
2340 003766 000241 5$: CLC ;INDICATE "NO ERROR"
2341 003770 000205 6$: RTS R5 ;RETURN
2342
2343
2344
2345

```

....GETBSR -- GET BYTE SELECT REGISTERS

2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362

```

.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      @@ @   @@@@ @   @@ @
;
;   EXIT CONDITIONS - NONE      @ @ @   @ @ @   @ @ @
;
;   REGISTERS DESTROYED - NONE   @@ @@@@ @@@@ @ @ @ @
;
;-----*

```

2363 003772 117737 176424 002206
 2364 004000 117737 176420 002210
 2365 004006 117737 176414 002212
 2366 004014 117737 176410 002214
 2367 004022 117737 176404 002216
 2368 004030 117737 176400 002220
 2369 004036 117737 176374 002222
 2370 004044 117737 176370 002224
 2371 004052 117737 176364 002226
 2372 004060 117737 176360 002230
 2373 004066 117737 176354 002232
 2374 004074 117737 176350 002234
 2375 004102 117737 176344 002236
 2376 004110 117737 176340 002240
 2377 004116 117737 176334 002242
 2378 004124 117737 176330 002244
 2379 004132 000207

```

GETBSR: MOVB @BSR0,BSR0 ;PUT THE CURRENT CSR VALUES INTO THE PRINT-OUT
;TABLE
MOVB @BSR1,BSR1
MOVB @BSR2,BSR2
MOVB @BSR3,BSR3
MOVB @BSR4,BSR4
MOVB @BSR5,BSR5
MOVB @BSR6,BSR6
MOVB @BSR7,BSR7
MOVB @BSR10,BSR10
MOVB @BSR11,BSR11
MOVB @BSR12,BSR12
MOVB @BSR13,BSR13
MOVB @BSR14,BSR14
MOVB @BSR15,BSR15
MOVB @BSR16,BSR16
MOVB @BSR17,BSR17
RTS PC ;RETURN TO CALLER

```

2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392

```

.SBTTL ....GETWSR -- GET WORD SELECT REGISTERS
; "WORD" VERSION OF ABOVE SUBROUTINE
GETWSR: MOV @WSR0,WSR0 ;MOVE THE 4 WORD REGISTERS TO THE OTHERWISE
;BYTE TABLE
MOV @WSR2,WSR2
MOV @WSR4,WSR4
MOV @WSR6,WSR6
MOV @WSR10,WSR10
MOV @WSR12,WSR12
MOV @WSR14,WSR14
MOV @WSR16,WSR16
RTS PC ;RETURN TO CALLER

```

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

```

2394 .SBTTL ....STUREG -- STATIC TEST OF SPECIFIED USYRT REGISTER
2395 ;.....
2396 ; STUREG -- PERFORM A STATIC TEST OF THE SPECIFIED USYRT REGISTER
2397 ;
2398 ; CALLING SEQUENCE:
2399 ;
2400 ; <R0 CONTAINS THE ADDRESS OF THE REGISTER TO BE TESTED>
2401 ; <"TDATA" CONTAINS THE TEST BYTE>
2402 ; <"GDATA" CONTAINS THE EXPECTED DATA>
2403 ; <"REGNUM" CONTAINS REG INDEX FOR POSSIBLE ERRORS>
2404 ;
2405 ; JSR PC,STUREG
2406 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2407 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2408 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2409 ;
2410 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2411 ;
2412 ;-----
2413
2414 004216 010037 004232 STUREG: MOV R0,2$ ;PUT SPECIFIED REGISTER'S ADDRESS IN I/O CALLS
2415 004222 010037 004250 MOV R0,4$
2416
2417 004226 004537 003646 JSR R5,WRITE ;WRITE IT
2418 004232 000000 2$: .WORD 0 ;*** MODIFIED FROM ABOVE ***
2419 004234 002326 .WORD TDATA ;
2420 004236 103431 BCS 10$ ;ON ERROR, EXIT
2421
2422 004240 005037 002332 CLR BDATA ;CLEAR BOTH BYTES -- JUST IN CASE....
2423 004244 004537 003422 JSR R5,READ ;READ IT BACK AGAIN
2424 004250 000000 4$: .WORD 0 ;*** MODIFIED FROM ABOVE ***
2425 004252 002332 .WORD BDATA ;
2426 004254 103422 BCS 10$ ;ON ERROR, EXIT
2427
2428 004256 123737 002330 002332 CMPB GDATA,BDATA ;DID WE READ WHAT WE WROTE?
2429 004264 000241 CLC ; (THIS ISN'T NEEDED FOR THE ERROR TEST BUT
2430 ; MUST BE CLEARED ON EXIT IF NO ERROR OCCURED)
2431 004266 001415 BEQ 10$ ;YES, EXIT FROM SUBTEST
2432 004270 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
004270 012737 000001 002176
004276 012737 000005 002200
004304 012737 014347 002202
004312 012737 021420 002204
2433 004320 000261 10$: SEC ;INDICATE THAT AN ERROR WAS DETECTED
2434 004322 000207 RTS PC
2435
2436
2437 .SBTTL ....STALL -- DELAY FOR 10.5 MICRO-SEC'S (ON LSI-11)
2438 ;.....
2439 ; STALL -- THIS SUBROUTINE STALLS FOR ABOUT 10.5 MICRO-SECONDS
2440 ;-----
2441
2442 004324 000207 STALL: RTS PC
2443
2444
2445

```

```

2447      .SBTTL
2448
2449      ;*****
2450      ;* GETURS - LOAD IN'0 THE 8 WORD STORAGE AREA (UREGS) THE CONTENTS OF THE
2451      ;*   VARIOUS USYRT REGISTERS
2452      ;*
2453      ;*   CALLING SEQUENCE:
2454      ;*
2455      ;*****
2456 004326 012737 002246 004370 GETURS: MOV   @UREGS,5$ ;INIT POINTER TO REG STORAGE TABLE
2457 004334 012737 120400 004366      MOV   @USYRT,4$ ;INIT POINTER TO REGISTER ADDRESSES
2458
2459 004342 005037 002264      CLR   UREGS+14. ;CLEAR STORAGE WORD
2460 004346 004537 003422      JSR   R5,READ   ;READ THE USYRT STATUS REGISTER
2461 004352 122000      .WORD USTATR   ;STATUS REGISTER'S ADDRESS WITHIN DMV-11
2462 004354 002264      .WORD UREGS+14. ;ADDRESS ALLOCATED TO THAT REG. W/IN "UREGS"
2463
2464 004356 005077 000006      3$: CLR   @5$ ;CLEAR STORAGE WORD
2455 004362 004537 003422      JSR   R5,READ   ;READ A LINE UNIT REG
2466 004366 000000      4$: .WORD 0 ;REGISTER ADDRESS GOES HERE
2467 004370 000000      5$: .WORD 0 ;STORAGE ADRS IN TABLE GOES HERE
2468
2469 004372 005237 004366      6$: INC   4$ ;INCREMENT REG NO.
2470 004376 023727 004366 120406      CMP   4$,@USYRT+6 ;THIS IS NOT A VALID REGISTER ADDRESS
2471 004404 001772      BEQ   6$ ;SO IT MUST BE BYPASSED
2472
2473 004406 062737 000002 004370      ADD   @2,5$ ;ADVANCE ADDRESS OF STORAGE AREA POINTER
2474 004414 023727 004366 120410      CMP   4$,@USYRT+10 ;SEE IF ALL REGS READ YET
2475 004422 001355      BNE   3$ ;BR IF NOT
2476
2477 004424 000207      RTS   PC ;RETURN
2478
2479
2480
2481      ;*****
2482      ;* GETVRS: - LOAD INTO THE 16 WORD STORAGE AREA (VREGS) THE CONTENTS OF THE
2483      ;*   VARIOUS VIA REGISTERS.
2484      ;*
2485      ;*   CALLING SEQUENCE :
2486      ;*****
2487 004426 012737 002266 004454 GETVRS: MOV   @VREGS,5$ ;INIT POINTER TO REG STORAGE TABLE
2488 004434 012737 120000 004452      MOV   @VIA,4$ ;INIT POINTER TO REGISTER ADDRESSES
2489 004442 005077 000006      3$: CLR   @5$ ;CLEAR STORAGE WORD
2490 004446 004537 003422      JSR   R5,READ   ;READ A VIA REG
2491 004452 000000      4$: .WORD 0 ;REGISTER ADDRESS GOES HERE
2492 004454 000000      5$: .WORD 0 ;STORAGE ADRS IN TABLE GOES HERE
2493 004456 005237 004452      6$: INC   4$ ;INCREMENT REG NO.
2494 004462 062737 000002 004454      ADD   @2,5$ ;INCREMENT STORAGE ADRS
2495 004470 023727 004452 120020      CMP   4$,@VIA+16. ;SEE IF ALL VIA REGS READ YET
2496 004476 001361      BNE   3$ ;BR IF NOT
2497 004500 000207      RTS   PC ;RETURN

```

....INITT1 -- INITIALIZE TIMER #1

2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526 004502 010146
2527 004504 012537 004626
2528 004510 012537 004654
2529 004514 111501
2530 004516 143701 000077
2531 004522 010137 004616
2532 004526 112501
2533
2534 004530 106301
2535 004532 106301
2536
2537
2538
2539
2540 004534 143701 000177
2541 004540 153701 000100
2542 004544 010137 004556
2543
2544 004550 004537 003660
2545 004554 120016
2546 004556 000000
2547
2548 004560 004537 003534
2549 004564 120013
2550 004566 000000
2551
2552 004570 013701 004566
2553 004574 143701 000300
2554 004600 053701 004616
2555 004604 010137 004616

```

.SBTTL ....INITT1 -- INITIALIZE TIMER #1
;*****
;* INITT1 - INITIALIZE TIMER # 1
;*
;*      CALLING SEQUENCE:
;*
;*      JSR      R5,INITT1
;*      .WORD   <VALUE LOADED INTO THE T1 LATCH @ VIAT1C & VIAT1D>
;*      .WORD   <VALUE LOADED INTO "T1L-L" & "T1C-H">
;*      .BYTE   <BITS 6 & 7 WILL BE LOADED INTO "ACR", BIT 5 WILL BE
;*              USED TO SET OR CLEAR BIT 6 ("T1") OF THE INTERRUPT
;*              ENABLE REGISTER ("IER")>
;*      .BYTE   <UNUSED>
;*
;* NOTE:
;*
;* BEFORE LOADING AND STARTING THE COUNTER, THE LATCH REGISTER (ACCESSED THRU
;* "VIAT1C") IS LOADED. THEN, T1L-L IS LOADED AND NEXT, T1C-H. THIS LAST
;* LOAD WILL RESET THE TIMEOUT BIT AND COUNTER LOGIC. IT IS EXPECTED AT THIS
;* TIME (5/25/79) THAT THE INTERRUPT FACILITY OF THE VIA CHIP WILL NOT BE USED
;* -- HOWEVER, ACCESS TO THE INTERRUPT ENABLE BIT IS GIVEN THROUGH THE THIRD
;* PARAMETER IN THE CALLING SEQUENCE (BIT 5 = 0 WILL CAUSE THIS ROUTINE TO
;* CLEAR THE ENABLE BIT ("T1") IN "IER".)
;*****
INITT1: MOV      R1,-(SP)      ;SAVE THE REGISTER WE WILL BE USING
        MOV      (R5)+,7#  ;SETUP VALUE TO BE WRITTEN IN LATCH
        MOV      (R5)+,10# ;SETUP VALUE TO BE WRITTEN IN COUNTER
        MOV      (R5),R1   ;GET & PROCESS BITS FOR ACR 6 & 7
        BICB    077,R1
        MOV      R1,4#     ;SETUP CALL SET ACR'S BITS 6 & 7
        MOV      (R5)+,R1  ;NOW, GET THE BIT TO BE USED IN SETTING OR
                           ;CLEARING BIT 6 OF "IER"
        ASLB    R1         ;THE PASSED BIT IS IN THE WRONG POSITION
        ASLB    R1         ;BUT, THE PASSED BIT SHOULD CONTROL THE OPERATION.
                           ;WE KNOW WE ARE SETTING OR CLEARING BIT 6 --
                           ;THUS, THE PASSED BIT WILL BECOME THE CONTROLLING
                           ;BIT 7 AND WE WILL "OR" IN THE BIT WE WISH TO
                           ;BE CONTROLLED (BIT 6).
                           ;FIRST, MAKE SURE ALL UNWANTED BITS ARE CLEARED
        BICB    177,R1     ;THEN SET BIT 6
        BISB    100,R1
        MOV      R1,2#     ;THE CALL WILL NOW WRITE THE APPROPRIATE VALUE
        JSR      R5,WRITEI ;WRITE TO
                           ;THE VIA'S IER
2$:      VIAIER 0          ;INTERRUPT ENABLE/DISABLE INFORMATION
        .WORD   0
        JSR      R5,READI  ;READ THE CURRENT SETTING OF
                           ;THE VIA'S ACR
3$:      VIAACR 0          ;INTO "3$"
        .WORD   0
        MOV      3$,R1     ;GET THAT VALUE
        BICB    300,R1    ;CLEAR THE CURRENT SETTING OF BITS 6 & 7
        BIS     4$,R1     ;SET THEM ACCORDING TO THE PASSED VALUES
        MOV      R1,4#    ;PASS THE NEW REG. SETTING TO APPROPRIATE CALL

```


....INITI1 -- INITIALIZE TIMER #1

```

2556
2557 004610 004537 003660      JSR      R5,WRITEI      ;WRITE TO
2558 004614 120013              VIAACR                    ;THE VIA'S ACR
2559 004616 000000      4$:  .WORD      0      ;THE NEW REGISTER SETTING
2560
2561 004620 004537 003660      JSR      R5,WRITEI      ;WRITE TO
2562 004624 120006              VIAT1C                    ;LOW ORDER LATCH REGISTER (T1L-L)
2563 004626 000000      7$:  .WORD      0      ;THE VALUE PASSED
2564
2565 004630 113737 004627 004644  MOVB     7$+1,8$         ;SETUP FOR AND
2566 004636 004537 003660      JSR      R5,WRITEI      ;WRITE TO
2567 004642 120007              VIAT1D                    ;HIGH ORDER LATCH REGISTER (T1L-H)
2568 004644 000000      8$:  .WORD      0      ;THE VALUE PASSED
2569
2570 004646 004537 003660      JSR      R5,WRITEI      ;WRITE TO
2571 004652 120004              VIAT1A                    ;LOW ORDER LATCH & COUNTER (T1L-L & T1C-L)
2572 004654 000000      10$: .WORD      0      ;THE VALUE PASSED
2573
2574 004656 113737 004655 004672  MOVB     10$+1,11$      ;SETUP FOR AND
2575 004664 004537 003660      JSR      R5,WRITEI      ;WRITE TO
2576 004670 120005              VIAT1B                    ;HIGH ORDER COUNTER (T1C-H) <ALSO STARTS CTR>
2577 004672 000000      11$: .WORD      0      ;THE VALUE PASSED
2578
2579      ; DON'T WAIT AROUND FOR ANYTHING TO HAPPEN -- JUST (JEST) RETURN!
2580
2581 004674 012601      MOV      (SP)+,R1      ;BUT FIRST RESTORE R1
2582 004676 005205      INC      R5            ;AND PUT R5 BACK ON A WORD BOUNDARY (THE LAST
2583                                     ;PASSED PARAM. WAS A BYTE, NOT A WORD!)
2584
2585 004700 000205      RTS      R5            ;NOW, RETURN
2586
2587

```

....INITT2 -- INITIALIZE TIMER #2

2589
 2590
 2591
 2592
 2593
 2594
 2595
 2596
 2597
 2598
 2599
 2600
 2601
 2602
 2603
 2604
 2605
 2606
 2607
 2608
 2609
 2610
 2611
 2612
 2613
 2614 004702 010146
 2615 004704 012537 005024
 2616 004710 111501
 2617 004712 143701 000337
 2618 004716 010137 005014
 2619 004722 112501
 2620
 2621 004724 106301
 2622 004726 106301
 2623 004730 106301
 2624
 2625
 2626
 2627
 2628 004732 143701 000177
 2629 004736 153701 000040
 2630 004742 010137 004754
 2631
 2632 004746 004537 003660
 2633 004752 120016
 2634 004754 000000
 2635
 2636 004756 004537 003534
 2637 004762 120013
 2638 004764 000000
 2639
 2640 004766 013701 004764
 2641 004772 143701 000040
 2642 004776 053701 005014
 2643 005002 010137 005014
 2644
 2645 005006 004537 003660

```

.SBTTL ....INITT2 -- INITIALIZE TIMER #2
;*****
;* INITT2 - INITIALIZE TIMER # 2
;*
;*      CALLING SEQUENCE:
;*
;*          JSR      R5,INITT2
;*          .WORD    <VALUE LOADED INTO "T2L-L" & "T2C-H">
;*          .BYTE    <BIT 5 WILL BE LOADED INTO "ACR", BIT 4 WILL BE USED
;*                  TO SET OR CLEAR BIT 5 ("T2") OF THE INTERRUPT ENABLE
;*                  REGISTER ("IER")>
;*          .BYTE    <UNUSED>
;*
;* NOTE:
;*
;* FIRST T2L-L IS LOADED, THEN T2C-H. THIS SECOND LOAD WILL RESET THE TIMEOUT
;* BIT AND COUNTER LOGIC. IT IS EXPECTED AT THIS TIME (5/25/79) THAT THE
;* INTERRUPT FACILITY OF THE VIA CHIP WILL NOT BE USED -- HOWEVER, ACCESS TO
;* THE INTERRUPT ENABLE BIT IS GIVEN THROUGH THE SECOND PARAMETER IN THE
;* CALLING SEQUENCE (BIT 4 = 0 WILL CAUSE THIS ROUTINE TO CLEAR THE ENABLE BIT
;* ("T2") IN "IER".)
;*
;*****
INITT2: MOV      R1,-(SP)          ;SAVE THE REGISTER WE WILL BE USING
        MOV      (R5)+,10$     ;SETUP VALUE TO BE WRITTEN IN COUNTER
        MOV      (R5),R1      ;GET & PROCESS BIT FOR ACR 5
        BICB    337,R1
        MOV      R1,4$        ;SETUP CALL TO SET OR CLEAR ACR'S BIT 5
        MOV      (R5)+,R1     ;NOW, GET THE BIT TO BE USED IN SETTING OR
                                ;CLEARING BIT 5 OF "IER"
        ASLB    R1            ;THE PASSED BIT IS IN THE WRONG POSITION
        ASLB    R1            ;BUT, THE PASSED BIT SHOULD CONTROL THE
                                ;OPERATION.
        ASLB    R1            ;WE KNOW WE ARE SETTING OR CLEARING BIT 5 --
                                ;THUS, THE PASSED BIT WILL BECOME THE CONTROLLING
                                ;BIT 7 AND WE WILL "OR" IN THE BIT WE WISH TO
                                ;BE CONTROLLED (BIT 5).
        BICB    177,R1        ;FIRST, MAKE SURE ALL UNWANTED BITS ARE CLEARED
        BISB    040,R1        ;THEN SET BIT 5
        MOV      R1,2$        ;THE CALL WILL NOW WRITE THE APPROPRIATE VALUE
        JSR     R5,WRITEI     ;WRITE TO
                                ;THE VIA'S IER
        VIAIER  .WORD 0      ;INTERRUPT ENABLE/DISABLE INFORMATION
2$:
        JSR     R5,READI      ;READ THE CURRENT SETTING OF
                                ;THE VIA'S ACR
        VIAACR  .WORD 0      ;INTO "3$"
3$:
        MOV      3$,R1        ;GET THAT VALUE
        BICB    040,R1        ;CLEAR THE CURRENT SETTING OF BIT 5
        BIS     4$,R1         ;SET IT ACCORDING TO THE PASSED VALUE
        MOV      R1,4$        ;PASS NEW REG. SETTING TO APPROPRIATE CALL
        JSR     R5,WRITEI     ;WRITE TO

```

....INITT2 -- INITIALIZE TIMER #2

```

2646 005012 120013
2647 005014 000000      4$: VIAACH          ;THE VIA'S ACR
                          .WORD 0      ;THE NEW REGISTER SETTING
2648
2649 005016 004537 003660 JSR R5,WRITEI      ;WRITE TO
2650 005022 120010      VIAT2A          ;LOW ORDER LATCH & COUNTER (T2L-L & T2C-L)
2651 005024 000000      10$: .WORD 0      ;THE VALUE PASSED
2652
2653 005026 113737 005025 005042 MOVB 10$,1,11$     ;SETUP FOR AND
2654 005034 004537 003660 JSR R5,WRITEI      ;WRITE TO
2655 005040 120011      VIAT2B          ;HIGH ORDER COUNTER (T2C-H) <ALSO STARTS CTR>
2656 005042 000000      11$: .WORD 0      ;THE VALUE PASSED
2657
2658                      ; DON'T WAIT AROUND FOR ANYTHING TO HAPPEN -- JUST (JEST) RETURN!
2659
2660 005044 012601      MOV (SP)+,R1      ;BUT FIRST RESTORE R1
2661 005046 005205      INC R5              ;AND PUT R5 BACK ON A WORD BOUNDARY (THE LAST
2662                      ;PASSED PARAM. WAS A BYTE, NOT A WORD!)
2663
2664 005050 000205      RTS R5              ;THEN RETURN
2665

```

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

```

2667 .SBTTL ....RSTCHK -- RESET USYRT/VERIFY ALL USYRT REGS @ RESET STATE
2668 ;*****
2669 ; RSTCHK - MANUALLY RESET THE USYRT AND VERIFY THAT ALL USYRT REGISTERS
2670 ; ARE IN THEIR RESET STATE. AN ERROR MESSAGE IDENTIFYING THE
2671 ; FAILING REGISTER IS STACKED IF ONE IS ENCOUNTERED.
2672 ;
2673 ; CALLING SEQUENCE:
2674 ; JSR R5,RSTCHK
2675 ;*****
2676
2677 RSTCHK:
2678 005052 010146 MOV R1,-(SP) ;SAVE R1
2679 005054 010246 MOV R2,-(SP) ;SAVE R2
2680
2681 005056 004537 003660 JSR R5,WRITEI ;SET PROGRAM RESET BIT IN VIA ORB REG
2682 005062 120000 VIAORB
2683 005064 00C031 DTR!RTSND!PRESET
2684 005066 004537 003660 JSR R5,WRITEI ;CLEAR PROGRAM RESET BIT IN VIA ORB REG
2685 005072 120000 VIAORB
2686 005074 000030 DTR!RTSND
2687
2688 005076 005001 CLR R1 ;INIT USYRT REG ADRS PTR
2689 005100 012702 002612 MOV #PATF,R2 ;INIT DATA PATTERN POINTER
2690 005104 016137 002502 005116 6$: MOV USYREG(R1),7$ ;SET USYRT READ ADDRESS
2691 005112 004537 003534 JSR R5,READI ;READ A USYRT REG
2692 005116 000000 7$: .WORD 0 ;USYRT REG ADRS GOES HERE
2693 005120 000000 8$: .WORD 0 ;DATA READ IS RETURNED HERE
2694 005122 123722 005120 CMPB 8$,(R2)+ ;SEE IF REG CONTAINS EXPECTED DATA
2695 005126 001432 BEQ 9$ ;BR IF MATCH
2696
2697 005130 010137 002342 MOV R1,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
2698 005134 006237 002342 ASR REGNUM ;GET WORD OFFSET
2699 005140 005037 002330 CLR GDATA ;GET EXPECTED DATA
2700 005144 116237 177777 002330 MOVB -1(R2),GDATA
2701 005152 013737 005120 002332 MOV 8$,BDATA ;GET ACTUAL DATA
2702 ;STACK "USYRT NOT CLEARED BY PROGRAM RESET" MSG
2703 005160 GTDF EM2,ERR10
; QUEUE "DEVICE FATAL" ERROR # 6
; MOV #T.EDF,ERRTYP
; MOV #6,ERRNBR
; MOV #EM2,ERRMSG
; MOV #ERR10,ERRBLK
005160 012737 000001 002176
005166 012737 000006 002200
005174 012737 014046 002202
005202 012737 021540 002204
2704 005210 000261 SEC ;SET C BIT TO FLAG ERROR
2705 005212 000406 BR 10$ ;TAKE ERROR EXIT
2706
2707 005214 062701 000002 9$: ADD #2,R1 ;INCR USYRT REG ADRS PTR
2708 005220 020127 000020 CMP R1,#16. ;SEE IF ALL REGS READ YET
2709 005224 002727 BLT 6$ ;BR IF NOT
2710 005226 000241 CLC ;** CLEAR C BIT FOR NO ERRORS
2711 005230 012602 10$: MOV (SP)+,R2 ;RESTORE R2
2712 005232 012601 MOV (SP)+,R1 ;RESTORE R1
2713 005234 000205 RTS R5 ;** RETURN
2714
2715

```

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

```

2717 ;*****
2718 ;* WAIT50 - THIS SUBROUTINE STALLS FOR AT LEAST 50 MICRO-SEC, AND THEN RETURNS.
2719 ;*****
2720 005236 010146      000005 WAIT50: MOV     R1, -(SP)      ;SAVE R1
2721 005240 012701      MOV     #5, R1      ;INIT COUNTER
2722 005244 077101      3$: SOB     R1, 3$      ;DELAY HERE FOR 23.8 MICRO-SEC'S
2723 005246 012601      MOV     (SP)+, R1      ;RESTORE R1
2724 005250 000207      RTS     PC      ;RETURN
2725
2726 ; OVERHEAD (JSR, MOV, MOV, MOV, & RTS) ADD UP TO 25.25 MICRO-SEC'S
2727
2728 ; THEREFORE, ACTUAL TOTAL DELAY IS 49.35 MICRO-SECONDS
2729
2730
2731
2732
2733 .SBTTL ....SETVIA -- SET UP VIA REGISTERS
2734 ;*****
2735 ;* SETVIA - SET UP THE VIA REGISTERS
2736 ;*
2737 ;* THIS SUBROUTINE PROGRAMS THE VIA REGISTERS FOR NORMAL OPERATION, BY
2738 ;* LOADING THE DDRB, DDRA, ORB, ACR, PCR, IER.
2739 ;*
2740 ;* CALLING SEQUENCE :
2741 ;* JSR PC,SETVIA
2742 ;*****
2743 005252 SETVIA: JSR     R5,WRITEI      ;SET PORT B FOR OUTPUT MODE
2744 005252 004537 003660 VIADPB
2745 005256 120002 377
2746 005260 000377 JSR     R5,WRITEI      ;SET PORT A FOR INPUT MODE
2747 005262 004537 003660 VIADPA      ; (BIT0 IS ONLY OUTPUT BIT)
2748 005266 120003 001
2749 005270 000001 JSR     R5,WRITEI      ;DISABLE USYRT INTERNAL LOOPBACK
2750 005272 004537 003660 VIAORA
2751 005276 120017 000
2752 005300 000000 JSR     R5,WRITEI      ;INIT PORT B
2753 005302 004537 003660 VIAORB
2754 005306 120000 DTR!RTSND
2755 005310 000030 JSR     R5,WRITEI      ;SET ACR FOR : T1 SQUARE WAVE OUTPUT MODE,
2756 005312 004537 003660 VIAACR      ; T2 ONE-SHOT OUTPUT MODE,
2757 005316 120013 350      ; SR AT SYS CLOCK RATE ON CB1
2758 005320 000350 JSR     R5,WRITEI      ;SET PCR FOR : CB1 NEG TRANS INPUT MODE,
2759 005322 004537 003660 VIAPCR      ; CA2 NEG TRANS INPUT MODE,
2760 005326 120014 022      ; CA1 NEG TRANS INPUT MODE
2761 005330 000022 JSR     R5,WRITEI      ;DISABLE ALL MICRO-INTRPTS
2762 005332 004537 003660 VIAIER
2763 005336 120016 177
2764 005340 000177 RTS     PC      ;RETURN
2765 005342 000207
2766
2767

```

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

2769
 2770
 2771
 2772
 2773
 2774
 2775
 2776
 2777
 2778 005344 004737 003320
 2779 005350 004737 005252
 2780 005354 000207
 2781
 2782
 2783
 2784
 2785
 2786
 2787
 2788
 2789
 2790
 2791
 2792 005356
 2793 005356 004537 003534
 2794 005362 122000
 2795 005364 000000
 2796 005366 122537 005364
 2797 005372 000241
 2798 005374 001430
 2799 005376 012737 000007 002342
 2800 005404 016537 177777 002330
 2801 005412 005037 002332
 2802 005416 113737 005364 002332
 2803
 2804 005424
 005424 012737 000001 002176
 005432 012737 000007 002200
 005440 012737 015500 002202
 005446 012737 021540 002204
 2805 005454 000261
 2806 005456 005205
 2807 005460 000205
 2808
 2809
 2810
 2811

```

.SBTTL ....INIDMV -- INIT DMV (MCLR, VIA SETUP)
;*****
;* INIDMV - THIS SUBROUTINE INITIALIZES THE DMV-11, BY DOING A MASTER CLEAR,
;* ENTERING THE M-LOOP, AND PROGRAMMING THE VIA REGS FOR DEFAULT
;* OPERATION.
;*
;* CALLING SEQUENCE :
;* JSR PC,INIDMV
;*****
INIDMV: JSR PC,MSTCLR ;MASTER CLR, M-LOOP
        JSR PC,SETVIA ;PROGRAM VIA
        RTS PC ;RETURN
  
```

```

.SBTTL ....CKUSTS -- CHECK USYRT STATUS REGISTERS
;*****
;* CKUSTS - THIS SUBROUTINE CHECKS THE USYRT STATUS BY READING THE USYRT
;* STATUS REGISTER AND COMPARING IT TO THE LOW BYTE OF THE WORD FOLLOWING
;* THE CALL. IF THERE IS A MISMATCH, THE SUBROUTINE STACKS THE ERROR
;* INFORMATION, AND SETS THE "C" BIT AND RETURNS.
;*****
CKUSTS: JSR R5,READI ;READ USYRT STATUS REGISTER
        USTATR
1$: .WORD 0
        CMPB (R5)+,1$ ;SEE IF STATUS MATCHES EXPECTED
        CLC ;CLEAR C BIT
        BEQ 2$ ;BR IF STATUS OK
        MOV #7,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
        MOV -1(R5),GDATA ;GET EXPECTED DATA
        CLR BDATA ;GET ACTUAL DATA
        MOVB 1$,BDATA
;STACK "USYRT STATUS INCORRECT" ERROR
        GTDF EM68,ERR10
; QUEUE "DEVICE FATAL" ERROR # 7
        MOV #T.EDF,ERRTYP
        MOV #7,ERRNBR
        MOV #EM68,ERRMSG
        MOV #ERR10,ERRBLK
2$: SEC ;SET C BIT FOR ERROR
        INC R5 ;INCREMENT R5 PAST ARGUMENT
        RTS R5 ;RETURN
  
```

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

```

2813 .SBTTL ....CKTACT -- CHECK TRANSMITTER ACTIVE (TXACT)
2814 ;*****
2815 ;* CKTACT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TXACT IN THE USYRT
2816 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2817 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2818 ;*
2819 ;* CALLING SEQUENCE :
2820 ;* JSR R5,CKTACT
2821 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TXACT>
2822 ;*****
2823 005462 CKTACT:
2824 005462 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2825 005470 004537 003534 JSR R5,READI ;READ USYRT STATUS
2826 005474 122000 USTATR
2827 005476 000000 1$: .WORD 0
2828 005500 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TXACT
2829 005504 001422 BEQ 2$ ;BR IF EXPECTED TXACT = 0
2830 005506 132737 000004 005476 BITB #TXACT,1$ ;SEE IF TXACT = 1
2831 005514 001040 BNE 3$ ;BR IF TXACT = 1
2832 ;STACK "TXACT NOT SET" MSG
2833 005516 GTDF EM69,ERR12
; QUEUE "DEVICE FATAL" ERROR # 8
MOV #T.EDF,ERRTYP
MOV #8,ERRNBR
MOV #EM69,ERRMSG
MOV #ERR12,ERRBLK
005516 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
005524 012737 000010 002200 BR 4$ ;TAKE ERROR EXIT
005532 012737 015527 002202 2$: BITB #TXACT,1$ ;SEE IF TXACT = 0
005540 012737 021714 002204 BEQ 3$ ;BR IF TXACT = 0
2834 005546 000261 ;STACK "TXACT NOT CLEARED" MSG
2835 005550 000423 GTDF EM70,ERR12
2836 005552 132737 000004 005476 3$: SEC ;SET C BIT TO FLAG ERROR
2837 005560 001416 BR 4$ ;TAKE ERROR EXIT
2838 ;STACK "TXACT NOT CLEARED" MSG
2839 005562 GTDF EM70,ERR12
; QUEUE "DEVICE FATAL" ERROR # 9
MOV #T.EDF,ERRTYP
MOV #9,ERRNBR
MOV #EM70,ERRMSG
MOV #ERR12,ERRBLK
005562 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
005570 012737 000011 002200 BR 4$ ;TAKE ERROR EXIT
005576 012737 015545 002202 3$: CLC ;CLEAR C BIT FOR NO ERRORS
005604 012737 021714 002204 4$: RTS R5 ;RETURN
2840 005612 000261
2841 005614 000401
2842 005616 000241
2843 005620 000205
2844
2845
2846
2847

```

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

```

2849 .SBTTL ....CKRACT -- CHECK RECEIVER ACTIVE (RXACT)
2850 ;*****
2851 ;* CKRACT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RXACT IN THE USYRT
2852 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2853 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2854 ;*
2855 ;* CALLING SEQUENCE :
2856 ;* JSR R5,CKRACT
2857 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RXACT>
2858 ;*****
2859 005622 CKRACT:
2860 005622 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2861 005630 004537 003534 JSR R5,READI ;READ USYRT STATUS
2862 005634 122000 USTATR
2863 005636 000000 1$: .WORD 0
2864 005640 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF RXACT
2865 005644 001422 BEQ 2$ ;BR IF EXPECTED RXACT = 0
2866 005646 132737 000040 005636 BITB #RXACT,1$ ;SEE IF RXACT = 1
2867 005654 001040 BNE 3$ ;BR IF RXACT = 1
2868 ;STACK "RXACT NOT SET" MSG
2869 005656 GTDF EM71,ERR12
; QUEUE "DEVICE FATAL" ERROR # 10
MOV #T.EDF,ERRTYP
MOV #10,ERRNBR
MOV #EM71,ERRMSG
MOV #ERR12,ERRBLK
2870 005656 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
2871 005664 012737 000012 002200 BR 4$ ;TAKE ERROR EXIT
2872 005672 012737 015567 002202 2$: BITB #RXACT,1$ ;SEE IF RXACT = 0
2873 005700 012737 021714 002204 BEQ 3$ ;BR IF RXACT = 0
2874 ;STACK "RXACT NOT CLEARED" MSG
2875 005722 GTDF EM72,ERR12
; QUEUE "DEVICE FATAL" ERROR # 11
MOV #T.EDF,ERRTYP
MOV #11,ERRNBR
MOV #EM72,ERRMSG
MOV #ERR12,ERRBLK
2876 005722 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
2877 005730 012737 000013 002200 BR 4$ ;TAKE ERROR EXIT
2878 005736 012737 015605 002202 3$: CLC ;CLEAR C BIT FOR NO ERRORS
2879 005744 012737 021714 002204 4$: RTS R5 ;RETURN
2880
2881
2882
2883

```


....CKTBMT -- CHECK TRANSMIT BUFFER EMPTY

```

2885 .SBTTL ....CKTBMT -- CHECK TRANSMIT BUFFER EMPTY
2886 ;*****
2887 ;* CKTBMT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TBMT IN THE USYRT
2888 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2889 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2890 ;*
2891 ;* CALLING SEQUENCE :
2892 ;* JSR R5,CKTBMT
2893 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TBMT>
2894 ;*****
2895 005762 CKTBMT:
2896 005762 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2897 005770 004537 003534 JSR R5,READI ;READ USYRT STATUS
2898 005774 122000 USTATR
2899 005776 000000 1$: .WORD 0
2900 006000 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TBMT
2901 006004 001422 BEQ 2$ ;BR IF EXPECTED TBMT = 0
2902 006006 132737 000100 005776 BITB #TBMT,1$ ;SEE IF TBMT = 1
2903 006014 001040 BNE 3$ ;BR IF TBMT = 1
2904 ;STACK "TBMT NOT SET" MSG
2905 006016 GTDF EM73,ERR12
; QUEUE "DEVICE FATAL" ERROR # 12
MOV #T.EDF,ERRTYP
MOV #12,ERRNBR
MOV #EM73,ERRMSG
MOV #ERR12,ERRBLK
2906 006046 000261 SEC ;SET C BIT TO FLAG ERROR
2907 006050 000423 BR 4$ ;TAKE ERROR EXIT
2908 006052 132737 000100 005776 2$: BITB #TBMT,1$ ;SEE IF TBMT = 0
2909 006060 001416 BEQ 3$ ;BR IF TBMT = 0
2910 ;STACK "TBMT NOT CLEARED" MSG
2911 006062 GTDF EM74,ERR12
; QUEUE "DEVICE FATAL" ERROR # 13
MOV #T.EDF,ERRTYP
MOV #13,ERRNBR
MOV #EM74,ERRMSG
MOV #ERR12,ERRBLK
2912 006112 000261 SEC ;SET C BIT TO FLAG ERROR
2913 006114 000401 BR 4$ ;TAKE ERROR EXIT
2914 006116 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
2915 006120 000205 4$: RTS R5 ;RETURN
2916
2917
2918
2919

```

....CKRDA -- CHECK RECEIVE DATA AVAILABLE

