

4
5 .TITLE CNDMEAO DMV11 LINE UNIT DIAG3
6 .SBTTL PROGRAM DOCUMENT
7 .REM @

8
9 IDENTIFICATION

10
11 PRODUCT CODE: AC T834A MC
12
13 PRODUCT NAME: CNDMEAO DMV 11 LINE UNIT STATIC DIAGNOSTIC PART 03
14
15 PRODUCT DATE: APRIL 1984
16
17 MAINTAINER: ISS DIAGNOSTICS
18
19 AUTHORS: CHRIS BRIENEN
20 DAVE HOFFMAN
21 RAY MARSHALL
22
23
24 MODIFIED BY: JAKI BERG 9 APR 1984
25
26 PURPOSE: THIS DIAGNOSTIC IS DESIGNED TO PERFORM STATIC LOGIC TESTS FOR
27 THE M8053 OR M8064 (HEREAFTER REFERRED TO AS THE DMV OR DMV-11)
28
29

30 THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
31 NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
32 EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO
33 RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.
34

35 NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF
36 SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS
37 AFFILIATED COMPANIES.
38

39 COPYRIGHT (C) 1984 BY DIGITAL EQUIPMENT CORPORATION

40 THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

41
42
43 DIGITAL PDP UNIBUS MASSBUS
44 DEC DECUS DECTAPE
45

PROGRAM DOCUMENT

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

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

REV A: ORIGINAL RELEASE	BRIENEN, HOFFMAN, MARSHALL	14 JAN-81
REV B: INSTALLED OUTSTANDING PATCHES		11 JUL-83
CVDMEB -> CNDMEA	JAKI BERG	9-APR-84
CHANGES WERE MADE TO CVDMEB TO PRODUCE CNDMEA 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

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

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

110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166

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, USYRT (BCP/BOP MODES), AND LINE DRIVERS ON THESE BOARDS. NOTE THAT ALL EXTERNAL LOOPBACK (XLB) TESTS ARE CONTAINED HERE. THE FOLLOWING FUNCTIONS WILL BE PERFORMED: MODEM LOOPBACK AND ASSORTED EXTERNAL LOOPBACK TESTS (INCLUDING BCP:CRC 16/ODD VRC/EVEN VRC; BOP:CRC-CCITT-1'S/O'S).

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

THE FOLLOWING HARDWARE IS REQUIRED TO FULLY TEST THE DMV 11 LINE DRIVERS:

M3254, M3255 LOOPBACK CONNECTORS

3.0 PRELIMINARY PROGRAM REQUIREMENTS

PROGRAM DOCUMENT

167
168

THIS PROGRAM (CNDME) SHOULD BE THE LAST OF THE FIVE DMV 11
STATIC DIAGNOSTICS TO BE RUN.

PROGRAM DOCUMENT

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

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 DIAGNOSTIC SUPERVISOR

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

4.2 EXECUTION TIME

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

4.3 XXDP.

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

4.4 ACT/SLIDE

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

4.5 APT

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

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM.

4.7 ERROR LOGGING

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

5.0 PROGRAM LOAD MEDIA

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

PROGRAM DOCUMENT

227 THE DIAGNOSTIC PROGRAM.
228
229
230 6.0 OPERATING INSTRUCTIONS
231
232
233 6.1 LOADING AND STARTING PROCEDURES
234
235
236 6.1.1 LOADING PROCEDURES
237
238 THIS PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE
239 ABSOLUTE LOADER. IT MAY ALSO BE LOADED FROM ANY XXDP. LOAD
240 MEDIA. WHEN LOADED UNDER XXDP., THE DIAGNOSTIC SUPERVISOR
241 WILL BE LOADED AUTOMATICALLY.
242
243
244 6.1.2 STARTING PROCEDURES
245
246 THE PROGRAM STARTS AT LOCATION 200. USE STANDARD DEC
247 PROCEDURES TO START THE PROGRAM.
248
249
250 6.1.3 STEPS FOR QUICK AND SIMPLE EXECUTION
251
252 THE DIAGNOSTIC CAN BE EXECUTED STANDALONE UNDER XXDP.,
253 WITHOUT READING THE REMAINDER OF THIS DOCUMENT, AS FOLLOWS:
254
255 A) LOAD AND START DIAGNOSTIC USING RUN COMMAND
256 B) RECEIVE DIAGNOSTIC SUPERVISOR IDENTIFICATION AND PROMPT (DRS C>)
257 C) ENTER STA<CR>
258 D) ANSWER HARDWARE AND SOFTWARE QUESTIONS
259 E) GET END OF PASS MESSAGES OR ERROR MESSAGES
260 F) TO END EXECUTION, ENTER CONTROL/C
261
262
263 6.2 INITIAL DIALOGUE
264
265 AFTER THE PROGRAM AND THE SUPERVISOR ARE LOADED AND THE PROGRAM
266 IS STARTED, THE FOLLOWING IDENTIFICATION IS TYPED :
267
268 DRS LOADED
269 DIAG. RUN TIME SERVICES
270 CNDME A-0
271 DMV-11 LINE UNIT TESTS PART 3 OF 3
272 UNIT IS M8053 OR M8064
273 DR>
274
275 THE OPERATOR THEN PROCEEDS BY TYPING ONE OR MORE OF THE
276 COMMANDS DESCRIBED IN THE FOLLOWING SECTION 6.3. (FOR MORE
277 DETAILED INFORMATION, REFER TO THE DIAGNOSTIC SUPERVISOR
278 FUNCTIONAL SPECIFICATION).
279
280
281 6.3 PROGRAM OPTIONS
282
283

PROGRAM DOCUMENT

284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340

6.3.1 START COMMAND

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 SFLECTED TESTS) AGAINST ALL UNITS SUBMITTED. THE DEFAULT IS NON ENDING EXECUTION. IN THIS CASE EXIT FROM THE PROGRAM IS ACCOMPLISHED EITHER BY TYPING A CONTROL/C OR BY OCCURANCE OF AN ERROR WITH THE HALT ON ERROR FLAG BEING SET. THE EXIT IS A RETURN TO COMMAND MODE. SEE EXAMPLE AT END OF 6.3.1.5.

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

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

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

PROGRAM DOCUMENT

341
342
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
395
396
397

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 "N 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 "N 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

398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454

THREE LETTERS ARE SCANNED.