```

2921 .SBTTL ....CKRDA -- CHECK RECEIVE DATA AVAILABLE
2922 ;*****
2923 ;* CKRDA - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RDA IN THE USYRT
2924 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2925 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2926 ;*
2927 ;* CALLING SEQUENCE :
2928 ;* JSR R5,CKRDA
2929 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RDA>
2930 ;*****
2931 006122 CKRDA:
2932 006122 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2933 006130 004537 003534 JSR R5,READI ;READ USYRT STATUS
2934 006134 122000 USTATR
2935 006136 000000 1$: .WORD 0
2936 006140 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF RDA
2937 006144 001422 BEQ 2$ ;BR IF EXPECTED RDA = 0
2938 006146 132737 000200 006136 BITB #RDA,1$ ;SEE IF RDA = 1
2939 006154 001040 BNE 3$ ;BR IF RDA = 1
2940 ;STACK "RDA NOT SET" MSG
2941 006156 GTDF EM75,ERR12
; QUEUE "DEVICE FATAL" ERROR # 14
MOV #T.EDF,ERRTYP
MOV #14,ERRNBR
MOV #EM75,ERRMSG
MOV #ERR12,ERRBLK
006156 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
006164 012737 000016 002200 BR 4$ ;TAKE ERROR EXIT
006172 012737 015665 002202 2$: BITB #RDA,1$ ;SEE IF RDA = 0
006200 012737 021714 002204 BEQ 3$ ;BR IF RDA = 0
2942 006206 000261 SEC ;SET C BIT TO FLAG ERROR
2943 006210 000423 BR 4$ ;TAKE ERROR EXIT
2944 006212 132737 000200 006136 2$: BITB #RDA,1$ ;SEE IF RDA = 0
2945 006220 001416 BEQ 3$ ;BR IF RDA = 0
2946 ;STACK "RDA NOT CLEARED" MSG
2947 006222 GTDF EM76,ERR12
; QUEUE "DEVICE FATAL" ERROR # 15
MOV #T.EDF,ERRTYP
MOV #15,ERRNBR
MOV #EM76,ERRMSG
MOV #ERR12,ERRBLK
006222 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
006230 012737 000017 002200 BR 4$ ;TAKE ERROR EXIT
006236 012737 015701 002202 3$: CLC ;CLEAR C BIT FOR NO ERRORS
006244 012737 021714 002204 4$: RTS R5 ;RETURN
2948 006252 000261 SEC ;SET C BIT TO FLAG ERROR
2949 006254 000401 BR 4$ ;TAKE ERROR EXIT
2950 006256 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
2951 006260 000205 4$: RTS R5 ;RETURN
2952
2953
2954
2955

```

....CKRSA -- CHECK RECEIVER STATUS AVAILABLE

```

2957 .SBTTL ....CKRSA -- CHECK RECEIVER STATUS AVAILABLE
2958 ;*****
2959 ;* CKRSA - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RSA IN THE USYRT
2960 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2961 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2962 ;*
2963 ;* CALLING SEQUENCE :
2964 ;* JSR R5,CKRSA
2965 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RSA>
2966 ;*****
2967 006262 CKRSA:
2968 006262 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2969 006270 004537 003534 JSR R5,READI ;READ USYRT STATUS
2970 006274 122000 USTATR
2971 006276 000000 1#: .WORD 0
2972 006300 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF RSA
2973 006304 001422 BEQ 2# ;BR IF EXPECTED RSA = 0
2974 006306 132737 000020 006276 BITB #RSA,1# ;SEE IF RSA = 1
2975 006314 001040 BNE 3# ;BR IF RSA = 1
2976 ;STACK "RSA NOT SET" MSG
2977 006316 GTDF EM77,ERR12
; QUEUE "DEVICE FATAL" ERROR # 16
MOV #T.EDF,ERRTYP
MOV #16,ERRNBR
MOV #EM77,ERRMSG
MOV #ERR12,ERRBLK
006316 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
006324 012737 000020 002200 BR 4# ;TAKE ERROR EXIT
006332 012737 015721 002202 2#: BITB #RSA,1# ;SEE IF RSA = 0
006340 012737 021714 002204 BEQ 3# ;BR IF RSA = 0
2978 006346 000261 ;STACK "RSA NOT CLEARED" MSG
2979 006350 000423 GTDF EM78,ERR12
2980 006352 132737 000020 006276 2#: BITB #RSA,1#
2981 006360 001416 BEQ 3#
2982 ;STACK "RSA NOT CLEARED" MSG
2983 006362 GTDF EM78,ERR12
; QUEUE "DEVICE FATAL" ERROR # 17
MOV #T.EDF,ERRTYP
MOV #17,ERRNBR
MOV #EM78,ERRMSG
MOV #ERR12,ERRBLK
006362 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
006370 012737 000021 002200 BR 4# ;TAKE ERROR EXIT
006376 012737 015735 002202 3#: CLC ;CLEAR C BIT FOR NO ERRORS
006404 012737 021714 002204 4#: RTS R5 ;RETURN
2984 006412 000261
2985 006414 000401
2986 006416 000241
2987 006420 000205
2988
2989

```

....CKROR -- CHECK RECEIVER OVERRUN

2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011

006456
006464
006472
006500
3012
3013
3014
3015
3016
3017

006522
006530
006536
006544
3018
3019
3020
3021
3022
3023
3024

```

.SBTTL ....CKROR -- CHECK RECEIVER OVERRUN
;*****
;* CKROR - THIS SUBROUTINE CHECKS FOR THE OCCURANCE OF RECEIVER OVERRUN IN THE
;* USYRT RECEIVER STATUS REGISTER (RDSRH), AND REPORTS AN ERROR IF IT IS
;* NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
;*
;* CALLING SEQUENCE :
;* JSR R5,CKROR
;* .WORD <BIT 0 IS EXPECTED VALUE OF ROR>
;*****
CKROR:
MOV #1,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
JSR R5,READI ;READ RECEIVER STATUS
RDSRH
1$: .WORD 0
BIT #BIT0,(R5) ;GET EXPECTED STATE OF ROR
BEQ 2$ ;BR IF EXPECTED ROR = 0
BITB #ROR,1$ ;SEE IF ROR = 1
BNE 3$ ;BR IF ROR = 1
;STACK "RECEIVER OVRN NOT SET" MSG
GDF EM90,ERR12
; QUEUE "DEVICE FATAL" ERROR # 18
MOV #T.EDF,ERRTYP
MOV #18,ERRNBR
MOV #EM90,ERRMSG
MOV #ERR12,ERRBLK
SEC ;SET C BIT TO FLAG ERROR
BR 4$ ;TAKE ERROR EXIT
2$: BITB #ROR,1$ ;SEE IF ROR = 0
BEQ 3$ ;BR IF ROR = 0
;STACK "ROR NOT CLEARED" MSG
GDF EM91,ERR12
; QUEUE "DEVICE FATAL" ERROR # 19
MOV #T.EDF,ERRTYP
MOV #19,ERRNBR
MOV #EM91,ERRMSG
MOV #ERR12,ERRBLK
3$: CLC
4$: RTS R5 ;SET C BIT TO FLAG ERROR
;TAKE ERROR EXIT
;CLEAR C BIT FOR NO ERRORS
;RETURN

```

....CKSEOM -- CHECK RSOM, REOM

```

3026 .SBTTL ....CKSEOM -- CHECK RSOM, REOM
3027 ;*****
3028 ;* CKSEOM - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF RSOM, REOM IN THE
3029 ;* USYRT RECEIVER STATUS REG (RDSRH) AND REPORTS AN ERROR IF THEY ARE NOT
3030 ;* PROPERLY SET TO THE STATES OF BITS 0,1 IN THE WORD FOLLOWING THE CALL.
3031 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3032 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3033 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3034 ;*
3035 ;* CALLING SEQUENCE :
3036 ;* JSR R5,CKSEOM
3037 ;* <BIT 0 IS EXPECTED VALUE OF RSOM, BIT 1 IS VALUE OF REOM>
3038 ;*****
3039 006562 CKSEOM:
3040 006562 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3041 006570 004537 003534 JSR R5,READI ;READ USYRT RECEIVER STATUS
3042 006574 120401 RDSRH
3043 006576 000000 1$: .WORD 0
3044 006600 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF RSOM
3045 006604 001422 BEQ 2$ ;BR IF EXPECTED RSOM = 0
3046 006606 132737 000001 006576 BITB #RSOM,1$ ;SEE IF RSOM = 1
3047 006614 001040 BNE 3$ ;BR IF RSOM = 1
3048 ;STACK "RSOM NOT SET" MSG
3049 006616 GTDF EM29,ERR12
; QUEUE "DEVICE FATAL" ERROR # 20
; MOV #T.EDF,ERRTYP
; MOV #20,ERRNBR
; MOV #EM29,ERRMSG
; MOV #ERR12,ERRBLK
006616 012737 000001 002176
006624 012737 000024 002200
006632 012737 014501 002202
006640 012737 021714 002204
3050 006646 000261 SEC ;SET C BIT TO FLAG ERROR
3051 006650 000473 BR 6$ ;TAKE ERROR EXIT
3052 006652 132737 000001 006576 2$: BITB #RSOM,1$ ;SEE IF RSOM = 0
3053 006660 001416 BEQ 3$ ;BR IF RSOM = 0
3054 ;STACK "RSOM NOT CLEARED" MSG
3055 006662 GTDF EM28,ERR12
; QUEUE "DEVICE FATAL" ERROR # 21
; MOV #T.EDF,ERRTYP
; MOV #21,ERRNBR
; MOV #EM28,ERRMSG
; MOV #ERR12,ERRBLK
006662 012737 000001 002176
006670 012737 000025 002200
006676 012737 014460 002202
006704 012737 021714 002204
3056 006712 000261 SEC ;SET C BIT TO FLAG ERROR
3057 006714 000451 BR 6$ ;TAKE ERROR EXIT
3058 006716 032765 000002 177776 3$: BIT #BIT1,-2(R5) ;GET EXPECTED STATE OF REOM
3059 006724 001422 BEQ 4$ ;BR IF EXPECTED REOM = 0
3060 006726 132737 000002 006576 BITB #REOM,1$ ;SEE IF REOM = 1
3061 006734 001040 BNE 5$ ;BR IF REOM = 1
3062 ;STACK "REOM NOT SET" MSG
3063 006736 GTDF EM31,ERR12
; QUEUE "DEVICE FATAL" ERROR # 22
; MOV #T.EDF,ERRTYP
; MOV #22,ERRNBR
; MOV #EM31,ERRMSG
; MOV #ERR12,ERRBLK
006736 012737 000001 002176
006744 012737 000026 002200
006752 012737 014537 002202
006760 012737 021714 002204
3064 006766 000261 SEC ;SET C BIT TO FLAG ERROR
3065 006770 000423 BR 6$ ;TAKE ERROR EXIT
3066 006772 132737 000002 006576 4$: BITB #REOM,1$ ;SEE IF REOM = 0
3067 007000 001416 BEQ 5$ ;BR IF REOM = 0

```

E6

....CKSEOM -- CHECK RSOM, REOM

3068 ;STACK "REOM NOT CLEARED" MSG
3069 007002 GTDF EM30,ERR12

; QUEUE "DEVICE FATAL" ERROR # 23
MOV @T.EDF,ERR1YP
MOV @23,ERRNBR
MOV @EM30,ERRMSG
MOV @ERR12,ERRBLK

007002 012737 000001 002176
007010 012737 000027 002200
007016 012737 014516 002202
007024 012737 021714 002204

3070 007032 000261 SEC
3071 007034 000401 BR 6#
3072 007036 000241 5#: CLC
3073 007040 000205 6#: RTS R5
3074
3075

;SET C BIT TO FLAG ERROR
;TAKE ERROR EXIT
;CLEAR C BIT FOR NO ERRORS
;RETURN

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

```

3077 .SBTTL ....CHKTSO -- CHECK TRANSMIT SERIAL OUT BIT
3078 ;*****
3079 ;* CHKTSO - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TSO IN THE USYRT
3080 ;* STATUS REGISTER, AND SETS THE "C" BIT IF IT IS NOT SET TO THE STATE
3081 ;* OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3082 ;*
3083 ;* CALLING SEQUENCE :
3084 ;* JSR R5,CHKTSO
3085 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TSO>
3086 ;*****
3087 007042 CHKTSO:
3088 007042 012737 000007 002342 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3089 007050 004537 003534 JSR R5,READI ;READ USYRT STATUS
3090 007054 122000 USTATR
3091 007056 000000 1$: .WORD 0
3092 007060 032725 000001 BIT #BIT0,(R5)+ ;GET EXPECTED STATE OF TSO
3093 007064 001422 BEQ 2$ ;BR IF EXPECTED TSO = 0
3094 007066 132737 000010 007056 BITB #TSO,1$ ;SEE IF TSO = 1
3095 007074 001040 BNE 3$ ;BR IF TSO = 1
3096 ;*** STACK "TSO NOT SET" ERROR ***
3097 007076 GTDF EM100,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 24
; MOV #T.EDF,ERRTYP
; MOV #24,ERRNBR
; MOV #EM100,ERRMSG
; MOV #ERR12,ERRBLK
007076 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
007104 012737 000030 002200 BR 4$ ;TAKE ERROR EXIT
007112 012737 016422 002202
007120 012737 021714 002204
3098 007126 000261 SEC ;SET C BIT TO FLAG ERROR
3099 007130 000423 BR 4$ ;TAKE ERROR EXIT
3100
3101 007132 132737 000010 007056 2$: BITB #TSO,1$ ;SEE IF TSO = 0
3102 007140 001416 BEQ 3$ ;BR IF TSO = 0
3103 ;*** STACK "TSO NOT CLEARED" ERROR ***
3104 007142 GTDF EM101,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 25
; MOV #T.EDF,ERRTYP
; MOV #25,ERRNBR
; MOV #EM101,ERRMSG
; MOV #ERR12,ERRBLK
007142 012737 000001 002176 SEC ;SET C BIT TO FLAG ERROR
007150 012737 000031 002200 BR 4$ ;TAKE ERROR EXIT
007156 012737 016442 002202
007164 012737 021714 002204
3105 007172 000261 SEC ;SET C BIT TO FLAG ERROR
3106 007174 000401 BR 4$ ;TAKE ERROR EXIT
3107 007176 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3108 007200 000205 4$: RTS R5 ;RETURN
3109

```

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

```

3111 .SBTTL ....SERIAL -- READ/CHECK TX CHARACTER VIA TSO BIT
3112 ;*****
3113 ;* SERIAL - THIS SUBROUTINE SERIALLY READS/CLOCKS/CHECKS A CHARACTER FROM
3114 ;* THE TRANSMIT SERIAL OUT (TSO) BIT OF THE USYRT STATUS REGISTER,
3115 ;* AND STACKS MESSAGE/SETS "C" BIT IF AN INCORRECT CHARACTER IS READ.
3116 ;* NOTE: "EXPECTED VALUE" ARGUMENT IS ALWAYS READ RIGHT-TO-LEFT.
3117 ;*
3118 ;* CALLING SEQUENCE :
3119 ;* JSR R5,SERIAL
3120 ;* .WORD <# OF BITS TO BE READ>
3121 ;* .WORD <EXPECTED VALUE OF SERIAL BIT STREAM>
3122 ;*****
3123 SERIAL:
3124 007202 010146 MOV R1,-(SP) ;SAVE R1
3125 007204 010246 MOV R2,-(SP) ;SAVE R2 (TICKS)
3126 007206 010346 MOV R3,-(SP) ;SAVE R3 (EXPECTED_WORD)
3127
3128 007210 005001 CLR R1 ;CLEAR ASSEMBLED_WORD
3129 007212 012502 MOV (R5)+,R2 ;GET # OF TICKS
3130
3131 007214 006301 1$: ASL R1 ;SHIFT ASSEMBLED_WORD
3132 007216 004537 011540 JSR R5,STEPLU ;CLOCK USYRT ONCE
3133 007222 000001 1
3134
3135 007224 004537 007042 JSR R5,CHKTSO ;CHECK FOR TSO=1
3136 007230 000001 1
3137 007232 103401 BCS 2$ ;BR IF TSO=0
3138 007234 005201 INC R1 ;TSO=1: SET LSB OF ASSEMBLED_WORD
3139 007236 077212 2$: SOB R2,1$ ;LOOP UNTIL NO MORE TICKS
3140
3141 007240 012503 MOV (R5)+,R3 ;GET EXPECTED_WORD
3142 007242 020103 CMP R1,R3 ;COMPARE EXPECTED_ AND ASSEMBLED_WORD
3143 007244 001422 BEQ 3$ ;BR IF CORRECT VALUE READ
3144
3145 007246 010337 002330 MOV R3,GDATA ;EXPECTED_WORD => GDATA
3146 007252 010137 002332 MOV R1,BDATA ;ASSEMBLED_WORD => BDATA
3147 ;*** STACK "TRANSMISSION ERROR" MSG ***
3148 007256 GTDF EM106,ERR13
; QUEUE "DEVICE FATAL" ERROR # 26
; MOV #T.EDF,ERRTYP
; MOV #26,ERRNBR
; MOV #EM106,ERRMSG
; MOV #ERR13,ERRBLK
007256 012737 000001 002176
007264 012737 000032 002200
007272 012737 016735 002202
007300 012737 022030 002204
3149 007306 000261 SEC ;SET C BIT TO FLAG ERROR
3150 007310 000401 BR .+4 ;TAKE ERROR EXIT
3151
3152 007312 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3153 007314 012603 MOV (SP)+,R3 ;RESTORE REGISTERS
3154 007316 012602 MOV (SP)+,R2
3155 007320 012601 MOV (SP)+,R1
3156 007322 000205 4$: RTS R5 ;RETURN
3157
3158

```


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

```

3160 .SBTTL ....INITRN -- INIT TRANSMISSION OF A MESSAGE
3161 ;*****
3162 ;* INITRN - THIS SUBROUTINE INITIATES TRANSMISSION OF A MESSAGE, BY LOADING
3163 ;* THE USYRT PCSARL,H AND THE PCR WITH THE DATA PASSED IN THE 2 WORDS
3164 ;* FOLLOWING THE CALL ; LOADING AND CLOCKING 1 SOM UNTIL THE FIRST
3165 ;* SYNCH OR FLAG HAS BEEN SERIALIZED IN THE USYRT. THE PROGRAM MONITORS
3166 ;* ALL THE FLAGS IN THE USYRT STATUS REGISTER THROUGHOUT THE PROCESS.
3167 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION IS STACKED
3168 ;* AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE DISCRETION
3169 ;* OF THE CALLING ROUTINE OR SUBROUTINE.
3170 ;*
3171 ;* CALLING SEQUENCE :
3172 ;* JSR R5,INITRN
3173 ;* .WORD <VALUE TO LOAD INTO USYRT PCSARL,H>
3174 ;* .WORD <VALUE TO LOAD INTO USYRT PCR (PASSED IN LO BYTE)>
3175 ;* <SPECIAL VIAORB MASKING VALUE (PASSED IN HI BYTE)>
3176 ;*****
3177 007324 INITRN:
3178 007324 010146 MOV R1,-(SP) ;SAVE R1
3179 007326 004537 003660 JSR R5,WRITEI ;RESET THE USYRT
3180 007332 120000 VIAORB
3181 007334 000031 RTSND!DTR!PRESET
3182 007336 004537 003660 JSR R5,WRITEI ;CLEAR USYRT RESET BIT
3183 007342 120000 VIAORB
3184 007344 000030 RTSND!DTR
3185 007346 112537 007360 MOVB (R5)+,1$ ;GET VALUE TO LOAD INTO USYRT PCSARL
3186 007352 004537 003660 JSR R5,WRITEI ;LOAD USYRT PCSARL
3187 007356 120404 PCSARL
3188 007360 000000 1$: .WORD 0
3189 007362 112537 007374 MOVB (R5)+,2$ ;GET VALUE TO LOAD INTO PCSARH
3190 007366 004537 003660 JSR R5,WRITEI ;LOAD USYRT PCSARH
3191 007372 120405 PCSARH
3192 007374 000000 2$: .WORD 0
3193 007376 112537 007422 MOVB (R5)+,3$ ;GET VALUE TO LOAD INTO PCR
3194 007402 005037 002406 CLR SAVLEN
3195 007406 113737 007422 002406 MOVB 3$,SAVLEN ;SAVE CHAR LENGTH BITS
3196 007414 004537 003660 JSR R5,WRITEI ;LOAD USYRT PCR
3197 007420 120407 PCR
3198 007422 000000 3$: .WORD 0
3199 007424 004537 003660 JSR R5,WRITEI ;SET ACR FOR T1 ONE-SHOT MODE
3200 007430 120013 VIAACR
3201 007432 000200 200
3202 007434 004537 003660 JSR R5,WRITEI ;LOAD VIA T1L-L
3203 007440 120006 VIAT1C
3204 007442 000300 300
3205 007444 004537 003660 JSR R5,WRITEI ;LOAD VIA T1L-H
3206 007450 120007 VIAT1D
3207 007452 000000 000
3208 007454 004537 005356 JSR R5,CKUSTS ;CHK USYRT STATUS FOR INIT'D STATE
3209 007460 000110 110 ; TBMT = 1, TSO = 1
3210 007462 103454 BCS 7$ ;IF ERROR, EXIT SUBROUTINE
3211
3212 007464 013737 007620 007504 MOV 20$,13$ ;* SET UP DEFAULT VIAORB PARAMETERS
3213 007472 142537 007504 BICB (R5)+,13$ ;* CLEAR ANY SPECIFIED VIAORB BITS.
3214
3215 007476 004537 003660 JSR R5,WRITEI ;SET UP USYRT
3216 007502 120000 VIAORB

```

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

```

3217 007504 000142          13$:  TXEN!RXEN!TTLOOP          ;* THIS VALUE MIGHT BE MODIFIED ABOVE
3218
3219 007506 004537 003660    JSR      R5,WRITEI          ;SET TSOM IN USYRT
3220 007512 120403          TDSRH
3221 007514 000001          TSOM
3222 007516 004537 003660    JSR      R5,WRITEI          ;LOAD SYNCH CHAR INTO TX BUF
3223 007522 120402          TDSRL
3224 007524 000226          SYNCH
3225 007526 004537 005762    JSR      R5,CKTBMT          ;CHK FOR TBMT = 0
3226 007532 000000          0
3227 007534 103427          BCS      7$                ;IF ERROR, EXIT SUBROUTINE
3228 007536 005001          CLR      R1                ;INIT CYCLE COUNTER
3229 007540 004537 011540    4$:  JSR      R5,STEPLU          ;CLOCK LU FOR 1 CYCLE
3230 007544 000001          1
3231 007546 004537 003534    JSR      R5,READI          ;READ USYRT STATUS REG
3232 007552 122000          USTATR
3233 007554 000000          .WORD  0
3234 007556 132737 000100 007554 5$:  BITB    @TBMT,5$          ;SEE IF TBMT IS SET YET
3235 007564 001010          BNE     6$                ;BR IF YES
3236 007566 005201          INC     R1                ;INCR CYCLE COUNTER
3237 007570 020127 000003    CMP     R1,@3             ;SEE IF 3 CYCLES DONE YET
3238 007574 002761          BLT     4$                ;BR IF LESS THAN 3 CYCLES
3239 007576 004537 005762    JSR      R5,CKTBMT          ;GO STACK "TBMT NOT SET" MSG
3240 007602 000001          1
3241 007604 103403          BCS     7$                ;IF ERROR, EXIT SUBROUTINE
3242 007606 004537 005462    6$:  JSR      R5,CKTACT          ;CHK FOR TXACT = 1
3243 007612 000001          1
3244 007614 012601          7$:  MOV     (SP)+,R1          ;RESTORE R1
3245 007616 000205          RTS     R5                ;RETURN (IF C = 1, WE HAD AN ERROR)
3246
3247 007620 000142          20$:  TXEN!RXEN!TTLOOP          ;DEFAULT VALUE FOR VIAORB: ENABLE
3248
3249

```

....TXCHAR -- TRANSMIT A CHARACTER

```

3251 .SBTTL ....TXCHAR -- TRANSMIT A CHARACTER
3252 ;*****
3253 ;* TXCHAR - THIS SUBROUTINE INITIATES TRANSMISSION OF A CHAR BY LOADING
3254 ;* THE USYRT TDSRL WITH THE DATA PASSED IN THE LO BYTE OF THE WORD
3255 ;* FOLLOWING THE CALL, AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES
3256 ;* PASSED IN THE SECOND WORD FOLLOWING THE CALL. THE PROGRAM CONTINUALLY
3257 ;* MONITORS TBMT AND TXACT THROUGHOUT THE PROCESS.
3258 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3259 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3260 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3261 ;*
3262 ;* CALLING SEQUENCE :
3263 ;* JSR R5,TXCHAR
3264 ;* .WORD <DATA FOR TDSRL IN LO BYTE>
3265 ;* .WORD <NUMBER OF CYCLES TO CLOCK (IN LO BYTE)>
3266 ;* <SWITCH TO DISABLE INITIAL TBMT=0 CHECK (MSB IN HI BYTE)>
3267 ;*****
3268 007622 TXCHAR:
3269 007622 010146 MOV R1,-(SP) ;SAVE R1
3270 007624 010246 MOV R2,-(SP) ;SAVE R2
3271 007626 012537 007640 MOV (R5)+,1$ ;GET DATA FOR TDSRL
3272 007632 004537 003660 JSR R5,WRITEI ;LOAD DATA INTO TDSRL
3273 007636 120402 TDSRL
3274 007640 000000 1$: .WORD 0
3275 007642 005001 CLR R1 ;INIT CYCLE COUNT AND CLEAR C BIT
3276 007644 005002 CLR R2 ;CLEAR REQ'D CYCLE COUNT
3277 007646 112502 MOVB (R5)+,R2 ;GET DESIRED NO. OF CYCLES
3278 007650 001425 BEQ 6$ ;BR IF NO CLOCKING DONE
3279 007652 004537 005462 3$: JSR R5,CKTACT ;CHECK TXACT = 1
3280 007656 000001 1
3281 007660 103421 BCS 6$ ;BR TO EXIT IF ERROR
3282 007662 020102 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
3283 007664 001414 BEQ 5$ ;BR IF YES
3284
3285 007666 131527 000200 BITB (R5),#NCTBMT ;* CHECK FOR "TBMT=0 CHECK" DISABLE
3286 007672 001004 BNE 7$ ;* BR IF MSB IS NOT SET
3287
3288 007674 004537 005762 JSR R5,CKTBMT ;CHECK FOR TBMT = 0
3289 007700 000000 0
3290 007702 103410 BCS 6$ ;BR TO EXIT IF ERROR
3291 007704 004537 011540 7$: JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
3292 007710 000001 1
3293 007712 005201 INC R1 ;INCR CYCLE COUNT
3294 007714 000756 BR 3$ ;KEEP CLOCKING
3295 007716 004537 005762 5$: JSR R5,CKTBMT ;CHK TBMT = 1
3296 007722 000001 1
3297 007724 012602 6$: MOV (SP)+,R2 ;RESTORE R2
3298 007726 012601 MOV (SP)+,R1 ;RESTORE R1
3299 007730 005205 INC R5 ;ADJUST R5 FOR SANE RETURN
3300 007732 000205 RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)
3301
3302
3303
3304

```

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

```

3306 .SBTTL ....TXCTRL -- CONTROL MESSAGE TRANSMISSION (TDSRH)
3307 :*****
3308 :* TXCTRL - THIS SUBROUTINE ALLOWS CONTROL OF MESSAGE TRANSMISSION BY LOADING
3309 :* THE USYRT TDSRH WITH THE DATA PASSED IN THE LO BYTE OF THE WORD
3310 :* FOLLOWING THE CALL, AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES
3311 :* PASSED IN THE SECOND WORD FOLLOWING THE CALL. THE PROGRAM CONTINUALLY
3312 :* MONITORS TBMT AND TXACT THROUGHOUT THE PROCESS.
3313 :* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3314 :* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3315 :* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3316 :*
3317 :* CALLING SEQUENCE :
3318 :* JSR R5,TXCTRL
3319 :* .WORD <DATA FOR TDSRH IN LO BYTE>
3320 :* .WORD <NUMBER OF CYCLES TO CLOCK>
3321 :*****
3322 TXCTRL:
3323 007734 010146 MOV R1,-(SP) ;SAVE R1
3324 007736 010246 MOV R2,-(SP) ;SAVE R2
3325 007740 012537 007752 MOV (R5)+,2$ ;GET DATA FOR TDSRH
3326 007744 004537 003660 JSR R5,WRITEI ;LOAD DATA INTO TDSRH
3327 007750 120403 TDSRH
3328 007752 000000 2$: .WORD 0
3329 007754 005001 CLR R1 ;INIT CYCLE COUNT AND CLEAR C BIT
3330 007756 012502 MOV (R5)+,R2 ;GET DESIRED NO. OF CYCLES
3331 007760 001422 BEQ 6$ ;BR IF NO CLOCKING DONE
3332 007762 004537 005462 3$: JSR R5,CKTACT ;CHECK TXACT = 1
3333 007766 000001 1
3334 007770 103416 BCS 6$ ;BR TO EXIT IF ERROR
3335 007772 020102 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
3336 007774 001411 BEQ 5$ ;BR IF YES
3337 007776 004537 005762 JSR R5,CKTBMT ;CHECK FOR TBMT = 0
3338 010002 000000 0
3339 010004 103410 BCS 6$ ;BR TO EXIT IF ERROR
3340 010006 004537 011540 JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
3341 010012 000001 1
3342 010014 005201 INC R1 ;INCR CYCLE COUNT
3343 010016 000761 BR 3$ ;KEEP CLOCKING
3344 010020 004537 005762 5$: JSR R5,CKTBMT ;CHK TBMT = 1
3345 010024 000001 1
3346 010026 012602 6$: MOV (SP)+,R2 ;RESTORE R2
3347 010030 012601 MOV (SP)+,R1 ;RESTORE R1
3348 010032 000205 RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)
3349
3350

```

....RXCHAR -- RECEIVE A CHARACTER

```

3352 .SBTTL ....RXCHAR -- RECEIVE A CHARACTER
3353 ;*****
3354 ;* RXCHAR - THIS SUBROUTINE READS THE USYRT RDSR AND CHECKS THE CONTENTS
3355 ;* AGAINST THE DATA PASSED IN THE WORD FOLLOWING THE CALL.
3356 ;* IF BIT0 = 0 IN THE SECOND WORD FOLLOWING THE CALL, THE RERR BIT IS
3357 ;* NOT CHECKED AGAINST THE EXPECTED VALUE. THEN, IT CLOCKS
3358 ;* THE LINE UNIT FOR THE NO. OF CYCLES PASSED IN THE THIRD WORD
3359 ;* FOLLOWING THE CALL. THE PROGRAM CONTINUALLY MONITORS RDA AND RXACT.
3360 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3361 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3362 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3363 ;*
3364 ;* CALLING SEQUENCE :
3365 ;* JSR R5,RXCHAR
3366 ;* .WORD <EXPECTED RDSRL IN LO BYTE, RDSRH IN HI BYTE>
3367 ;* .WORD <=0 FOR NO RERR CHK, =1 FOR RERR CHK>
3368 ;* .WORD <NUMBER OF CYCLES TO CLOCK (IN LO BYTE)>
3369 ;* <SPECIAL DISABLE SWITCHES: NOCRDA,NFCRDA,NCRACK(IN HI BYTE)>
3370 ;*****
3371 RXCHAR:
3372 010034 010146 MOV R1,-(SP) ;SAVE R1
3373 010036 010246 MOV R2,-(SP) ;SAVE R2
3374 010040 004537 003534 JSR R5,READI ;READ RDSRH
3375 010044 120401 RDSRH
3376 010046 000000 2$: .WORD 0
3377 010050 004537 003534 JSR R5,READI ;READ RDSRL
3378 010054 120400 RDSRL
3379 010056 000000 1$: .WORD 0
3380 010060 111501 MOVB (R5),R1 ;GET EXPECTED RDSRL
3381 010062 042701 177400 BIC #177400,R1 ;MASK OFF UNUSED BITS
3382 010066 023727 002406 000347 CMP SAVLEN,#TXDL!RXDL ;SEE IF 7-BIT CHARS BEING USED
3383 010074 001005 BNE 3$ ;BR IF NOT 7-BIT CHARS
3384 010076 142737 000200 010056 BICB #BIT7,1$ ;CLEAR 8TH BIT FOR COMPARE
3385 010104 142701 000200 BICB #BIT7,R1
3386 010110 123701 010056 3$: CMPB 1$,R1 ;COMPARE RCV'D CHAR TO EXPECTED
3387 010114 001462 BEQ 6$ ;BR IF MATCH
3388 010116 004537 003534 JSR R5,READI ;READ USYRT STATUS REG
3389 010122 122000 USTATR
3390 010124 000000 4$: .WORD 0
3391 010126 132737 000002 010124 BITB #TXU,4$ ;SEE IF TX UNDERRUN OCCURRED
3392 010134 001421 BEQ 5$ ;BR IF NOT
3393 010136 012737 000007 002342 MOV #7,REGNUM ;SET USYRT REG NO. FOR STATUS REG
3394 ;STACK "TX UNDERRUN" ERROR
3395 010144 GTDF EM54,ERR12
; QUEUE "DEVICE FATAL" ERROR # 27
; MOV #T.EDF,ERRTYP
; MOV #27,ERRNBR
; MOV #EM54,ERRMSG
; MOV #ERR12,ERRBLK
3396 010174 000137 011274 JMP 20$ ;TAKE ERROR EXIT
3397 010200 005037 002342 5$: CLR REGNUM ;SET USYRT REG NO. FOR RDSRL
3398 010204 005037 002330 CLR GDATA ;SET EXPECTED DATA
3399 010210 110137 002330 MOVB R1,GDATA
3400 010214 005037 002332 CLR BDATA ;SET ACTUAL DATA
3401 010220 113737 010056 002332 MOVB 1$,BDATA
3402 ;STACK "RCV'D DATA MISCOMPARE" ERROR
3403 010226 GTDF EM34,ERR10

```

....RXCHAR -- RECEIVE A CHARACTER

```

010226 012737 000001 002176
010234 012737 000034 002200
010242 012737 014626 002202
010250 012737 021540 002204
3404 010256 000137 011274
3405 010262 116501 000001
3406 010266 042701 177400
3407 010272 123701 010046
3408 010276 001016
3409 010300 000137 011160
3410 010304 012737 000001 002342
3411 010312 005037 002330
3412 010316 110137 002330
3413 010322 005037 002332
3414 010326 113737 010046 002332
3415 010334 012737 000001 002342
3416 010342 032765 000001 000002
3417 010350 001447
3418
3419 010352 132701 000200
3420 010356 001022
3421 010360 132737 000200 010046
3422 010366 001440
3423
3424 010370

010370 012737 000001 002176
010376 012737 000035 002200
010404 012737 014654 002202
010412 012737 021714 002204
3425 010420 000137 011274
3426 010424 132737 000200 010046
3427 010432 001016
3428
3429 010434

010434 012737 000001 002176
010442 012737 000036 002200
010450 012737 014675 002202
010456 012737 021714 002204
3430 010464 000137 011274
3431
3432 010470 132701 000010
3433 010474 001022
3434 010476 132737 000010 010046
3435 010504 001440
3436
3437 010506

010506 012737 000001 002176
010514 012737 000037 002200
010522 012737 014327 002202
010530 012737 021714 002204
3438 010536 000137 011274
3439 010542 132737 000010 010046
3440 010550 001016

; QUEUE "DEVICE FATAL" ERROR # 28
; TAKE ERROR EXIT
; GET RDSRH
; MASK OFF UNUSED BITS
; COMPARE RCV'D 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 RCV ERROR BIT SHOULD BE IGNORED
; BR IF YES
; SEE IF EXPECTED BIT = 1
; BR IF YES
; SEE IF ACTUAL BIT = 0
; BR IF YES
; 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
; QUEUE "DEVICE FATAL" ERROR # 31
; TAKE ERROR EXIT
; SEE IF ACTUAL BIT = 1
; BR IF YES

6$: JMP 20$
MOV 1(R5),R1
BIC #177400,R1
CMPB 2$,R1
BNE 7$
JMP 17$
MOV #1,REGNUM
CLR GDATA
MOV R1,GDATA
CLR BDATA
MOV 2$,BDATA
7$: MOV #1,REGNUM
BIT #RERCHK,2(R5)
BEQ 9$
;CHECK RERR BIT
BITB #RERR,R1
BNE 8$
BITB #RERR,2$
BEQ 9$
;STACK "RERR NOT CLEARED" MSG
GDF EM35,ERR12

8$: JMP 20$
BITB #RERR,2$
BNE 9$
;STACK "RERR NOT SET" MSG
GDF EM36,ERR12

9$: JMP 20$
;CHECK ROR BIT
BITB #ROR,R1
BNE 10$
BITB #ROR,2$
BEQ 11$
;STACK "ROR NOT CLEARED" MSG
GDF EM16,ERR12

10$: JMP 20$
BITB #ROR,2$
BNE 11$

```

....RXCHAR -- RECEIVE A CHARACTER

```

3441
3442 010552          ;STACK "ROR NOT SET" MSG
                        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
                                ;
                                ;   TAKE ERROR EXIT
                                ;   SEE IF ACTUAL BIT = 1
                                ;   BR IF YES
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
010552 012737 000001 002176
010560 012737 000040 002200
010566 012737 014265 002202
010574 012737 021714 002204
3443 010602 000137 011274
3444          ;CHECK RABGA BIT
3445 010606 132701 000004 11$: BITB  #RABGA,R1
3446 010612 001022          BNE  12$
3447 010614 132737 000004 010046 BITB  #RABGA,2$
3448 010622 001440          BEQ  13$
3449          ;STACK "RABGA NOT CLEARED" MSG
3450 010624          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
                                ;
                                ;   QUEUE "DEVICE FATAL" ERROR # 34
                                ;
                                ;   MOV      #T.EDF,ERRTYP
                                ;   MOV      #34,ERRNBR
                                ;   MOV      #EM40,ERRMSG
                                ;   MOV      #ERR12,ERRBLK
                                ;
                                ;   TAKE ERROR EXIT
                                ;   SEE IF ACTUAL BIT = 1
                                ;   BR IF YES
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
010624 012737 000001 002176
010632 012737 000041 002200
010640 012737 014712 002202
010646 012737 021714 002204
3451 010654 000137 011274
3452 010660 132737 000004 010046 12$: BITB  #RABGA,2$
3453 010666 001016          BNE  13$
3454          ;STACK "RABGA NOT SET" MSG
3455 010670          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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
010670 012737 000001 002176
010676 012737 000042 002200
010704 012737 014734 002202
010712 012737 021714 002204
3456 010720 000137 011274
3457          ;CHECK REOM BIT
3458 010724 132701 000002 13$: BITB  #REOM,R1
3459 010730 001022          BNE  14$
3460 010732 132737 000002 010046 BITB  #REOM,2$
3461 010740 001440          BEQ  15$
3462          ;STACK "REOM NOT CLEARED" MSG
3463 010742          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
                                ;
                                ;   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
                                ;
                                ;   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
                                ;
010742 012737 000001 002176
010750 012737 000043 002200
010756 012737 014516 002202
010764 012737 021714 002204
3464 010772 000137 011274
3465 010776 132737 000002 010046 14$: BITB  #REOM,2$
3466 011004 001016          BNE  15$
3467          ;STACK "REOM NOT SET" MSG
3468 011006          GTDF      EM31,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
                                ;
                                ;   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
                                ;
011006 012737 000001 002176
011014 012737 000044 002200
011022 012737 014537 002202
011030 012737 021714 002204
3469 011036 000137 011274
3470          ;CHECK RSOM BIT
3471 011042 132701 000001 15$: BITB  #RSOM,R1
3472 011046 001022          BNE  16$

```

....RXCHAR -- RECEIVE A CHARACTER

```

3473 011050 132737 000001 010046      BITB   @RSOM,2#      ;SEE IF ACTUAL BIT = 0
3474 011056 001440                      BEQ    17#          ;BR IF YES
3475                                     ;STACK "RSOM NOT CLEARED" MSG
3476 011060                      GTDF   EM28,ERR12

      011060 012737 000001 002176      ;
      011066 012737 000045 002200      MOV    @T.EDF,ERRTYP
      011074 012737 014460 002202      MOV    @37,ERRNBR
      011102 012737 021714 002204      MOV    @EM28,ERRMSG
      011110 000137 011274 010046      MOV    @ERR12,ERRBLK
3477 011110 000137 011274 010046      JMP    20#          ;TAKE ERROR EXIT
3478 011114 132737 000001 010046      16# : BITB   @RSOM,2#      ;SEE IF ACTUAL BIT = 1
3479 011122 001016                      BNE   17#          ;BR IF YES
3480                                     ;STACK "RSOM NOT SET" MSG
3481 011124                      GTDF   EM29,ERR12

      011124 012737 000001 002176      ;
      011132 012737 000046 002200      MOV    @T.EDF,ERRTYP
      011140 012737 014501 002202      MOV    @38,ERRNBR
      011146 012737 021714 002204      MOV    @EM29,ERRMSG
      011154 000137 011274 010046      MOV    @ERR12,ERRBLK
3482 011154 000137 011274 010046      JMP    20#          ;TAKE ERROR EXIT
3483
3484 011160 116502 000004 010046      17# : MOVB   4(R5),R2      ;GET DESIRED NO. OF CYCLES
3485 011164 005001                      CLR    R1          ;INIT CYCLE COUNT
3486
3487 011166 136527 000005 000040      18# : BITB   5(R5),@BIT5      ;* IS RXACT CHECK TO BE DISABLED ?
3488 011174 001004                      BNE   31#          ;* BR IF YES
3489 011176 004537 005622                      JSR   R5,CKRACT    ;CHK FOR RACT = 1
3490 011202 000001                      1
3491 011204 103433                      BCS   20#          ;BR TO EXIT IF ERROR
3492
3493 011206 020102 010046      31# : CMP    R1,R2          ;SEE IF REQUIRED CYCLES DONE YET
3494 011210 001415                      BEQ   19#          ;BR IF YES
3495
3496 011212 136527 000005 000200      BITB   5(R5),@BIT7      ;* SEE IF INITIAL RDA CHECK DESIRED
3497 011220 001004                      BNE   22#          ;* BR IF NO
3498 011222 004537 006122                      JSR   R5,CKRDA     ;CHK FOR RDA = 0
3499 011226 000000                      0
3500 011230 103421                      BCS   20#          ;BR TO EXIT IF ERROR
3501
3502 011232 004537 011540      22# : JSR   R5,STEPLU      ;CLOCK LU FOR 1 CYCLE
3503 011236 000001                      1
3504 011240 005201                      INC   R1           ;INCR CYCLE COUNT
3505 011242 000751                      BR    18#          ;CONTINUE CLOCKING
3506
3507 011244 136527 000005 000100      19# : BITB   5(R5),@BIT6      ;* IS FINAL RDA CHECK TO BE SKIPPED ?
3508 011252 001004                      BNE   30#          ;* BR IF YES
3509 011254 004537 006122                      JSR   R5,CKRDA     ;CHK RDA = 1
3510 011260 000001                      1
3511 011262 103404                      BCS   20#          ;BR IF ERROR
3512
3513 011264 062705 000006      30# : ADD    @6,R5          ;FIX UP RETURN ADRS
3514 011270 000241                      CLC
3515 011272 000403                      BR    21#          ;SET C = 0 FOR NO ERROR
3516 011274 062705 000006      20# : ADD    @6,R5          ;TAKE ERROR-FREE EXIT
3517 011300 000261                      SEC
3518 011302 012602                      21# : MOV    (SP)+,R2      ;FIX UP RETURN ADDRESS
3519 011304 012601                      MOV    (SP)+,R1    ;SET C BIT FOR ERROR
                                     ;RESTORE R2
                                     ;RESTORE R1

```


C7

....RXCHAR -- RECEIVE A CHARACTER

3520 011306 000205
3521

RTS R5 ;RETURN

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

3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538 011310
3539 011310 010146
3540 011312 010246
3541 011314 005001
3542 011316 012502
3543 011320 062702 000003
3544 011324 004537 005622
3545 011330 000000
3546 011332 103446
3547 011334 004537 006122
3548 011340 000000
3549 011342 103442
3550 011344 004537 006562
3551 011350 000000
3552 011352 103436
3553 011354 004537 011540
3554 011360 000001
3555 011362 005201
3556 011364 004537 003534
3557 011370 122000
3558 011372 000000
3559 011374 132737 000200 011372
3560 011402 001006
3561 011404 020102
3562 011406 002762
3563 011410 004537 006122
3564 011414 000001
3565 011416 103414
3566 011420 020165 177776
3567 011424 002004
3568 011426 004537 006122
3569 011432 000000
3570 011434 103405
3571 011436 004537 005622
3572 011442 000001
3573 011444 103401
3574 011446 000241
3575 011450 012602
3576 011452 012601
3577 011454 000205
3578
3579

```

.SBTTL ....RCV1ST -- RECEIVE FIRST CHARACTER OF MESSAGE
*****
;* RCV1ST - THIS SUBROUTINE RECEIVES THE FIRST CHAR OF A MESSAGE AND MONITORS
;* THE STATUS OF THE RECEIVER. FIRST, A CHECK IS MADE FOR RXACT = 0,
;* RDA = 0, RSA = 0, RSOM = 0. THEN, THE LINE UNIT IS CLOCKED UNTIL
;* RDA = 1. THE PROGRAM CHECKS FOR THIS TO OCCUR WITHIN 3 CYCLES AFTER
;* THE NO. OF CYCLES PASSED IN THE SECOND WORD FOLLOWING THE CALL.
;* 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,RCV1ST
;* .WORD <EXPECTED RECEIVER CYCLE COUNT>
*****
RCV1ST:
MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
CLR R1 ;INIT CYCLE COUNT
MOV (R5)+,R2 ;GET CYCLE COUNT LIMIT
ADD #3,R2
JSR R5,CKRACT ;CHK FOR RXACT = 0
0
BCS 6# ;BR TO EXIT IF ERROR
JSR R5,CKRDA ;CHK FOR RDA = 0
0
BCS 6# ;BR TO EXIT IF ERROR
JSR R5,CKSEOM ;CHK FOR RSOM = 0, REOM = 0
0
BCS 6# ;BR TO EXIT IF ERROR
1# JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
INC R1 ;INCREMENT CYCLE COUNT
JSR R5,READI ;READ USYRT STATUS REG
USTATR
2# .WORD 0
BITB #RDA,2# ;SEE IF RDA SET YET
BNE 3# ;BR IF YES
CMP R1,R2 ;SEE IF LIMIT EXCEEDED
BLT 1# ;BR IF NOT YET
1# JSR R5,CKRDA ;GO STACK "RDA NOT SET" MSG
BCS 6# ;BR TO EXIT IF ERROR
3# CMP R1,-2(R5) ;SEE IF LESS THAN REQUIRED CYCLES
BGE 4# ;BR IF NOT
4# JSR R5,CKRDA ;GO STACK "RDA NOT CLEARED" MSG
0
BCS 6# ;BR TO EXIT IF ERROR
4# JSR R5,CKRACT ;CHK FOR RXACT = 1
1
BCS 6# ;BR TO EXIT IF ERROR
5# CLC ;CLEAR C BIT FOR NO ERRORS
6# MOV (SP)+,R2 ;RESTORE R2
MOV (SP)+,R1 ;RESTORE R1
RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)

```

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

3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596 011456
3597 011456 012537 011516
3598 011462 004537 005462
3599 011466 000001
3600 011470 103422
3601 011472 004537 005622
3602 011476 000001
3603 011500 103416
3604 011502 004537 003660
3605 011506 120000
3606 011510 000002
3607 011512 004537 011540
3608 011516 000000
3609 011520 004537 005462
3610 011524 000000
3611 011526 103403
3612 011530 004537 005622
3613 011534 000000
3614 011536 000205
3615
3616

```

.SBTTL ....ENDTRN -- SHUT DOWN TRANSMITTER/RECEIVER
;*****
;* ENDTRN - THIS SUBROUTINE TERMINATES A MESSAGE BY CLEARING TXEN AND RXEN,
;* CLOCKING THE LINE UNIT FOR THE NUMBER OF CYCLES PASSED IN THE WORD
;* FOLLOWING THE CALL, AND CHECKING FOR THE USYRT TRANSMITTER AND
;* RECEIVER TO BE SHUT DOWN.
;* 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.
;* NOTE: THIS ROUTINE ASSUMES THAT TTLOOP SHOULD BE ENABLED.
;*
;* CALLING SEQUENCE :
;*   JSR   R5,ENDTRN
;*   <NO. OF CYCLES TO CLOCK>
;*****
ENDTRN:
MOV      (R5)+,2#           ;GET DESIRED NO. OF CYCLES TO CLOCK
JSR      R5,CKTACT         ;CHK FOR TXACT = 1
BCS     1
JSR      R5,CKRACT        ;BR IF ERROR
BCS     1                 ;CHK FOR RXACT = 1
JSR      R5,WRITEI        ;CLEAR TXEN AND RXEN IN USYRT
VIAORB  TTLOOP            ; ** BUT LEAVE TTLOOP ENABLED **
;
;CLOCK LU FOR DESIRED NO. OF CYCLES
2# : JSR      R5,STEPLU
      .WORD  0
      JSR      R5,CKTACT   ;CHK FOR TXACT = 0
      BCS     6#
      JSR      R5,CKRACT   ;CHK FOR RXACT = 0
      BCS     6#
6# : RTS      R5

```

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

3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631 011540
3632 011540 010146
3633 011542 012501
3634 011544 004537 003660
3635 011550 120005
3636 011552 000000
3637 011554 005301
3638 011556 001372
3639 011560 012601
3640 011562 000205
3641
3642
3643
3644
3645
3646

```
.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$: JSR R5,WRITEI ;LOAD T1C-H, START COUNTER, CLOCK 1 CYCLE
VIAT1B
000
DEC R1 ;DECR CYCLE COUNTER
BNE 1$ ;BR IF ALL CYCLES NOT DONE YET
MOV (SP),R1 ;RESTORE R1
RTS R5 ;RETURN
```

GLOBAL ERROR REPORT SECTION

```

3648          .SBTTL GLOBAL ERROR REPORT SECTION
3649
3650          ;////////////////////////////////////
3651          ;/ THE GLOBAL ERROR REPORT SECTION CONTAINS ERROR MESSAGES
3652          ;/ THAT ARE USED IN MORE THAN ONE TEST.
3653          ;////////////////////////////////////
3654
3655          .NLIST BEX
3656 011564      045      116      045 ENDEMB: .ASCIZ /#N#N/
3657 011571      045      116      000 NEWLIN: .ASCIZ /#N/          ;USED TO TERMINATE ERROR MESSAGES
3658
3659 011574      045      116      045 FMT2:  .ASCIZ /#N#AFAILING REG = #T#ASEL#01/
3660 011631      045      116      045 FMT3:  .ASCIZ /#N#A EXPECTED: #03#A ACTUAL: #03#A XOR: #03/
3661 011715      045      116      045 FMT4:  .ASCIZ /#N#ATHE CONTENTS OF ALL#T#N#T/
3662 011753      045      116      045 FMT4A: .ASCIZ /#N#S1#03#S5#03#S5#03#S5#03/
3663 012006      045      116      045 FMT4B: .ASCIZ /#N#T/
3664 012013      045      116      045 FMT4C: .ASCIZ /#N#S5#03#S5#03#S5#03#S5#03/
3665 012046      045      116      045 FMT5:  .ASCIZ /#N#A WHEN #03#A LOADED INTO BSEL1/
3666 012111      045      116      045 FMT5A: .ASCIZ /#N#A ATTEMPTING "M-LOOP" FUNCTION CODE #02#A (#T#A)/
3667 012176      045      116      045 FMT7:  .ASCIZ /#N#AMDIAG #03#A FAILED/
3668
3669 012226      045      116      045 FMT10: .ASCIZ /#N#A EXPECTED:#08#A ACTUAL:#08#A XOR:#08/
3670 012302      045      101      040 FMT10A: .ASCIZ /#A LSI ADDR:#08/
3671 012323      045      116      045 FMT11: .ASCIZ /#N#08#08#08#08/
3672 012342      045      116      045 FMT12: .ASCIZ /#N#N#T/
3673 012351      045      116      045 FMT13: .ASCIZ /#N#T#03#S2#03#S2#03#S2#03#S2#03#S2#03/
3674 012417      045      123      062 FMT14: .ASCIZ /#S2#03#S2#03/
3675 012434      045      101      040 FMT15: .ASCIZ /#A DETECTED IN #T#T#A --/
3676 012466      045      101      040 FMT15A: .ASCIZ /#A DETECTED @ TEST PATTERN ELEMENT # #02/
3677 012540      045      116      045 FMT16: .ASCIZ /#N#T#03#S4#03#S#03/
3678 012563      045      116      045 FMT16A: .ASCIZ /#N#T#03#S#03#S#03#S4#03#S#03#S#03/
3679 012625      045      101      040 FMT17: .ASCIZ /#A VALUE SENT TO NPR CONTROL REGISTER: #03/
3680 012704      045      116      045 FMT17A: .ASCIZ /#N#A VALUE READ FROM CONTROL REGISTER: #03/
3681 012765      045      116      045 FMT17B: .ASCIZ /#N#A LSI-11 MEMORY ADDRESS ACCESSED:#08/
3682 013040      045      116      045 FMT17C: .ASCIZ /#N#A INFORMATION ON THE FIRST OF #05#A ERRORS:/
3683
3684 013120      045      116      045 FMT19: .ASCIZ /#N#ATEST #02#A NOT RUN#N/
3685 013151      045      124      045 FMT21: .ASCIZ /#T#06#N/
3686 013161      045      116      045 FMT22: .ASCIZ /#N#AFAILING REG: /
3687 013203      045      101      105 FMT23: .ASCIZ /#AEXPECTED: #03#S5#A ACTUAL: #03#S5#A XOR: #03#N/
3688 013262      045      116      045 FMT24: .ASCIZ /#N#T#N#T#N/
3689 013275      045      117      063 FMT25: .ASCIZ /#03#S5#03#S5#03#S5#03#N/
3690 013325      045      123      064 FMT26: .ASCIZ /#S4#03#S5#03#S5#03#S5#03#N/
3691 013360      045      124      045 FMT27: .ASCIZ /#T#T#N/
3692 013367      045      101      105 FMT28: .ASCIZ /#AEXTENDED REG AX#01#A-#T#N/
3693 013423      045      124      045 FMT29: .ASCIZ /#T#N/
3694 013430      045      116      045 FMT30: .ASCIZ /#N#AFOR BAUD RATE SPECIFIED./
3695 013465      045      116      045 FMT31: .ASCIZ /#N#AIMPROPER CONNECTOR TYPE SPECIFIED/
3696 013533      045      116      045 FMT32: .ASCIZ /#N#AFOR OPTION SPECIFIED./
3697 013565      045      116      045 FMT39: .ASCIZ /#N#ATEST #02#A NOT RUN#N/
3698 013616      045      116      045 FMT40: .ASCIZ /#N#AFAILING RAM ADRS: #06#A (OCT)#N/
3699 013662      045      116      045 FMT50: .ASCIZ /#N#ARESPONDING ADRS: #03#A (OCT)#N/
3700 013725      045      116      045 FMT51: .ASCIZ /#N#AEXPECTED COUNT: #01#A ACTUAL COUNT: #01#N/
3701
3702 014007      122      105      107 EM1:  .ASCIZ /REG NOT INITIALIZED BY MST CLR/
3703 014046      125      123      131 EM2:  .ASCIZ /USYRT NOT INITIALIZED BY PROGRAM RESET/
3704 014115      115      111      103 EM3:  .ASCIZ /MICRO-DIAG. FAILURE/

```

GLOBAL ERROR REPORT SECTION

3705	014141	115	122	104	EM4:	.ASCIZ	/MRDY TIMEOUT/
3706	014156	116	125	114	EM5:	.ASCIZ	/NULL CLK BIT STUCK AT 0/
3707	014206	116	125	114	EM6:	.ASCIZ	/NULL CLK BIT STUCK AT 1/
3708	014236	122	117	122	EM13:	.ASCIZ	/ROR NOT CLEARED BY SOM/
3709	014265	122	117	122	EM14:	.ASCIZ	/ROR NOT SET/
3710	014301	122	117	122	EM15:	.ASCIZ	/ROR NOT CLEARED BY OC/
3711	014327	122	117	122	EM16:	.ASCIZ	/ROR NOT CLEARED/
3712	014347	122	105	101	EM25:	.ASCIZ	'READ/WRITE DATA ERROR'
3713	014375	111	116	103	EM26:	.ASCIZ	/INCORRECT DATA CHAR RCV'D/
3714	014427	111	116	103	EM27:	.ASCIZ	/INCORRECT CRC BYTE RCV'D/
3715	014460	122	123	117	EM28:	.ASCIZ	/RSOM NOT CLEARED/
3716	014501	122	123	117	EM29:	.ASCIZ	/RSOM NOT SET/
3717	014516	122	105	117	EM30:	.ASCIZ	/REOM NOT CLEARED/
3718	014537	122	105	117	EM31:	.ASCIZ	/REOM NOT SET/
3719	014554	124	130	104	EM32:	.ASCIZ	/TXDATA BIT NOT CLEARED/
3720	014603	124	130	104	EM33:	.ASCIZ	/TXDATA BIT NOT SET/
3721	014626	122	103	126	EM34:	.ASCIZ	/RCV'D DATA MISCOMPARE/
3722	014654	122	105	122	EM35:	.ASCIZ	/RERR NOT CLEARED/
3723	014675	122	105	122	EM36:	.ASCIZ	/RERR NOT SET/
3724	014712	122	101	102	EM39:	.ASCIZ	/RABGA NOT CLEARED/
3725	014734	122	101	102	EM40:	.ASCIZ	/RABGA NOT SET/
3726	014752	117	126	122	EM41:	.ASCIZ	/OVRR NOT CLEARED/
3727	014773	117	126	122	EM42:	.ASCIZ	/OVRR NOT SET/
3728	015010	123	127	040	EM43:	.ASCIZ	/SW PACK #1 INCORRECT/
3729	015035	123	127	040	EM44:	.ASCIZ	/SW PACK #2 INCORRECT/
3730	015062	123	127	040	EM45:	.ASCIZ	/SW PACK #3 INCORRECT/
3731	015107	101	123	123	EM47:	.ASCIZ	/ASSEMB BIT COUNT INCORRECT/
3732	015142	117	104	104	EM48:	.ASCIZ	/ODD VRC PARITY BIT NOT SET/
3733	015175	117	104	104	EM49:	.ASCIZ	/ODD VRC PARITY BIT NOT CLEARED/
3734	015234	105	126	105	EM50:	.ASCIZ	/EVEN VRC PARITY BIT NOT SET/
3735	015270	105	126	105	EM51:	.ASCIZ	/EVEN VRC PARITY BIT NOT CLEARED/
3736	015330	124	130	040	EM54:	.ASCIZ	/TX UNDERRUN ERROR/
3737	015352	122	124	123	EM60:	.ASCIZ	/RTS NOT SET/
3738	015366	122	124	123	EM65:	.ASCIZ	/RTS NOT CLEARED/
3739	015406	122	105	107	EM66:	.ASCIZ	/REG MISCOMPARE/
3740	015425	122	105	107	EM67:	.ASCIZ	/REG NOT INITIALIZED BY UNIBUS RESET (INIT)/
3741	015500	125	123	131	EM68:	.ASCIZ	/USYRT STATUS INCORRECT/
3742	015527	124	130	101	EM69:	.ASCIZ	/TXACT NOT SET/
3743	015545	124	130	101	EM70:	.ASCIZ	/TXACT NOT CLEARED/
3744	015567	122	130	101	EM71:	.ASCIZ	/RXACT NOT SET/
3745	015605	122	130	101	EM72:	.ASCIZ	/RXACT NOT CLEARED/
3746	015627	124	102	115	EM73:	.ASCIZ	/TBMT NOT SET/
3747	015644	124	102	115	EM74:	.ASCIZ	/TBMT NOT CLEARED/
3748	015665	122	104	101	EM75:	.ASCIZ	/RDA NOT SET/
3749	015701	122	104	101	EM76:	.ASCIZ	/RDA NOT CLEARED/
3750	015721	122	123	101	EM77:	.ASCIZ	/RSA NOT SET/
3751	015735	122	123	101	EM78:	.ASCIZ	/RSA NOT CLEARED/
3752	015755	122	101	115	EM79:	.ASCIZ	/RAM ERROR LOADING MICROCODE/
3753	016011	103	101	122	EM80:	.ASCIZ	/CARRIER NOT SET/
3754	016031	103	101	122	EM81:	.ASCIZ	/CARRIER NOT CLEARED/
3755	016055	111	116	126	EM82:	.ASCIZ	/INVALID ERROR CODE FROM 6502/
3756	016112	115	117	104	EM83:	.ASCIZ	/MODEM STATUS INCORRECT/
3757	016141	103	124	123	EM84:	.ASCIZ	/CTS NOT CLRD/
3758	016156	103	124	123	EM85:	.ASCIZ	/CTS NOT SET/
3759	016172	103	101	122	EM86:	.ASCIZ	/CARRIER NOT CLRD/
3760	016213	103	101	122	EM87:	.ASCIZ	/CARRIER NOT SET/
3761	016233	115	117	104	EM88:	.ASCIZ	/MODEM RDY NOT CLRD/

GLOBAL ERROR REPORT SECTION

```

3762 016256      115      117      104 EM89:  .ASCIZ  /MODEM RDY NOT SET/
3763 016300      122      105      103 EM90:  .ASCIZ  /RECEIVER OVERRUN NOT SET/
3764 016331      122      105      103 EM91:  .ASCIZ  /RECEIVER OVERRUN NOT CLEARED/
3765 016366      124      102      115 EM92:  .ASCIZ  /TBMT INTERRUPT TEST FAILURE/
3766 016422      124      123      117 EM100: .ASCIZ  /TSO BIT NOT SET/
3767 016442      124      123      117 EM101: .ASCIZ  /TSO BIT NOT CLEARED/
3768 016466      125      123      131 EM102: .ASCIZ  /USYRT RESPONDED TO THE WRONG ADDR/
3769 016530      125      123      131 EM103: .ASCIZ  /USYRT DIDN'T RESPOND TO SECONDARY STATION ADDR/
3770 016607      125      123      131 EM104: .ASCIZ  /USYRT DIDN'T RESPOND TO ALL-PARTIES-ADDR(377)/
3771 016665      125      123      131 EM105: .ASCIZ  /USYRT ASSEMBLED BIT COUNT WAS INCORRECT/
3772 016735      124      122      101 EM106: .ASCIZ  /TRANSMISSION ERROR (AS READ BY TSO BIT)/
3773
3774
3775
3776
3777
3778
3779 017005      102      123      105 TXT1:  .ASCIZ  /BSEL0 BSEL1 BSEL2 BSEL3/
3780 017043      040      040      040 TXT2:  .ASCIZ  / BSEL4 BSEL5 BSEL6 BSEL7/
3781 017105      102      123      105 TXT2A: .ASCIZ  /BSEL10 BSEL11 BSEL12 BSEL13/
3782 017144      040      040      040 TXT2B: .ASCIZ  / BSEL14 BSEL15 BSEL16 BSEL17/
3783 017207      040      102      131 TXT3:  .ASCIZ  / BYTE SELECT REG'S ARE:/
3784 017237      040      040      040 TXT4:  .ASCIZ  / SELO SEL2 SEL4 SEL6/
3785 017277      040      040      040 TXT4A: .ASCIZ  / SEL10 SEL12 SEL14 SEL16/
3786 017340      102      000
3787 017342      040      123      105 TXT5:  .ASCIZ  /B/
3788 017365      040      122      105 TXT6:  .ASCIZ  / SELECT REG'S ARE:/
3789 017455      040      040      040 TXT7:  .ASCIZ  / REGISTERS ORB ORA DDRB DDRA T1CL T1CH T1LL T1LH /
3790 017545      040      105      130 TXT7A: .ASCIZ  / T2CL T2CH SR ACR PCR IFR IER ORA /
3791 017565      040      101      103 TXT8:  .ASCIZ  / EXPECTED: /
3792 017605      040      130      117 TXT9:  .ASCIZ  / ACTUAL: /
3793 017625      040      040      116 TXT10: .ASCIZ  / XOR: /
3794 017677      040      040      040 TXT11: .ASCIZ  / N P R R E G I S T E R S:/
3795 017735      040      040      040 TXT11A: .ASCIZ  / CONTROL DATA/
3796 020005      104      105      126 TXT11B: .ASCIZ  / OUT ADDR. IN ADDR./
3797 020033      125      123      131 TXT12: .ASCIZ  /DEVICE CSR ADDRESS : /
3798 020050      122      104      123 TXT13: .ASCIZ  /USYRT REGS :/
3799 020106      040      040      040 TXT14: .ASCIZ  /RDSRL RDSRH TDSRL TDSRH/
3800 020150      126      111      101 TXT15: .ASCIZ  / PCSARL PCSARH PCR USTAT/
3801 020163      117      122      102 TXT16: .ASCIZ  /VIA REGS :/
3802 020220      040      040      040 TXT17: .ASCIZ  /ORB ORA DDRB DDRA/
3803 020261      124      062      103 TXT18: .ASCIZ  / T1CL T1CH T1LL T1LH/
3804 020315      040      040      040 TXT19: .ASCIZ  /T2CL T2CH SR ACR/
3805
3806 020355      021      000      TXTNUL: .BYTE 21.0 ;CTL-Q -- THIS (WE HOPE) IS HARMLESS
3807
3808 020357      116      117      120 TXTML0: .ASCIZ  /NOP/
3809 020363      122      105      101 TXTML1: .ASCIZ  /READ 1 BYTE/
3810 020377      127      122      111 TXTML2: .ASCIZ  /WRITE 1 BYTE/
3811 020414      116      120      122 TXTML3: .ASCIZ  /NPR-OUT 256 BYTES/
3812 020436      116      120      122 TXTML4: .ASCIZ  /NPR-IN 256 BYTES/
3813 020457      123      105      124 TXTML5: .ASCIZ  /SET MICROPROCESSOR'S PC/
3814 020507      125      116      104 TXTML6: .ASCIZ  /UNDEFINED/
3815 020521      101      114      114 TXTML7: .ASCIZ  /ALLOW U-PROCESSOR INTERRUPTS/
3816
3817 020556      126      111      101 TXTVR:  .ASCIZ  /VIA REGISTER /
3818 020574      117      122      102 TXTVR0: .ASCIZ  /ORB/

```

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

3819	020600	117	122	101	TXTVR1:	.ASCIZ	/ORA/
3820	020604	104	104	122	TXTVR2:	.ASCIZ	/DDR/
3821	020611	104	104	122	TXTVR3:	.ASCIZ	/DDRA/
3822	020616	124	061	103	TXTVR4:	.ASCIZ	/T1CL/
3823	020623	124	061	103	TXTVR5:	.ASCIZ	/T1CH/
3824	020630	124	061	114	TXTVR6:	.ASCIZ	/T1LL/
3825	020635	124	061	114	TXTVR7:	.ASCIZ	/T1LH/
3826	020642	124	062	103	TXTVR8:	.ASCIZ	/T2CL/
3827	020647	124	062	103	TXTVR9:	.ASCIZ	/T2CH/
3828	020654	123	122	000	TXTVRA:	.ASCIZ	/SR/
3829	020657	101	103	122	TXTVRB:	.ASCIZ	/ACR/
3830	020663	120	103	122	TXTVRC:	.ASCIZ	/PCR/
3831	020667	111	106	122	TXTVRD:	.ASCIZ	/IFR/
3832	020673	111	105	122	TXTVRE:	.ASCIZ	/IER/
3833	020677	117	122	101	TXTVRF:	.ASCIZ	/ORA/
3834							
3835	020703	116	120	122	TXTNP:	.ASCIZ	/NPR /
3836	020710	103	117	116	TXTNP0:	.ASCIZ	/CONTROL/
3837	020720	104	101	124	TXTNP1:	.ASCIZ	/DATA HI/
3838	020730	104	101	124	TXTNP2:	.ASCIZ	/DATA LO/
3839	020740	101	104	104	TXTNP3:	.ASCIZ	/ADDR. OUT EX/
3840	020755	101	104	104	TXTNP4:	.ASCIZ	/ADDR. OUT HI/
3841	020772	101	104	104	TXTNP5:	.ASCIZ	/ADDR. OUT LO/
3842	021007	101	104	104	TXTNP6:	.ASCIZ	/ADDR. IN EX/
3843	021023	101	104	104	TXTNP7:	.ASCIZ	/ADDR. IN HI/
3844	021037	101	104	104	TXTNP8:	.ASCIZ	/ADDR. IN LO/
3845							
3846	021053	125	123	131	TXTUR:	.ASCIZ	/USYRT REG /
3847	021066	122	104	123	TXTURO:	.ASCIZ	/RDSRL/
3848	021074	122	104	123	TXTUR1:	.ASCIZ	/RDSRH/
3849	021102	124	104	123	TXTUR2:	.ASCIZ	/TDSRL/
3850	021110	124	104	123	TXTUR3:	.ASCIZ	/TDSRH/
3851	021116	120	103	123	TXTUR4:	.ASCIZ	/PCSARL/
3852	021125	120	103	123	TXTUR5:	.ASCIZ	/PCSARH/
3853	021134	120	103	122	TXTUR6:	.ASCIZ	/PCR/
3854	021140	125	123	124	TXTUR7:	.ASCIZ	/USTAT/
3855					.LIST	BEX	
3856					.EVEN		
3857							
3858							
3859							

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

 ;----- TEXT ADDRESS TABLES USED BY ERROR HANDLERS -----
 ;-----

3864	021146	020357	020363	020377	TXTMLT:	.WORD	TXTML0,TXTML1,TXTML2,TXTML3,TXTML4,TXTML5,TXTML6,TXTML7
	021154	020414	020436	020457			
	021162	020507	020521				
3865							
3866	021166	020556				.WORD	TXTVR
3867	021170	020574	020600	020604	TXTVRT:	.WORD	TXTVR0,TXTVR1,TXTVR2,TXTVR3,TXTVR4,TXTVR5,TXTVR6,TXTVR7
	021176	020611	020616	020623			
	021204	020630	020635				
3868	021210	020642	020647	020654		.WORD	TXTVR8,TXTVR9,TXTVRA,TXTVRB,TXTVRC,TXTVRD,TXTVRE,TXTVRF
	021216	020657	020663	020667			
	021224	020673	020677				
3869							

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

3870	021230	020703				.WORD	TXTNP
3871	021232	020710	020720	020730	TXTNPT:	.WORD	TXTNPO, TXTNP1, TXTNP2, TXTNP3, TXTNP4, TXTNP5, TXTNP6, TXTNP7, TXTNP8
	021240	020740	020755	020772			
	021246	021007	021023	021037			
3872	021254	021066	021074	021102	TXTURT:	.WORD	TXTURO, TXTUR1, TXTUR2, TXTUR3, TXTUR4, TXTUR5, TXTUR6, TXTUR7
	021262	021110	021116	021125			
	021270	021134	021140				
3873							
3874							

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

```

3876
3877
3878
3879 021274
      021274
3880 021274 105037 002331
3881 021300 010146
3882 021302 013701 002330
3883 021306 022701 000017
3884 021312 002012
3885 021314
      021314 010146
      021316 012746 012046
      021322 012746 000002
      021326 010600
      021330 104415
      021332 062706 000006
3886 021336 000424
3887
3888 021340 001001
3889 021342 005001
3890 021344 022701 000007
3891 021350 002002
3892 021352 012701 000006
3893 021356 006301
3894 021360
      021360 016146 021146
      021364 013746 002330
      021370 012746 012111
      021374 012746 000003
      021400 010600
      021402 104415
      021404 062706 000010
3895
3896 021410 012601
3897 021412 004737 022724
3898 021416
      021416
      021416 104423
3899
3900
3901
3902 021420
      021420
3903 021420 113701 002342
3904 021424 006301
3905 021426
      021426 016146 021254
      021432 012746 021053
      021436 012746 012434
      021442 012746 000003
      021446 010600
      021450 104414
      021452 062706 000010
3906 021456 004737 022352
3907 021462
      021462 013746 002334

```

```

-----
.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,RO
      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$: CMP #7,R1
      BGE 7$
      MOV #6,R1
      ;NO, THEN WE CAN HANDLE IT
      ;YES, THEN IT'S UNDEFINED -- SAY SO
7$: ASL R1
      PRINTX #FMT5A,GDATA,TEXTMLT(R1)
      ;CONVERT TO A WORD OFFSET
      ;REPORT THE FAILING FUNCTION
      MOV TEXTMLT(R1),-(SP)
      MOV GDATA,-(SP)
      MOV #FMT5A,-(SP)
      MOV #3,-(SP)
      MOV SP,RO
      TRAP C$PNTX
      ADD #10,SP
20$: MOV (SP)+,R1
      JSR PC,ERR5$
      ;RESTORE THE WORKING REGISTER
      ;DUMP THE SELECT REGISTERS
      L10002: TRAP C$MSG
-----
.SBTTL ....ERROR HANDLER -- ERR7A -- USYRT REGISTER ERRORS
-----
      BGNMSG ERR7A
      ERR7A::
      MOV REGNUM,R1
      ASL R1
      ;AS PASSED, THIS WAS A BYTE OFFSET
      PRINTB #FMT15,#TXTUR,TEXTURT(R1)
      MOV TEXTURT(R1),-(SP)
      MOV #TXTUR,-(SP)
      MOV #FMT15,-(SP)
      MOV #3,-(SP)
      MOV SP,RO
      TRAP C$PNTB
      ADD #10,SP
      JSR PC,XORGB
      PRINTB #FMT3,GDATA,BDATA,XDATA
      MOV XDATA,-(SP)

```

....ERROR HANDLER -- ERR7A -- USYRT REGISTER ERRORS

021466	013746	002332		MOV	BDATA,-(SP)
021472	013746	002330		MOV	GDATA,-(SP)
021476	012746	011631		MOV	#FMT3,-(SP)
021502	012746	000004		MOV	#4,-(SP)
021506	010600			MOV	SP,R0
021510	104414			TRAP	C\$PNTB
021512	062706	000012		ADD	#12,SP
3908 021516			PRINTB	#ENDEMB	
021516	012746	011564		MOV	#ENDEMB,-(SP)
021522	012746	000001		MOV	#1,-(SP)
021526	010600			MOV	SP,R0
021530	104414			TRAP	C\$PNTB
021532	062706	000004		ADD	#4,SP
3909 021536			ENDMSG		
021536					
021536	104423			L10003:	TRAP C\$MSG
3910			;-----		
3911			;SBTTLERROR HANDLER -- ERR10 -- USYRT REG ERROR (XOR, REG PRINTOUT)		
3912			;-----		
3913					
3914 021540			BGNMSG	ERR10	
021540					
3915 021540			PRINTB	#FMT21,#TXT12,MPCSR	ERR10::
021540	013746	002422		MOV	MPCSR,-(SP)
021544	012746	020005		MOV	#TXT12,-(SP)
021550	012746	013151		MOV	#FMT21,-(SP)
021554	012746	000003		MOV	#3,-(SP)
021560	010600			MOV	SP,R0
021562	104414			TRAP	C\$PNTB
021564	062706	000010		ADD	#10,SP
3916 021570			PRINTB	#FMT22	
021570	012746	013161		MOV	#FMT22,-(SP)
021574	012746	000001		MOV	#1,-(SP)
021600	010600			MOV	SP,R0
021602	104414			TRAP	C\$PNTB
021604	062706	000004		ADD	#4,SP
3917 021610	013701	002342	MOV	REGNUM,R1	
3918 021614	006301		ASL	R1	;GET PTR TO USYRT REG ASCII
3919 021616			PRINTB	#FMT27,#TXTUR,TXTURT(R1)	
021616	016146	021254		MOV	TXTURT(R1),-(SP)
021622	012746	021053		MOV	#TXTUR,-(SP)
021626	012746	013360		MOV	#FMT27,-(SP)
021632	012746	000003		MOV	#3,-(SP)
021636	010600			MOV	SP,R0
021640	104414			TRAP	C\$PNTB
021642	062706	000010		ADD	#10,SP
3920 021646	004737	022352	JSR	PC,XORGB	;COMPUTE XOR OF GOOD AND BAD DATA
3921 021652			PRINTB	#FMT23,GDATA,BDATA,XDATA	
021652	013746	002334		MOV	XDATA,-(SP)
021656	013746	002332		MOV	BDATA,-(SP)
021662	013746	002330		MOV	GDATA,-(SP)
021666	012746	013203		MOV	#FMT23,-(SP)
021672	012746	000004		MOV	#4,-(SP)
021676	010600			MOV	SP,R0
021700	104414			TRAP	C\$PNTB
021702	062706	000012		ADD	#12,SP
3922 021706	004737	023454	JSR	PC,ERR12\$;GET & PRINT USYRT REGISTERS

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