6.3.2 RESTART COMMAND

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 DIAGLOGUE. THE UNITS WHICH ARE
SELECTED MUST NOT HAVE BEEN DROPPED BY THE DROP COMMAND.
SEE THE DISCUSSION OF ADD AND DROP COMMANDS BELOW. DEFAULT
IS TO TEST ALL UNITS WHICH HAVE NOT BEEN DROPPED BY A DROP
COMMAND.

6.3.2.3 EFFECT OF RESTART COMMAND

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

6.3.3 CONTINUE COMMAND

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

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

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

PROGRAM DOCUMENT

455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511

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

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

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

6.3.3.3 EFFECT OF CONTINUE COMMAND

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

6.3.4 PROCEED COMMAND

PRO(CEED)/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

M1

PROGRAM DOCUMENT

512 HARDWARE DIALOGUE. THIS COMMAND MUST BE FOLLOWED BY A
513 RESTART OR CONTINUE. THE UNITS SWITCH MUST BE SPECIFIED.
514 THE ADD COMMAND IS MEANINGFUL ONLY FOR UNITS THAT WERE
515 PREVIOUSLY DROPPED.
516

517
518 6.3.6 DROP COMMAND

519 *****
520 DRO(P)/UNITS:<UNIT-LIST>
521 *****
522

523
524 6.3.6.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

525 <UNIT-LIST> IS AS IN THE RESTART COMMAND.
526

527
528 6.3.6.2 EFFECT OF DROP COMMAND

529 THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS
530 WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START
531 COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND
532 MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.
533

534
535 6.3.7 PRINT COMMAND

536 *****
537 PRI(NT)
538 *****
539

540
541 6.3.7.1 EFFECT OF PRINT COMMAND

542 THE TOTAL NUMBER OF ERRORS FOR EACH UNIT SINCE THE LAST
543 START OR RESTART COMMAND ARE PRINTED. THE ISR (INHIBIT
544 STATISTICAL REPORTING) FLAG IS CLEARED.
545

546
547 6.3.8 DISPLAY COMMAND

548 *****
549 DIS(PLAY)/UNITS:<UNIT-LIST>
550 *****
551

552
553 6.3.8.1 UNITS SWITCH (/UNITS:<UNIT LIST>)

554 <UNIT-LIST> IS AS IN THE RESTART COMMAND.
555

556
557 6.3.8.2 EFFECT OF DISPLAY COMMAND

558 THE HARDWARE P-TABLES FOR ALL UNITS UNDER TEST ARE PRINTED
559 OUT IN THE FORMAT IN WHICH THEY WERE ENTERED. ANY UNITS
560 THAT WERE DROPPED BY THE OPERATOR 'DROP' COMMAND ARE SO
561
562
563
564
565
566
567
568

PROGRAM DOCUMENT

569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625

DESIGNATED.

6.3.9 FLAGS COMMAND

FLA(GS)

6.3.9.1 EFFECT OF FLAGS COMMAND

THE CURRENT SETTINGS OF ALL FLAGS ARE PRINTED.

6.3.10 ZFLAGS COMMAND

ZFL(AGS)

6.3.10.1 EFFECT OF ZFLAGS COMMAND

ALL FLAGS ARE CLEARED.

6.3.11 CONTROL CHARACTERS

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

A CONTROL Z (Z) ENTERED DURING ONE OF THE THREE OPERATOR DIALOGUES- HARD CORE QUESTIONS (SEE 6.?), 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

626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682

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

3. DEVICE PRIORITY LEVEL : (0) 4 ?

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

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

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

5. TURNAROUND CONNECTOR TYPE -
(0=M3254&M3255, 1-INTEGRAL MODEM CABLE, 2=EIA CABLE,
3=V.35 CABLE, 4=NONE) : (0) 0 ?

THIS IS THE TYPE OF EXTERNAL LOOPBACK CONNECTOR BEING USED. IF NO LOOPBACK CONNECTOR IS PRESENT (4), THE EXTERNAL LOOPBACK TESTS WILL ALL BE RUN USING TTL INTERNAL LOOPBACK.

6.3.13 SOFTWARE PARAMETERS

NO SOFTWARE PARAMETER QUESTIONS ARE ASKED BY THIS TEST.

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.

PROGRAM DOCUMENT

683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739

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

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

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

THE FOLLOWING DIALOGUE WOULD ACCOMPLISH THIS GOAL:

```
# UNITS (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

741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797

7.0 TEST DESCRIPTIONS

: * TEST 1 <RX DATA FLUSHING TEST>
: *
: * IN BCP MODE/HALF DUPLEX IT IS DESIRABLE TO HAVE THE ABILITY TO FLUSH
: * THE USYRT OF ITS CRC CHARACTERS. THIS FLUSHING IS ACCOMPLISHED BY WRITING
: * TO THE VIA SHIFT REGISTER.
: * THIS TEST VERIFIES THAT WHEN THE VIA SR IS WRITTEN INTO, 8 PULSES WILL
: * BE GENERATED AT THE CB1 PIN (WHICH DIRECTLY FEEDS THE CHARACTER FIFO).
: *

: * TEST 2 <INTEGRAL MODEM INTERFACE TEST>
: *
: * THE INTEGRAL MODEM IS SELECTED BY THE PROGRAM AND A MESSAGE IS
: * TRANSMITTED, RECEIVED, AND CHECKED USING A TURNAROUND CONNECTOR ON
: * THE BOARD OR AT THE END OF A CABLE. THE FOLLOWING MESSAGE WILL BE
: * SENT IN BCP MODE WITH CRC 16 SPECIFIED:
: *
: * SYNC SYNC 000 125 252 377 000 CRC1 CRC2 SYNC
: *
: * IF THE P-TABLE FOR THE CURRENT UNIT INDICATES THAT NO EXTERNAL
: * TURNAROUND IS PROVIDED, THE TEST WILL BE SKIPPED FOR THAT UNIT.

: * TEST 3 <DATA TEST -- BCP XLB CRC 16>
: *
: * IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
: * RECEIVE IN BCP MODE WITH CRC 16 ERROR DETECTION THE FOLLOWING
: * MESSAGE:
: *
: * 125 252 000 377 001 002 004 010 020 040 100 200 376 375 373 367
: * 357 337 277 177
: *
: * THIS MESSAGE WILL BE PRECEDED BY 3 SYNC CHARACTERS AND REPEATED
: * THREE TIMES WITH CRC'S FOLLOWING EACH ONE. THE LAST TRANSMISSION OF
: * THE CRC WILL BE FOLLOWED BY SEVERAL SYNC CHARACTERS BEFORE DROPPING
: * TXE & RXE. 8 BIT CHARACTER LENGTHS ARE ALSO UTILIZED.
: *
: * IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
: * USING INTERNAL LOOPBACK (ITLOOP=1).

: * TEST 4 <DATA TEST - BCP XLB ODD VRC>
: *
: * IF XLB IS SPECIFIED IN THE P TABLE, THIS TEST WILL TRANSMIT &
: * RECEIVE IN BCP MODE WITH ODD VRC ERROR DETECTION THE FOLLOWING

PROGRAM DOCUMENT

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
841
842
843
844
845
846
847
848
849
850
851
852
853
854