```

3923 021712          ENDMSG
      021712
      021712 104423          L10004: TRAP C$MSG
3924
3925
3926 ;-----
3927 ;.SBTTL ....ERROR HANDLER -- ERR12 -- USYRT REG ERROR (USYRT PRINTOUT)
3928 ;-----
      BGNMSG ERR12          ERR12::
3929 021714          PRINTB #FMT21,#TXT12,MPCSR
      021714 013746 002422          MOV MPCSR,-(SP)
      021720 012746 020005          MOV #TXT12,-(SP)
      021724 012746 013151          MOV #FMT21,-(SP)
      021730 012746 000003          MOV #3,-(SP)
      021734 010600          MOV SP,RO
      021736 104414          TRAP C$PNTB
      021740 062706 000010          ADD #10,SP
3930 021744          PRINTB #FMT22
      021744 012746 013161          MOV #FMT22,-(SP)
      021750 012746 000001          MOV #1,-(SP)
      021754 010600          MOV SP,RO
      021756 104414          TRAP C$PNTB
      021760 062706 000004          ADD #4,SP
3931 021764 013701 002342          MOV REGNUM,R1
3932 021770 006301          ASL R1 ;GET PTR TO USYRT REG ASCII
3933 021772          PRINTB #FMT27,#TXTUR,TXTURT(R1)
      021772 016146 021254          MOV TXTURT(R1),-(SP)
      021776 012746 021053          MOV #TXTUR,-(SP)
      022002 012746 013360          MOV #FMT27,-(SP)
      022006 012746 000003          MOV #3,-(SP)
      022012 010600          MOV SP,RO
      022014 104414          TRAP C$PNTB
      022016 062706 000010          ADD #10,SP
3934 022022 004737 023454          JSR PC,ERR12$ ;GET & PRINT USYRT REGISTERS
3935 022026          ENDMSG
      022026          L10005: TRAP C$MSG
      022026 104423
3936
3937
3938 ;-----
3939 ;.SBTTL ....ERROR HANDLER -- ERR13 -- RAM ADDRESS ERRORS
3940 ;-----
      BGNMSG ERR13          ERR13::
3941 022030          PRINTB #FMT21,#TXT12,MPCSR
      022030 013746 002422          MOV MPCSR,-(SP)
      022034 012746 020005          MOV #TXT12,-(SP)
      022040 012746 013151          MOV #FMT21,-(SP)
      022044 012746 000003          MOV #3,-(SP)
      022050 010600          MOV SP,RO
      022052 104414          TRAP C$PNTB
      022054 062706 000010          ADD #10,SP
3943 022060          PRINTB #FMT40,REGNUM
      022060 013746 002342          MOV REGNUM,-(SP)
      022064 012746 013616          MOV #FMT40,-(SP)
      022070 012746 000002          MOV #2,-(SP)
      022074 010600          MOV SP,RO

```

....ERROR HANDLER -- ERR13 -- RAM ADDRESS ERRORS

```

022076 104414
022100 062706 000006
3944 022104 004737 022352 JSR PC,XORGB ;COMPUTE XOR OF GOOD AND BAD DATA
3945 022110 PRINTB #FMT23,GDATA,BDATA,XDATA
022110 013746 002334 MOV XDATA,-(SP)
022114 013746 002332 MOV BDATA,-(SP)
022120 013746 002330 MOV GDATA,-(SP)
022124 012746 013203 MOV #FMT23,-(SP)
022130 012746 000004 MOV #4,-(SP)
022134 010600 MOV SP,RO
022136 104414 TRAP C$PNTB
022140 062706 000012 ADD #12,SP
3946 022144 ENDMSG
022144 104423 L10006: TRAP C$MSG
3947
3948
3949 :-----
3950 :.SBTTL ....ERROR HANDLER -- ERR20 -- USYRT REG DUMP
3951 :-----
3951 022146 BGNMSG ERR20
3952 022146 PRINTB #FMT21,@TXT12,MPCSR ERR20::
022146 013746 002422 MOV MPCSR,-(SP)
022152 012746 020005 MOV @TXT12,-(SP)
022156 012746 013151 MOV #FMT21,-(SP)
022162 012746 000003 MOV #3,-(SP)
022166 010600 MOV SP,RO
022170 104414 TRAP C$PNTB
022172 062706 000010 ADD #10,SP
3953 022176 004737 023454 JSR PC,ERR12# ;GET & PRINT USYRT REGISTERS
3954 022202 ENDMSG
022202 104423 L10007: TRAP C$MSG
3955
3956 :-----
3957 :.SBTTL ....ERROR HANDLER -- ERR21 -- USYRT "WRONG ADDR" ERROR
3958 :-----
3959 022204 BGNMSG ERR21
3960 022204 PRINTB #FMT21,@TXT12,MPCSR ERR21::
022204 013746 002422 MOV MPCSR,-(SP)
022210 012746 020005 MOV @TXT12,-(SP)
022214 012746 013151 MOV #FMT21,-(SP)
022220 012746 000003 MOV #3,-(SP)
022224 010600 MOV SP,RO
022226 104414 TRAP C$PNTB
022230 062706 000010 ADD #10,SP
3961 022234 PRINTB #FMT50,R3 ;GET/PRINT RESPONDING ADDRESS
022234 010346 MOV R3,-(SP)
022236 012746 013662 MOV #FMT50,-(SP)
022242 012746 000002 MOV #2,-(SP)
022246 010600 MOV SP,RO
022250 104414 TRAP C$PNTB
022252 062706 000006 ADD #6,SP
3962 022256 004737 023454 JSR PC,ERR12# ;GET & PRINT USYRT REGISTERS
3963 022262 ENDMSG
022262 L10010:

```

....ERROR HANDLER -- ERR21 -- USYRT "WRONG ADDR" ERROR

```

022262 104423                                TRAP      C$MSG
3964
3965
3966
3967
3968 022264                                BGNMSG  ERR22
022264                                PRINTB  @FMT21,@TXT12,MPCSR
3969 022264                                ERR22::
022264 013746 002422                                MOV      MPCSR,-(SP)
022270 012746 020005                                MOV      @TXT12,-(SP)
022274 012746 013151                                MOV      @FMT21,-(SP)
022300 012746 000003                                MOV      @3,-(SP)
022304 010600                                MOV      SP,RO
022306 104414                                TRAP    C$PNTB
022310 062706 000010                                ADD     @10,SP
3970 022314                                PRINTB  @FMT51,GDATA,BDATA      ;GET/PRINT GOOD/BAD BIT COUNTS
022314 013746 002332                                MOV      BDATA,-(SP)
022320 013746 002330                                MOV      GDATA,-(SP)
022324 012746 013725                                MOV      @FMT51,-(SP)
022330 012746 000003                                MOV      @3,-(SP)
022334 010600                                MOV      SP,RO
022336 104414                                TRAP    C$PNTB
022340 062706 000010                                ADD     @10,SP
3971 022344 004737 023454                                JSR     PC,ERR12$      ;GET & PRINT USYRT REGISTERS
3972 022350                                ENDMSG
022350                                L10011:
022350 104423                                TRAP    C$MSG
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985 022352 010146                                XORGB: MOV      R1,-(SP)      ;PRESERVE WORKING REGISTER
3986 022354 013701 002330                                MOV      GDATA,R1      ;GET "GOOD" DATA
3987 022360 013737 002332 002334                                MOV      BDATA,XDATA    ;AND "BAD" DATA
3988 022366 074137 002334                                XOR      R1,XDATA      ;PERFORM EXCLUSIVE OR
3989 022372 012601                                MOV      (SP)+,R1      ;RESTORE R1
3990 022374 000207                                RTS      PC              ;RETURN
3991
3992
3993
3994
3995
3996
3997 022376                                SBTTL   .....ERROR HANDLER SUBROUTINE -- ERR4$
022376 012746 017005                                IDENTIFY & DUMP THE BYTE SELECT REGISTERS
022402 012746 017207                                ERR4$: PRINTX @FMT4,@TXT3,@TXT1
022406 012746 011715                                MOV      @TXT1,-(SP)
022412 012746 000003                                MOV      @TXT3,-(SP)
022416 010600                                MOV      @FMT4,-(SP)
                                MOV      @3,-(SP)
                                MOV      SP,RO

```

.....ERROR HANDLER SUBROUTINE -- ERR4\$

	022420	104415			TRAP	C\$PNTX
	022422	062706	000010		ADD	#10,SP
3998	022426			PRINTX	#FMT4A,BSR0,BSR1,BSR2,BSR3	
	022426	013746	002214		MOV	BSR3,-(SP)
	022432	013746	002212		MOV	BSR2,-(SP)
	022436	013746	002210		MOV	BSR1,-(SP)
	022442	013746	002206		MOV	BSR0,-(SP)
	022446	012746	011753		MOV	#FMT4A,-(SP)
	022452	012746	000005		MOV	#5,-(SP)
	022456	010600			MOV	SP,R0
	022460	104415			TRAP	C\$PNTX
	022462	062706	000014		ADD	#14,SP
3999	022466			PRINTX	#FMT4B,#TXT2	
	022466	012746	017043		MOV	#TXT2,-(SP)
	022472	012746	012006		MOV	#FMT4B,-(SP)
	022476	012746	000002		MOV	#2,-(SP)
	022502	010600			MOV	SP,R0
	022504	104415			TRAP	C\$PNTX
	022506	062706	000006		ADD	#6,SP
4000	022512			PRINTX	#FMT4C,BSR4,BSR5,BSR6,BSR7	
	022512	013746	002224		MOV	BSR7,-(SP)
	022516	013746	002222		MOV	BSR6,-(SP)
	022522	013746	002220		MOV	BSR5,-(SP)
	022526	013746	002216		MOV	BSR4,-(SP)
	022532	012746	012013		MOV	#FMT4C,-(SP)
	022536	012746	000005		MOV	#5,-(SP)
	022542	010600			MOV	SP,R0
	022544	104415			TRAP	C\$PNTX
	022546	062706	000014		ADD	#14,SP
4001	022552			PRINTX	#FMT4B,#TXT2A	
	022552	012746	017105		MOV	#TXT2A,-(SP)
	022556	012746	012006		MOV	#FMT4B,-(SP)
	022562	012746	000002		MOV	#2,-(SP)
	022566	010600			MOV	SP,R0
	022570	104415			TRAP	C\$PNTX
	022572	062706	000006		ADD	#6,SP
4002	022576			PRINTX	#FMT4A,BSR10,BSR11,BSR12,BSR13	
	022576	013746	002234		MOV	BSR13,-(SP)
	022602	013746	002232		MOV	BSR12,-(SP)
	022606	013746	002230		MOV	BSR11,-(SP)
	022612	013746	002226		MOV	BSR10,-(SP)
	022616	012746	011753		MOV	#FMT4A,-(SP)
	022622	012746	000005		MOV	#5,-(SP)
	022626	010600			MOV	SP,R0
	022630	104415			TRAP	C\$PNTX
	022632	062706	000014		ADD	#14,SP
4003	022636			PRINTX	#FMT4B,#TXT2B	
	022636	012746	017144		MOV	#TXT2B,-(SP)
	022642	012746	012006		MOV	#FMT4B,-(SP)
	022646	012746	000002		MOV	#2,-(SP)
	022652	010600			MOV	SP,R0
	022654	104415			TRAP	C\$PNTX
	022656	062706	000006		ADD	#6,SP
4004	022662			PRINTX	#FMT4C,BSR14,BSR15,BSR16,BSR17	
	022662	013746	002244		MOV	BSR17,-(SP)
	022666	013746	002242		MOV	BSR16,-(SP)
	022672	013746	002240		MOV	BSR15,-(SP)

.....ERROR HANDLER SUBROUTINE -- ERR4\$

022676 013746 002236
 022702 012746 012013
 022706 012746 000005
 022712 010600
 022714 104415
 022716 062706 000014
 4005 022722 000207

RTS PC

MOV BSR14, -(SP)
 MOV #FMT4C, -(SP)
 MOV #5, -(SP)
 MOV SP, RO
 TRAP C\$PNTX
 ADD #14, SP

4006
 4007
 4008
 4009
 4010

```

;-----
;SBTTL .....ERROR HANDLER SUBROUTINE -- ERR5$
;-----
;
;COMMON ERROR SUBROUTINE TO PRINT SELECT REGISTERS
ERR5$:

```

4011 022724
 4012 022724
 022724 012746 017237
 022730 012746 017342
 022734 012746 011715
 022740 012746 000003
 022744 010600
 022746 104415
 022750 062706 000010
 4013 022754
 022754 013746 002214
 022760 013746 002212
 022764 013746 002210
 022770 013746 002206
 022774 012746 012323
 023000 012746 000005
 023004 010600
 023006 104415
 023010 062706 000014
 4014 023014
 023014 012746 017277
 023020 012746 012006
 023024 012746 000002
 023030 010600
 023032 104415
 023034 062706 000006
 4015 023040
 023040 013746 002224
 023044 013746 002222
 023050 013746 002220
 023054 013746 002216
 023060 012746 012323
 023064 012746 000005
 023070 010600
 023072 104415
 023074 062706 000014
 4016 023100
 023100 012746 011564
 023104 012746 000001
 023110 010600
 023112 104414
 023114 062706 000004
 4017 023120 000207

PRINTX #FMT4, #TXT6, #TXT4

MOV #TXT4, -(SP)
 MOV #TXT6, -(SP)
 MOV #FMT4, -(SP)
 MOV #3, -(SP)
 MOV SP, RO
 TRAP C\$PNTX
 ADD #10, SP

PRINTX #FMT11, WSR0, WSR2, WSR4, WSR6 ;DUMP THE SELECT REGISTERS

MOV WSR6, -(SP)
 MOV WSR4, -(SP)
 MOV WSR2, -(SP)
 MOV WSR0, -(SP)
 MOV #FMT11, -(SP)
 MOV #5, -(SP)
 MOV SP, RO
 TRAP C\$PNTX
 ADD #14, SP

PRINTX #FMT4B, #TXT4A

MOV #TXT4A, -(SP)
 MOV #FMT4B, -(SP)
 MOV #2, -(SP)
 MOV SP, RO
 TRAP C\$PNTX
 ADD #6, SP

PRINTX #FMT11, WSR10, WSR12, WSR14, WSR16 ;DUMP THE SELECT REGISTERS

MOV WSR16, -(SP)
 MOV WSR14, -(SP)
 MOV WSR12, -(SP)
 MOV WSR10, -(SP)
 MOV #FMT11, -(SP)
 MOV #5, -(SP)
 MOV SP, RO
 TRAP C\$PNTX
 ADD #14, SP

PRINTB #ENDEMB

MOV #ENDEMB, -(SP)
 MOV #1, -(SP)
 MOV SP, RO
 TRAP C\$PNTB
 ADD #4, SP

RTS PC

4018
 4019

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

```

4020      .SBTTL .....ERROR HANDLER SUBROUTINE -- ERR11$
4021      ;-----
4022      ;      COMMON ERROR SUBROUTINE TO GET/PRINT VIA REGISTERS
4023
4024      ERR11$: JSR      PC,GETVRS      ;GET VIA REGS FOR PRINTOUT
4025      PRINTX  #FMT24,#TXT16,#TXT17
4026      PRINTX  #FMT25,VREGS+0,VREGS+2,VREGS+4,VREGS+6
4027      PRINTX  #FMT29 #TXT18
4028      PRINTX  #FMT26,VREGS+8.,VREGS+10.,VREGS+12.,VREGS+14.
4029      PRINTX  #FMT29,#TXT19
4030      PRINTX  #FMT25,VREGS+16.,VREGS+18.,VREGS+20.,VREGS+22.

```

023122	004737	004426	MOV	#TXT17,-(SP)
023126	012746	020163	MOV	#TXT16,-(SP)
023132	012746	020150	MOV	#FMT24,-(SP)
023136	012746	013262	MOV	#3,-(SP)
023142	012746	000003	MOV	SP,R0
023146	010600		TRAP	C#PNTX
023150	104415		ADD	#10,SP
023152	062706	000010	MOV	VREGS+6,-(SP)
023156	013746	002274	MOV	VREGS+4,-(SP)
023162	013746	002272	MOV	VREGS+2,-(SP)
023166	013746	002270	MOV	VREGS+0,-(SP)
023172	013746	002266	MOV	#FMT25,-(SP)
023176	012746	013275	MOV	#5,-(SP)
023202	012746	000005	MOV	SP,R0
023206	010600		TRAP	C#PNTX
023210	104415		ADD	#14,SP
023212	062706	000014	MOV	#TXT18,-(SP)
023216	012746	020220	MOV	#FMT29,-(SP)
023222	012746	013423	MOV	#2,-(SP)
023226	012746	000002	MOV	SP,R0
023232	010600		TRAP	C#PNTX
023234	104415		ADD	#6,SP
023236	062706	000006	MOV	VREGS+14,-(SP)
023242	013746	002304	MOV	VREGS+12,-(SP)
023246	013746	002302	MOV	VREGS+10,-(SP)
023252	013746	002300	MOV	VREGS+8,-(SP)
023256	013746	002276	MOV	#FMT26,-(SP)
023262	012746	013325	MOV	#5,-(SP)
023266	012746	000005	MOV	SP,R0
023272	010600		TRAP	C#PNTX
023274	104415		ADD	#14,SP
023276	062706	000014	MOV	#TXT19,-(SP)
023302	012746	020261	MOV	#FMT29,-(SP)
023306	012746	013423	MOV	#2,-(SP)
023312	012746	000002	MOV	SP,R0
023316	010600		TRAP	C#PNTX
023320	104415		ADD	#6,SP
023322	062706	000006	MOV	VREGS+22,-(SP)
023326	013746	002314	MOV	VREGS+20,-(SP)
023332	013746	002312	MOV	VREGS+18,-(SP)
023336	013746	002310	MOV	VREGS+16,-(SP)
023342	013746	002306	MOV	#FMT25,-(SP)
023346	012746	013275	MOV	#5,-(SP)
023352	012746	000005	MOV	SP,R0
023356	010600		TRAP	C#PNTX
023360	104415		ADD	#14,SP
023362	062706	000014		

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

```

4031 023366          PRINTX  #FMT29,#TXT20
      023366 012746 020315
      023372 012746 013423
      023376 012746 000002
      023402 010600
      023404 104415
      023406 062706 000006
4032 023412          PRINTX  #FMT26,VREGS+24.,VREGS+26.,VREGS+28.,VREGS+30.
      023412 013746 002324
      023416 013746 002322
      023422 013746 002320
      023426 013746 002316
      023432 012746 013325
      023436 012746 000005
      023442 010600
      023444 104415
      023446 062706 000014
4033 023452          RTS      PC
4034
4035
4036 ;-----
4037 ;.SBTTL .....ERROR HANDLER SUBROUTINE -- ERR12$
4038 ;
4039 ;      COMMON ERROR ROUTINE TO GET AND PRINTOUT USYRT REGISTERS
4040 ERR12$: JSR      PC,GETURS          ;GET USYRT REGS FOR PRINTOUT
4041          PRINTX  #FMT24,#TXT13,#TXT14
      023460 012746 020050
      023464 012746 020033
      023470 012746 013262
      023474 012746 000003
      023500 010600
      023502 104415
      023504 062706 000010
4042          PRINTX  #FMT25,UREGS+0,UREGS+2,UREGS+4,UREGS+6
      023510 013746 002254
      023514 013746 002252
      023520 013746 002250
      023524 013746 002246
      023530 012746 013275
      023534 012746 000005
      023540 010600
      023542 104415
      023544 062706 000014
4043          PRINTX  #FMT29,#TXT15
      023550 012746 020106
      023554 012746 013423
      023560 012746 000002
      023564 010600
      023566 104415
      023570 062706 000006
4044          PRINTX  #FMT26,UREGS+10,UREGS+12,UREGS+14,UREGS+16
      023574 013746 002264
      023600 013746 002262
      023604 013746 002260
      023610 013746 002256
      023614 012746 013325
      023620 012746 000005

```

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

023624 010600
023626 104415
023630 062706 000014
4045 023634 000207
4046
4047

RTS PC

.EVEN

MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

LOAD DEVICE PROTECTION TABLE

4049
 4050
 4051
 4052
 4053
 4054
 4055
 4056 023636
 023636
 4057 023636 177777
 4058 023640 177777
 4059 023642 177777
 4060 023644

.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

L\$PROT::

```

.WORD -1 ;DON'T CHK CSR ADRS
.WORD -1 ;DON'T CHK MASSBUS UNIT NO.
.WORD -1 ;DON'T CHK DRIVE NO.
ENDPROT

```

INITIALIZE SECTION

```

4062          .SBTTL  INITIALIZE SECTION
4063
4064          ;////////////////////////////////////
4065          ;/ THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
4066          ;/ AT THE BEGINNING OF THE TEST SEQUENCE ON THE NEXT UNIT.
4067          ;////////////////////////////////////
4068
4069 023644          BGNINIT
4069 023644
4070
4071 023644          SETVEC  #140,#170000,#340          ;ODT ROM ADDRESS
4071 023644 012746 000340          MOV          ;JB REV A-0
4071 023650 012746 170000          MOV          #340,-(SP)
4071 023654 012746 000140          MOV          #170000,-(SP)
4071 023660 012746 000003          MOV          #140,-(SP)
4071 023664 104437          TRAP          #3,-(SP)
4071 023666 062706 000010          ADD          C$SVEC
4071 023666 062706 000010          ADD          #10,SP
4072
4073
4074 023672 010637 002344          MOV          SP,PSTACK          ;SAVE BASE-LEVEL STACK POINTER
4075 023676 005037 002350          CLR          SUBRPC          ;CLEAR SUBR CALL PC
4076 023702 005037 002404          CLR          CHPTYP          ;CLEAR USYRT CHIP TYPE INDICATOR
4077 023706 005037 002402          CLR          ERROR1          ;CLEAR ERROR FLAG
4078 023712 005037 002406          CLR          SAVLEN          ;CLEAR CHAR LENGTH FROM SETUP
4079 023716 005737 002374          TST          FRSTIM          ;SEE IF FIRST TIME THROUGH AFTER LOAD
4080 023722 001007          BNE          6$          ;BR IF NOT
4081 023724 013737 000004 002376          MOV          @#4,SAVE4          ;SAVE ERROR TRAP VECTOR
4082 023732 013737 000006 002400          MOV          @#6,SAVE6
4083 023740 000406          BR          9$
4084
4085 023742 013737 002376 000004 6$: MOV          SAVE4,@#4          ;RESTORE ERROR TRAP VECTOR
4086 023750 013737 002400 000006          MOV          SAVE6,@#6
4087
4088 023756 012737 000001 002374 9$: MOV          #1,FRSTIM          ;MARK FLAG FOR NEXT TIME THROUGH
4089
4090          ;SEE IF PROGRAM JUST STARTED, BR IF YES
4091 023764          READEF  #EF.START
4091 023764 012700 000040          MOV          #EF.START,R0
4091 023770 104447          TRAP          C$REFG
4092 023772          BCOMPLETE          STARST
4092 023772 103415          BCS          STARST
4093
4094          ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
4095 023774          READEF  #EF.RESTART
4095 023774 012700 000037          MOV          #EF.RESTART,R0
4095 024000 104447          TRAP          C$REFG
4096 024002          BCOMPLETE          STARST
4096 024002 103411          BCS          STARST
4097
4098          ;SEE IF THIS IS A NEW PASS, BR IF YES
4099 024004          READEF  #EF.NEW
4099 024004 012700 000035          MOV          #EF.NEW,R0
4099 024010 104447          TRAP          C$REFG
4100 024012          BCOMPLETE          NEWST
4100 024012 103411          BCS          NEWST
4101
4102          ;SEE IF PROGRAM WAS JUST CONTINUED

```

INITIALIZE SECTION

```

4103 024014          READEF  #EF.CONTINUE
      024014 012700 000036
      024020 104447
4104 024022          BCOMPLETE      ENDIT
      024022 103473
4105 024024 000414  BR      GETPRM
4106
4107 024026          STARST:
4108 024026 005037 002416  CLR      STARES          ;CLEAR FLAG TO SHOW JUST HAD STA OR RES
4109
4110          ;CLEAR DEVICE MAP
4111 024032 005037 002410  CLR      DEVMAP
4112 024036
4113 024036 012737 177777 002340  NEWST:  MOV      #-1,LOGDEV      ;RESET LOGICAL DEVICE TO -1
4114 024044 005237 002416          INC      STARES          ;INCREMENT NO. OF PASSES SINCE STA OR RES
4115 024050 012737 000001 002412  MOV      #BITO,DEVPTR      ;INIT DEVICE MAP BIT POINTER
4116
4117          ; GET UNIBUS ADDRESS, VECTOR, PRIORITY LEVEL, SWITCH PACKS, TEST
4118          ; CONNECTOR INFORMATION FOR THIS LOGICAL DEVICE
4119 024056          GETPRM:
4120 024056 005237 002340          INC      LOGDEV          ;INCREMENT LOGICAL DEVICE NUMBER
4121 024062          GPHARD  LOGDEV,R1      ;GET P-TABLE POINTER INTO R1
      024062 013700 002340
      024066 104442
      024070 010001
4122 024072          BCOMPLETE      10$      ;BR IF DEVICE AVAILABLE
      024072 103403
4123 024074 006337 002412
4124 024100 000766
4125 024102 053737 002412 002410 10$:  BIS      DEVPTR,DEVMAP      ;SET BIT FOR THIS DEVICE
4126 024110 006337 002412  ASL      DEVPTR          ;SHIFT BIT POINTER
4127
4128 024114 012102
4129 024116 012703 002422  MOV      (R1)+,R2          ;R2=CSR ADDR VALUE
      MOV      #MPCSR,R3      ;R3=POINTER TO CSR ADDR STORAGE AREA
4130
4131 024122 010223          11$:  MOV      R2,(R3)+          ;PUT CSR ADDRESSES IN 'BSEL' AREA
4132 024124 005202          INC      R2              ;BUMP BSEL ADDR
4133 024126 022703 002462  CMP      #BSEL17+2,R3      ;ALL 16 ADDRESSES MOVED ?
4134 024132 001373          BNE      11$              ;NO: DO ANOTHER ADDRESS
4135
4136
4137 024134 011137 002462  MOV      (R1),MPIVEC      ;GET DMV11 INPUT INTRPT VECTOR
4138 024140 012137 002464  MOV      (R1)+,MPOVEC      ;GET DMV11 OUTPUT INTRPT VECTOR
4139 024144 062737 000004 002464  ADD      #4,MPOVEC
4140 024152 012137 002466  MOV      (R1)+,MPRIOR      ;GET DMV11 DEVICE PRIORITY
4141 024156 012137 002470  MOV      (R1)+,LUSWI1      ;GET LU SWITCH PACK #1
4142 024162 012137 002472  MOV      (R1)+,LUSWI2      ;GET LU SWITCH PACK #2
4143 024166 012137 002474  MOV      (R1)+,BRDTYP      ;GET DMV-11 BOARD TYPE
4144 024172 012137 002476  MOV      (R1)+,TSTCON      ;GET TEST CONNECTOR INDICATOR
4145 024176 011137 002500  MOV      (R1)+,BDRATE      ;GET BAUD RATE FOR THIS DEVICE
4146          ;ISSUE LSI BUS RESET, TO INIT DMV11
4147 024202          BRESET
      024202 104433
4148 024204 005000          TRAP      C$RESET
4149 024206 000240          15$:  CLR      RO              ;# TIME DELAY TO ALLOW COMPLETION
4150 024210 077002          NOP
4151 024212          SOB      RO,15$      ;# OF DMV11 MICRODIAGNOSTICS.
      ENDIT:

```

INITIALIZE SECTION

4152 024212
024212
024212 104411

ENDINIT

L10013: TRAP C\$INIT

AUTO DROP UNIT SECTION

.SBTTL AUTO DROP UNIT SECTION

;/ THE AUTO DROP CODING DETERMINES WHETHER OR NOT THE DEVICE WHOSE P-TABLE WAS JUST OBTAINED IS READY FOR TESTING, AND IT IS DROPPED IF NOT READY.

THIS ALGORITHM IS THE SAME A CNDMA TEST # 1 EXCEPT THAT TEST WILL JUST REPORT THE FAILURE AND GO ON -- THIS ROUTINE WILL CAUSE THE DEVICE TO BE DROPPED IF A BUS-TIMEOUT OCCURS WHEN ANY OF THE CSR'S ARE ACCESSED WITH EITHER A "TST" OR "TSTB" INSTRUCTION.

4154 .SBTTL AUTO DROP UNIT SECTION
4155
4156 ;// THE AUTO DROP CODING DETERMINES WHETHER OR NOT THE DEVICE WHOSE P-TABLE
4157 ;/ WAS JUST OBTAINED IS READY FOR TESTING, AND IT IS DROPPED IF NOT READY.
4158 ;//
4159
4160
4161 ;+*****
4162 ;
4163 ; THIS ALGORITHM IS THE SAME A CNDMA TEST # 1 EXCEPT THAT TEST
4164 ; WILL JUST REPORT THE FAILURE AND GO ON -- THIS ROUTINE WILL CAUSE THE
4165 ; DEVICE TO BE DROPPED IF A BUS-TIMEOUT OCCURS WHEN ANY OF THE CSR'S
4166 ; ARE ACCESSED WITH EITHER A "TST" OR "TSTB" INSTRUCTION.
4167 ;
4168 ; -*****
4169
4170 024214 BGNAUTO
4171 024214 L\$AUTO::
4172 024214 SETVEC #4,#AD.HIT,#0 ;SETUP INVALID-ADDRESS TRAP VECTOR
024214 012746 000000 MOV #0,-(SP)
024220 012746 024332 MOV #AD.HIT,-(SP)
024224 012746 000004 MOV #4,-(SP)
024230 012746 000003 MOV #3,-(SP)
024234 104437 TRAP C\$SVEC
024236 062706 000010 ADD #10,SP
4173 024242 CLR TMP0 ;INITIALIZE TRAP FLAG REGISTER
4174 024246 MOV #1,R2 ;FLAG BIT
4175 024252 MOV BSELO,R3 ;INIT ADDRESS POINTER
4176
4177 024256 105723 1\$: TSTB (R3)+ ;ACCESS THE CSR'S BY BYTES.
4178 024260 006302 ASL R2
4179 024262 103375 BCC 1\$
4180
4181 024264 013703 002422 MOV BSELO,R3 ;RE-INIT ADDRESS POINTER
4182 024270 012702 000001 MOV #1,R2 ;RE-INIT FLAG BIT
4183 024274 005723 2\$: TST (R3)+ ;ACCESS THE CSR'S BY WORDS.
4184 024276 006302 ASL R2
4185 024300 006302 ASL R2
4186 024302 103374 BCC 2\$
4187
4188 024304 CLRVEC #4 ;RESTORE THE VECTOR TO DS
024304 012700 000004 MOV #4,R0
024310 104436 TRAP C\$CVEC
4189 024312 005737 002552 TST TMP0 ;DID WE GET HIT WITH AN INVALID ADDRESS TRAP?
4190 024316 001403 BEQ AD.OK ;NO, EXIT TEST
4191 024320 DODU LOGDEV ;YES, DROP THIS LOGICAL DEV.
024320 013700 002340 MOV LOGDEV,R0
024324 104451 TRAP C\$DODU
4192
4193 024326 000240 AD.OK: NOP ;(FOR PATCHING IN A HALT IF NECESSARY)
4194
4195 024330 ENDAUTO
024330 L10014:
024330 104461 TRAP C\$AUTO
4196
4197 024332 050237 002552 AD.HIT: BIS R2,TMP0 ;FLAG THE HIT IF WE GET IT!

N8

CNDMDAO DMV11 LINE UNIT DIAG2 MACRO M1200 22-FEB-84 15:39 PAGE 59-1

SEQ 0104

AUTO DROP UNIT SECTION

4198 024336 000002
4199

RTI

;RETURN

CLEANUP CODING SECTION

4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211

024340
024340

024340
024340
024340 104412

.SBTTL CLEANUP CODING SECTION

:/ THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
:/ AT THE END OF THE TEST SEQUENCE ON A PARTICULAR UNIT.

BGNCLN

ENDCLN

L\$CLEAN::

L10015: TRAP C\$CLEAN

DROP UNIT SECTION

4213
4214
4215
4216
4217
4218
4219

.SBTTL DROP UNIT SECTION

:/ THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
:/ TO NO LONGER BE TESTED.

4220 024342
024342
4221
4222 024342 104433
024342
4223 024344
024344 104453

BGNDU
:ISSUE UNIBUS RESET TO CLEAN UP
BRESET
ENDDU

L\$DU::
TRAP C\$RESET
L10016:
TRAP C\$DU

ADD UNIT SECTION

4225
4226
4227
4228
4229
4230
4231
4232

.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.
;/ ;

4233 024346
024346
4234 024346
024346
024346 104452

BGNAU
ENDAU

L\$AU::
L10017: TRAP C\$AU

TEST 1 -- VRC PARITY GENERATION TEST

4258

.SBTTL TEST 1 -- VRC PARITY GENERATION TEST

```

*****
;*
;* TEST 1 -- VRC PARITY GENERATION TEST
;*
;* SUBTEST 1 - TEST OF CORRECT ODD VRC PARITY GENERATION :
;* THE LINE UNIT IS PLACED IN CHAR MODE, WITH ODD VRC. AND 7-BIT CHARS SELECTED.
;* THE DATA CHARS IN PATTERN Q ARE LOADED/TRANSMITTED/READ. AS THE 8TH BIT
;* (PARITY BIT) OF EACH DATA CHAR IS SENT THE PROGRAM CHECKS TSO FOR THE PROPER
;* STATE. FOR THE FIRST 4 CHARS IN PATTERN Q THE PARITY BIT SHOULD = 1, FOR THE
;* LAST 4 CHARACTERS IT SHOULD = 0.
;*
;* SUBTEST 2 - TEST OF CORRECT EVEN VRC PARITY GENERATION :
;* THE LINE UNIT IS PLACED IN CHAR MODE, WITH EVEN VRC AND 7-BIT CHARS SELECTED.
;* THE DATA CHARS IN PATTERN Q ARE LOADED/TRANSMITTED/READ. AS THE 8TH BIT
;* (PARITY BIT) OF EACH DATA CHAR IS SENT THE PROGRAM CHECKS TSO FOR THE PROPER
;* STATE. FOR THE FIRST 4 CHARS IN PATTERN Q THE PARITY BIT SHOULD = 0, FOR THE
;* LAST 4 CHARACTERS IT SHOULD = 1.
;*
;* DATA PATTERN Q = 000,003,014,060,001,007,037,177
;*
;* NOTE: SINCE THE ROUTINE "SERIAL" TREATS THE FIRST BIT RECEIVED FROM THE
;* USYRT AS THE MSB, THE "EXPECTED BIT SEQUENCE" WILL HAVE A REVERSED
;* BIT ORDER.

```

```

;*
;* BGNTST
;*
;* T1::

```

```

;-----
; SUBTEST #1: ODD VRC PARITY CHECK
;-----

```

```

BGNSUB
;-----
; T1.1:
; TRAP C$BSUB
JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
;-----
JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
DDCMP!OVRC!226 ;SET DDCMP,ODD VRC CHECK,SYNCH=226
TXDL ;USE 7 BIT TX CHARS
BCC .+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR
; TRAP C$ERROR
ESCAPE SUB ;SKIP REMAINDER OF THIS SUBTEST
; TRAP C$ESCAPE
; .WORD L10021-.
;-----
JSR R5,TXCTRL ;CLEAR TSOM
OOO
O
;-----
JSR R5,WRITEI ;LOAD 1ST DATA CHARACTER (000)
TDSRL
OOO
BCC .+8. ;BR IF NO ERROR
ERROR ;PRINT STACKED ERROR MESSAGE

```

```

024350
4259
4260
4261
4262 024350
024350
024350 104402
4263 024352 004737 005344
4264
4265 024356 004537 007324
4266 024362 042226
4267 024364 000340
4268 024366 103003
4269 024370
024370 104460
4270 024372
024372 104410
024374 000310
4271
4272 024376 004537 007734
4273 024402 000000
4274 024404 000000
4275
4276 024406 004537 003660
4277 024412 120402
4278 024414 000000
4279 024416 103003
4280 024420

```

TEST 1 -- VRC PARITY GENERATION TEST

```

4281 024420 104460                                     TRAP  C$ERROR
      024422                                     ESCAPE SUB ;AND EXIT SUBTEST
      024422 104410                                     TRAP  C$ESCAPE
      024424 000260                                     .WORD L10021-.

4282 ;----- READ SYNCH CHARACTER -----
4283 024426 004537 007042      JSR  R5,CHKTSO ;CHECK 1ST BIT OF EXPECTED "SYNCH"
4284 024432 000000      0 ; CHARACTER (SHOULD BE 0)
4285 024434 103003      BCC  .+8. ;BR IF NO ERROR
4286 024436                                     ERROR ;REPORT STACKED ERROR
      024436 104460                                     TRAP  C$ERROR
4287 024440                                     ESCAPE SUB ;AND EXIT SUBTEST
      024440 104410                                     TRAP  C$ESCAPE
      024442 000242                                     .WORD L10021-.

4288
4289 024444 004537 007202      JSR  R5,SERIAL ;READ REMAINING 7 BITS OF "SYNCH" CHARACTER
4290 024450 000007      7. ; (OFF OF TSO BIT)
4291 024452 00C150      150 ; EXPECTED BIT SEQUENCE (0010110)
4292 024454 103003      BCC  .+8. ;BR IF NO ERROR
4293 024456                                     ERROR ;REPORT STACKED ERROR
      024456 104460                                     TRAP  C$ERROR
4294 024460                                     ESCAPE SUB ;AND EXIT SUBTEST
      024460 104410                                     TRAP  C$ESCAPE
      024462 000222                                     .WORD L10021-.