```

: * MESSAGE:
: *
: *   125 252 000 377 001 002 004 010 020 040 100 200 376 375 373 367
: *   357 337 277 177
: *
: * THIS MESSAGE WILL BE PRECEDED BY 3 SYNC CHARACTERS AND REPEATED
: * THREE TIMES. AFTER THE LAST MESSAGE, SEVERAL SYNC CHARACTERS ARE
: * SENT BEFORE DROPIING TXE & RXE. 7-BIT CHARACTER LENGTHS ARE ALSO
: * UTILIZED.
: *
: * IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
: * USING INTERNAL LOOPBACK (TTLOOP=1).
: *
: *
: *
: *
: * *****
: * TEST 5 <DATA TEST -- BCP XLB EVEN VRC>
: *
: * IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
: * RECEIVE IN BCP MODE WITH EVEN VRC ERROR DETECTION THE FOLLOWING
: * MESSAGE:
: *
: *   125 252 000 377 001 002 004 010 020 040 100 200 376 375 373 367
: *   357 337 277 177
: *
: * THIS MESSAGE WILL BE PRECEDED BY 3 SYNC CHARACTERS AND REPEATED
: * THREE TIMES. AFTER THE LAST MESSAGE, SEVERAL SYNC CHARACTERS ARE
: * SENT BEFORE DROPIING TXE & RXE. 7-BIT CHARACTER LENGTHS ARE ALSO
: * UTILIZED.
: *
: * IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
: * USING INTERNAL LOOPBACK (TTLOOP=1).
: *
: *
: *
: *
: * *****
: * TEST 6 <DATA TEST -- BOP XLB CRC-CCITT-1>
: *
: * IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
: * RECEIVE IN BOP MODE WITH CRC-CCITT-1 ERROR DETECTION THE FOLLOWING
: * SHORT MESSAGE: 125 252 000 377 001
: *
: * THIS MESSAGE WILL BE PRECEDED BY FLAG CHARACTERS AND REPEATED
: * THREE TIMES WITH CRC AND FLAG'S FOLLOWING EACH ONE. 8-BIT CHARACTER
: * LENGTHS ARE ALSO UTILIZED.
: *
: * IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
: * USING INTERNAL LOOPBACK (TTLOOP=1).
: *
: *
: *
: *
: * *****
: * TEST 7 <DATA TEST -- BOP XLB CRC-CCITT-0>
: *
: * IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
: * RECEIVE IN BOP MODE WITH CRC CCITT-0 ERROR DETECTION THE FOLLOWING
: * SHORT MESSAGE: 125 252 000 377 001

```

PROGRAM DOCUMENT

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
898
899
900
901
902
903
904
905
906
907
908
909

```

;*
;* THIS MESSAGE WILL BE PRECEDED BY FLAG CHARACTERS AND REPEATED
;* THREE TIMES WITH CRC AND FLAG'S FOLLOWING EACH ONE. 8-BIT CHARACTER
;* LENGTHS ARE ALSO UTILIZED.
;*
;* IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
;* USING INTERNAL LOOPBACK (TTLOOP=1).
;*****
;*****
;* TEST 8 <MODEM CONTROL SIGNAL LOOPBACK TEST>
;*
;* FIRST, THE DMV-11 IS INITIALIZED. THEN, TTL LOOPBACK IS SELECTED,
;* AND THE FOLLOWING CHECKS ARE PERFORMED INVOLVING THE MODEM STATUS
;* REGISTER :
;* - RING, CARRIER, MODEM READY, TEST MODE, CTS ARE CHECKED FOR 1 STATE.
;* - RTS IS DE-ASSERTED AND CTS IS CHECKED FOR 0.
;* - RTS IS ASSERTED AND CTS IS CHECKED FOR 1.
;*
;* NEXT, IF THE OPTION IS AN M8053 WITH AN M3254 TEST CONNECTOR INSTALLED,
;* THE DMV-11 IN INITIALIZED AGAIN, (TTL LOOPBACK IS CLEARED), AND
;* THE FOLLOWING CHECKS ARE PERFORMED :
;* - RING, CARRIER, MODEM READY, CTS ARE CHECKED FOR 1, TEST MODE IS CHECKED
;* FOR 0.
;* - RTS IS DE-ASSERTED, AND CARRIER AND CTS ARE CHECKED FOR 0.
;* - RTS IS ASSERTED, AND CARRIER AND CTS ARE CHECKED FOR 1.
;* - DTR IS DE-ASSERTED, AND MODEM READY IS CHECKED FOR 0.
;* - DTR IS ASSERTED, AND MODEM READY IS CHECKED FOR 1.
;*****
;*****
;* TEST 9 <DDCMP MESSAGE TEST>
;*
;* THIS TEST WILL USE XLB IF IT IS ENABLED - OTHERWISE TTL LOOPBACK
;* WILL BE UTILIZED. THIS ASSURES THAT IT CAN ALWAYS BE RUN AS A
;* GENERAL "RINGOUT" OF THE M8053.
;*
;* INITIALIZATION: BCP MODE, CRC 16, IDLE = 0, SYNC (S/AR) = 226 OCT.
;* (96 HEX.), RXCL & TXCL = 0 (CHAR. LENGTH = 8).
;*
;* THE FOLLOWING SAMPLE DDCMP MESSAGE IS TRANSMITTED & RECEIVED AND ALL
;* DATA AND CRC CHARACTERS ARE CHECKED FOR ERRORS:
;*
;* ----- HEADER ----- --- DATA (PATTERN K) -----
;* SYNC SYNC 201 000 075 003 002 001 CRC CRC 000 377 ... 25? 000 CRC CRC
;*
;* THE ATTEMPT HERE IS TO PROVIDE A TEST JUST BELOW THE LEVEL OF THE
;* FUNCTIONAL DIAGNOSTIC. THE USYRT WILL BE RESPONSIBLE FOR ALL CRC
;* GENERATION AND VERIFICATION BUT THE CRC'S WILL ALSO BE VERIFIED BY
;* SOFTWARE.
;*****

```

PROGRAM DOCUMENT

911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952

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 P. 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

```

954          .SBTTL GENERAL EQUATES AND DS INVOCATION & SETUP
955
956
957          000000      HELP=0          ; CONTROL LISTING OF HELP INFORMATION
958                                     ;
959                                     ; HELP=0   NO LIST
960                                     ; HELP=1   LIST
961
962          002000      .=-2000
963
964          .MCALL      SVC
965          002000      SVC          ; INITIALIZE SUPERVISOR MACROS
966
967
968          002000      BGNMOD      LU1MOD
969
970          000001      $LSTIN= 1
971          000001      $LSTTAG= 1
972          000001      SVCINS= 1      ; LIST INSTRUCTIONS, SHIFTED RIGHT
973          000001      SVCTST= 1     ; LIST TEST TAGS, SHIFTED RIGHT
974          000001      SVCSUB= 1     ; LIST SUBTEST TAGS, SHIFTED RIGHT
975          000001      SVCGBL= 1     ; LIST GLOBAL TAGS, SHIFTED RIGHT
976          000001      SVCTAG= 1    ; LIST OTHER TAGS, SHIFTED RIGHT
977
978          ;          CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
979          ;          TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
980          ;          SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
981          ;          CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
982
983
984
985
986
987

```

PROGRAM HEADER

```

989      .SBTTL  PROGRAM HEADER
990      ;**
991      ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
992      ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
993      ;--
994
995 002000      POINTER BGNAU,BGNDU,ERRTBL
996
1004
1005 002000      HEADER  CNDME,A,0,15.,0

```

```

002000      103
002001      116
002002      104
002003      115
002004      105
002005      000
002006      000
002007      000
002010
002010      101
002011
002011      060
002012
002012      000000
002014
002014      000017
002016
002016      033700
002020
002020      000000
002022
002022      002150
002024
002024      000000
002026
002026      034542
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

```

```

L$NAME::
          .ASCII /C/
          .ASCII /N/
          .ASCII /D/
          .ASCII /M/
          .ASCII /E/
          .BYTE  0
          .BYTE  0
          .BYTE  0
L$REV::
          .ASCII /A/
L$DEPO::
          .ASCII /O/
L$UNIT::
          .WORD  0
L$TIML::
          .WORD  15.
L$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  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 003264
 002062
 002062 000000
 002064
 002064 000000
 002066
 002066 000000
 002070
 002070 023256
 002072
 002072 023252
 002074
 002074 000000
 002076
 002076 003304
 002100
 002100 104035
 002102
 002102 002172
 002104
 002104 022554
 002106
 002106 023250
 002110
 002110 023124
 002112
 002112 022546
 002114
 002114 000000
 002116
 002116 000000
 002120
 002120 000000

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

1006
 1012
 1013

.EVEN

DISPATCH TABLE

1015
1016
1017 002122

1018
1019
1020 002122

1021
1022 002122 000011
002124
002124 023260
002126 023560
002130 026264
002132 027142
002134 027704
002136 030446
002140 031130
002142 031612
002144 032556

1023

.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
;;//

DISPATCH 9.