4295 ;----- LOAD/TX/READ PARITY BIT=1 CHARACTERS -----
4296 024464 012703 003012      MOV  #PATQ+1,R3 ;SET UP TX CHARACTER POINTER
4297 024470 012704 003021      MOV  #PATQB,R4 ;SET UP RX CHARACTER POINTER
4298 024474 112337 024512      1$: MOV  (R3)+,2$ ;SET UP NEXT TX CHAR
4299 024500 112437 024532      MOV  (R4)+,3$ ;SET UP NEXT RX CHARACTER
4300
4301 024504 004537 003660      JSR  R5,WRITEI ;LOAD NEXT TX CHARACTER
4302 024510 120402      TDSRL
4303 024512 000000      2$: 000 ;** HOLE FOR TX CHARACTER
4304 024514 103003      BCC  .+8. ;BR IF NO ERROR
4305 024516                                     ERROR ;PRINT STACKED ERROR MESSAGE
      024516 104460                                     TRAP  C$ERROR
4306 024520                                     ESCAPE SUB ;AND EXIT SUBTEST
      024520 104410                                     TRAP  C$ESCAPE
      024522 000162                                     .WORD L10021-.

4307
4308 024524 004537 007202      JSR  R5,SERIAL ;CLOCK/CHECK PREVIOUS TX CHAR (1 CHAR BUFFER)
4309 024530 000007      7
4310 024532 000000      3$: 000 ;** HOLE FOR EXPECTED BIT SEQUENCE
4311 024534 103003      BCC  .+8. ;BR IF NO ERROR
4312 024536                                     ERROR ;REPORT STACKED ERROR
      024536 104460                                     TRAP  C$ERROR
4313 024540                                     ESCAPE SUB ;SKIP REMAINDER OF THIS SUBTEST
      024540 104410                                     TRAP  C$ESCAPE
      024542 000142                                     .WORD L10021-.

4314
4315 024544 004537 011540      JSR  R5,STEPLU ;CLOCK PARITY BIT TO TSO
4316 024550 000001      1
4317
4318 024552 004537 007042      JSR  R5,CHKTSO ;CHECK STATE OF PARITY BIT
4319 024556 000001      1 ; (SHOULD BE 1)
4320 024560 103006      BCC  4$ ;BR IF NO ERROR
4321 024562      GEDF EM48,ERR12 ;REPORT "ODD VRC PARITY BIT NOT SET"
; "DEVICE FATAL" ERROR # 39

```

TEST 1 -- VRC PARITY GENERATION TEST

```

024562 104455 TRAP C$ERDF
024564 000047 .WORD 39
024566 015142 .WORD EM48
024570 021714 .WORD ERR12
4322 024572 ESCAPE SUB ;SKIP REMAINDER OF THIS SUBTEST
024572 104410 TRAP C$ESCAPE
024574 000110 .WORD L10021-.

4323
4324 024576 020327 003016 4$: CMP R3,#PATQ+5 ;
4325 024602 001334 BNE 1$ ;BR IF TSO=1 CHECKS ARE NOT COMPLETE
4326 ;----- LOAD/TX/READ PARITY BIT=0 CHARACTERS -----
4327 024604 112337 024622 11$: MOVB (R3)+,12$ ;SET UP NEXT TX CHAR
4328 024610 112437 024632 MOVB (R4)+,13$ ;SET UP NEXT RX CHARACTER
4329
4330 024614 004537 003660 JSR R5,WRITEI ;LOAD NEXT TX CHARACTER
4331 024620 120402 TDSRL
4332 024622 000000 12$: 000 ;** HOLE FOR TX CHARACTER
4333
4334 024624 004537 007202 JSR R5,SERIAL ;CLOCK/CHECK PREVIOUS TX CHAR (1 CHAR BUFFER)
4335 024630 000007 7
4336 024632 000000 13$: 000 ;** HOLE FOR EXPECTED BIT SEQUENCE
4337 024634 103003 BCC .+8. ;BR IF NO ERROR
4338 024636 ERROR ;REPORT STACKED ERROR
024636 104460
4339 024640 ESCAPE SUB ;SKIP REMAINDER OF THIS SUBTEST TRAP C$ERROR
024640 104410 TRAP C$ESCAPE
024642 000042 .WORD L10021-.

4340
4341 024644 004537 011540 JSR R5,STEPLU ;CLOCK PARITY BIT TO TSO
4342 024650 000001 1
4343
4344 024652 004537 007042 JSR R5,CHKTSO ;CHECK STATE OF PARITY BIT
4345 024656 000000 0 ; (SHOULD BE 0)
4346 024660 103006 BCC 14$ ;BR IF NO ERROR
4347 024662 GEDF EM49,ERR12 ;REPORT "ODD VRC PARITY BIT NOT CLEARED"
; "DEVICE FATAL" ERROR # 40
024662 104455 TRAP C$ERDF
024664 000050 .WORD 40
024666 015175 .WORD EM49
024670 021714 .WORD ERR12
4348 024672 ESCAPE SUB ;SKIP REMAINDER OF SUBTEST
024672 104410 TRAP C$ESCAPE
024674 000010 .WORD L10021-.

4349
4350 024676 020327 003022 14$: CMP R3,#PATQ+9. ;
4351 024702 001340 BNE 11$ ;BR IF TSO=0 CHECKS ARE NOT COMPLETE
4352 024704 ENDSUB
024704 L10021: TRAP C$ESUB
024704 104403

4353
4354 ;-----
4355 ; SUBTEST #2: EVEN VRC PARITY CHECK
4356 ;-----
4357 024706 BGNSUB
024706 T1.2: TRAP C$BSUB
024706 104402
4358 024710 004737 005344 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP

```

TEST 1 -- VRC PARITY GENERATION TEST

```

4359
4360 024714 004537 007324      JSR      R5,INITRN      ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
4361 024720 042626              DDCMP!EVRC!226        ;SET DDCMP,EVEN VRC CHECK,SYNCH=226
4362 024722 000340              TXDL                  ;USE 7 BIT TX CHARS
4363 024724 103003              BCC      .+8.         ;BR IF NO ERROR
4364 024726 104460              ERROR                ;REPORT STACKED ERROR
                                TRAP      C$ERROR
4365 024730 104410              ESCAPE  SUB          ;SKIP TO END OF SUBTEST
                                TRAP      C$ESCAPE
                                .WORD    L10022-.
                                024732 000320
4366
4367 024734 004537 007734      JSR      R5,TXCTRL     ;CLEAR TSOM
4368 024740 000000              000
4369 024742 000000              0
4370 024744 004537 003660      JSR      R5,WRITEI     ;LOAD 1ST DATA CHARACTER (000)
4371 024750 120402              TDSRL
4372 024752 000000              000
4373 024754 103003              BCC      .+8.         ;BR IF NO ERROR
4374 024756 104460              ERROR                ;PRINT STACKED ERROR MESSAGE
                                TRAP      C$ERROR
4375 024760 104410              ESCAPE  SUB          ;AND EXIT SUBTEST
                                TRAP      C$ESCAPE
                                .WORD    L10022-.
4376
4377 024764 004537 007042      ;----- READ SYNCH CHARACTER -----
4378 024770 000000              JSR      R5,CHKTSO     ;CHECK 1ST BIT OF EXPECTED "SYNCH"
4379 024772 103003              0                    ; CHARACTER (SHOULD BE 0)
4380 024774 104460              BCC      .+8.         ;BR IF NO ERROR
4381 024776 104410              ERROR                ;REPORT STACKED ERROR
                                TRAP      C$ERROR
4382 025000 000252              ESCAPE  SUB          ;AND EXIT SUBTEST
                                TRAP      C$ESCAPE
                                .WORD    L10022-.
4383 025002 004537 007202      JSR      R5,SERIAL     ;READ REMAINING 7 BITS OF "SYNCH" CHARACTER
4384 025006 000007              7.                   ; (OFF OF TSO BIT)
4385 025010 000151              151                  ; EXPECTED BIT SEQUENCE (0010110)
4386 025012 103003              BCC      .+8.         ;BR IF NO ERROR
4387 025014 104460              ERROR                ;REPORT STACKED ERROR
                                TRAP      C$ERROR
4388 025016 104410              ESCAPE  SUB          ;AND EXIT SUBTEST
                                TRAP      C$ESCAPE
                                .WORD    L10022-.
4389
4390 025022 012703 003012      ;----- LOAD/TX/READ PARITY BIT=0 CHARACTERS -----
4391 025026 012704 003021      MOV      #PATQ+1,R3    ;SET UP TX CHARACTER POINTER
4392 025032 112337 025050      MOV      #PATQB,R4     ;SET UP RX CHARACTER POINTER
4393 025036 112437 025070      1$:      MOV      (R3)+,2$   ;SET UP NEXT TX CHAR
4394                                MOV      (R4)+,3$   ;SET UP NEXT RX CHARACTER
4395 025042 004537 003660      JSR      R5,WRITEI     ;LOAD NEXT TX CHARACTER
4396 025046 120402              TDSRL
4397 025050 000000              000                  ;** HOLE FOR TX CHARACTER
4398 025052 103003              BCC      .+8.         ;BR IF NO ERROR
4399 025054 104460              ERROR                ;PRINT STACKED ERROR MESSAGE
                                TRAP      C$ERROR
4400 025056 104410              ESCAPE  SUB          ;AND EXIT SUBTEST
                                TRAP      C$ESCAPE
                                .WORD    L10022-.
                                025060 000172

```


TEST 1 -- VRC PARITY GENERATION TEST

```

4401
4402 025062 004537 007202          JSR    R5,SERIAL          ;CLOCK/CHECK PREVIOUS TX CHAR (1 CHAR BUFFER)
4403 025066 000007                    7
4404 025070 000000          3$: 000          ;** HOLE FOR EXPECTED BIT SEQUENCE
4405 025072 103003          BCC    .+8.          ;BR IF NO ERROR
4406 025074 104460          ERROR          ;REPORT STACKED ERROR
4407 025076 104410          ESCAPE SUB          ;SKIP REMAINDER OF THIS SUBTEST      TRAP    C$ERROR
      025100 000152                    .WORD    L10022-.
4408
4409 025102 004537 011540          JSR    R5,STEPLU        ;CLOCK PARITY BIT TO TSO
4410 025106 000001                    1
4411
4412 025110 004537 007042          JSR    R5,CHKTSO        ;CHECK STATE OF PARITY BIT
4413 025114 000000                    0          ; (SHOULD BE 0)
4414 025116 103006          BCC    4$          ;BR IF NO ERROR
4415 025120 104455          GEDF   EM51,ERR12      ;REPORT "EVEN VRC PARITY NOT CLEARED"
      025122 000051                    ;          "DEVICE FATAL" ERROR # 41
      025124 015270                    .WORD    C$ERDF
      025126 021714                    .WORD    41
4416 025130 104410          ESCAPE SUB          ;SKIP REMAINDER OF THIS SUBTEST      TRAP    C$ERDF
      025132 000120                    .WORD    41
4417
4418 025134 020327 003016          4$:  CMP    R3,#PATQ*5          ;
4419 025140 001334          BNE    1$          ;BR IF TSO=0 CHECKS ARE NOT COMPLETE
4420          ;----- LOAD/TX/READ PARITY BIT=1 CHARACTERS -----
4421 025142 112337 025160          11$: MOVB   (R3)+,12$          ;SET UP NEXT TX CHAR
4422 025146 112437 025200          MOVB   (R4)+,13$          ;SET UP NEXT RX CHARACTER
4423
4424 025152 004537 003660          JSR    R5,WRITEI        ;LOAD NEXT TX CHARACTER
4425 025156 120402          TDSRL
4426 025160 000000          12$: 000          ;** HOLE FOR TX CHARACTER
4427 025162 103003          BCC    .+8.          ;BR IF NO ERROR
4428 025164 104460          ERROR          ;PRINT STACKED ERROR MESSAGE
4429 025166 104410          ESCAPE SUB          ;AND EXIT SUBTEST      TRAP    C$ERROR
      025170 000062                    .WORD    C$ESCAPE
4430
4431 025172 004537 007202          JSR    R5,SERIAL          ;CLOCK/CHECK PREVIOUS TX CHAR (1 CHAR BUFFER)
4432 025176 000007                    7
4433 025200 000000          13$: 000          ;** HOLE FOR EXPECTED BIT SEQUENCE
4434 025202 103003          BCC    .+8.          ;BR IF NO ERROR
4435 025204 104460          ERROR          ;REPORT STACKED ERROR
4436 025206 104410          ESCAPE SUB          ;SKIP REMAINDER OF THIS SUBTEST      TRAP    C$ERROR
      025210 000042                    .WORD    C$ESCAPE
4437
4438 025212 004537 011540          JSR    R5,STEPLU        ;CLOCK PARITY BIT TO TSO
4439 025216 000001                    1
4440
4441 025220 004537 007042          JSR    R5,CHKTSO        ;CHECK STATE OF PARITY BIT

```

TEST 1 -- VRC PARITY GENERATION TEST

```

4442 025224 000001          1          ; (SHOULD BE 1)
4443 025226 103006        BCC      14$          ;BR IF NO ERROR
4444 025230          GEDF    EM50,ERR12 ;REPORT "EVEN VRC PARITY NOT SET"
;          "DEVICE FATAL" ERROR # 42
          025230 104455          TRAP      C$ERDF
          025232 000052          .WORD    42
          025234 015234          .WORD    EM50
          025236 021714          .WORD    ERR12
4445 025240          ESCAPE  SUB          ;SKIP REMAINDER OF SUBTEST
          025240 104410          TRAP      C$ESCAPE
          025242 000010          .WORD    L10022-.
4446
4447 025244 020327 003022  14$:    CMP      R3,#PATQ.9. ;
4448 025250 001334          BNE      11$          ;BR IF TSO=1 CHECKS ARE NOT COMPLETE
4449 025252          ENDSUB
          025252          L10022:
          025252 104403          TRAP      C$ESUB
4450 025254          ENDTST          L10020:
          025254 104401          TRAP      C$ETST

```

TEST 2 -- VRC ERROR DETECTION TEST

4472

.SBTTL TEST 2 -- VRC ERROR DETECTION TEST

```

*****
;*
;* TEST 2 -- VRC ERROR DETECTION TEST
;*
;* SUBTEST 1 - FORCING OF RERR USING ODD VRC
;* THE USYRT IS PLACED IN CHAR MODE WITH ODD VRC AND BOTH TX AND RX CHAR
;* LENGTH=7 BITS. THE RECEIVER AND TRANSMITTER ARE THEN SYNC'D. WHEN THE FIRST
;* DATA CHARACTER IS LOADED INTO TXDB, THE RX CHAR LENGTH IS CHANGED TO 6 BITS.
;* TWO 7 BIT CHARACTERS (+PARITY) ARE THEN TRANSMITTED, RESULTING IN A 16 BIT
;* STREAM WHICH THE RECEIVER WILL READ AS TWO 6 BIT CHARS (+PARITY + 2 LEFT).
;* THE FIRST "CHARACTER" READ WILL HAVE THE CORRECT PARITY; THE SECOND WILL
;* NOT.
;*
;* SUBTEST 2 - FORCING OF RERR USING EVEN VRC
;* THE USYRT IS PLACED IN CHAR MODE WITH EVEN VRC AND BOTH TX AND RX CHAR
;* LENGTH=7 BITS. THE RECEIVER AND TRANSMITTER ARE THEN SYNC'D. WHEN THE FIRST
;* DATA CHARACTER IS LOADED INTO TXDB, THE RX CHAR LENGTH IS CHANGED TO 6 BITS.
;* TWO 7 BIT CHARACTERS (+PARITY) ARE THEN TRANSMITTED, RESULTING IN A 16 BIT
;* STREAM WHICH THE RECEIVER WILL READ AS TWO 6 BIT CHARS (+PARITY + 2 LEFT).
;* THE FIRST "CHARACTER" READ WILL HAVE THE CORRECT PARITY; THE SECOND WILL
;* NOT.
;*
*****

```

```

: BGNTST
:
:-----T2:-----
: SUBTEST #1: FORCING ODD VRC ERROR
:-----
: BGNSUB
:
: T2.1:
: TRAP C$BSUB
: JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
: JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
: DDCMP!OVRC!226 ;SET DDCMP,ODD VRC CHECK,SYNCH=226
: TXDL!RXDL ;TX/RX CHAR LENGTH=7 BITS
: BCC .+8. ;BR IF NO ERROR
: ERROR ;REPORT STACKED ERROR
: TRAP C$ERROR
: ESCAPE SUB ;SKIP TO END OF TEST
: TRAP C$ESCAPE
: .WORD L10024-.
:
: JSR R5,TXCTRL ;SET TSOM
: TSOM
: 7.
: JSR R5,TXCTRL ;SET TSOM AGAIN (KNOCK DOWN TBMT)
: TSOM
: 8.
: JSR R5,TXCTRL ;CLEAR TSOM
: 000
: 0
: JSR R5,TXCHAR ;LOAD 043, TX 3RD SYNCH
: 043

```

```

025256
4473
4474
4475
4476 025256
025256 104402
4477 025260 004737 005344
4478 025264 004537 007324
4479 025270 042226
4480 025272 000347
4481 025274 103C03
4482 025276
025276 104460
4483 025300
025300 104410
025302 000256
4484
4485 025304 004537 007734
4486 025310 000001
4487 025312 000007
4488 025314 004537 007734
4489 025320 000001
4490 025322 000010
4491 025324 004537 007734
4492 025330 000000
4493 025332 000000
4494 025334 004537 007622
4495 025340 000043

```

TEST 2 -- VRC ERROR DETECTION TEST

4496	025342	000010		8.					
4497	025344	103003		BCC	+.8.		;BR IF NO ERROR		
4498	025346			ERROR			;REPORT STACKED ERROR		
	025346	104460						TRAP	C\$ERROR
4499	025350			ESCAPE	SUB		;SKIP TO END OF TEST		
	025350	104410						TRAP	C\$ESCAPE
	025352	000206						.WORD	L10024-.
4500									
4501	025354	004537	003660	JSR	R5,WRITEI		;SET RX CHAR LENGTH=6 BITS		
4502	025360	120407		PCR					
4503	025362	000346		TXDL!6			;TXCL=7, RXCL=6		
4504	025364	103003		BCC	+.8.		;BR IF NO ERROR		
4505	025366			ERROR			;PRINT STACKED ERROR MESSAGE		
	025366	104460						TRAP	C\$ERROR
4506	025370			ESCAPE	SUB		;AND EXIT SUBTEST		
	025370	104410						TRAP	C\$ESCAPE
	025372	00C166						.WORD	L10024-.
4507									
4508	025374	004537	007622	JSR	R5,TXCHAR		;LOAD 036		
4509	025400	000036		036					
4510	025402	000010		8.					
4511	025404	103003		BCC	+.8.		;BR IF NO ERROR		
4512	025406			ERROR			;REPORT STACKED ERROR		
	025406	104460						TRAP	C\$ERROR
4513	025410			ESCAPE	SUB		;SKIP TO END OF TEST		
	025410	104410						TRAP	C\$ESCAPE
	025412	000146						.WORD	L10024-.
4514									
4515	025414	004537	007622	JSR	R5,TXCHAR		;LOAD FILLER (000)		
4516	025420	000000		000					
4517	025422	000010		8.					
4518	025424	103003		BCC	+.8.		;BR IF NO ERROR		
4519	025426			ERROR			;REPORT STACKED ERROR		
	025426	104460						TRAP	C\$ERROR
4520	025430			ESCAPE	SUB		;SKIP TO END OF TEST		
	025430	104410						TRAP	C\$ESCAPE
	025432	000126						.WORD	L10024-.
4521									
4522	025434	004537	007622	JSR	R5,TXCHAR		;LOAD FILLER (000)		
4523	025440	000000		000					
4524	025442	000010		8.					
4525	025444	103003		BCC	+.8.		;BR IF NO ERROR		
4526	025446			ERROR			;REPORT STACKED ERROR		
	025446	104460						TRAP	C\$ERROR
4527	025450			ESCAPE	SUB		;SKIP TO END OF TEST		
	025450	104410						TRAP	C\$ESCAPE
	025452	000106						.WORD	L10024-.
4528									
4529	025454	004537	010034	JSR	R5,RXCHAR		;READ/CHK SYNCH CHARACTER		
4530	025460	000026		026					
4531	025462	000001		RERCHK			;CHECK RERR (NO VRC ERROR EXPECTED)		
4532	025464	100000		NOCRDA			;NO INITIAL CHECK OF RDA=0		
4533	025466	103003		BCC	+.8.		;BR IF NO ERROR		
4534	025470			ERROR			;REPORT STACKED ERROR		
	025470	104460						TRAP	C\$ERROR
4535	025472			ESCAPE	SUB		;SKIP TO END OF TEST		
	025472	104410						TRAP	C\$ESCAPE

TEST 2 -- VRC ERROR DETECTION TEST

```

025474 000064 .WORD L10024-.
4536
4537 025476 004537 010034 JSR R5,RXCHAR ;READ/CHK 6 BIT CHARACTER
4538 025502 000043 043 ;EXPECTED 1ST "CHARACTER" (043)
4539 025504 000001 RERCHK ;CHECK RERR (NO VRC ERROR EXPECTED)
4540 025506 100000 NOCRDA ;DON'T CHECK INITIAL RDA=0
4541 025510 103003 BCC .+8. ;BR IF NO ERROR
4542 025512 ERROR ;REPORT STACKED ERROR
025512 104460 TRAP C$ERROR
4543 025514 ESCAPE SUB ;SKIP TO END OF TEST
025514 104410 TRAP C$ESCAPE
025516 000042 .WORD L10024-.
4544
4545 025520 004537 010034 JSR R5,RXCHAR ;READ/CHK 6 BIT CHARACTER
4546 025524 100074 RXERR!074 ;EXPECTED 2ND "CHARACTER" (074)
4547 025526 000001 RERCHK ;CHECK RERR (VRC ERROR IS EXPECTED)
4548 025530 100000 NOCRDA ;DON'T CHECK INITIAL RDA=0
4549 025532 103003 BCC .+8. ;BR IF NO ERROR
4550 025534 ERROR ;REPORT STACKED ERROR
025534 104460 TRAP C$ERROR
4551 025536 ESCAPE SUB ;SKIP TO END OF TEST
025536 104410 TRAP C$ESCAPE
025540 000020 .WORD L10024-.
4552
4553 025542 004537 011456 JSR R5,ENDTRN ;SHUT DOWN TRANSMITTER, RECEIVER
4554 025546 000011 9.
4555 025550 103003 BCC .+8. ;BR IF NO ERROR
4556 025552 ERROR ;REPORT STACKED ERROR
025552 104460 TRAP C$ERROR
4557 025554 ESCAPE SUB ;SKIP TO NEXT SUBTEST
025554 104410 TRAP C$ESCAPE
025556 000002 .WORD L10024-.
4558 025560 ENDSUB
025560 L10024: TRAP C$ESUB
025560 104403
4559
4560 ;-----
4561 ; SUBTEST #2: FORCING EVEN VRC ERROR
4562 ;-----
4562 025562 BGNSUB
025562 T2.2: TRAP C$BSUB
025562 104402
4563 025564 004737 005344 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
4564 025570 004537 007324 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
4565 025574 042626 DDCMP!EVRT!226 ;SET DDCMP,EVEN VRC CHECK,SYNCH=226
4566 025576 000347 TXDL!RXDL ;TX/RX CHAR LENGTH=7 BITS
4567 025600 103003 BCC .+8. ;BR IF NO ERROR
4568 025602 ERROR ;REPORT STACKED ERROR
025602 104460 TRAP C$ERROR
4569 025604 ESCAPE SUB ;SKIP TO END OF TEST
025604 104410 TRAP C$ESCAPE
025606 000256 .WORD L10025-.
4570
4571 025610 004537 007734 JSR R5,TXCTRL ;SET TSOM
4572 025614 000001 TSOM
4573 025616 000007 7.
4574 025620 004537 007734 JSR R5,TXCTRL ;SET TSOM AGAIN (KNOCK DOWN TBMT)
4575 025624 000001 TSOM

```

TEST 2 -- VRC ERROR DETECTION TEST

4576	025626	000010		8.					
4577	025630	004537	007734	JSR	R5,TXCTRL	;CLEAR TSOM			
4578	025634	000000		000					
4579	025636	000000		0					
4580	025640	004537	007622	JSR	R5,TXCHAR	;LOAD 143, TX 3RD SYNCH			
4581	025644	000143		143					
4582	025646	000010		8.					
4583	025650	103003		BCC	.+8.	;BR IF NO ERROR			
4584	025652			ERROR		;REPORT STACKED ERROR			
		104460					TRAP	C\$ERROR	
4585	025654			ESCAPE	SUB	;SKIP TO END OF TEST			
		104410					TRAP	C\$ESCAPE	
	025656	000206					.WORD	L10025-.	
4586									
4587	025660	004537	003660	JSR	R5,WRITEI	;SET RX CHAR LENGTH=6 BITS			
4588	025664	120407		PCR					
4589	025666	000346		TXDL!6		;TXCL=7, RXCL=6			
4590	025670	103003		BCC	.+8.	;BR IF NO ERROR			
4591	025672			ERROR		;PRINT STACKED ERROR MESSAGE			
		104460					TRAP	C\$ERROR	
4592	025674			ESCAPE	SUB	;AND EXIT SUBTEST			
		104410					TRAP	C\$ESCAPE	
	025676	000166					.WORD	L10025-.	
4593									
4594	025700	004537	007622	JSR	R5,TXCHAR	;LOAD 026			
4595	025704	000026		026					
4596	025706	000010		8.					
4597	025710	103003		BCC	.+8.	;BR IF NO ERROR			
4598	025712			ERROR		;REPORT STACKED ERROR			
		104460					TRAP	C\$ERROR	
4599	025714			ESCAPE	SUB	;SKIP TO END OF TEST			
		104410					TRAP	C\$ESCAPE	
	025716	000146					.WORD	L10025-.	
4600									
4601	025720	004537	007622	JSR	R5,TXCHAR	;LOAD FILLER (000)			
4602	025724	000000		000					
4603	025726	000010		8.					
4604	025730	103003		BCC	.+8.	;BR IF NO ERROR			
4605	025732			ERROR		;REPORT STACKED ERROR			
		104460					TRAP	C\$ERROR	
4606	025734			ESCAPE	SUB	;SKIP TO END OF TEST			
		104410					TRAP	C\$ESCAPE	
	025736	000126					.WORD	L10025-.	
4607									
4608	025740	004537	007622	JSR	R5,TXCHAR	;LOAD FILLER (000)			
4609	025744	000000		000					
4610	025746	000010		8.					
4611	025750	103003		BCC	.+8.	;BR IF NO ERROR			
4612	025752			ERROR		;REPORT STACKED ERROR			
		104460					TRAP	C\$ERROR	
4613	025754			ESCAPE	SUB	;SKIP TO END OF TEST			
		104410					TRAP	C\$ESCAPE	
	025756	000106					.WORD	L10025-.	
4614									
4615	025760	004537	010034	JSR	R5,RXCHAR	;READ/CHK SYNCH CHARACTER			
4616	025764	000026		026					
4617	025766	000001		RERCHK		;CHECK RERR (NO VRC ERROR EXPECTED)			

TEST 2 -- VRC ERROR DETECTION TEST

```

4618 025770 100000
4619 025772 103003
4620 025774 104460
4621 025776 104410
025774 104460
025776 104410
026000 000064
4622
4623 026002 004537 010034
4624 026006 000043
4625 026010 000001
4626 026012 100000
4627 026014 103003
4628 026016 104460
026016 104460
4629 026020 104410
026020 104410
026022 000042
4630
4631 026024 004537 010034
4632 026030 100054
4633 026032 000001
4634 026034 100000
4635 026036 103003
4636 026040 104460
026040 104460
4637 026042 104410
026042 104410
026044 000020
4638
4639 026046 004537 011456
4640 026052 000011
4641 026054 103003
4642 026056 104460
026056 104460
4643 026060 104410
026060 104410
026062 000002
4644 026064 104403
026064
026064 104403
4645 026066 104401
026066
026066 104401

```

ENDTST

```

NOCRDA ;NO INITIAL CHECK OF RDA=0
BCC .+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR
ESCAPE SUB ;SKIP TO END OF TEST
TRAP C#ERROR
WORD C#ESCAPE
L10025-.

JSR R5,RXCHAR ;READ/CHK 6 BIT CHARACTER
043 ;EXPECTED 1ST "CHARACTER" (043)
RERCHK ;CHECK RERR (NO VRC ERROR EXPECTED)
NOCRDA ;DON'T CHECK INITIAL RDA=0
BCC .+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR
TRAP C#ERROR
ESCAPE SUB ;SKIP TO END OF TEST
TRAP C#ESCAPE
WORD L10025-.

JSR R5,RXCHAR ;READ/CHK 6 BIT CHARACTER
RXERR!054 ;EXPECTED 2ND "CHARACTER" (054)
RERCHK ;CHECK RERR (VRC ERROR IS EXPECTED)
NOCRDA ;DON'T CHECK INITIAL RDA=0
BCC .+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR
TRAP C#ERROR
ESCAPE SUB ;SKIP TO END OF TEST
TRAP C#ESCAPE
WORD L10025-.

JSR R5,ENDTRN ;SHUT DOWN TRANSMITTER, RECEIVER
9.
BCC .+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR
TRAP C#ERROR
ESCAPE SUB ;SKIP TO NEXT SUBTEST
TRAP C#ESCAPE
WORD L10025-.

ENDSUB
L10025: TRAP C#ESUB
L10023: TRAP C#ETST

```

TEST 3 -- BCP CRC GENERATION/DETECTION TEST

4656

.SBTTL TEST 3 -- BCP CRC GENERATION/DETECTION TEST

```

;*****
;*
;* TEST 3 -- BCP CRC GENERATION/DETECTION TEST
;*
;* THIS TEST IS COMPOSED OF 2 SUBTESTS -- #1 EXPECTS GOOD CRC
;* GENERATION AND REPORT ERRORS -- #2 FORCES AN ERROR AND ONLY
;* REPORT WHEN THE CRC IS ACCEPTED AS GOOD. EACH IS
;* RUN AT THE CHARACTER LENGTHS OF 8 BITS FOR THE ENTIRITY
;* OF EACH MESSAGE. BOTH THE TRANSMITTER AND RECEIVER WILL BE SET TO
;* THE SAME CHARACTER LENGTH. ERROR LOOPING WILL BE ON THE FAILING
;* SUBTEST. TEXT STRINGS WILL BE LIMITED TO 5 CHARACTERS.
;*
;*****

```

; BGNTST

T3::

; SUBTEST #1 : GOOD CRC-16 GENERATION

; BGNSUB

T3.1:

TRAP C\$BSUB

```

026070
4657 026070
4658 026070
4659 026070
4660 026070
026070 104402
4661 026072 004737 005344
4662 026076 004537 007324
4663 026102 065626
4664 026104 000000
4665 026106 103003
4666 026110
026110 104460
4667 026112
026112 104410
026114 000544
4668
4669 026116 004537 007734
4670 026122 000001
4671 026124 000007
4672 026126 004537 007734
4673 026132 000000
4674 026134 000000
4675
4676
4677
4678 026136 012703 026662
4679 026142 112337 026152
4680
4681 026146 004537 007622
4682 026152 000000
4683 026154 000010
4684 026156 103003
4685 026160
026160 104460
4686 026162
026162 104410
026164 000474
4687

```

```

JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
DDCMP!STRIPS!IDLES!CRC16!SYNCH ;SET DDCMP, STRIP, IDLE, CRC-16, SYNCH=226
0 ;USE 8 BIT CHARS
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-.

JSR R5, TXCTRL ;SET TSOM, TX 1ST SYNCH
TSOM
7.
JSR R5, TXCTRL ;CLEAR TSOM
000
0

```

; NOW TRANSMIT THE FIVE 8-BIT DATA CHARACTERS TO THE RECEIVER/FIFO

```

10$: MOV #T01TBL,R3 ;SET UP DATA TABLE POINTER
MOV (R3)+,1# ;INSTALL NEXT TX CHARACTER

1$: JSR R5, TXCHAR ;TRANSMIT CHARACTER ( ==> RX/FIFO )
000 ;** HOLE FOR NEXT CHARACTER **
8.
BCC .+8. ;BR IF NO ERROR
ERROR ;REPORT STACKED ERROR
ESCAPE TST ;SKIP TO END OF TEST
TRAP C$ERROR
TRAP C$ESCAPE
.WORD L10026-.

```


TEST 3 -- BCP CRC GENERATION/DETECTION TEST

4688	026166	022703	026667	CMP	#T01TBL+5,R3	;ALL CHARACTERS TRANSMITTED ?		
4689	026172	001363		BNE	10#	; IF NOT, TX ANOTHER ONE		
4690				-----				
4691	026174	004537	007734	JSR	R5, TXCTRL	;LOAD 1ST TEOM		
4692	026200	000002		TEOM				
4693	026202	000010		8.				
4694	026204	004537	007734	JSR	R5, TXCTRL	;LOAD 2ND TEOM		
4695	026210	000002		TEOM				
4696	026212	000010		8.				
4697	026214	004537	011540	JSR	R5, STEPLU			
4698	026220	000016		14.				
4699								
4700	026222	004537	010034	JSR	R5, RXCHAR	;READ & CHK 000, RCV 125		
4701	026226	000000		000				
4702	026230	000000		0				
4703	026232	100000		NOCRDA		;NO INITIAL CHECK OF RDA=0		
4704	026234	103003		BCC	.+8.	;BR IF NO ERROR		
4705	026236			ERROR		;REPORT STACKED ERROR		
	026236	104460					TRAP	C#ERROR
4706	026240			ESCAPE	TST	;SKIP TO END OF TEST		
	026240	104410					TRAP	C#ESCAPE
	026242	000416					.WORD	L10026-.
4707								
4708	026244	004537	010034	JSR	R5, RXCHAR	;READ & CHK 125, RCV 252		
4709	026250	000125		125				
4710	026252	000000		0				
4711	026254	100000		NOCRDA		;NO INITIAL CHECK OF RDA=0		
4712	026256	103003		BCC	.+8.	;BR IF NO ERROR		
4713	026260			ERROR		;REPORT STACKED ERROR		
	026260	104460					TRAP	C#ERROR
4714	026262			ESCAPE	TST	;SKIP TO END OF TEST		
	026262	104410					TRAP	C#ESCAPE
	026264	000374					.WORD	L10026-.
4715								
4716	026266	004537	010034	JSR	R5, RXCHAR	;READ & CHK 252, RCV 377		
4717	026272	000252		252				
4718	026274	000000		0				
4719	026276	100000		NOCRDA		;NO INITIAL CHECK OF RDA=0		
4720	026300	103003		BCC	.+8.	;BR IF NO ERROR		
4721	026302			ERROR		;REPORT STACKED ERROR		
	026302	104460					TRAP	C#ERROR
4722	026304			ESCAPE	TST	;SKIP TO END OF TEST		
	026304	104410					TRAP	C#ESCAPE
	026306	000352					.WORD	L10026-.
4723								
4724	026310	004537	010034	JSR	R5, RXCHAR	;READ & CHK 377, RCV 000		
4725	026314	000377		377				
4726	026316	000000		0				
4727	026320	100010		NOCRDA!8.		;NO INITIAL CHECK OF RDA=0		
4728	026322	103003		BCC	.+8.	;BR IF NO ERROR		
4729	026324			ERROR		;REPORT STACKED ERROR		
	026324	104460					TRAP	C#ERROR
4730	026326			ESCAPE	TST	;SKIP TO END OF TEST		
	026326	104410					TRAP	C#ESCAPE
	026330	000330					.WORD	L10026-.
4731								
4732	026332	004537	010034	JSR	R5, RXCHAR	;READ & CHK 000, CHECK CRC :		

TEST 3 -- BCP CRC GENERATION/DETECTION TEST

```

4733 026336 100000          RXERR!000          ; RERR=1 (IF CRC-16 WAS OK).
4734 026340 000001          RERCHK
4735 026342 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
4736 026344 103003          BCC      .+8.    ;BR IF NO ERROR
4737 026346          ERROR          ;REPORT STACKED ERROR
          026346 104460          . TRAP      C$ERROR
4738 026350          ESCAPE TST      ;SKIP TO END OF TEST
          026350 104410          TRAP      C$ESCAPE
          026352 000306          .WORD    L10026-.
4739
4740 026354 004537 011456    JSR      R5,ENDTRN ;SHUT DOWN TRANSMITTER, RECEIVER
4741 026360 000011          9.
4742 026362 103003          BCC      .+8.    ;BR IF NO ERROR
4743 026364          ERROR          ;REPORT STACKED ERROR
          026364 104460          TRAP      C$ERROR
4744 026366          ESCAPE TST      ;SKIP TO END OF TEST
          026366 104410          TRAP      C$ESCAPE
          026370 000270          .WORD    L10026-.
4745 026372          ENDSUB
          026372          L10027:
          026372 104403          TRAP      C$ESUB
4746
4747          ;-----
4748          ; SUBTEST #2 : BAD CRC-16 GENERATION
4749          ;-----
          026374          BGNSUB
          026374          T3.2:
          026374 104402          TRAP      C$BSUB
4750 026376 004737 005344    JSR      PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
4751 026402 004537 007324    JSR      R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
4752 026406 065626          DDCMP!STRIPS!IDLES!CRC16 ;SYNCH ;SET DDCMP, STRIP, IDLE, CRC-16, SYNCH=226
4753 026410 000000          0          ;USE 8 BIT CHARS
4754 026412 103003          BCC      .+8.    ;BR IF NO ERROR
4755 026414          ERROR          ;REPORT STACKED ERROR
          026414 104460          TRAP      C$ERROR
4756 026416          ESCAPE TST      ;SKIP TO END OF TEST
          026416 104410          TRAP      C$ESCAPE
          026420 000240          .WORD    L10026-.
4757
4758 026422 004537 007734    JSR      R5, TXCTRL ;SET TSOM, TX 1ST SYNCH
4759 026426 000001          TSOM
4760 026430 000007          7.
4761 026432 004537 007734    JSR      R5, TXCTRL ;CLEAR TSOM
4762 026436 000000          000
4763 026440 000000          0
4764
4765          ;-----
4766          ; NOW TRANSMIT THE FIVE 8-BIT DATA CHARACTERS PLUS THE ADDITIONAL
4767          ; TWO BAD CRC (ALL 1'S) CHARACTERS TO THE RECEIVER/FIFO
4768          ;-----
4768 026442 012703 026662    MOV      @T01TBL,R3 ;SET UP DATA TABLE POINTER
4769 026446 112337 026456    10$:    MOVB   (R3)+,1$    ;INSTALL NEXT TX CHARACTER
4770
4771 026452 004537 007622    JSR      R5, TXCHAR ;TRANSMIT CHARACTER ( ==> RX/FIFO )
4772 026456 000000          000          ;** HOLE FOR NEXT CHARACTER **
4773 026460 000010          8.
4774 026462 103003          BCC      .+8.    ;BR IF NO ERROR
4775 026464          ERROR          ;REPORT STACKED ERROR
          026464 104460          TRAP      C$ERROR

```

TEST 3 -- BCP CRC GENERATION/DETECTION TEST