.WORD 9
L\$DISPATCH:;
.WORD T1
.WORD T2
.WORD T3
.WORD T4
.WORD T5
.WORD T6
.WORD T7
.WORD T8
.WORD T9

DEFAULT HARDWARE P-TABLE

```

1031          .SBTTL  DEFAULT HARDWARE P-TABLE
1032
1033          ;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
1034          ;/ THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
1035          ;/ THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
1036          ;/ IS IDENTICAL TO THE STRUCTURE OF THE RUN TIME P-TABLE.
1037          ;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
1038
1039          002146          BGNHW  DFPTBL
1040          002146          000010          .WORD  L10000-L$HW/2
1041          002150          160020          L$HW::
1042          002152          000300          DFPTBL::
1043          002154          004000          ;DMV11 CSR UNIBUS ADDRESS
1044          002156          000000          ;DMV11 INTERRUPT VECTOR
1045          002160          000000          ;DMV11 INTERRUPT PRIORITY LEVEL = 4
1046          002162          000000          ;SWITCH REG. #1 (BOOT ADDRESS)
1047          002164          000000          ;SWITCH REG. #2 (DDCMP ADDRESS)
1048          002166          000001          ;MODULE IS M8064
1049          ;
1050          ;
1051          ;
1052          002170          ENDPHW          ;H3254&H3255 USED
1053          002170          ;BAUD RATE = 56 K
1054          ;
1055          ;
1056          ;
1057          ;
1058          ;
1059          ;
1060          ;
1061          ;
1062          ;
1063          ;
1064          ;
1065          ;
1066          ;
1067          ;
1068          ;
1069          ;
1070          ;
1071          ;
1072          ;
1073          ;
1074          ;
1075          ;
1076          ;
1077          ;
1078          ;
1079          ;
1080          ;
1081          ;
1082          ;
1083          ;
1084          ;
1085          ;
1086          ;
1087          ;
1088          ;
1089          ;
1090          ;
1091          ;
1092          ;
1093          ;
1094          ;
1095          ;
1096          ;
1097          ;
1098          ;
1099          ;
1100          ;
1101          ;
1102          ;
1103          ;
1104          ;
1105          ;
1106          ;
1107          ;
1108          ;
1109          ;
1110          ;
1111          ;
1112          ;
1113          ;
1114          ;
1115          ;
1116          ;
1117          ;
1118          ;
1119          ;
1120          ;
1121          ;
1122          ;
1123          ;
1124          ;
1125          ;
1126          ;
1127          ;
1128          ;
1129          ;
1130          ;
1131          ;
1132          ;
1133          ;
1134          ;
1135          ;
1136          ;
1137          ;
1138          ;
1139          ;
1140          ;
1141          ;
1142          ;
1143          ;
1144          ;
1145          ;
1146          ;
1147          ;
1148          ;
1149          ;
1150          ;
1151          ;
1152          ;
1153          ;
1154          ;
1155          ;
1156          ;
1157          ;
1158          ;
1159          ;
1160          ;
1161          ;
1162          ;
1163          ;
1164          ;
1165          ;
1166          ;
1167          ;
1168          ;
1169          ;
1170          ;
1171          ;
1172          ;
1173          ;
1174          ;
1175          ;
1176          ;
1177          ;
1178          ;
1179          ;
1180          ;
1181          ;
1182          ;
1183          ;
1184          ;
1185          ;
1186          ;
1187          ;
1188          ;
1189          ;
1190          ;
1191          ;
1192          ;
1193          ;
1194          ;
1195          ;
1196          ;
1197          ;
1198          ;
1199          ;
1200          ;
1201          ;
1202          ;
1203          ;
1204          ;
1205          ;
1206          ;
1207          ;
1208          ;
1209          ;
1210          ;
1211          ;
1212          ;
1213          ;
1214          ;
1215          ;
1216          ;
1217          ;
1218          ;
1219          ;
1220          ;
1221          ;
1222          ;
1223          ;
1224          ;
1225          ;
1226          ;
1227          ;
1228          ;
1229          ;
1230          ;
1231          ;
1232          ;
1233          ;
1234          ;
1235          ;
1236          ;
1237          ;
1238          ;
1239          ;
1240          ;
1241          ;
1242          ;
1243          ;
1244          ;
1245          ;
1246          ;
1247          ;
1248          ;
1249          ;
1250          ;
1251          ;
1252          ;
1253          ;
1254          ;
1255          ;
1256          ;
1257          ;
1258          ;
1259          ;
1260          ;
1261          ;
1262          ;
1263          ;
1264          ;
1265          ;
1266          ;
1267          ;
1268          ;
1269          ;
1270          ;
1271          ;
1272          ;
1273          ;
1274          ;
1275          ;
1276          ;
1277          ;
1278          ;
1279          ;
1280          ;
1281          ;
1282          ;
1283          ;
1284          ;
1285          ;
1286          ;
1287          ;
1288          ;
1289          ;
1290          ;
1291          ;
1292          ;
1293          ;
1294          ;
1295          ;
1296          ;
1297          ;
1298          ;
1299          ;
1300          ;
1301          ;
1302          ;
1303          ;
1304          ;
1305          ;
1306          ;
1307          ;
1308          ;
1309          ;
1310          ;
1311          ;
1312          ;
1313          ;
1314          ;
1315          ;
1316          ;
1317          ;
1318          ;
1319          ;
1320          ;
1321          ;
1322          ;
1323          ;
1324          ;
1325          ;
1326          ;
1327          ;
1328          ;
1329          ;
1330          ;
1331          ;
1332          ;
1333          ;
1334          ;
1335          ;
1336          ;
1337          ;
1338          ;
1339          ;
1340          ;
1341          ;
1342          ;
1343          ;
1344          ;
1345          ;
1346          ;
1347          ;
1348          ;
1349          ;
1350          ;
1351          ;
1352          ;
1353          ;
1354          ;
1355          ;
1356          ;
1357          ;
1358          ;
1359          ;
1360          ;
1361          ;
1362          ;
1363          ;
1364          ;
1365          ;
1366          ;
1367          ;
1368          ;
1369          ;
1370          ;
1371          ;
1372          ;
1373          ;
1374          ;
1375          ;
1376          ;
1377          ;
1378          ;
1379          ;
1380          ;
1381          ;
1382          ;
1383          ;
1384          ;
1385          ;
1386          ;
1387          ;
1388          ;
1389          ;
1390          ;
1391          ;
1392          ;
1393          ;
1394          ;
1395          ;
1396          ;
1397          ;
1398          ;
1399          ;
1400          ;
1401          ;
1402          ;
1403          ;
1404          ;
1405          ;
1406          ;
1407          ;
1408          ;
1409          ;
1410          ;
1411          ;
1412          ;
1413          ;
1414          ;
1415          ;
1416          ;
1417          ;
1418          ;
1419          ;
1420          ;
1421          ;
1422          ;
1423          ;
1424          ;
1425          ;
1426          ;
1427          ;
1428          ;
1429          ;
1430          ;
1431          ;
1432          ;
1433          ;
1434          ;
1435          ;
1436          ;
1437          ;
1438          ;
1439          ;
1440          ;
1441          ;
1442          ;
1443          ;
1444          ;
1445          ;
1446          ;
1447          ;
1448          ;
1449          ;
1450          ;
1451          ;
1452          ;
1453          ;
1454          ;
1455          ;
1456          ;
1457          ;
1458          ;
1459          ;
1460          ;
1461          ;
1462          ;
1463          ;
1464          ;
1465          ;
1466          ;
1467          ;
1468          ;
1469          ;
1470          ;
1471          ;
1472          ;
1473          ;
1474          ;
1475          ;
1476          ;
1477          ;
1478          ;
1479          ;
1480          ;
1481          ;
1482          ;
1483          ;
1484          ;
1485          ;
1486          ;
1487          ;
1488          ;
1489          ;
1490          ;
1491          ;
1492          ;
1493          ;
1494          ;
1495          ;
1496          ;
1497          ;
1498          ;
1499          ;
1500          ;
1501          ;
1502          ;
1503          ;
1504          ;
1505          ;
1506          ;
1507          ;
1508          ;
1509          ;
1510          ;
1511          ;
1512          ;
1513          ;
1514          ;
1515          ;
1516          ;
1517          ;
1518          ;
1519          ;
1520          ;
1521          ;
1522          ;
1523          ;
1524          ;
1525          ;
1526          ;
1527          ;
1528          ;
1529          ;
1530          ;
1531          ;
1532          ;
1533          ;
1534          ;
1535          ;
1536          ;
1537          ;
1538          ;
1539          ;
1540          ;
1541          ;
1542          ;
1543          ;
1544          ;
1545          ;
1546          ;
1547          ;
1548          ;
1549          ;
1550          ;
1551          ;
1552          ;
1553          ;
1554          ;
1555          ;
1556          ;
1557          ;
1558          ;
1559          ;
1560          ;
1561          ;
1562          ;
1563          ;
1564          ;
1565          ;
1566          ;
1567          ;
1568          ;
1569          ;
1570          ;
1571          ;
1572          ;
1573          ;
1574          ;
1575          ;
1576          ;
1577          ;
1578          ;
1579          ;
1580          ;
1581          ;
1582          ;
1583          ;
1584          ;
1585          ;
1586          ;
1587          ;
1588          ;
1589          ;
1590          ;
1591          ;
1592          ;
1593          ;
1594          ;
1595          ;
1596          ;
1597          ;
1598          ;
1599          ;
1600          ;
1601          ;
1602          ;
1603          ;
1604          ;
1605          ;
1606          ;
1607          ;
1608          ;
1609          ;
1610          ;
1611          ;
1612          ;
1613          ;
1614          ;
1615          ;
1616          ;
1617          ;
1618          ;
1619          ;
1620          ;
1621          ;
1622          ;
1623          ;
1624          ;
1625          ;
1626          ;
1627          ;
1628          ;
1629          ;
1630          ;
1631          ;
1632          ;
1633          ;
1634          ;
1635          ;
1636          ;
1637          ;
1638          ;
1639          ;
1640          ;
1641          ;
1642          ;
1643          ;
1644          ;
1645          ;
1646          ;
1647          ;
1648          ;
1649          ;
1650          ;
1651          ;
1652          ;
1653          ;
1654          ;
1655          ;
1656          ;
1657          ;
1658          ;
1659          ;
1660          ;
1661          ;
1662          ;
1663          ;
1664          ;
1665          ;
1666          ;
1667          ;
1668          ;
1669          ;
1670          ;
1671          ;
1672          ;
1673          ;
1674          ;
1675          ;
1676          ;
1677          ;
1678          ;
1679          ;
1680          ;
1681          ;
1682          ;
1683          ;
1684          ;
1685          ;
1686          ;
1687          ;
1688          ;
1689          ;
1690          ;
1691          ;
1692          ;
1693          ;
1694          ;
1695          ;
1696          ;
1697          ;
1698          ;
1699          ;
1700          ;
1701          ;
1702          ;
1703          ;
1704          ;
1705          ;
1706          ;
1707          ;
1708          ;
1709          ;
1710          ;
1711          ;
1712          ;
1713          ;
1714          ;
1715          ;
1716          ;
1717          ;
1718          ;
1719          ;
1720          ;
1721          ;
1722          ;
1723          ;
1724          ;
1725          ;
1726          ;
1727          ;
1728          ;
1729          ;
1730          ;
1731          ;
1732          ;
1733          ;
1734          ;
1735          ;
1736          ;
1737          ;
1738          ;
1739          ;
1740          ;
1741          ;
1742          ;
1743          ;
1744          ;
1745          ;
1746          ;
1747          ;
1748          ;
1749          ;
1750          ;
1751          ;
1752          ;
1753          ;
1754          ;
1755          ;
1756          ;
1757          ;
1758          ;
1759          ;
1760          ;
1761          ;
1762          ;
1763          ;
1764          ;
1765          ;
1766          ;
1767          ;
1768          ;
1769          ;
1770          ;
1771          ;
1772          ;
1773          ;
1774          ;
1775          ;
1776          ;
1777          ;
1778          ;
1779          ;
1780          ;
1781          ;
1782          ;
1783          ;
1784          ;
1785          ;
1786          ;
1787          ;
1788          ;
1789          ;
1790          ;
1791          ;
1792          ;
1793          ;
1794          ;
1795          ;
1796          ;
1797          ;
1798          ;
1799          ;
1800          ;
1801          ;
1802          ;
1803          ;
1804          ;
1805          ;
1806          ;
1807          ;
1808          ;
1809          ;
1810          ;
1811          ;
1812          ;
1813          ;
1814          ;
1815          ;
1816          ;
1817          ;
1818          ;
1819          ;
1820          ;
1821          ;
1822          ;
1823          ;
1824          ;
1825          ;
1826          ;
1827          ;
1828          ;
1829          ;
1830          ;
1831          ;
1832          ;
1833          ;
1834          ;
1835          ;
1836          ;
1837          ;
1838          ;
1839          ;
1840          ;
1841          ;
1842          ;
1843          ;
1844          ;
1845          ;
1846          ;
1847          ;
1848          ;
1849          ;
1850          ;
1851          ;
1852          ;
1853          ;
1854          ;
1855          ;
1856          ;
1857          ;
1858          ;
1859          ;
1860          ;
1861          ;
1862          ;
1863          ;
1864          ;
1865          ;
1866          ;
1867          ;
1868          ;
1869          ;
1870          ;
1871          ;
1872          ;
1873          ;
1874          ;
1875          ;
1876          ;
1877          ;
1878          ;
1879          ;
1880          ;
1881          ;
1882          ;
1883          ;
1884          ;
1885          ;
1886          ;
1887          ;
1888          ;
1889          ;
1890          ;
1891          ;
1892          ;
1893          ;
1894          ;
1895          ;
1896          ;
1897          ;
1898          ;
1899          ;
1900          ;
1901          ;
1902          ;
1903          ;
1904          ;
1905          ;
1906          ;
1907          ;
1908          ;
1909          ;
1910          ;
1911          ;
1912          ;
1913          ;
1914          ;
1915          ;
1916          ;
1917          ;
1918          ;
1919          ;
1920          ;
1921          ;
1922          ;
1923          ;
1924          ;
1925          ;
1926          ;
1927          ;
1928          ;
1929          ;
1930          ;
1931          ;
1932          ;
1933          ;
1934          ;
1935          ;
1936          ;
1937          ;
1938          ;
1939          ;
1940          ;
1941          ;
1942          ;
1943          ;
1944          ;
1945          ;
1946          ;
1947          ;
1948          ;
1949          ;
1950          ;
1951          ;
1952          ;
1953          ;
1954          ;
1955          ;
1956          ;
1957          ;
1958          ;
1959          ;
1960          ;
1961          ;
1962          ;
1963          ;
1964          ;
1965          ;
1966          ;
1967          ;
1968          ;
1969          ;
1970          ;
1971          ;
1972          ;
1973          ;
1974          ;
1975          ;
1976          ;
1977          ;
1978          ;
1979          ;
1980          ;
1981          ;
1982          ;
1983          ;
1984          ;
1985          ;
1986          ;
1987          ;
1988          ;
1989          ;
1990          ;
1991          ;
1992          ;
1993          ;
1994          ;
1995          ;
1996          ;
1997          ;
1998          ;
1999          ;
2000          ;