```

4776 026466          ESCAPE TST          ;SKIP TO END OF TEST
      026466 104410
      026470 000170          TRAP      C$ESCAPE
                                .WORD      L10026-.
4777
4778 026472 022703 026671          CMP      #T01TBL+7,R3      ;ALL CHARACTERS TRANSMITTED ?
4779 026476 001363          BNE      10$              ; IF NOT, TX ANOTHER ONE
4780          ;-----
4781 026500 004537 011540          JSR      R5,STEPLU
4782 026504 000010          10
4783
4784 026506 004537 010034          JSR      R5,RXCHAR          ;READ & CHK 000, RCV 125
4785 026512 000000          000
4786 026514 000000          0
4787 026516 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
4788 026520 103003          BCC      .+8.          ;BR IF NO ERROR
4789 026522          ERROR          ;REPORT STACKED ERROR
      026522 104460          TRAP      C$ERROR
4790 026524          ESCAPE TST          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      026524 104410          .WORD      L10026-.
      026526 000132
4791
4792 026530 004537 010034          JSR      R5,RXCHAR          ;READ & CHK 125, RCV 252
4793 026534 000125          125
4794 026536 000000          0
4795 026540 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
4796 026542 103003          BCC      .+8.          ;BR IF NO ERROR
4797 026544          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      026544 104460          TRAP      C$ERROR
4798 026546          ESCAPE TST          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      026546 104410          .WORD      L10026-.
      026550 000110
4799
4800 026552 004537 010034          JSR      R5,RXCHAR          ;READ & CHK 252, RCV 377
4801 026556 000252          252
4802 026560 000000          0
4803 026562 100010          NOCRDA!8.          ;NO INITIAL CHECK OF RDA=0
4804 026564 103003          BCC      .+8.          ;BR IF NO ERROR
4805 026566          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      026566 104460          TRAP      C$ERROR
4806 026570          ESCAPE TST          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      026570 104410          .WORD      L10026-.
      026572 000066
4807
4808 026574 004537 010034          JSR      R5,RXCHAR          ;READ & CHK 377, RCV 000
4809 026600 000377          377
4810 026602 000000          0
4811 026604 100010          NOCRDA!8.          ;NO INITIAL CHECK OF RDA=0
4812 026606 103003          BCC      .+8.          ;BR IF NO ERROR
4813 026610          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      026610 104460          TRAP      C$ERROR
4814 026612          ESCAPE TST          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      026612 104410          .WORD      L10026-.
      026614 000044
4815
4816 026616 004537 010034          JSR      R5,RXCHAR          ;READ & CHK 000, CHECK CRC :
4817 026622 000000          000          ; RERR=0 IF BAD CRC-16 (EXPECTED).
4818 026624 000001          RERCHK

```

TEST 3 -- BCP CRC GENERATION/DETECTION TEST

```

4819 026626 100000          NOCRDA          ;NO INITIAL CHECK OF RDA=0
4820 026630 103003          BCC          .+8.      ;BR IF NO ERROR
4821 026632          ERROR          ;REPORT STACKED ERROR
4822 026634 104460          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
                                026634 104410          ;                               TRAP C$ESCAPE
                                026636 000022          .WORD          L10026-.
4823
4824 026640 004537 011456    JSR          R5,ENDTRN ;SHUT DOWN TRANSMITTER, RECEIVER
4825 026644 000011          9.
4826 026646 103003          BCC          .+8.      ;BR IF NO ERROR
4827 026650          ERROR          ;REPORT STACKED ERROR
4828 026652 104460          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
                                026652 104410          ;                               TRAP C$ESCAPE
                                026654 000004          .WORD          L10026-.
4829 026656          ENDSUB
                                026656          ;                               L10030:
                                026656 104403          ;                               TRAP C$ESUB
4830 026660          ENDTST
                                026660          ;                               L10026:
                                026660 104401          ;                               TRAP C$ETST
4831
4832 026662 000          ;-----
4833 026663 125          T01TBL: .BYTE 000      ;D1
4834 026664 252          .BYTE 125      ;D2
4835 026665 377          .BYTE 252      ;D3
4836 026666 000          .BYTE 377      ;D4
4837 026667 377          .BYTE 000      ;D5
4838 026670 377          .BYTE 377      ;BAD CRC1
                                .BYTE 377      ;BAD CRC2
                                .EVEN
;-----

```

TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST

4859

.SBTTL TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST

```

*****
;*
;* TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST
;*
;* THE USYRT IS INITIALIZED FOR BOP MODE WITH TTL LOOPBACK SELECTED.
;* "SECONDARY STATION ADDRESS" IS NOT USED AND NO CRC/VRC IS CALCULATED.
;* A PATTERN IS TRANSMITTED AND TERMINATED FOLLOWED BY A SECOND MESSAGE.
;* TERMINATION OF THE FIRST MESSAGE IS ACCOMPLISHED WITH A FLAG
;* CHARACTER BUT RXE IS NOT DROPPED SO THAT THE SECOND MESSAGE CAN BE
;* SENT WITHOUT RE-SYNCRONIZATION. SEVERAL FLAG'S ARE IDLED BETWEEN THE
;* TWO MESSAGES. DURING THE SECOND MESSAGE A RECEIVER OVERRUN CONDITION
;* IS FORCED. THROUGHOUT THIS TEST, BASIC RECEIVER OPERATION AND TIMING
;* IS CHECKED. TRANSMITTED INFORMATION IS VERIFIED BY CHECKING THE DATA
;* MADE AVAILABLE AT RXDB.
;*
;* TRANSMITTED PATTERN: FLAG FLAG 123 321 000 377 101 FLAG... FLAG
;*                      321 123 377 000 276.
;*
;* RECEIVED PATTERN: 123 321 000 377 101 ..... 321 123.
*****

```

```

:
: BGNTST
:
: JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T4::
4860 026672 004737 005344
4861
4862 026676 004537 007324 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
4863 026702 003626 NOCHK!SYNCH ;SET BOP MODE,SYNCH REG=226
4864 026704 000000 0 ;USE 8 BIT CHARS
4865 026706 103003 BCC .+8. ;BR IF NO ERROR
4866 026710 104460 ERROR ;REPORT STACKED ERROR
: TRAP C$ERROR
4867 026712 ESCAPE TST ;SKIP TO END OF TEST
: TRAP C$ESCAPE
: .WORD L10031-.
4868
4869 026716 004537 007734 JSR R5,TXCTRL ;LOAD 2ND FLAG,TX 1ST FLAG
4870 026722 000001 TSOM
4871 026724 000007 7.
4872 026726 004537 007734 JSR R5,TXCTRL ;CLEAR TSOM
4873 026732 000000 000
4874 026734 000000 0
4875
4876 026736 004537 007622 JSR R5,TXCHAR ;LOAD 123(DATA1), TX 2ND FLAG
4877 026742 000123 123
4878 026744 000010 8.
4879 026746 103003 BCC .+8. ;BR IF NO ERROR
4880 026750 104460 ERROR ;REPORT STACKED ERROR
: TRAP C$ERROR
4881 026752 ESCAPE TST ;SKIP TO END OF TEST
: TRAP C$ESCAPE
: .WORD L10031-.
4882
4883 026756 004537 007622 JSR R5,TXCHAR ;LOAD 321(DATA2), TX 123(DATA1)
4884 026762 000321 321

```

TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST

4885	026764	000010		8.					
4886	026766	103003		BCC	..8.				
4887	026770			ERROR					
	026770	104460							
4888	026772			ESCAPE	TST				
	026772	104410							
	026774	000522							
4889									
4890	026776	004537	007622	JSR	R5, TXCHAR				
4891	027002	000000		000					
4892	027004	000010		8.					
4893	027006	103003		BCC	..8.				
4894	027010			ERROR					
	027010	104460							
4895	027012			ESCAPE	TST				
	027012	104410							
	027014	000502							
4896									
4897	027016	004537	007622	JSR	R5, TXCHAR				
4898	027022	000377		377					
4899	027024	000000		0					
4900	027026	103003		BCC	..8.				
4901	027030			ERROR					
	027030	104460							
4902	027032			ESCAPE	TST				
	027032	104410							
	027034	000462							
4903									
4904	027036	004537	011310	JSR	R5, RCV1ST				
4905	027042	000000		0					
4906	027044	103003		BCC	..8.				
4907	027046			ERROR					
	027046	104460							
4908	027050			ESCAPE	TST				
	027050	104410							
	027052	000444							
4909									
4910	027054	004537	010034	JSR	R5, RXCHAR				
4911	027060	000523		RXSOM:123					
4912	027062	000000		0					
4913	027064	000010		8.					
4914	027066	103003		BCC	..8.				
4915	027070			ERROR					
	027070	104460							
4916	027072			ESCAPE	TST				
	027072	104410							
	027074	000422							
4917									
4918	027076	004537	007622	JSR	R5, TXCHAR				
4919	027102	000101		101					
4920	027104	000000		0					
4921	027106	103003		BCC	..8.				
4922	027110			ERROR					
	027110	104460							
4923	027112			ESCAPE	TST				
	027112	104410							
	027114	000402							

TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST

```

4924
4925 027116 004537 010034 JSR R5,RXCHAR ;READ/CHECK 321(DATA2),RCV 000(DATA3)
4926 027122 000321 321
4927 027124 000000 0
4928 027126 000010 8.
4929 027130 103003 BCC .+8. ;BR IF NO ERROR
4930 027132 ERROR ;REPORT STACKED ERROR
027132 104460
4931 027134 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
027134 104410 TRAP C$ESCAPE
027136 000360 .WORD L10031-.
4932
4933 027140 004537 007734 JSR R5, TXCTRL ;LOAD TEOM
4934 027144 000002 TEOM
4935 027146 000000 0
4936
4937 027150 004537 010034 JSR R5,RXCHAR ;READ/CHECK 000(DATA3),RCV 377(DATA4)
4938 027154 000000 000
4939 027156 000000 0
4940 027160 000010 8.
4941 027162 103003 BCC .+8. ;BR IF NO ERROR
4942 027164 ERROR ;REPORT STACKED ERROR
027164 104460
4943 027166 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
027166 104410 TRAP C$ESCAPE
027170 000326 .WORD L10031-.
4944
4945 027172 004537 007734 JSR R5, TXCTRL ;LOAD TEOM
4946 027176 000002 TEOM
4947 027200 000000 0
4948
4949 027202 004537 010034 JSR R5,RXCHAR ;READ/CHECK 377(DATA4),RCV 101(DATA5)
4950 027206 000377 377 ; AND TX (FLAG1)
4951 027210 000000 0
4952 027212 020010 NCRACT!8. ;DON'T CHECK RECEIVER ACTIVE
4953 027214 103003 BCC .+8. ;BR IF NO ERROR
4954 027216 ERROR ;REPORT STACKED ERROR
027216 104460
4955 027220 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
027220 104410 TRAP C$ESCAPE
027222 000274 .WORD L10031-.
4956
4957 027224 004537 007734 JSR R5, TXCTRL ;LOAD TEOM
4958 027230 000002 TEOM
4959 027232 000000 0
4960
4961 027234 004537 010034 JSR R5,RXCHAR ;READ/CHECK 101(DATA5),RCV (FLAG1)
4962 027240 001101 RXEOM!101 ; TX (FLAG2) & CHECK REOM
4963 027242 000000 0
4964 027244 060010 NFCRDA!NCRACT!8. ;DON'T CHECK FOR FINAL RDA=RXACT=1
4965 027246 103003 BCC .+8. ;BR IF NO ERROR
4966 027250 ERROR ;REPORT STACKED ERROR
027250 104460
4967 027252 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
027252 104410 TRAP C$ESCAPE
027254 000242 .WORD L10031-.
4968

```

TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST

4969	027256	004537	007734	JSR	R5, TXCTRL	;CLEAR TEOM		
4970	027262	000000		000				
4971	027264	000000		0				
4972								
4973	027266	004537	007622	JSR	R5, TXCHAR	;LOAD 321(DATA6), TX (FLAG3)		
4974	027272	000321		321				
4975	027274	100010		NCTBMT*256.!8.		;DON'T CHECK TBMT		
4976	027276	103003		BCC	+.8.	;BR IF NO ERROR		
4977	027300			ERROR		;REPORT STACKED ERROR		
	027300	104460					TRAP	C\$ERROR
4978	027302			ESCAPE	TST	;SKIP TO END OF TEST		
	027302	104410					TRAP	C\$ESCAPE
	027304	000212					.WORD	L10031-.
4979								
4980	027306	004537	007622	JSR	R5, TXCHAR	;LOAD 123(DATA7), TX(DATA6)		
4981	027312	000123		123				
4982	027314	100010		NCTBMT*256.!8.		;DON'T CHECK TBMT		
4983	027316	103003		BCC	+.8.	;BR IF NO ERROR		
4984	027320			ERROR		;REPORT STACKED ERROR		
	027320	104460					TRAP	C\$ERROR
4985	027322			ESCAPE	TST	;SKIP TO END OF TEST		
	027322	104410					TRAP	C\$ESCAPE
	027324	000172					.WORD	L10031-.
4986								
4987	027326	004537	007622	JSR	R5, TXCHAR	;LOAD 377(DATA8), TX(DATA7)		
4988	027332	000377		377				
4989	027334	100010		NCTBMT*256.!8.		;DON'T CHECK FINAL TBMT		
4990	027336	103003		BCC	+.8.	;BR IF NO ERROR		
4991	027340			ERROR		;REPORT STACKED ERROR		
	027340	104460					TRAP	C\$ERROR
4992	027342			ESCAPE	TST	;SKIP TO END OF TEST		
	027342	104410					TRAP	C\$ESCAPE
	027344	000152					.WORD	L10031-.
4993								
4994	027346	004537	007622	JSR	R5, TXCHAR	;LOAD 000(DATA9)		
4995	027352	000000		000				
4996	027354	000000		0				
4997	027356	103003		BCC	+.8.	;BR IF NO ERROR		
4998	027360			ERROR		;REPORT STACKED ERROR		
	027360	104460					TRAP	C\$ERROR
4999	027362			ESCAPE	TST	;SKIP TO END OF TEST		
	027362	104410					TRAP	C\$ESCAPE
	027364	000132					.WORD	L10031-.
5000								
5001	027366	004537	010034	JSR	R5, RXCHAR	;READ/CHECK 321(DATA6), RCV 123(DATA7)		
5002	027372	000721		RXSOM!321				
5003	027374	000000		0				
5004	027376	000010		8.				
5005	027400	103003		BCC	+.8.	;BR IF NO ERROR		
5006	027402			ERROR		;REPORT STACKED ERROR		
	027402	104460					TRAP	C\$ERROR
5007	027404			ESCAPE	TST	;SKIP TO END OF TEST		
	027404	104410					TRAP	C\$ESCAPE
	027406	000110					.WORD	L10031-.
5008								
5009	027410	004537	007622	JSR	R5, TXCHAR	;LOAD 276(DATA10)		
5010	027414	000276		276				

TEST 4 -- BOP RX BASIC RECEIVE/FLAG RECOGNITION TEST

```

5011 027416 000000      0
5012 027420 103003      BCC      .+8.      ;BR IF NO ERROR
5013 027422      ERROR      ;REPORT STACKED ERROR
      027422 104460
5014 027424      ESCAPE TST      ;SKIP TO END OF TEST      TRAP      C$ERROR
      027424 104410      ;
      027426 000070      ;
5015
5016 027430 004537 010034      JSR      R5,RXCHAR      ;READ/CHECK 123(DATA7),RCV 377(DATA8)
5017 027434 000123      123
5018 027436 000000      0
5019 027440 100014      NOCRDA!12.      ;NO CHECK OF INITIAL RDA=0
5020 027442 103003      BCC      .+8.      ;BR IF NO ERROR
5021 027444      ERROR      ;REPORT STACKED ERROR
      027444 104460      ;
5022 027446      ESCAPE TST      ;SKIP TO END OF TEST      TRAP      C$ERROR
      027446 104410      ;
      027450 000046      ;
5023
5024 027452 012704 000010      ;-----
5025 027456 004537 007622      5$:  MOV      #8.,R4      ;INIT CHARACTER COUNT
5026 027462 000000      JSR      R5, TXCHAR      ;LOAD/TX FILLER (OVERFLOW FIFO)
5027 027464 100010      000
5028 027466 103003      NCTBMT*256.!8.      ;DON'T CHECK FINAL TBMT
5029 027470      BCC      .+8.      ;BR IF NO ERROR
      027470 104460      ERROR      ;REPORT STACKED ERROR      TRAP      C$ERROR
5030 027472      ESCAPE TST      ;SKIP TO END OF TEST      TRAP      C$ESCAPE
      027472 104410      ;
      027474 000022      ;
5031
5032 027476 077411      SOB      R4,5$      ;FILL TO OVERFLOW
5033
5034
5035 027500 004537 006422      JSR      R5,CKROR      ;CHECK RECEIVER OVERRUN BIT
5036 027504 000001      1      ;(IT SHOULD BE SET)
5037 027506 103003      BCC      .+8.      ;BR IF NO ERROR
5038 027510      ERROR      ;REPORT STACKED ERROR      TRAP      C$ERROR
      027510 104460
5039 027512      ESCAPE TST      ;SKIP TO END OF TEST      TRAP      C$ESCAPE
      027512 104410      ;
      027514 000002      ;
5040 027516      ENDTST
      027516
      027516 104401      L10031:      TRAP      C$ETST

```

TEST 5 -- BOP RX SECONDARY STATION ADDRESSING

```

5055 .SBTTL TEST 5 -- BOP RX SECONDARY STATION ADDRESSING
;*****
;*
;* TEST 5 -- BOP RX SECONDARY STATION ADDRESSING
;*
;* THE USYRT IS INITIALIZED FOR BOP MODE WITH TTL LEVEL LOOPBACK,
;* SAM = 1, APA=0, AND ECM = 7. USING SHORT MESSAGES, THE ADDRESSES
;* 000, 125, 252, 176, AND 177 ARE CHECKED TO SEE THAT THE RECEIVER
;* RECOGNIZES THEM CORRECTLY. IN EACH CASE (AT EACH ADDRESS), A SERIES OF
;* 20 DIFFERENT MESSAGES ARE SENT TO VERIFY THAT THE USYRT WILL ONLY
;* RESPOND TO THE SPECIFIED VALUE.
;*
;* TEST PATTERN: ADR 000 OCR ADR
;* WHERE ADR IS THE ADDRESS BEING TESTED AND OCA IS THE ONE'S
;* COMPLEMENT OF THAT ADDRESS.
;*
;-----
;
; BGNTST
;
5056 027520 004737 005344 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
5057 027524 005004 CLR R4 ;CLEAR TEST ADDR INDEX (0,125,252,176,177)
5058
5059 027526 OLOOP: BGNSUB
;
; T5.1:
; TRAP C$BSUB
5060 027526 104402 CLR R2 ;CLEAR TX ADDRESS INDEX (0 => 20.)
5061 027530 005002 MOV B,ADPAT(R4),NWSAR ;INSTALL NEW S/AR VALUE IN "INNER LOOP"
5062 027532 116437 030224 027546 ;*****
5063 ; INNER LOOP: TEST ONE "TEST ADDRESS" (0,125,252,176, OR 177)
5064 ;*****
5065 027540 BGNSEG
; TRAP C$BSEG
5066 027540 104404 ILOOP: JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5067 027542 004537 007324 NWSAR: SECADR!NOCHK!000 ;SET BOP MODE,SAM=1,###S/AR IS VARIABLE###
5068 027550 013400 0 ;USE 8 BIT CHARS
5069 027552 103003 BCC .+8. ;BR IF NO ERROR
5070 027554 104460 ERROR ;REPORT STACKED ERROR
; TRAP C$ERROR
5071 027556 104410 ESCAPE SEG ;SKIP TO END OF TEST
; TRAP C$ESCAPE
5072 027560 000422 .WORD 10000$-.
;
;-----
; SETUP TX/RX STRINGS (ADR 000 OCA ADR) -- ADR TAKEN FROM PATX1
;-----
5073
5074
5075 027562 116203 002651 MOV B,PATX1(R2),R3 ;ADR => R3
5076 027566 110337 027650 MOV B,R3,1$ ;ADDRESS(ADR) => 1$,3$,4$,6$
5077 027572 110337 027730 MOV B,R3,3$
5078 027576 110337 030010 MOV B,R3,4$
5079 027602 110337 030126 MOV B,R3,6$
5080 027606 110337 027710 MOV B,R3,2$ ;ADDRESS_NOT(OCA) => 2$,5$
5081 027612 105137 027710 COMB 2$
5082 027616 113737 027710 030074 MOV B,2$,5$
5083
5084
;-----

```

TEST 5 -- BOP RX SECONDARY STATION ADDRESSING

```

5085 ; NOW TRANSMIT TEST PATTERN
5086 -----
5087 027624 004537 007734 JSR R5,TXCTRL ;LOAD 2ND FLAG,TX 1ST FLAG
5088 027630 000001 TSOM
5089 027632 000007 7.
5090 027634 004537 007734 JSR R5,TXCTRL ;CLEAR TSOM
5091 027640 000000 000
5092 027642 000000 0
5093 027644 004537 007622 JSR R5,TXCHAR ;LOAD ADDRESS, TX 2ND FLAG
5094 027650 000000 1$: 000 ;** HOLE FOR SECONDARY STATION ADDRESS
5095 027652 000010 8.
5096 027654 103003 BCC .+8. ;BR IF NO ERROR
5097 027656 ERROR ;REPORT STACKED ERROR
5098 027656 104460 TRAP C$ERROR
5098 027660 ESCAPE SEG ;SKIP TO END OF TEST TRAP C$ESCAPE
5098 027660 104410 .WORD 10000$-.
5098 027662 000320
5099 027664 004537 007622 JSR R5,TXCHAR ;LOAD 000, TX ADDRESS
5100 027670 000000 000
5101 027672 100011 NCTBMT*256.!9. ;DON'T CHECK TBMT
5102 027674 103003 BCC .+8. ;BR IF NO ERROR
5103 027676 ERROR ;REPORT STACKED ERROR
5104 027676 104460 TRAP C$ERROR
5104 027700 ESCAPE SEG ;SKIP TO END OF TEST TRAP C$ESCAPE
5104 027700 104410 .WORD 10000$-.
5104 027702 000300
5105 027704 004537 007622 JSR R5,TXCHAR ;LOAD ADDR_NOT, TX 000
5106 027710 000000 2$: 000 ;** HOLE FOR COMPLEMENTED ADDRESS
5107 027712 100010 NCTBMT*256.!8. ;DON'T CHECK TBMT
5108 027714 103003 BCC .+8. ;BR IF NO ERROR
5109 027716 ERROR ;REPORT STACKED ERROR
5110 027716 104460 TRAP C$ERROR
5110 027720 ESCAPE SEG ;SKIP TO END OF TEST TRAP C$ESCAPE
5110 027720 104410 .WORD 10000$-.
5110 027722 000260
5111 027724 004537 007622 JSR R5,TXCHAR ;LOAD ADDRESS AGAIN
5112 027730 000000 3$: 000 ;** HOLE FOR ADDRESS (AGAIN)
5113 027732 000000 0
5114 027734 103003 BCC .+8. ;BR IF NO ERROR
5115 027736 ERROR ;REPORT STACKED ERROR
5116 027736 104460 TRAP C$ERROR
5116 027740 ESCAPE SEG ;SKIP TO END OF TEST TRAP C$ESCAPE
5116 027740 104410 .WORD 10000$-.
5116 027742 000240
5117 027744 004537 011540 JSR R5,STEPLU ;CLOCK/RCV ADDRESS FIELD
5118 027750 000003 3
5119
5120 027752 004537 006122 JSR R5,CKRDA ;DID USYRT RESPOND TO ADDRESS ??
5121 027756 000001 1
5122 027760 103471 BCS 10$ ;BR IF RDA=0
5123 -----
5124 ; USYRT RESPONDED TO MESSAGE (RDA=1): SHOULD IT HAVE?
5125 -----
5126 027762 123703 027546 CMPB NWSAR,R3 ;IS ADDRESS = S/AR ?
5127 027766 001406 BEQ 40$ ;BR IF YES
5128 -----
5129 ; ...NO, REPORT ERROR : "USYRT RESPONDED TO WRONG ADDRESS"

```

TEST 5 -- BOP RX SECONDARY STATION ADDRESSING

```

5130
5131 027770      GEDF      EM102,ERR21      ;
;-----;
; "DEVICE FATAL" ERROR # 43
;
; TRAP      C$ERDF
; .WORD    43
; .WORD    EM102
; .WORD    ERR21
5132 030000      ESCAPE  SEG
;
; TRAP      C$ESCAPE
; .WORD    10000$-.
5133
5134
5135
;-----;
; ...YES, READ AND VERIFY RECEIVED MESSAGE
;-----;
5136 030004 004537 010034 40$: JSR      R5,RXCHAR      ;READ & CHK ADDRESS, RCV 000
5137 030010 000400      4$: RXSOP:000      ; & CHECK RSOM=1
5138 030012 000000      0
5139 030014 10C010      NOCRDA!8.      ;NO INITIAL CHECK OF RDA=0
5140 030016 103003      BCC      .+8.      ;BR IF NO ERROR
5141 030020      ERROR      ;REPORT STACKED ERROR
; TRAP      C$ERROR
5142 030022      ESCAPE  SEG      ;SKIP TO END OF TEST
; TRAP      C$ESCAPE
; .WORD    10000$-.
5143 030026 004537 007734      JSR      R5,TXCTRL      ;SET TEOM
5144 030032 000002      TEOM
5145 030034 000000      0
5146 030036 004537 010034      JSR      R5,RXCHAR      ;READ/CHECK 000
5147 030042 000000      000
5148 030044 000000      0
5149 030046 100010      NOCRDA!8.      ;NO INITIAL CHECK OF RDA=0
5150 030050 103003      BCC      .+8.      ;BR IF NO ERROR
5151 030052      ERROR      ;REPORT STACKED ERROR
; TRAP      C$ERROR
5152 030054      ESCAPE  SEG      ;SKIP TO END OF TEST
; TRAP      C$ESCAPE
; .WORD    10000$-.
5153 030060 004537 007734      JSR      R5,TXCTRL      ;SET TEOM
5154 030064 000002      TEOM
5155 030066 000000      0
5156 030070 004537 010034      JSR      R5,RXCHAR      ;READ/CHECK COMPLEMENTED ADDRESS
5157 030074 000000      5$: 000      ;** HOLE FOR ADDRESS_NOT
5158 030076 000000      0      ;NO INITIAL CHECK OF RDA=0
5159 030100 120010      NOCRDA!NCRCT!8. ;DON'T CHECK FINAL RXACT=1
5160 030102 103003      BCC      .+8.      ;BR IF NO ERROR
5161 030104      ERROR      ;REPORT STACKED ERROR
; TRAP      C$ERROR
5162 030106      ESCAPE  SEG      ;SKIP TO END OF TEST
; TRAP      C$ESCAPE
; .WORD    10000$-.
5163 030112 004537 007734      JSR      R5,TXCTRL      ;SET TSOM
5164 030116 000001      TSOM
5165 030120 000000      0
5166 030122 004537 010034      JSR      R5,RXCHAR      ;READ/CHECK ADDRESS (AGAIN)
5167 030126 001000      6$: RXEOM!000      ;** HOLE FOR FINAL ADDRESS
5168 030130 000000      0
5169 030132 060000      NFCRDA!NCRCT      ;DON'T CHECK FOR FINAL RDA=RXACT=1
5170 030134 103014      BCC      50$      ;BR IF NO ERROR (TO CONTINUE TEST)

```

TEST 5 - BOP RX SECONDARY STATION ADDRESSING

```

5171 030136          ERROR          ;REPORT STACKED ERROR
      030136 104460
5172 030140          ESCAPE SEG      ;SKIP TO END OF TEST          TRAP   C$ERROR
      030140 104410
      030142 000040          .WORD   10000$-.
5173
5174 ;-----
5175 ; USYRT DIDN'T RESPOND TO MESSAGE (RDA=0): SHOULD IT HAVE ?
5176 030144 123703 027546 10$:  CMPB   NWSAR,R3      ;WAS NON-RESPONDING ADDR=S/AR ?
5177 030150 001006          BNE    50$          ;BR IF NO
5178 ;-----
5179 ; ...NO, REPORT ERROR : "USYRT DIDN'T RESPOND TO SECONDARY STATION ADDR"
5180 ;-----
5181 030152          GEDF    EM103,ERR20      ; "DEVICE FATAL" ERROR # 44
      030152 104455          .WORD   44          TRAP   C$ERDF
      030154 00C054          .WORD   EM103        .WORD   44
      030156 016530          .WORD   ERR20        .WORD   EM103
5182 030162          ESCAPE SEG      ;                               TRAP   C$ESCAPE
      030162 104410          .WORD   10000$-.    .WORD   10000$-.
      030164 000016
5183 ;-----
5184 ;...YES, UPDATE ADDRESS AND CONTINUE TESTING
5185 ;-----
5186 030166 005202          50$:  INC    R2          ;INCREMENT TESTING ADDRESS INDEX
5187 030170 022702 000025  CMP    #21.,R2
5188 030174 001402          BEQ    .+6
5189 030176 000137 027542  JMP    ILOOP      ;IF INDEX .LE. 20 THEN CHECK IT
5190 030202          ENDSEG      ;OTHERWISE END INNER LOOP...
      030202          .WORD   10000$:
      030202 104405          TRAP   C$ESEG
5191 ;*****
5192 030204 005204          INC    R4          ;INCREMENT ACTUAL TEST ADDRESS INDEX
5193 030206 020427 000005  CMP    R4,#5      ;ALL 5 TEST ADDRESSES CHECKED?
5194 030212 001402          BEQ    .+6          ; BR IF DONE
5195 030214 000137 027526  JMP    OLOOP      ; NOT DONE: DO NEXT ADDRESS
5196 030220          ENDSUB
      030220          L10033:
5197 030222 104403          ENDTST          TRAP   C$ESUB
      030222          L10032:
      030222 104401          TRAP   C$ETST
5198 ;-----
5199 030224          ADPAT:  .BYTE   000
5200 030225          .BYTE   125
5201 030226          .BYTE   252
5202 030227          .BYTE   176
5203 030230          .BYTE   177
5204          .EVEN
5205 ;-----

```

TEST 6 -- BOP RX ALL PARTIES ADDRESS TEST

5218

.SBTTL TEST 6 -- BOP RX ALL PARTIES ADDRESS TEST

```

*****
:
: *
: * TEST 6 -- BOP RX ALL PARTIES ADDRESS TEST
: *
: * INITIALIZE THE USYRT FOR BOP MODE WITH TTL LEVEL LOOPBACK
: * SAM = 1, S/AR = 123(OCT.), APA = 1, AND ECM = 7.
: * A SERIES OF 256 DIFFERENT SHORT MESSAGES ARE SENT TO VERIFY THAT
: * THE USYRT WILL ONLY RESPOND TO THE SPECIFIED VALUE AND ALSO 377 (FF
: * HEX.).
: *
: * TEST PATTERN: ADR 000 OCA ADR
: * WHERE ADR IS THE ADDRESS BEING TESTED AND OCA IS THE ONE'S
: * COMPLEMENT OF THAT ADDRESS.
: *
:
: *****

```

```

:
: BGNTST
:
: JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T6::
: CLR R3 ;CLEAR ADDRESS
:
: LOOP: BGNSUB
:
: JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE T6.1: TRAP C$BSUB
: APAD!SECADR!NOCHK!123 ;SET BOP MODE,APA=1,SAM=1,ECM=7,S/AR=123
: 0 ;USE 8 BIT CHARS
: BCC .+8. ;BR IF NO ERROR
: ERROR ;REPORT STACKED ERROR
:
: ESCAPE SUB ;SKIP TO END OF TEST TRAP C$ERROR
:
: ;TRAP C$ESCAPE
: ;.WORD L10035-.
:
: -----
: SETUP TX/RX STRINGS (ADR 000 OCA ADR)
: -----
:
: MOVB R3,1$ ;ADDRESS(ADR) => 1$,3$,4$,6$
: MOVB R3,3$
: MOVB R3,4$
: MOVB R3,6$
: MOVB R3,2$ ;ADDRESS_NOT(OCA) => 2$,5$
: COMB 2$
: MOVB 2$,5$
:
: -----
: NOW TRANSMIT TEST PATTERN
: -----
:
: JSR R5,TXCTRL ;LOAD 2ND FLAG, TX 1ST FLAG
: TSOM
: 7.
: JSR R5,TXCTRL ;CLEAR TSOM
: 000
: 0
: JSR R5,TXCHAR ;LOAD ADDRESS, TX 2ND FLAG
: 1$: 000 ;** HOLE FOR SECONDARY STATION ADDRESS
: 8.

```

```

5219 030232 004737 005344
5220 030236 005003
5221
5222 030240
030240
030240 104402
5223 030242 004537 007324
5224 030246 113523
5225 030250 000000
5226 030252 103003
5227 030254
030254 104460
5228 030256
030256 104410
030260 000444
5229
5230
5231
5232 030262 110337 030344
5233 030266 110337 030424
5234 030272 110337 030512
5235 030276 110337 030630
5236 030302 110337 030404
5237 030306 105137 030404
5238 030312 113737 030404 030576
5239
5240
5241
5242 030320 004537 007734
5243 030324 000001
5244 030326 000007
5245 030330 004537 007734
5246 030334 000000
5247 030336 000000
5248 030340 004537 007622
5249 030344 000000
5250 030346 000010

```

TEST 6 -- BOP RX ALL PARTIES ADDRESS TEST

```

5251 030350 103003          BCC      .+8.          ;BR IF NO ERROR
5252 030352          ERROR          ;REPORT STACKED ERROR
      030352 104460
5253 030354          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP      C$ERROR
      030354 104410          ;                                TRAP      C$ESCAPE
      030356 000346          ;                                .WORD     L10035-.
5254 030360 004537 007622    JSR      R5,TXCHAR      ;LOAD 000, TX ADDRESS
5255 030364 000000          000
5256 030366 100011          NCTBMT*256.!9.
5257 030370 103003          BCC      .+8.          ;BR IF NO ERROR
5258 030372          ERROR          ;REPORT STACKED ERROR
      030372 104460          ;                                TRAP      C$ERROR
5259 030374          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      030374 104410          ;                                .WORD     L10035-.
      030376 000326          ;
5260 030400 004537 007622    JSR      R5,TXCHAR      ;LOAD ADDR_NOT, TX 000
5261 030404 000000          000          ;** HOLE FOR COMPLEMENTED ADDRESS
5262 030406 100010          2$:  NCTBMT*256.!8.
5263 030410 103003          BCC      .+8.          ;BR IF NO ERROR
5264 030412          ERROR          ;REPORT STACKED ERROR
      030412 104460          ;                                TRAP      C$ERROR
5265 030414          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      030414 104410          ;                                .WORD     L10035-.
      030416 000306          ;
5266 030420 004537 007622    JSR      R5,TXCHAR      ;LOAD ADDRESS AGAIN
5267 030424 000000          000          ;** HOLE FOR ADDRESS (AGAIN)
5268 030426 000000          3$:  0
5269 030430 103003          BCC      .+8.          ;BR IF NO ERROR
5270 030432          ERROR          ;REPORT STACKED ERROR
      030432 104460          ;                                TRAP      C$ERROR
5271 030434          ESCAPE SUB          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
      030434 104410          ;                                .WORD     L10035-.
      030436 000266          ;
5272 030440 004537 011540    JSR      R5,STEPLU      ;CLOCK/RCV ADDRESS FIELD
5273 030444 000002          2
5274
5275 030446 004537 006122    JSR      R5,CKRDA       ;DID USYRT RESPOND TO ADDRESS ??
5276 030452 000001          1
5277 030454 103475          BCS      10$          ;BR IF RDA=0
5278
5279          ;-----
5280          ; USYRT RESPONDED TO MESSAGE (RDA=1): SHOULD IT HAVE?
5281          ;-----
5281 030456 022703 000123    CMP      #123,R3        ;ADDRESS = 123 ?
5282 030462 001411          BEQ      40$          ;BR IF YES
5283 030464 022703 000377    CMP      #377,R3        ;ADDRESS = 377 ?
5284 030470 001406          BEQ      40$
5285          ;-----
5286          ; ...NO, REPORT ERROR : "USYRT RESPONDED TO WRONG ADDRESS"
5287          ;-----
5288 030472          GEDF      EM102,ERR21          ; "DEVICE FATAL" ERROR # 45
      030472 104455          ;                                TRAP      C$ERDF
      030474 000055          ;                                .WORD     45
      030476 016466          ;                                .WORD     EM102
      030500 022204          ;                                .WORD     ERR21
5289 030502          ESCAPE SUB
      030502 104410          ;                                TRAP      C$ESCAPE

```

TEST 6 -- BOP RX ALL PARTIES ADDRESS TEST

```

030504 000220                                     .WORD  L10035-.
5290
5291 ; -----
5292 ; ...YES, READ AND VERIFY RECEIVED MESSAGE
5293 030506 004537 010034 40$: JSR    R5,RXCHAR      ;READ & CHK ADDRESS, RCV 000
5294 030512 000400 4$:   RXSOM!000      ; & CHECK RSOM=1
5295 030514 000000      0
5296 030516 100010      0 NOCRDA!8.      ;NO INITIAL CHECK OF RDA=0
5297 030520 103003      BCC    .+8.      ;BR IF NO ERROR
5298 030522 104460      ERROR      ;REPORT STACKED ERROR
5299 030524      ESCAPE  SUB      ;SKIP TO END OF TEST          TRAP    C$ERROR
030524 104410      TRAP    C$ESCAPE
030526 000176      .WORD  L10035-.
5300 030530 004537 007734      JSR    R5, TXCTRL  ;SET TEOM
5301 030534 000002      TEOM
5302 030536 000000      0
5303 030540 004537 010034      JSR    R5,RXCHAR  ;READ/CHECK 000
5304 030544 000000      000
5305 030546 000000      0
5306 030550 100010      0 NOCRDA!8.      ;NO INITIAL CHECK OF RDA=0
5307 030552 103003      BCC    .+8.      ;BR IF NO ERROR
5308 030554 104460      ERROR      ;REPORT STACKED ERROR          TRAP    C$ERROR
5309 030556      ESCAPE  SUB      ;SKIP TO END OF TEST          TRAP    C$ESCAPE
030556 104410      TRAP    C$ESCAPE
030560 000144      .WORD  L10035-.
5310 030562 004537 007734      JSR    R5, TXCTRL  ;SET TEOM
5311 030566 000002      TEOM
5312 030570 000000      0
5313 030572 004537 010034      JSR    R5,RXCHAR  ;READ/CHECK COMPLEMENTED ADDRESS
5314 030576 000000 5$:   000      ;** HOLE FOR ADDRESS_NOT
5315 030600 000000      0      ;NO INITIAL CHECK OF RDA=0
5316 030602 120010      0 NOCRDA!NCRACT!8. ;DON'T CHECK FINAL RXACT=1
5317 030604 103003      BCC    .+8.      ;BR IF NO ERROR
5318 030606 104460      ERROR      ;REPORT STACKED ERROR          TRAP    C$ERROR
5319 030610      ESCAPE  SUB      ;SKIP TO END OF TEST          TRAP    C$ESCAPE
030610 104410      TRAP    C$ESCAPE
030612 000112      .WORD  L10035-.
5320 030614 004537 007734      JSR    R5, TXCTRL  ;SET TSOM
5321 030620 000001      TSOM
5322 030622 000000      0
5323 030624 004537 010034      JSR    R5,RXCHAR  ;READ/CHECK ADDRESS (AGAIN)
5324 030630 001000 6$:   RXEOM!000      ;** HOLE FOR FINAL ADDRESS
5325 030632 000000      0
5326 030634 060000      0 NOCRDA!NCRACT  ;DON'T CHECK FOR FINAL RDA=RXACT=1
5327 030636 103003      BCC    .+8.      ;BR IF NO ERROR
5328 030640 104460      ERROR      ;REPORT STACKED ERROR          TRAP    C$ERROR
5329 030642      ESCAPE  SUB      ;SKIP TO END OF TEST          TRAP    C$ESCAPE
030642 104410      TRAP    C$ESCAPE
030644 000060      .WORD  L10035-.
5330 030646 000422      BR     20$      ;BR TO CONTINUE TEST
5331 ; -----
5332 ; USYRT DIDN'T RESPOND TO MESSAGE (RDA=0): SHOULD IT HAVE ?
5333 ; -----

```


TEST 7 -- BOP RX BIT STUFFING TEST

5371

.SBTTL TEST 7 -- BOP RX BIT STUFFING TEST

*
* TEST 7 -- BOP RX BIT STUFFING TEST
*
* THE USYRT IS INITIALIZED AND THE FOLLOWING TEXT IS TRANSMITTED
* (DELIMITED BY THE APPROPRIATE CONTROL CHARACTERS -- OF COURSE):
*
* 000, 017, 036, 074, 170, 360, 037, 076, 174, 370, 077, 176, 374,
* 177, 376, 377.
*
* NOTE THAT THIS PATTERN CONSISTS OF CHARACTERS WHICH REQUIRE BIT
* STUFFING BOTH INDIVIDUALLY AND IN COMBINATION WITH ADJACENT
* CHARACTERS. THERE ARE ALSO CHARACTERS WHICH REQUIRE NO BIT STUFFING
* AT ALL. ALL 16 CHARACTERS ARE READ BY THE RECEIVER AND COMPARED AS
* THEY ARE MADE AVAILABLE AT RXDB.
*
*

```

:
:          BGNTST
:
5372 030730 004737 005344          JSR    PC,INIDMV          ;INIT DMV-11, ENTER M-LOOP          T7::
5373
5374 030734 004537 007324          JSR    R5,INITRN          ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5375 030740 003626          NOCHK!SYNCH          ;SET BOP MODE,SYNCH REG=226
5376 030742 000000          0          ;USE 8 BIT CHARS
5377 030744 103003          BCC    .+8.          ;BR IF NO ERROR
5378 030746          ERROR          ;REPORT STACKED ERROR
5379 030750          ESCAPE TST          ;SKIP TO END OF TEST          TRAP    C$ERROR
5379 030750 104410          .WORD    L10036-.
5379 030752 001120
5380
5381 030754 004537 007734          JSR    R5,TXCTRL          ;LOAD 2ND FLAG,TX 1ST FLAG
5382 030760 000001          TSOM
5383 030762 000007          7.
5384 030764 004537 007734          JSR    R5,TXCTRL          ;CLEAR TSOM
5385 030770 000000          000
5386 030772 000000          0
5387
5388 030774 004537 007622          JSR    R5,TXCHAR          ;LOAD 000(DATA1), TX 2ND FLAG
5389 031000 000000          000
5390 031002 000010          8.
5391 031004 103003          BCC    .+8.          ;BR IF NO ERROR
5392 031006          ERROR          ;REPORT STACKED ERROR
5392 031006 104460          TRAP    C$ERROR
5393 031010          ESCAPE TST          ;SKIP TO END OF TEST          TRAP    C$ESCAPE
5393 031010 104410          .WORD    L10036-.
5393 031012 001060
5394
5395 031014 004537 007622          JSR    R5,TXCHAR          ;LOAD 017(DATA2), TX 000(DATA1)
5396 031020 000017          017
5397 031022 000010          8.
5398 031024 103003          BCC    .+8.          ;BR IF NO ERROR
5399 031026          ERROR          ;REPORT STACKED ERROR          TRAP    C$ERROR
5399 031026 104460
```

TEST 7 -- BOP RX BIT STUFFING TEST

5400	031030		ESCAPE	TST	;SKIP TO END OF TEST		
	031030	104410				TRAP	C\$ESCAPE
	031032	001040				.WORD	L10036--
5401							
5402	031034	004537	007622	JSR	R5, TXCHAR	;LOAD 036(DATA3), TX 017(DATA2)	
5403	031040	000036		036			
5404	031042	000010		8.			
5405	031044	103003		BCC	.+8.	;BR IF NO ERROR	
5406	031046			ERROR		;REPORT STACKED ERROR	
	031046	104460					TRAP
5407	031050			ESCAPE	TST	;SKIP TO END OF TEST	C\$ERROR
	031050	104410					TRAP
	031052	001020				.WORD	C\$ESCAPE
5408							L10036--
5409	031054	004537	007622	JSR	R5, TXCHAR	;LOAD 074(DATA4)	
5410	031060	000074		074			
5411	031062	000000		0			
5412	031064	103003		BCC	.+8.	;BR IF NO ERROR	
5413	031066			ERROR		;REPORT STACKED ERROR	
	031066	104460					TRAP
5414	031070			ESCAPE	TST	;SKIP TO END OF TEST	C\$ERROR
	031070	104410					TRAP
	031072	001000				.WORD	C\$ESCAPE
5415							L10036--
5416	031074	004537	011310	JSR	R5, RCV1ST	;CLOCK AND RCV 000(DATA1)	
5417	031100	000000		0			
5418	031102	103003		BCC	.+8.	;BR IF NO ERROR	
5419	031104			ERROR		;REPORT STACKED ERROR	
	031104	104460					TRAP
5420	031106			ESCAPE	TST	;SKIP TO END OF TEST	C\$ERROR
	031106	104410					TRAP
	031110	000762				.WORD	C\$ESCAPE
5421							L10036--
5422	031112	004537	010034	JSR	R5, RXCHAR	;READ & CHK 000(DATA1), RCV 017(DATA2)	
5423	031116	000400		RXSOM!000		; & CHECK RSOM=1	
5424	031120	000000		0			
5425	031122	000010		8.			
5426	031124	103003		BCC	.+8.	;BR IF NO ERROR	
5427	031126			ERROR		;REPORT STACKED ERROR	
	031126	104460					TRAP
5428	031130			ESCAPE	TST	;SKIP TO END OF TEST	C\$ERROR
	031130	104410					TRAP
	031132	000740				.WORD	C\$ESCAPE
5429							L10036--
5430	031134	004537	007622	JSR	R5, TXCHAR	;LOAD 170(DATA5)	
5431	031140	000170		170			
5432	031142	000000		0			
5433	031144	103003		BCC	.+8.	;BR IF NO ERROR	
5434	031146			ERROR		;REPORT STACKED ERROR	
	031146	104460					TRAP
5435	031150			ESCAPE	TST	;SKIP TO END OF TEST	C\$ERROR
	031150	104410					TRAP
	031152	000720				.WORD	C\$ESCAPE
5436							L10036--
5437	031154	004537	010034	JSR	R5, RXCHAR	;READ/CHECK 017(DATA2), RCV 036(DATA3)	
5438	031160	000017		017			
5439	031162	000000		0			

TEST 7 -- BOP RX BIT STUFFING TEST

5440	031164	000010		8.					
5441	031166	103003		BCC	+.8.		;BR IF NO ERROR		
5442	031170			ERROR			;REPORT STACKED ERROR		
	031170	104460						TRAP	C\$ERROR
5443	031172			ESCAPE	TST		;SKIP TO END OF TEST		
	031172	104410						TRAP	C\$ESCAPE
	031174	000676						.WORD	L10036-.
5444									
5445	031176	004537	007622	JSR	R5, TXCHAR		;LOAD 360(DATA6)		
5446	031202	000360		360					
5447	031204	000000		0					
5448	031206	103003		BCC	+.8.		;BR IF NO ERROR		
5449	031210			ERROR			;REPORT STACKED ERROR		
	031210	104460						TRAP	C\$ERROR
5450	031212			ESCAPE	TST		;SKIP TO END OF TEST		
	031212	104410						TRAP	C\$ESCAPE
	031214	000656						.WORD	L10036-.
5451									
5452	031216	004537	010034	JSR	R5, RXCHAR		;READ/CHECK 036(DATA3),RCV 074(DATA4)		
5453	031222	000036		036					
5454	031224	000000		0					
5455	031226	000010		8.					
5456	031230	103003		BCC	+.8.		;BR IF NO ERROR		
5457	031232			ERROR			;REPORT STACKED ERROR		
	031232	104460						TRAP	C\$ERROR
5458	031234			ESCAPE	TST		;SKIP TO END OF TEST		
	031234	104410						TRAP	C\$ESCAPE
	031236	000634						.WORD	L10036-.
5459									
5460	031240	004537	007622	JSR	R5, TXCHAR		;LOAD 037(DATA7)		
5461	031244	000037		037					
5462	031246	000000		0					
5463	031250	103003		BCC	+.8.		;BR IF NO ERROR		
5464	031252			ERROR			;REPORT STACKED ERROR		
	031252	104460						TRAP	C\$ERROR
5465	031254			ESCAPE	TST		;SKIP TO END OF TEST		
	031254	104410						TRAP	C\$ESCAPE
	031256	000614						.WORD	L10036-.
5466									
5467	031260	004537	010034	JSR	R5, RXCHAR		;READ/CHECK 074(DATA4),RCV 170(DATA5)		
5468	031264	000074		074					
5469	031266	000000		0					
5470	031270	000010		8.					
5471	031272	103003		BCC	+.8.		;BR IF NO ERROR		
5472	031274			ERROR			;REPORT STACKED ERROR		
	031274	104460						TRAP	C\$ERROR
5473	031276			ESCAPE	TST		;SKIP TO END OF TEST		
	031276	104410						TRAP	C\$ESCAPE
	031300	000572						.WORD	L10036-.
5474									
5475	031302	004537	007622	JSR	R5, TXCHAR		;LOAD 076(DATA8)		
5476	031306	000076		076					
5477	031310	000000		0					
5478	031312	103003		BCC	+.8.		;BR IF NO ERROR		
5479	031314			ERROR			;REPORT STACKED ERROR		
	031314	104460						TRAP	C\$ERROR
5480	031316			ESCAPE	TST		;SKIP TO END OF TEST		

TEST 7 -- BOP RX BIT STUFFING TEST

	031316	104410					TRAP	C\$ESCAPE
	031320	000552					.WORD	L10036-.
5481								
5482	031322	004537	010034	JSR	R5,RXCHAR	;READ/CHECK 170(DATA5),RCV 360(DATA6)		
5483	031326	000170		170				
5484	031330	000000		0				
5485	031332	000010		8.				
5486	031334	103003		BCC	+.8.	;BR IF NO ERROR		
5487	031336			ERROR		;REPORT STACKED ERROR		
	031336	104460					TRAP	C\$ERROR
5488	031340			ESCAPE	TST	;SKIP TO END OF TEST		
	031340	104410					TRAP	C\$ESCAPE
	031342	000530					.WORD	L10036-.
5489								
5490	031344	004537	007622	JSR	R5, TXCHAR	;LOAD 174(DATA9)		
5491	031350	000174		174				
5492	031352	000000		0				
5493	031354	103003		BCC	+.8.	;BR IF NO ERROR		
5494	031356			ERROR		;REPORT STACKED ERROR		
	031356	104460					TRAP	C\$ERROR
5495	031360			ESCAPE	TST	;SKIP TO END OF TEST		
	031360	104410					TRAP	C\$ESCAPE
	031362	000510					.WORD	L10036-.
5496								
5497	031364	004537	010034	JSR	R5,RXCHAR	;READ/CHECK 360(DATA6),RCV 037(DATA7)		
5498	031370	000360		360				
5499	031372	000000		0				
5500	031374	000010		8.				
5501	031376	103003		BCC	+.8.	;BR IF NO ERROR		
5502	031400			ERROR		;REPORT STACKED ERROR		
	031400	104460					TRAP	C\$ERROR
5503	031402			ESCAPE	TST	;SKIP TO END OF TEST		
	031402	104410					TRAP	C\$ESCAPE
	031404	000466					.WORD	L10036-.
5504								
5505	031406	004537	007622	JSR	R5, TXCHAR	;LOAD 370(DATA10)		
5506	031412	000370		370				
5507	031414	000000		0				
5508	031416	103003		BCC	+.8.	;BR IF NO ERROR		
5509	031420			ERROR		;REPORT STACKED ERROR		
	031420	104460					TRAP	C\$ERROR
5510	031422			ESCAPE	TST	;SKIP TO END OF TEST		
	031422	104410					TRAP	C\$ESCAPE
	031424	000446					.WORD	L10036-.
5511								
5512	031426	004537	010034	JSR	R5,RXCHAR	;READ/CHECK 037(DATA7),RCV 076(DATA8)		
5513	031432	000037		037				
5514	031434	000000		0				
5515	031436	000014		12.		;(EXTRA 4 TICKS FOR BIT-STUFF & FIFO)		
5516	031440	103003		BCC	+.8.	;BR IF NO ERROR		
5517	031442			ERROR		;REPORT STACKED ERROR		
	031442	104460					TRAP	C\$ERROR
5518	031444			ESCAPE	TST	;SKIP TO END OF TEST		
	031444	104410					TRAP	C\$ESCAPE
	031446	000424					.WORD	L10036-.
5519								
5520	031450	004537	007622	JSR	R5, TXCHAR	;LOAD 077(DATA11)		

TEST 7 -- BOP RX BIT STUFFING TEST

5521	031454	000077		077					
5522	031456	000000		0					
5523	031460	103003		BCC	+.8.		;BR IF NO ERROR		
5524	031462			ERROR			;REPORT STACKED ERROR		
	031462	104460						TRAP	C\$ERROR
5525	031464			ESCAPE	TST		;SKIP TO END OF TEST		
	031464	104410						TRAP	C\$ESCAPE
	031466	000404						.WORD	L10036-.
5526									
5527	031470	004537	010034	JSR	R5,RXCHAR		;READ/CHECK 076(DATA8),RCV 174(DATA9)		
5528	031474	000076		076					
5529	031476	000000		0					
5530	031500	000010		8.					
5531	031502	103003		BCC	+.8.		;BR IF NO ERROR		
5532	031504			ERROR			;REPORT STACKED ERROR		
	031504	104460						TRAP	C\$ERROR
5533	031506			ESCAPE	TST		;SKIP TO END OF TEST		
	031506	104410						TRAP	C\$ESCAPE
	031510	000362						.WORD	L10036-.
5534									
5535	031512	004537	007622	JSR	R5, TXCHAR		;LOAD 176(DATA12)		
5536	031516	000176		176					
5537	031520	000000		0					
5538	031522	103003		BCC	+.8.		;BR IF NO ERROR		
5539	031524			ERROR			;REPORT STACKED ERROR		
	031524	104460						TRAP	C\$ERROR
5540	031526			ESCAPE	TST		;SKIP TO END OF TEST		
	031526	104410						TRAP	C\$ESCAPE
	031530	000342						.WORD	L10036-.
5541									
5542	031532	004537	010034	JSR	R5,RXCHAR		;READ/CHECK 174(DATA9),RCV 370(DATA10)		
5543	031536	000174		174					
5544	031540	000000		0					
5545	031542	000010		8.					
5546	031544	103003		BCC	+.8.		;BR IF NO ERROR		
5547	031546			ERROR			;REPORT STACKED ERROR		
	031546	104460						TRAP	C\$ERROR
5548	031550			ESCAPE	TST		;SKIP TO END OF TEST		
	031550	104410						TRAP	C\$ESCAPE
	031552	000320						.WORD	L10036-.
5549									
5550	031554	004537	007622	JSR	R5, TXCHAR		;LOAD 374(DATA13)		
5551	031560	000374		374					
5552	031562	000000		0					
5553	031564	103003		BCC	+.8.		;BR IF NO ERROR		
5554	031566			ERROR			;REPORT STACKED ERROR		
	031566	104460						TRAP	C\$ERROR
5555	031570			ESCAPE	TST		;SKIP TO END OF TEST		
	031570	104410						TRAP	C\$ESCAPE
	031572	000300						.WORD	L10036-.
5556									
5557	031574	004537	010034	JSR	R5,RXCHAR		;READ/CHECK 370(DATA10),RCV 077(DATA11)		
5558	031600	000370		370					
5559	031602	000000		0					
5560	031604	000010		8.					
5561	031606	103003		BCC	+.8.		;BR IF NO ERROR		
5562	031610			ERROR			;REPORT STACKED ERROR		

TEST 7 -- BOP RX BIT STUFFING TEST

```

5602 031742 004537 010034      JSR      R5,RXCHAR      ;READ/CHECK 374(DATA13),RCV 177(DATA14)
5603 031746 000374              374
5604 031750 000000              0
5605 031752 000010              8.
5606 031754 103003      BCC      .+8.          ;BR IF NO ERROR
5607 031756              ERROR          ;REPORT STACKED ERROR
                    031756 104460
5608 031760              ESCAPE  TST          ;SKIP TO END OF TEST
                    031760 104410
                    031762 000110
                                TRAP      C$ERROR
                                .WORD    C$ESCAPE
                                L10036-.
5609
5610 031764 004537 007734      JSR      R5,TXCTRL     ;LOAD 1ST TEOM
5611 031770 000002              TEOM
5612 031772 000000              0
5613 031774 103003      BCC      .+8.          ;BR IF NO ERROR
5614 031776              ERROR          ;REPORT STACKED ERROR
                    031776 104460
5615 032000              ESCAPE  TST          ;SKIP TO END OF TEST
                    032000 104410
                    032002 000070
                                TRAP      C$ERROR
                                .WORD    C$ESCAPE
                                L10036-.
5616
5617 032004 004537 010034      JSR      R5,RXCHAR     ;READ/CHECK 177(DATA14),RCV 376(DATA15)
5618 032010 000177              177
5619 032012 000000              0
5620 032014 000010              8.
5621 032016 103003      BCC      .+8.          ;BR IF NO ERROR
5622 032020              ERROR          ;REPORT STACKED ERROR
                    032020 104460
5623 032022              ESCAPE  TST          ;SKIP TO END OF TEST
                    032022 104410
                    032024 000046
                                TRAP      C$ERROR
                                .WORD    C$ESCAPE
                                L10036-.
5624
5625 032026 004537 010034      JSR      R5,RXCHAR     ;READ/CHECK 376(DATA15),RCV 377(DATA16)
5626 032032 000376              376
5627 032034 000000              0
5628 032036 020014      NCRACT!12.          ;DON'T CHECK FOR FINAL RXACT=1
5629 032040 103003      BCC      .+8.          ;(EXTRA 4 TICKS FOR BIT-STUFF/FIFO)
5630 032042              ERROR          ;BR IF NO ERROR
                    032042 104460
5631 032044              ESCAPE  TST          ;REPORT STACKED ERROR
                    032044 104410
                    032046 000024
                                TRAP      C$ERROR
                                .WORD    C$ESCAPE
                                L10036-.
5632
5633 032050 004537 010034      JSR      R5,RXCHAR     ;READ/CHECK 377(DATA16)
5634 032054 001377      RXEOM!377          ; & CHECK REOM
5635 032056 000000              0
5636 032060 060000      NFCRDA!NCRACT      ;DON'T CHECK FOR FINAL RDA=RXACT=1
5637 032062 103003      BCC      .+3.          ;BR IF NO ERROR
5638 032064              ERROR          ;REPORT STACKED ERROR
                    032064 104460
5639 032066              ESCAPE  TST          ;SKIP TO END OF TEST
                    032066 104410
                    032070 000002
                                TRAP      C$ERROR
                                .WORD    C$ESCAPE
                                L10036-.
5640 032072              ENDTST
                    032072 104401
                                L10036:
                                TRAP      C$ETST

```


TEST 8 -- BOP RX UNDERRUN IDLE ABORTS/FLAGS

5649

.SBTTL TEST 8 -- BOP RX UNDERRUN IDLE ABORTS/FLAGS

```

:*****
:*
:* TEST 8 -- BOP RX UNDERRUN IDLE ABORTS/FLAGS
:*
:* THE USYRT IS INITIALIZED AND A MESSAGE IS STARTED. THEN, A
:* TRANSMITTER UNDERRUN IS FORCED WITH IDLE = 0 -- CAUSING ABORT
:* CHARACTER(S) TO BE IDLED. THE RECEIVER SHOULD BE RESET BY THE ABORT
:* CHARACTER(S). VERIFY THAT RAB/GA BIT=1.
:* REPEAT THE ABOVE WITH IDLE=1.
:*
:*****

```

```

:
: BGN1ST
:
: T8::
:***** SUBTEST # 1 *****
: BGNSUB
: T8.1:
: TRAP C$BSUB
5650 032074
5651 032074
5652 032074 104402 004737 005344 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
5653
5654 032102 004537 007324 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5655 032106 003626 NOCHK!SYNCH ;SET BOP MODE,SYNCH REG=226
5656 032110 000000 0 ;USE 8 BIT CHARS
5657 032112 103003 BCC .+8. ;BR IF NO ERROR
5658 032114 ERROR ;REPORT STACKED ERROR
5659 032114 104460 ESCAPE SUB ;SKIP TO END OF TEST TRAP C$ERROR
5659 032116 104410 TRAP C$ESCAPE
5659 032120 000300 .WORD L10040-.
5660
5661 032122 004537 007734 JSR R5,TXCTRL ;LOAD 2ND FLAG,TX 1ST FLAG
5662 032126 000001 TSOM
5663 032130 000007 7.
5664 032132 004537 007734 JSR R5,TXCTRL ;CLEAR TSOM
5665 032136 000000 000
5666 032140 000000 0
5667
5668 032142 004537 007622 JSR R5,TXCHAR ;LOAD 123(DATA1), TX 2ND FLAG
5669 032146 000123 123
5670 032150 000010 8.
5671 032152 103003 BCC .+8. ;BR IF NO ERROR
5672 032154 ERROR ;REPORT STACKED ERROR
5673 032154 104460 ESCAPE SUB ;SKIP TO END OF TEST TRAP C$ERROR
5673 032156 104410 TRAP C$ESCAPE
5673 032160 000240 .WORD L10040-.
5674
5675 032162 004537 007622 JSR R5,TXCHAR ;LOAD 321(DATA2), TX 123(DATA1)
5676 032166 000321 321
5677 032170 000010 8.
5678 032172 103003 BCC .+8. ;BR IF NO ERROR
5679 032174 ERROR ;REPORT STACKED ERROR
5680 032174 104460 ESCAPE SUB ;SKIP TO END OF TEST TRAP C$ERROR
5680 032176 104410 TRAP C$ESCAPE

```


TEST 8 -- BOP RX UNDERRUN IDLE ABORTS/FLAGS

```

5722 032342          GEDF   EM40,ERR12      ;** REPORT RAB/GA BIT NOT SET!!!
;          "DEVICE FATAL" ERROR # 48
          032342 104455          TRAP      C$ERDF
          032344 000060          .WORD     48
          032346 014734          .WORD     EM40
          032350 021714          .WORD     ERR12
5723 032352          ESCAPE SUB          ;** AND EXIT TEST
          032352 104410          TRAP      C$ESCAPE
          032354 000044          .WORD     L10040-.
5724 032356 132737 000002 032330 10$: BITB   @REOM,1$      ;*** CHECK FOR RXEOM BIT = 1 ***
5725 032364 001006          BNE     15$
5726 032366          GEDF   EM31,ERR12      ;** REPORT REOM BIT NOT SET!!!
;          "DEVICE FATAL" ERROR # 49
          032366 104455          TRAP      C$ERDF
          032370 000061          .WORD     49
          032372 014537          .WORD     EM31
          032374 021714          .WORD     ERR12
5727 032376          ESCAPE SUB          ;** AND EXIT TEST
          032376 104410          TRAP      C$ESCAPE
          032400 000020          .WORD     L10040-.
5728
5729 032402 004537 005356 15$: JSR     R5,CKUSTS      ;** CHECK USYRT STATUS **
5730 032406 000116          TBMT!TSO!TXACT!TXU
5731 032410 103003          BCC     .+8.
5732 032412          ERROR
          032412 104460          ;BR IF NO ERROR
          ;REPORT STACKED ERROR
          ;SKIP TO END OF TEST
          TRAP      C$ERROR
5733 032414          ESCAPE SUB          ;SKIP TO END OF TEST
          032414 104410          TRAP      C$ESCAPE
          032416 000002          .WORD     L10040-.
5734 032420          ENDSUB
          032420          L10040:
          032420 104403          TRAP      C$ESUB
5735
5736 032422          ;***** SUBTEST # 2 *****
          032422          BGNSUB
          032422 104402          T8.2:
          032424 004737 005344          TRAP      C$BSUB
5737 032424          JSR     PC,INIDMV      ;INIT DMV-11, ENTER M-LOOP
5738
5739 032430 004537 007324          JSR     R5,INITRN      ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5740 032434 007626          IDLES!NOCHK!SYNCH    ;SET BOP MODE, IDLE=1, SYNCH REG=226
5741 032436 000000          0
5742 032440 103003          BCC     .+8.
5743 032442          ERROR
          032442 104460          ;USE 8 BIT CHARS
          ;BR IF NO ERROR
          ;REPORT STACKED ERROR
          TRAP      C$ERROR
5744 032444          ESCAPE SUB          ;SKIP TO END OF TEST
          032444 104410          TRAP      C$ESCAPE
          032446 000242          .WORD     L10041-.
5745
5746 032450 004537 007734          JSR     R5,TXCTRL      ;LOAD 2ND FLAG, TX 1ST FLAG
5747 032454 000001          TSOM
5748 032456 000007          7.
5749 032460 004537 007734          JSR     R5,TXCTRL      ;CLEAR TSOM
5750 032464 000000          000
5751 032466 000000          0
5752
5753 032470 004537 007622          JSR     R5,TXCHAR      ;LOAD 123(DATA1), TX 2ND FLAG
5754 032474 000123          123

```

TEST 8 -- BOP RX UNDERRUN IDLE ABORTS/FLAGS

5755	032476	000010		8.					
5756	032500	103003		BCC	.+8.				
5757	032502			ERROR					
	032502	104460							
5758	032504			ESCAPE	SUB				
	032504	104410							
	032506	000202							
5759									
5760	032510	004537	007622	JSR	R5, TXCHAR				
				321					
5761	032514	000321							
5762	032516	000010		8.					
5763	032520	103003		BCC	.+8.				
5764	032522			ERROR					
	032522	104460							
5765	032524			ESCAPE	SUB				
	032524	104410							
	032526	00C162							
5766									
5767	032530	004537	007622	JSR	R5, TXCHAR				
				000					
5768	032534	000000							
5769	032536	000010		8.					
5770	032540	103003		BCC	.+8.				
5771	032542			ERROR					
	032542	104460							
5772	032544			ESCAPE	SUB				
	032544	104410							
	032546	000142							
5773									
5774	032550	004537	011310	JSR	R5, RCV1ST				
				0					
5775	032554	000000							
5776	032556	103003		BCC	.+8.				
5777	032560			ERROR					
	032560	104460							
5778	032562			ESCAPE	SUB				
	032562	104410							
	032564	000124							
5779									
5780	032566	004537	010034	JSR	R5, RXCHAR				
				RXSOM!123					
5781	032572	000523		0					
5782	032574	000000		8.					
5783	032576	000010		BCC	.+8.				
5784	032600	103003		ERROR					
5785	032602								
	032602	104460							
5786	032604			ESCAPE	SUB				
	032604	104410							
	032606	000102							
5787									
5788	032610	004537	010034	JSR	R5, RXCHAR				
				321					
5789	032614	000321		0					
5790	032616	000000		NCRACK!8.					
5791	032620	020010		BCC	.+8.				
5792	032622	103003		ERROR					
5793	032624								
	032624	104460							
5794	032626			ESCAPE	SUB				
	032626	104410							

TEST 8 -- BOP RX UNDERRUN IDLE ABORTS/FLAGS

```

032630 000060                                .WORD  L10041-.
5795
5796 032632 004537 005356                    JSR      R5,CKUSTS          ;+++ CHECK FOR TXU=1 +++
5797 032636 000336                            RDA!TBMT!RSA!TSO!TXACT!TXU
5798 032640 103003                            BCC      .+8.              ;BR IF NO ERROR
5799 032642                                ERROR                      ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5800 032644                                ESCAPE  SUB                ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD  L10041-.
032644 104410
032646 000042
5801
5802 032650 004537 010034                    JSR      R5,RXCHAR         ;READ/CHECK 000(DATA3)
5803 032654 001000                            RXEOM!000                 ; & CHECK REOM
5804 032656 000000                            0
5805 032660 060010                            NFCRDA!NCRACT!8.        ;DON'T CHECK FOR FINAL RDA=RXACT=1
5806 032662 103003                            BCC      .+8.              ;BR IF NO ERROR
5807 032664                                ERROR                      ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5808 032666                                ESCAPE  SUB                ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD  L10041-.
032666 104410
032670 000020
5809
5810 032672 004537 005356                    JSR      R5,CKUSTS          ;++ CHECK USYRT STATUS ++
5811 032676 000116                            TBMT!TSO!TXACT!TXU
5812 032700 103003                            BCC      .+8.              ;BR IF NO ERROR
5813 032702                                ERROR                      ;REPORT STACKED ERROR
                                TRAP      C$ERROR
5814 032704                                ESCAPE  SUB                ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD  L10041-.
032704 104410
032706 000002
5815 032710                                ENDSUB
                                L10041:
032710 104403                                TRAP      C$ESUB
5816 032712                                ENDTST
                                L10037:
032712 104401                                TRAP      C$ETST

```

TEST 9 -- BOP RX LOST RXE TEST

5823

.SBTTL TEST 9 -- BOP RX LOST RXE TEST

```

:*****
:*
:* TEST 9 -- BOP RX LOST RXE TEST
:*
:* THE USYRT IS INITIALIZED AND A MESSAGE IS STARTED. WHILE IN THE
:* MIDDLE OF TEXT, RXE IS DROPPED AND THE REACTION OF THE RECEIVER IS
:* MONITORED.
:*
:--*****
:
: BGNTST

```

```

032714
5824
5825 032714 004737 005344 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
5826
5827 032720 004537 007324 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5828 032724 007626 IDLES!NOCHK!SYNCH ;SET BOP MODE, IDLE=1, SYNCH REG=226
5829 032726 000000 0 ;USE 8 BIT CHARS
5830 032730 103003 BCC .+8. ;BR IF NO ERROR
5831 032732 104460 ERROR ;REPORT STACKED ERROR
5832 032734 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
032734 104410 TRAP C$ESCAPE
032736 000216 .WORD L10042-.
5833
5834 032740 004537 007734 JSR R5, TXCTRL ;LOAD 2ND FLAG, TX 1ST FLAG
5835 032744 000001 TSOM
5836 032746 000007 7.
5837 032750 004537 007734 JSR R5, TXCTRL ;CLEAR TSOM
5838 032754 000000 000
5839 032756 000000 0
5840
5841 032760 004537 007622 JSR R5, TXCHAR ;LOAD 123(DATA1), TX 2ND FLAG
5842 032764 000123 123
5843 032766 000010 8.
5844 032770 103003 BCC .+8. ;BR IF NO ERROR
5845 032772 104460 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
5846 032774 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
032774 104410 TRAP C$ESCAPE
032776 000156 .WORD L10042-.
5847
5848 033000 004537 007622 JSR R5, TXCHAR ;LOAD 321(DATA2), TX 123(DATA1)
5849 033004 000321 321
5850 033006 000010 8.
5851 033010 103003 BCC .+8. ;BR IF NO ERROR
5852 033012 104460 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
5853 033014 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
033014 104410 TRAP C$ESCAPE
033016 000136 .WORD L10042-.
5854
5855 033020 004537 007622 JSR R5, TXCHAR ;LOAD 000(DATA3), TX 321(DATA2)
5856 033024 000000 000
5857 033026 000010 8.

```

TEST 9 -- BOP RX LOST RXE TEST

```

5858 033030 103003          BCC      .+8.          ;BR IF NO ERROR
5859 033032          ERROR          ;REPORT STACKED ERROR
      033032 104460
5860 033034          ESCAPE  TST          ;SKIP TO END OF TEST          TRAP      C$ERROR
      033034 104410          ;                                TRAP      C$ESCAPE
      033036 000116          ;                                .WORD    L10042-.
5861
5862 033040 004537 011310    JSR      R5,RCV1ST      ;CLOCK AND RCV 123(DATA1)
5863 033044 000000          0
5864 033046 103003          BCC      .+8.          ;BR IF NO ERROR
5865 033050          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      033050 104460          ;                                TRAP      C$ESCAPE
5866 033052          ESCAPE  TST          ;SKIP TO END OF TEST          .WORD    L10042-.
      033052 104410
      033054 000100
5867
5868 033056 004537 010034    JSR      R5,RXCHAR      ;READ & CHK 123(DATA1), RCV 321(DATA2)
5869 033062 000523          RXSOM!123          ; & CHECK RSOM=1
5870 033064 000000          0
5871 033066 000010          8.          ; 8 TICKS OF THE CLOCK
5872 033070 103003          BCC      .+8.          ;BR IF NO ERROR
5873 033072          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      033072 104460          ;                                TRAP      C$ESCAPE
5874 033074          ESCAPE  TST          ;SKIP TO END OF TEST          .WORD    L10042-.
      033074 104410
      033076 000056
-----
5875
5876 033100 004537 003660    JSR      R5,WRITEI      ;DROP RECEIVER ENABLE (RXEN)
5877 033104 120000          VIAORB
5878 033106 000072          TXEN!DTR!RTSND!TTLOOP
5879
5880 033110 004537 005356    JSR      R5,CKUSTS      ;+++ CHECK USYRT STATUS REGISTER +++
5881 033114 000116          TBMT!TSO!TXACT!TXU
5882 033116 103003          BCC      .+8.          ;BR IF NO ERROR
5883 033120          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      033120 104460          ;                                TRAP      C$ESCAPE
5884 033122          ESCAPE  TST          ;SKIP TO END OF TEST          .WORD    L10042-.
      033122 104410
      033124 000030
5885
5886 033126 004537 007734    JSR      R5,TXCTRL      ;LOAD 2ND FLAG,TX 1ST FLAG
5887 033132 000001          TSOM
5888 033134 000010          8.
5889
5890 033136 004537 005356    JSR      R5,CKUSTS      ;+++ CHECK USYRT STATUS REGISTER +++
5891 033142 000104          TBMT!TXACT
5892 033144 103003          BCC      .+8.          ;BR IF NO ERROR
5893 033146          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
      033146 104460          ;                                TRAP      C$ESCAPE
5894 033150          ESCAPE  TST          ;SKIP TO END OF TEST          .WORD    L10042-.
      033150 104410
      033152 000002
5895 033154          ENDTST          ;                                L10042:
      033154 104401          ;                                TRAP      C$ETST

```

TEST 10 -- BOP RX GA (GO-AHEAD) RECOGNITION

5903

.SBTTL TEST 10 -- BOP RX GA (GO-AHEAD) RECOGNITION

```

:*****
:*
:* TEST 10 -- BOP RX GA (GO-AHEAD) RECOGNITION
:*
:* A SHORT MESSAGE IS TRANSMITTED FOLLOWED BY A GA CHARACTER (INSTEAD
:* OF A FLAG CHARACTER). THE RECEIVER IS OBSERVED FOR PROPER HANDLING
:* OF BOTH THE MESSAGE AND THE GA CHARACTER. THE RAB/GA STATUS BIT
:* SHOULD BE SET BY THE RECEIVER UPON RECOGNITION OF THE GA CHARACTER.
:*
:-----

```