```

SOFTWARE P TABLE

```

1054          .SBTTL  SOFTWARE P-TABLE
1055
1056          ;////////////////////////////////////
1057          ;/ THE SOFTWARE P-TABLE CONTAINS THE VALUES OF THE PROGRAM
1058          ;/ PARAMETERS THAT CAN BE CHANGED BY THE OPERATOR.
1059          ;////////////////////////////////////
1060
1061 002170          BGNSW  SFPTBL
1061 002170 000000          .WORD  L10001-L$SW/2
1061 002172          L$SW::
1061 002172          SFPTBL::
1062
1063 002172          ENDSW
1063 002172          L10001:

```


GLOBAL EQUATES SECTION BASIC EQUATES

1065
1066
1067
1068
1069
1070
1071
1072
1073 002172

.SBTTL GLOBAL EQUATES SECTION - BASIC EQUATES

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

EQUALS

; BIT DEFINITIONS

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

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

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

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

; PRIORITY LEVEL DEFINITIONS

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

GLOBAL EQUATES SECTION BASIC EQUATES

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

REGISTER DEFINITIONS MAINTENANCE REGISTERS -- SELN & BSELN

```

1075 .SBTTL REGISTER DEFINITIONS -- MAINTENANCE REGISTERS -- SELN & BSELN
1076
1077 ;*****
1078 ;* MAINTENANCE REGISTER # 0 - BSEL0
1079 ;*****
1080 IEO = BIT4 ;"INTERRUPT ENABLE OUT"
1081 IEI = BIT0 ;"INTERRUPT ENABLE IN"
1082
1083 ; BIT 7 IS ALSO USED BY THE MICROCODE. ITS LABEL IS "RQI" WHICH STANDS FOR
1084 ; "REQUIST IN". IT'S PART OF THE HANDSHAKING FOR USING THE SEL & BSEL REG'S.
1085 ; HOWEVER, THE MAINT. LOOP DOES NOT MAKE USE OF THIS BIT AND IT IS THEREFORE
1086 ; UNNECESSARY TO DEFINE IT HERE.
1087
1088 ;*****
1089 ;* MAINTENANCE REGISTER # 1 BSEL1
1090 ;*****
1091 RUN = BIT7 ;"RUN" & ALSO CONTROLS 6502 MICROPROCESSOR'S RDY STATE
1092 MCLR = BIT6 ;MASTER CLEAR
1093 MREQ = BIT0 ;M-LOOP ACCESS
1094 STRMLOP= RUN!MCLR!MREQ ;INITIATE M-LOOP
1095
1096 ;*****
1097 ;* MAINTENANCE REGISTER # 2 - BSEL2
1098 ;*****
1099 MRDY = BIT7 ;M-LOOP READY
1100
1101 ;*****
1102 ;* MAINTENANCE LOOP COMMAND DEFINITIONS
1103 ;*****
1104 REDLOC = 1 ;READ LOC. W/IN DMV-11 ---- (SEL4) ---> BSEL6
1105 WRILOC = 2 ;WRITE LOC. W/IN DMV-11 --- BSEL6 ---> (SEL4)
1106 REDPAG = 3 ;READ BLOCK W/IN DMV-11 - - (SEL6) ---> (SEL4)
1107 WRIPAG = 4 ;WRITE BLOCK W/IN DMV-11 -- (SEL4) ---> (SEL6)
1108 EXECUT = 5 ;SET 6502'S PC AND EXECUTE - SEL6 ---> PC
1109 DOTBMT = 7 ;SET MAINTENANCE INTERRUPT DISABLE IN PROCESSOR
1110 ;STATUS - [KB7] --> BSEL3
1111

```

REGISTER DEFINITIONS USYRT

```

1113 .SBTTL REGISTER DEFINITIONS - USYRT
1114
1115
1116 120400 USYRT = 120400 ;USYRT BASE ADDRESS = A100 (HEX)
1117
1118 ;*****
1119 ;* USYRT "RECEIVER DATA BUFFER" REGISTER -- READ ONLY
1120 ;*****
1121
1122 120400 RDSRL = 120400 ;ADDRESS OF THIS REG
1123
1124 ;*****
1125 ;* USYRT "RECEIVER STATUS" REGISTER -- READ ONLY
1126 ;*****
1127
1128 120401 RDSRH = 120401 ;ADDRESS OF THIS REG
1129
1130 ;BIT DEFINITIONS ON BYTE BASIS :
1131 000200 RERR = BIT7 ;ERROR CHECK
1132 000160 ABC = BIT6!BIT5!BIT4 ;ASSEMBLED BIT COUNT
1133 000010 ROR = BIT3 ;RECEIVER OVER RUN
1134 000004 RABGA = BIT2 ;RECEIVED ABORT/GA CHARACTER
1135 000002 REOM = BIT1 ;RECEIVED END-OF-MESSAGE
1136 000001 RSOM = BIT0 ;RECEIVED START-OF-MESSAGE
1137
1138 ;BIT DEFINITIONS ON WORD BASIS :
1139 100000 RXERR = BIT15 ;RECEIVED CRC/VRC ERROR
1140 004000 RXOR = BIT11 ;RECEIVER OVER RUN
1141 002000 RXABGA = BIT10 ;RECEIVED ABORT/GO AHEAD CHARACTER
1142 001000 RXEOM = BIT9 ;RECEIVED END-OF-MESSAGE
1143 000400 RXSOM = BIT8 ;RECEIVED START-OF-MESSAGE
1144
1145 000001 RERCHK = BIT0 ;FLAG TO INVOKE RERR CHK IN SUBROUTINE RXCHAR
1146
1147 ;*****
1148 ;* USYRT "TRANSMITTER DATA BUFFER" REGISTER
1149 ;*****
1150
1151 120402 TDSRL = 120402 ;ADDRESS OF THIS REG
1152
1153 ;*****
1154 ;* USYRT "TX STATUS AND CONTROL" REGISTER
1155 ;*****
1156
1157 120403 TDSRH = 120403 ;ADDRESS OF THIS REG
1158
1159 ;BIT DEFINITIONS ON BYTE BASIS :
1160 000200 TERR = BIT7 ;TRANSMITTER UNDERRUN ERROR
1161 000010 TGA = BIT3 ;TRANSMIT GO AHEAD
1162 000004 TAB = BIT2 ;TRANSMIT ABORT
1163 000002 TEOM = BIT1 ;TRANSMIT END OF-MESSAGE
1164 000001 TSOM = BIT0 ;TRANSMIT START-OF-MESSAGE
1165
1166 ;BIT DEFINITIONS ON WORD BASIS :
1167 100000 TXERR = BIT15 ;TRANSMITTER UNDERRUN ERROR
1168 004000 TXGA = BIT11 ;TRANSMIT GO AHEAD
1169 002000 TXAB = BIT10 ;TRANSMIT ABORT