```

:
: BGNTST
:
5904 033156 004737 005344 JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP T10::
5905
5906 033162 004537 007324 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5907 033166 023400 STRIPS!NOCHK ;SET BOP MODE,NO ERROR CHECKING,SS/GA=1
5908 033170 000000 0 ;USE 8 BIT CHARS
5909 033172 103003 BCC .+8. ;BR IF NO ERROR
5910 033174 104460 ERROR ;REPORT STACKED ERROR
5911 033176 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
5911 033176 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5911 033200 000216 .WORD L10043-.
5912
5913 033202 004537 007734 JSR R5,TXCTRL ;LOAD 2ND FLAG,TX 1ST FLAG
5914 033206 000001 TSOM
5915 033210 000007 7.
5916 033212 004537 007734 JSR R5,TXCTRL ;CLEAR TSOM
5917 033216 000000 000
5918 033220 000000 0
5919
5920 033222 004537 007622 JSR R5,TXCHAR ;LOAD 123(DATA1), TX 2ND FLAG
5921 033226 000123 123
5922 033230 000010 8.
5923 033232 103003 BCC .+8. ;BR IF NO ERROR
5924 033234 104460 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
5925 033236 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5925 033236 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5925 033240 000156 .WORD L10043-.
5926
5927 033242 004537 007622 JSR R5,TXCHAR ;LOAD 321(DATA2), TX 123(DATA1)
5928 033246 000321 321
5929 033250 000010 8.
5930 033252 103003 BCC .+8. ;BR IF NO ERROR
5931 033254 104460 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
5932 033256 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5932 033256 104410 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5932 033260 000136 .WORD L10043-.
5933
5934 033262 004537 007622 JSR R5,TXCHAR ;LOAD 000(DATA3), TX 321(DATA2)
5935 033266 000000 000
5936 033270 000010 8.

```


TEST 10 -- BOP RX GA (GO-AHEAD) RECOGNITION

```

5937 033272 103003          BCC      .+8.          ;BR IF NO ERROR
5938 033274 104460          ERROR          ;REPORT STACKED ERROR
5939 033276 104410          ESCAPE  TST          ;SKIP TO END OF TEST
033276 104410
033300 000116          TRAP      C$ERROR
5940 033302 004537 011310    JSR      R5,RCV1ST    ;CLOCK AND RCV 123(DATA1)
5941 033306 000000          0
5942 033310 103003          BCC      .+8.          ;BR IF NO ERROR
5943 033312 104460          ERROR          ;REPORT STACKED ERROR
5944 033314 104410          ESCAPE  TST          ;SKIP TO END OF TEST
033314 104410
033316 000100          TRAP      C$ERROR
5945 033320 004537 010034    JSR      R5,RXCHAR    ;READ & CHK 123(DATA1), RCV 321(DATA2)
5946 033324 000523          RXSOM!123          ; & CHECK RSOM=1
5947 033326 000000          0
5948 033330 000010          8.
5949 033332 103003          BCC      .+8.          ; 8 TICKS OF THE CLOCK
5950 033334 104460          ERROR          ;BR IF NO ERROR
5951 033336 104410          ESCAPE  TST          ;REPORT STACKED ERROR
033336 104410
033340 000056          TRAP      C$ERROR
5952 033342 004537 007734    JSR      R5,TXCTRL    ;SET TEOM AND TGA
5953 033346 000012          TEOM!TGA
5954 033350 000000          0
5955 033352 004537 010034    JSR      R5,RXCHAR    ;READ/CHECK 321(DATA2),RCV 000(DATA3)
5956 033356 000321          321
5957 033360 000000          0
5958 033362 020010          NCRACT!8.
5959 033364 103003          BCC      .+8.          ;DON'T CHECK FOR FINAL RXACT=1
5960 033366 104460          ERROR          ;BR IF NO ERROR
5961 033370 104410          ESCAPE  TST          ;REPORT STACKED ERROR
033370 104410
033372 000024          TRAP      C$ERROR
5962 033374 004537 010034    JSR      R5,RXCHAR    ;READ/CHECK 000(DATA3)
5963 033400 003000          RXABGA!RXEOM!000    ;VERIFY REOM & RABGA = 1
5964 033402 000000          0
5965 033404 060000          NFCRDA!NCRACT
5966 033406 103003          BCC      .+8.          ;DON'T CHECK FOR FINAL RDA=RXACT=1
5967 033410 104460          ERROR          ;BR IF NO ERROR
5968 033412 104410          ESCAPE  TST          ;REPORT STACKED ERROR
033412 104410
033414 000002          TRAP      C$ERROR
5969 033416 104401          ENDTST
033416 104401          L10043:
033416 104401          TRAP      C$ETST

```

TEST 11 -- BOP RX "ABC" TEST

5988

.SBTTL TEST 11 -- BOP RX "ABC" TEST

```

*****
;*
;* TEST 11 -- BOP RX "ABC" TEST
;*
;* THIS TEST IS COMPOSED OF 7 SUBTESTS -- EACH ONE CHECKING A DIFFERENT
;* EXPECTED VALUE IN ABC (THE 3 BIT "ASSEMBLED BIT COUNT" FIELD WITHIN
;* RDSR). IN EACH SUBTEST THE USYRT IS INITIALIZED AND A SMALL MESSAGE
;* IS STARTED. THE LAST CHARACTER IS SENT WITH ITS LENGTH BEING
;* SPECIFIED FIRST AS 1 BIT, THEN AS 2 BITS, THEN AS 3 BITS, ETC. IN THE
;* TRANSMITTER SIDE OF THE USYRT. IN ALL CASES THE RECEIVER IS LEFT SET
;* TO 8 BITS IN LENGTH AND WHEN THE FLAG CHARACTER IS DETECTED, ABC IS
;* CHECKED AND SHOULD MATCH TXCL. ERROR LOOPING WILL BE ON THE FAILING
;* SUBTEST.
;*
;*
```

BGNTST

```

033420
5989 033420 004737 005344
5990 033424 012704 000001
5991
5992
5993
```

```

; T11::
; JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
; MOV #1,R4 ;INIT GENERAL PURPOSE INDEX
```

MAIN PROGRAM LOOP

T9.LP: BGNSUB

```

5994 033430
033430
033430 104402
5995 033432 116437 034056 034020
5996 033440 116437 034066 033574
5997
5998 033446 004537 007324
5999 033452 003626
6000 033454 000000
6001 033456 103003
6002 033460
033460 104460
6003 033462
033462 104410
033464 000352
6004
6005 033466 004537 007734
6006 033472 000001
6007 033474 000007
6008 033476 004537 007734
6009 033502 000000
6010 033504 000000
6011
6012 033506 004537 007622
6013 033512 000123
6014 033514 000010
6015 033516 103003
6016 033520
033520 104460
6017 033522
033522 104410
```

```

; T11.1:
; MOVB TABLR(R4),30$ ;SET UP EXPECTED FINAL VALUE TRAP C$BSUB
; MOVBL LNTBL(R4),5$ ;SET UP FINAL TX CHAR LENGTH (1 => 8 BITS)
; JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
; NOCHK!SYNCH ;SET BOP MODE,SYNCH REG=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 L10045-.

; JSR R5,TXCTRL ;LOAD 2ND FLAG,TX 1ST FLAG
; TSOM 7.
; JSR R5,TXCTRL ;CLEAR TSOM
; 000 0

; JSR R5,TXCHAR ;LOAD 123(DATA1), 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
```

TEST 11 -- BOP RX "ABC" TEST

6018	033524	000312					.WORD	L10045-.
6019	033526	004537	007622	JSR	R5, TXCHAR	;LOAD 321(DATA2), TX 123(DATA1)		
6020	033532	000321			321			
6021	033534	000010			8.			
6022	033536	103003		BCC	+.8.	;BR IF NO ERROR		
6023	033540			ERROR		;REPORT STACKED ERROR		
6024	033540	104460					TRAP	C\$ERROR
	033542	104410		ESCAPE	SUB	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	033544	000272					.WORD	L10045-.
6025								
6026	033546	004537	007622	JSR	R5, TXCHAR	;LOAD 000(DATA3), TX 321(DATA2)		
6027	033552	000000			000			
6028	033554	000010			8.			
6029	033556	103003		BCC	+.8.	;BR IF NO ERROR		
6030	033560			ERROR		;REPORT STACKED ERROR		
6031	033560	104460					TRAP	C\$ERROR
	033562	104410		ESCAPE	SUB	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	033564	000252					.WORD	L10045-.
6032								
6033	033566	004537	003660	JSR	R5, WRITEI	;CHANGE BIT LENGTH OF FINAL CHAR		
6034	033572	120407			PCR			
6035	033574	000000		5\$:	000	; ** HOLE FOR RX/TX CHAR LENGTH **		
6036								
6037	033576	004537	007622	JSR	R5, TXCHAR	;LOAD 377(DATA4); TX 000(DATA3)		
6038	033602	000377			377			
6039	033604	000010			8.			
6040	033606	103003		BCC	+.8.	;BR IF NO ERROR		
6041	033610			ERROR		;REPORT STACKED ERROR		
6042	033610	104460					TRAP	C\$ERROR
	033612	104410		ESCAPE	SUB	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	033614	000222					.WORD	L10045-.
6043								
6044	033616	004537	007734	JSR	R5, TXCTRL	;TX DATA4 (ONLY # OF BITS SPECIFIED IN		
6045	033622	000002			TEOM	; R4 WILL GET TRANSMITTED) + SOME OF THE		
6046	033624	000020			16.	; CRC CHARACTER		
6047	033626	004537	011540	JSR	R5, STEPLU	;TX REMAINING CRC CHAR + PUT SOME EXTRA BITS		
6048	033632	000040			32.	;ON THE FIFO		
6049								
6050	033634	004537	010034	JSR	R5, RXCHAR	;READ & CHK 123(DATA1), RCV 321(DATA2)		
6051	033640	000523			RXSOM!123	; & CHECK RSOM=1		
6052	033642	000000			0			
6053	033644	100000			NOCRDA			
6054	033646	103003		BCC	+.8.	;BR IF NO ERROR		
6055	033650			ERROR		;REPORT STACKED ERROR		
6056	033650	104460					TRAP	C\$ERROR
	033652	104410		ESCAPE	SUB	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	033654	000162					.WORD	L10045-.
6057								
6058	033656	004537	010034	JSR	R5, RXCHAR	;READ/CHECK 321(DATA2), RCV 000(DATA3)		
6059	033662	000321			321			
6060	033664	000000			0			
6061	033666	120000			NOCRDA!NCRACT			

TEST 11 -- BOP RX "ABC" TEST

```

6062 033670 103003          BCC      .+8.          ;BR IF NO ERROR
6063 033672          ERROR          ;REPORT STACKED ERROR
        033672 104460
6064 033674          ESCAPE  SUB          ;SKIP TO END OF TEST          TRAP      C$ERROR
        033674 104410          ;                                TRAP      C$ESCAPE
        033676 000140          ;                                .WORD    L10045-.
6065
6066 033700 004537 010034    JSR      R5,RXCHAR      ;READ/CHECK 000(DATA3),RCV DATA4
6067 033704 000000          000
6068 033706 000000          0
6069 033710 120000          NOCRDA!NCRACT
6070 033712 103003          BCC      .+8.          ;BR IF NO ERROR
6071 033714          ERROR          ;REPORT STACKED ERROR          TRAP      C$ERROR
        033714 104460          ;                                TRAP      C$ESCAPE
6072 033716          ESCAPE  SUB          ;SKIP TO END OF TEST          .WORD    L10045-.
        033716 104410
        033720 000116
6073
6074 033722 004537 003534    JSR      R5,READI      ;GET READ STATUS REGISTER
6075 033726 120401          RDSRH
6076 033730 000000          000          ;** HOLE FOR RDSRH VALUE **
6077 033732 042737 177617 033730 20$:    BIC      #177617,20$   ;MASK "ABC" VALUE
6078 033740 006237 033730          ASR      20$          ; AND RIGHT JUSTIFY IT
6079 033744 006237 033730          ASR      20$
6080 033750 006237 033730          ASR      20$
6081 033754 006237 033730          ASR      20$
6082 033760 020437 033730          CMP      R4,20$       ;IS ASSEMBLED BIT COUNT CORRECT ?
6083 033764 001413          BEQ      31$
6084
6085          ;-----
6086          ; ERROR REPORTING GOES HERE
6087 033766 010437 002330          MOV      R4,GDATA     ;EXPECTED BIT COUNT
6088 033772 013737 033730 002332    MOV      20$,BDATA    ;ACTUAL (ERRONEOUS) BIT COUNT
6089 034000          GEDF      EM105,ERR22
        ; "DEVICE FATAL" ERROR # 50
        ;                                TRAP      C$ERDF
        ;                                .WORD    50
        ;                                .WORD    EM105
        ;                                .WORD    ERR22
6090 034010          ESCAPE  SUB          ;                                TRAP      C$ESCAPE
        034010 104410          ;                                .WORD    L10045-.
        034012 000024
6091
6092 034014 004537 010034    31$:    JSR      R5,RXCHAR      ;READ/CHECK DATA4 (SHORT CHARACTER)
6093 034020 000001          001          ;** HOLE FOR DATA4 VALUE **
6094 034022 000000          0
6095 034024 060000          NCRACT!NFCRDA
6096 034026 103003          BCC      .+8.          ;DON'T CHECK RECEIVER ACTIVE/FINAL RDA.
6097 034030          ERROR          ;BR IF NO ERROR
        034030 104460          ;REPORT STACKED ERROR          TRAP      C$ERROR
6098 034032          ESCAPE  SUB          ;SKIP TO END OF TEST          TRAP      C$ESCAPE
        034032 104410          ;                                TRAP      C$ESCAPE
        034034 000002          ;                                .WORD    L10045-.
6099 034036          ENDSUB
        034036          ;                                L10045:
        034036 104403          ;                                TRAP      C$ESUB
6100 034040 005204          INC      R4          ;BUMP GENERAL PURPOSE INDEX

```

TEST 11 -- BOP RX "ABC" TEST

6101 034042 020427 000010
 6102 034046 001402
 6103 034050 000137 033430
 6104 034054
 034054
 034054 104401
 6105
 6106 034056 377
 6107 034057 001
 6108 034060 003
 6109 034061 007
 6110 034062 017
 6111 034063 037
 6112 034064 077
 6113 034065 177
 6114
 6115 034066 000
 6116 034067 040
 6117 034070 100
 6118 034071 140
 6119 034072 200
 6120 034073 240
 6121 034074 300
 6122 034075 340
 6123

ENDTST
 CMP R4,#8.
 BEQ .+6
 JMP T9.LP
 ;ARE WE DONE WITH THIS TEST ?
 ;EXIT IF YES
 ;OTHERWISE DO THE NEXT COUNT

L10044: TRAP C\$ETST

```

;-----
TABLR: .BYTE 377
       .BYTE 001
       .BYTE 003
       .BYTE 007
       .BYTE 017
       .BYTE 037
       .BYTE 077
       .BYTE 177
;-----
LNTBL: .BYTE 000
       .BYTE 040
       .BYTE 100
       .BYTE 140
       .BYTE 200
       .BYTE 240
       .BYTE 300
       .BYTE 340
;-----

```

HARDWARE PARAMETER CODING SECTION

.SBTTL HARDWARE PARAMETER CODING SECTION

6125
6126
6127
6128
6129
6130
6131
6132
6133
6134
6135
6136
6137

:/ THE HARDWARE 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.
:////

6138 034076 000015
034076
034100

BGNHRD

.WORD L10046-L\$HARD/2
L\$HARD::

6139 034100
6140 034100 000031
034102 034132
034104 160020
034106 177776

GPRMA ADDRES.0.0.160020.177776.YES

.WORD T\$CODE
.WORD ADDRES
.WORD T\$LLOLIM
.WORD T\$HILIM

6141 034110
034110 001031
034112 034160
034114 000000
034116 000674

GPRMA VECTOR.2.0.0.674.YES

.WORD T\$CODE
.WORD VECTOR
.WORD T\$LLOLIM
.WORD T\$HILIM

6142 034120
034120 002032
034122 034211
034124 007000
034126 000004
034130 000007

GPRMD PRIRTY.4.0.7000.4.7.YES

.WORD T\$CODE
.WORD PRIRTY
.WORD 7000
.WORD T\$LLOLIM
.WORD T\$HILIM

6143
6144 034132

ENDHRD

.EVEN
L10046:

6145
6146
6147 034132 104 105 126
6148 034160 104 105 126
6149 034211 104 105 126
6150
6151

.NLIST BEX
ADDRES: .ASCIZ /DEVICE CSR ADDRESS : /
VECTOR: .ASCIZ /DEVICE VECTOR ADDRESS : /
PRIRTY: .ASCIZ /DEVICE PRIORITY LEVEL : /
.LIST BEX
.EVEN

SOFTWARE PARAMETER CODING SECTION

6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165 034242
034242 000000
034244
6166
6167 034244
034244

.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 L10047-L$SOFT/2
L$SOFT::

```

ENDSFT

```

.EVEN
L10047:

```

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

6169
6170
6171 034244
6172 034344 034344
6173 034344 000240
6174 034346 000240
6175 034350 000240
6176
6177
6178
6179
6180 034352
6181
6182
6183 034352

034352 000000
034354 000000
034356
6184
6185 000001

.SBTTL ***** PATCH AREA FOR DEBUG *****

PATCH:

.....100
NOP
NOP
NOP

.SBTTL "ENDMOD" STATEMENT

ENDMOD

.SBTTL "LASTAD" STATEMENT & END OF PROGRAM
LASTAD

.EVEN
.WORD 0
.WORD 0

L\$LAST::

.END

SYMBOL TABLE

ABC	=	000160	BSR1	002210	C#ETST=	000001	EM25	014347	EM92	016366
ADDRES		034132	BSR10	002226	C#EXIT=	000032	EM26	014375	ENDEMB	011564
ADPAT		030224	BSR11	002230	C#GETB=	000026	EM27	014427	ENDIT	024212
ADR	=	000020	BSR12	002232	C#GETW=	000027	EM28	014460	ENDPAT	003031
AD.HIT		024332	BSR13	002234	C#GMAN=	000043	EM29	014501	ENDTRN	011456
AD.OK		024326	BSR14	002236	C#GPHR=	000042	EM3	014115	ERRBLK	002204 G
APA	=	000200	BSR15	002240	C#GPLD=	000030	EM30	014516	ERRFLG	002354
APAD	=	100000	BSR16	002242	C#GPRI=	000040	EM31	014537	ERRMSG	002202 G
ASSEMB=		000010	BSR17	002244	C#INIT=	000011	EM32	014554	ERRNBR	002200 G
BADDAT		002372	BSR2	002212	C#INLP=	000020	EM33	014603	ERROR1	002402
BADATA		002332	BSR3	002214	C#MANI=	000050	EM34	014626	ERRTYP	002176 G
BDRATE		002500	BSR4	002216	C#MEM =	000031	EM35	014654	ERR10	021540 G
BIT0	=	000001	BSR5	002220	C#MSG =	000023	EM36	014675	ERR11\$	023122
BIT00	=	000001	BSR6	002222	C#OPEN=	000034	EM39	014712	ERR12	021714 G
BIT01	=	000002	BSR7	002224	C#PNTB=	000014	EM4	014141	ERR12\$	023454
BIT02	=	000004	CARRIER=	000100	C#PNTF=	000017	EM40	014734	ERR13	022030 G
BIT03	=	000010	CA1CTL=	000001	C#PNTS=	000016	EM41	014752	ERR20	022146 G
BIT04	=	000020	CA2CTL=	000016	C#PNTX=	000015	EM42	014773	ERR21	022204 G
BIT05	=	000040	CB1CTL=	000020	C#QIO =	000377	EM43	015010	ERR22	022264 G
BIT06	=	000100	CB2CTL=	000340	C#RDBU=	000007	EM44	015035	ERR4	021274 G
BIT07	=	000200	CHKTS0	007042	C#REFG=	000047	EM45	015062	ERR4\$	022376
BIT08	=	000400	CHPTYP	002404	C#RESE=	000033	EM47	015107	ERR5\$	022724
BIT09	=	001000	CKRACT	005622	C#REVI=	000003	EM48	015142	ERR7A	021420 G
BIT1	=	000002	CKRDA	006122	C#RFLA=	000021	EM49	015175	EVL	= 000004 G
BIT10	=	002000	CKROR	006422	C#RPT =	000025	EM5	014156	EVRC	= 002400
BIT11	=	004000	CKRSA	006262	C#SEFG=	000046	EM50	015234	EXADD	= 000020
BIT12	=	010000	CKSEOM	006562	C#SPRI=	000041	EM51	015270	EXCON	= 000010
BIT13	=	020000	CKTACT	005462	C#SVEC=	000037	EM54	015330	EXECUT=	000005
BIT14	=	040000	CKTBMT	005762	C#TPRI=	000013	EM6	014206	E#END =	002100
BIT15	=	100000	CKUSTS	005356	DDCMP =	040000	EM60	015352	E#LOAD=	000035
BIT2	=	000004	CRCOS =	000400	DEVMAP	002410	EM65	015366	FLGCA1=	000002
BIT3	=	000010	CRC16 =	001400	DEVPTR	002412	EM66	015406	FLGCA2=	000001
BIT4	=	000020	CTS =	000010	DFPTBL	002154 G	EM67	015425	FLGCB1=	000020
BIT5	=	000040	C#AU =	000052	DIAGMC=	000000	EM68	015500	FLGCB2=	000010
BIT6	=	000100	C#AUTO=	000061	DOTBMT=	000007	EM69	015527	FLGIRQ=	000200
BIT7	=	000200	C#BRK =	000022	DTR =	000020	EM70	015545	FLGSR =	000004
BIT8	=	000400	C#BSEG=	000004	DTRL =	000000	EM71	015567	FLGT1 =	000100
BIT9	=	001000	C#BSUB=	000002	D.BUG =	000000	EM72	015605	FLGT2 =	000040
BOE	=	000400	C#CEFG=	000045	EF.CON=	000036 G	EM73	015627	FMT10	012226
BRDTYP		002474	C#CLCK=	000062	EF.NEW=	000035 G	EM74	015644	FMT10A	012302
BSEL0		002422	C#CLEA=	000012	EF.PWR=	000034 G	EM75	015665	FMT11	012323
BSEL1		002424	C#CLOS=	000035	EF.RES=	000037 G	EM76	015701	FMT12	012342
BSEL10		002442	C#CLP1=	000006	EF.STA=	000040 G	EM77	015721	FMT13	012351
BSEL11		002444	C#CVEC=	000036	EIAV35=	000002	EM78	015735	FMT14	012417
BSEL12		002446	C#DCLN=	000044	EM1	014007	EM79	015755	FMT15	012434
BSEL13		002450	C#DODU=	000051	EM100	016422	EM80	016011	FMT15A	012466
BSEL14		002452	C#DRPT=	000024	EM101	016442	EM81	016031	FMT16	012540
BSEL15		002454	C#DU =	000053	EM102	016466	EM82	016055	FMT16A	012563
BSEL16		002456	C#EDIT=	000003	EM103	016530	EM83	016112	FMT17	012625
BSEL17		002460	C#ERDF=	000055	EM104	016607	EM84	016141	FMT17A	012704
BSEL2		002426	C#ERHR=	000056	EM105	016665	EM85	016156	FMT17B	012765
BSEL3		002430	C#ERRO=	000060	EM106	016735	EM86	016172	FMT17C	013040
BSEL4		002432	C#ERSF=	000054	EM13	014236	EM87	016213	FMT19	013120
BSEL5		002434	C#ERSO=	000057	EM14	014265	EM88	016233	FMT2	011574
BSEL6		002436	C#ESCA=	000010	EM15	014301	EM89	016256	FMT21	013151
BSEL7		002440	C#ESEG=	000005	EM16	014327	EM90	016300	FMT22	013161
BSRO		002206	C#ESUB=	000003	EM2	014046	EM91	016331	FMT23	013203

SYMBOL TABLE

FMT24	013262	G\$OFFS=	000400	LUSWI1	002470	L10001	002176	NORXEN=	040000
FMT25	013275	G\$OFISI=	000376	LUSWI2	002472	L10002	021416	NULCLK=	000200
FMT26	013325	G\$PRMA=	000001	LU1MOD	002000 G	L10003	021536	NWSAR	027546
FMT27	013360	G\$PRMD=	000002	L\$ACP	002110 G	L10004	021712	OLOOP	027526
FMT28	013367	G\$PRML=	000000	L\$APT	002036 G	L10005	022026	OVRC	= 002000
FMT29	013423	G\$RADA=	000140	L\$AU	024346 G	L10006	022144	O\$APTS=	000000
FMT3	011631	G\$RADB=	000000	L\$AUT	002070 G	L10007	022202	O\$AU	= 000001
FMT30	013430	G\$RADD=	000040	L\$AUTO	024214 G	L10010	022262	O\$BGNR=	000000
FMT31	013465	G\$RADL=	000120	L\$CCP	002106 G	L10011	022350	O\$BGNS=	000000
FMT32	013533	G\$RADO=	000020	L\$CLEA	024340 G	L10013	024212	O\$DU	= 000001
FMT39	013565	G\$XFER=	000004	L\$CO	002032 G	L10014	024330	O\$ERRT=	000001
FMT4	011715	G\$YES	= 000010	L\$DEPO	002011 G	L10015	024340	O\$GNSW=	000000
FMT4A	011753	HDX	= 000004	L\$DESC	003252 G	L10016	024344	O\$POIN=	000001
FMT4B	012006	HELP	= 000000	L\$DESP	002076 G	L10017	024346	O\$SETU=	000000
FMT4C	012013	HOE	= 100000 G	L\$DEVP	002060 G	L10020	025254	PALENB=	000001
FMT40	013616	IBE	= 010000 G	L\$DISP	002124 G	L10021	024704	PATCH	034244
FMT5	012046	IDLE	= 000010	L\$DLY	002116 G	L10022	025252	PATE	002602
FMT5A	012111	IDLES	= 004000	L\$DTP	002040 G	L10023	026066	PATF	002612
FMT50	013662	IDU	= 000040 G	L\$DTYP	002034 G	L10024	025560	PATG	002622
FMT51	013725	IEI	= 000001	L\$DU	024342 G	L10025	026064	PATI	002676
FMT7	012176	IEO	= 000020	L\$DUT	002072 G	L10026	026660	PATJ	002737
FRSTIM	002374	IER	= 020000 G	L\$DVTY	003232 G	L10027	026372	PATK	002747
F\$AU	= 000015	ILOOP	027542	L\$EF	002052 G	L10030	026656	PATL	002770
F\$AUTO=	000020	INIDMV	005344	L\$ENVI	002044 G	L10031	027516	PATQ	003011
F\$BGN	= 000040	INITRN	007324	L\$ERRT	002176 G	L10032	030222	PATQB	003021
F\$CLEA=	000007	INITT1	004502	L\$ETP	002102 G	L10033	030220	PATX1	002651
F\$DU	= 000016	INITT2	004702	L\$EXP1	002046 G	L10034	030726	PBLENB=	000002
F\$END	= 000041	INTFLG	002352	L\$EXP4	002064 G	L10035	030724	PCR	= 120407
F\$HARD=	000004	INTGRL=	000001	L\$EXP5	002066 G	L10036	032072	PCSARH=	120405
F\$HW	= 000013	INTSC	= 000200	L\$HARD	034100 G	L10037	032712	PCSARL=	120404
F\$INIT=	000006	ISR	= 000100 G	L\$HIME	002120 G	L10040	032420	PNT	= 001000 G
F\$JMP	= 000050	IXE	= 004000 G	L\$HPCP	002016 G	L10041	032710	PRESET=	000001
F\$MOD	= 000000	I\$AU	= 000041	L\$HPTP	002022 G	L10042	033154	PRI	= 002000 G
F\$MSG	= 000011	I\$AUTO=	000041	L\$HW	002154 G	L10043	033416	PRIOR	002346
F\$PROT=	000021	I\$CLN	= 000041	L\$ICP	002104 G	L10044	034054	PRIPTY	034211
F\$PWR	= 000017	I\$DU	= 000041	L\$INIT	023644 G	L10045	034036	PRI00	= 000000 G
F\$RPT	= 000012	I\$HRD	= 000041	L\$LADP	002026 G	L10046	034132	PRI01	= 000040 G
F\$SEG	= 000003	I\$INIT=	000041	L\$LAST	034356 G	L10047	034244	PRI02	= 000100 G
F\$SOFT=	000005	I\$MOD	= 000041	L\$LOAD	002100 G	MCLR	= 000100	PRI03	= 000140 G
F\$SRV	= 000010	I\$MSG	= 000041	L\$LUN	002074 G	MDMRDY=	000040	PRI04	= 000200 G
F\$SUB	= 000002	I\$PROT=	000040	L\$MREV	002050 G	MLWRI	003670	PRI05	= 000240 G
F\$SW	= 000014	I\$PTAB=	000041	L\$NAME	002000 G	MPCSR	002422	PRI06	= 000300 G
F\$TEST=	000001	I\$PWR	= 000041	L\$PRIO	002042 G	MPIVEC	002462	PRI07	= 000340 G
GDATA	002330	I\$RPT	= 000041	L\$PROT	023636 G	MPOVEC	002464	PROTO	= 000100
GETBSR	003772	I\$SEG	= 000041	L\$PRT	002112 G	MPRIOR	002466	PSTACK	002344
GETPRM	024056	I\$SETU=	000041	L\$REPP	002062 G	MRDY	= 000200	RABGA	= 000004
GETURS	004326	I\$SFT	= 000041	L\$REV	002010 G	MREQ	= 000001	RAMADR=	001000
GETVRS	004426	I\$SRV	= 000041	L\$SOFT	034244 G	MSTCLR	003320	RCVBUF	003032
GETWSR	004134	I\$SUB	= 000041	L\$SPC	002056 G	NCRACT=	020000	RCVDAT=	000002
GOODAT	002370	I\$TST	= 000041	L\$SPCP	002020 G	NCTBMT=	000200	RCV1ST	011310
G\$CNTD=	000200	J\$JMP	= 000167	L\$SPTP	002024 G	NEWLIN	011571	RDA	= 000200
G\$DELM=	000372	LNTBL	034066	L\$STA	002030 G	NEWST	024036	RDSRH	= 120401
G\$DISP=	000003	LOADAT	002366	L\$SW	002176 G	NFCRDA=	040000	RDSRL	= 120400
G\$EXCP=	000400	LOE	= 040000 G	L\$TEST	002114 G	NOCHK	= 003400	READ	003422
G\$HILI=	000002	LOGDEV	002340	L\$TIML	002014 G	NOCRDA=	100000	READI	003534
G\$LOLI=	000001	LOOP	030240	L\$UNIT	002012 G	NOLoop=	001000	REDBYT	002362
G\$NO	= 000000	LOT	= 000010 G	L10000	002174	NOLP	030724	REDDAT	002522

SYMBOL TABLE

REDLOC = 000001	STRTML = 000301	TXTNP 020703	TXT3 017207	T1MODE = 000300
REDPAG = 000003	STUREG 004216	TXTNP1 020710	TXT4 017237	T1.1 024350
REGNUM 002342	SUBRPC 002350	TXTNP0 020720	TXT4A 017277	T1.2 024706
REG0 002532	SVCGBL = 000000	TXTNP1 020720	TXT5 017340	T10 033156 G
REG1 002534	SVCINS = 000001	TXTNP2 020730	TXT6 017342	T11 033420 G
REG2 002536	SVCSUB = 000001	TXTNP3 020740	TXT7 017365	T11.1 033430
REG3 002540	SVCTAG = 000001	TXTNP4 020755	TXT7A 017455	T2 025256 G
REG4 002542	SVCTST = 000001	TXTNP5 020772	TXT8 017545	T2MODE = 000040
REG5 002544	SWPBT = 121000	TXTNP6 021007	TXT9 017565	T2.1 025256
REG6 002546	SWPDDC = 121400	TXTNP7 021023	TXU = 000002	T2.2 025562
REG7 002550	SYNCH = 000226	TXTNP8 021037	T\$ARGC = 000005	T3 026070 G
REOM = 000002	S\$LSYM = 010000	TXTNUL 020355	T\$CODE = 002032	T3.1 026070
RERCHK = 000001	TAB = 000004	TXTUR 021053	T\$ERRN = 000062	T3.2 026374
RERR = 000200	TABLR 034056	TXTURT 021254	T\$EXCP = 000000	T4 026672 G
RETADR 002360	TBMT = 000100	TXTUR0 021066	T\$FLAG = 000040	T5 027520 G
RING = 000200	TCCHEK = 100000	TXTUR1 021074	T\$GMAN = 000000	T5.1 027526
ROR = 000010	TDATA 002326	TXTUR2 021102	T\$HILI = 000007	T6 030232 G
RSA = 000020	TDSRH = 120403	TXTUR3 021110	T\$LAST = 000001	T6.1 030240
RSON = 000001	TDSRL = 120402	TXTUR4 021116	T\$LOLI = 000004	T7 030730 G
RSTCHK 005052	TDSRNR 002601	TXTUR5 021125	T\$LSYM = 010000	T8 032074 G
RTSND = 000010	TEOM = 000002	TXTUR6 021134	T\$LTNO = 000013	T8.1 032074
RUN = 000200	TERR = 000200	TXTUR7 021140	T\$NEST = 177777	T8.2 032422
RXABGA = 002000	TGA = 000010	TXTVR 020556	T\$NS0 = 000000	T9 032714 G
RXACT = 000040	TIMFLG 002356	TXTVRA 020654	T\$NS1 = 000005	T9.LP 033430
RXCHAR 010034	TM = 000004	TXTVRB 020657	T\$NS2 = 000002	UAM = 000200 G
RXDL = 000007	TMP0 002552	TXTVRC 020663	T\$NS3 = 000003	UMAIN = 000001
RXEN = 000100	TMP1 002554	TXTVRD 020667	T\$PTNU = 000000	UNIT 002414
RXEOM = 001000	TMP2 002556	TXTVRE 020673	T\$SAVL = 177777	UPBITS 002572
RXERR = 100000	TMP3 002560	TXTVRF 020677	T\$SEGL = 177777	UREGS 002246
RXOR = 004000	TMP4 002562	TXTVRT 021170	T\$SEKO = 010000	USTATR = 122000
RXSOM = 000400	TMP5 002564	TXTVR0 020574	T\$SUBN = 000001	USYREG 002502
SAVE4 002376	TMP6 002566	TXTVR1 020600	T\$TAGL = 177777	USYRT = 120400
SAVE6 002400	TMP7 002570	TXTVR2 020604	T\$TAGN = 010050	VECTOR 034160
SAVLEN 002406	TSO = 000010	TXTVR3 020611	T\$TEMP = 000000	VIA = 120000
SCRACH 002336	TSOM = 000001	TXTVR4 020616	T\$TEST = 000013	VIAACR = 120013
SECAD = 000020	TSTCON 002476	TXTVR5 020623	T\$TSTM = 177777	VIADPA = 120003
SECADR = 010000	TSTNUM 002420	TXTVR6 020630	T\$TSTS = 000001	VIADPB = 120002
SEL0 002422	TTLOOP = 000002	TXTVR7 020635	T\$AU = 010017	VIAIER = 120016
SEL10 002442	TXAB = 002000	TXTVR8 020642	T\$AUT = 010014	VIAIFR = 120015
SEL12 002446	TXACT = 000004	TXTVR9 020647	T\$CLE = 010015	VIAIMS = 120001
SEL14 002452	TXCHAR 007622	TXT1 017005	T\$DU = 010016	VIAORA = 120017
SEL16 002456	TXCTRL 007734	TXT10 017605	T\$HAR = 010046	VIAORB = 120000
SEL2 002426	TXDL = 000340	TXT11 017625	T\$HW = 010000	VIAPCR = 120014
SEL4 002432	TXEN = 000040	TXT11A 017677	T\$INI = 010013	VIASR = 120012
SEL6 002436	TXEOM = 001000	TXT11B 017735	T\$MSG = 010011	VIAT1A = 120004
SERIAL 007202	TXERR = 100000	TXT12 020005	T\$PRO = 010012	VIAT1B = 120005
SETVIA 005252	TXGA = 004000	TXT13 020033	T\$SEG = 010000	VIAT1C = 120006
SFPTBL 002176 G	TXSOM = 000400	TXT14 020050	T\$SOF = 010047	VIAT1D = 120007
SFR = 000001	TXTMLT 021146	TXT15 020106	T\$SUB = 010045	VIAT2A = 120010
SPEED = 000020	TXTML0 020357	TXT16 020150	T\$SW = 010001	VIAT2B = 120011
SRMODE = 000034	TXTML1 020363	TXT17 020163	T\$TES = 010044	VREGS 002266
STALL 004324	TXTML2 020377	TXT18 020220	T.EDF = 000001	WAIT50 005236
STARES 002416	TXTML3 020414	TXT19 020261	T.EHRD = 000002	WRIBYT 002364
STARST 024026	TXTML4 020436	TXT2 017043	T.ESF = 000000	WRILOC = 000002
STEPLU 011540	TXTML5 020457	TXT2A 017105	T.ESFT = 000003	WRIPAG = 000004
STRIP = 000040	TXTML6 020507	TXT2B 017144	T01TBL 026662	WRITE 003646
STRIPS = 020000	TXTML7 020521	TXT20 020315	T1 024350 G	WRITEI 003660

SYMBOL TABLE

WSRO	002206	WSR16	002224	XDATA	002334	X\$FALS=	000040	\$LSTIN=	000001
WSR10	002216	WSR2	002210	XORGB	022352	X\$OFFS=	000400	\$LSTTA=	000001
WSR12	002220	WSR4	002212	XYZ	= 000007	X\$TRUE=	000020	\$T	= 000013
WSR14	002222	WSR6	002214	X\$ALWA=	000000	\$E	= 000062		

. ABS. 034356 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 31136 WORDS (122 PAGES)
DYNAMIC MEMORY: 19748 WORDS (75 PAGES)
ELAPSED TIME: 00:08:30
CNDMDA.BIC,CNDMDA.SEQ/CR/-SP=SVC34.MLB/ML,CNDMDA.P11