```

REGISTER DEFINITIONS USYRT

```

1170      001000      TXEOM   = BIT9           ; TRANSMIT END-OF-MESSAGE
1171      000400      TXSOM   = BIT8           ; TRANSMIT START-OF-MESSAGE
1172
1173      ;*****
1174      ;* USYRT "SYNC/SECONDARY ADDRESS" REGISTER
1175      ;*****
1176
1177      120404      PCSARL  = 120404          ; ADDRESS OF THIS REG
1178      000226      SYNCH   = 226           ; STANDARD SYNCH CHARACTER
1179
1180      ;*****
1181      ;* USYRT "MODE CONTROL"
1182      ;*****
1183
1184      120405      PCSARH  = 120405          ; ADDRESS OF THIS REG
1185
1186      ;BIT DEFINITIONS ON BYTE BASIS:
1187
1188      000200      APA     = BIT7           ; "ALL PARTIES ADDRESS" ENABLE
1189      000100      PROTO  = BIT6           ; SPECIFIES BOP/CCP PROTOCOL      0 = BOP
1190      000040      STRIP  = BIT5           ; STRIP EXTRA SYNC'S IN CCP MODE, SEE GA CHARS IN BOP
1191      000020      SECAD  = BIT4           ; SECONDARY ADDRESS MODE -- BOP MODE ONLY
1192      000010      IDLE   = BIT3           ; IDLE & SYNC CHAR. TRANSMISSION CONTROL
1193      000007      XYZ    = BIT2!BIT1!BIT0 ; CRC/PARITY SELECTION CONTROL
1194
1195      ;BIT DEFINITIONS ON WORD BASIS:
1196
1197      100000      APAD    = BIT15          ; "ALL PARTIES ADDRESS" ENABLE
1198      040000      DDCMP  = BIT14          ; CODE FOR DDCMP MODE
1199      020000      STRIPS  = BIT13          ; STRIP EXTRA SYNC'S IN CCP MODE, SEE GA CHARS IN BOP
1200      010000      SECADR = BIT12          ; SECONDARY ADDRESS MODE -- BOP MODE ONLY
1201      004000      IDLES  = BIT11          ; IDLE & SYNC CHAR. TRANSMISSION CONTROL
1202      000400      CRCOS  = BIT8           ; CODE FOR CRC-CCITT-0 SELECTION
1203      001400      CRC16 = BIT9!BIT8      ; CODE FOR CRC-16 SELECTION
1204      003400      NOCHK  = BIT10!BIT9!BIT8 ; CODE FOR NO ERROR CHECKING
1205      002400      EVRC   = BIT10!BIT8    ; CODE FOR VRC EVEN CHECK
1206      002000      OVRC   = BIT10         ; CODE FOR VRC ODD CHECK
1207
1208      ;*****
1209      ;* USYRT "DATA LENGTH SELECT" REGISTER
1210      ;*****
1211
1212      120407      PCR     = 120407          ; ADDRESS OF THIS REG
1213
1214      ;BIT DEFINITIONS:
1215
1216      000340      TXDL    = BIT7!BIT6!BIT5 ; TRANSMIT DATA LENGTH SELECTION
1217      000020      EXADD  = BIT4           ; EXTENDED ADDRESS FIELD - NOT USED OR TESTED
1218      000010      EXCON  = BIT3           ; EXTENDED CONTROL FIELD -- NOT USED OR TESTED
1219      000007      RXDL   = BIT2!BIT1!BIT0 ; RECEIVER DATA LENGTH SELECTION
1220
1221      ;*****
1222      ;* USYRT STATUS REGISTER (ADDR. A400)
1223      ;*****
1224      122000      USTATR = 122000          ; USYRT STATUS REGISTER ADDRESS = A400 (HEX)
1225
1226      ;BIT DEFINITIONS:

```

REGISTER DEFINITIONS -- USYRT

1227				
1228	000200	RDA	= BIT7	;RECEIVER DATA AVAILABLE
1229	000100	TBMT	= BIT6	;TRANSMITTER BUFFER EMPTY
1230	000040	RXACT	= BIT5	;RECEIVER ACTIVE
1231	000020	RSA	= BIT4	;RECEIVER STATUS AVAILABLE
1232	000010	TSO	= BIT3	;TRANSMITTER SERIAL OUTPUT
1233	000004	TXACT	= BIT2	;TRANSMITTER ACTIVE
1234	000002	TXU	= BIT1	;TRANSMITTER UNDERRUN
1235	000001	SFR	= BIT0	;SYNC/FLAG RECEIVED

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1237      .SBTTL REGISTER DEFINITIONS -- 6522 VIA CHIP
1238
1239      120000 VIA      = 120000      ;VIA BASE ADDRESS = A000 (HEX)
1240
1241      ;*****
1242      ;* MODEM & MAINTENANCE CONTROL -- "ORB" 8 BIT PORT B -- WRITE ONLY
1243      ;*****
1244
1245      120000 VIAORB = 120000      ;ADDRESS OF THIS REGISTER  HEX = A0X0
1246
1247      000200 NULCLK = BIT7      ;"NULL CLK L" -- NULL CLOCK
1248      000100 RXEN   = BIT6      ;"RXENL" -- USYRT RECEIVER ENABLE
1249      000040 TXEN   = BIT5      ;"TXENL" -- USYRT TRANSMITTER ENABLE
1250      000020 DTR    = BIT4      ;"DTR" -- DATA TERMINAL READY
1251      000010 RTSND  = BIT3      ;"RTSND" - REQUEST TO SEND
1252      000004 HDX    = BIT2      ;"HDX" -- HALF DUPLEX
1253      000002 TTLOOP = BIT1      ;"SELECT TTL LEVEL LOOPBACK"
1254      000001 PRESET = BIT0      ;"PRESET H" --
1255      000000 DTRL   = 0          ;DTR IS ASSERTED LOW
1256
1257      ;*****
1258      ;* MODEM STATUS REGISTER -- "ORA" 8 BIT PORT A -- READ ONLY
1259      ;*****
1260
1261      120001 VIAMS  = 120001      ;ADDRESS OF THIS REGISTER -- HEX = A0X1
1262
1263      000200 RING   = BIT7      ;"RING H" --
1264      000100 CARRIER = BIT6      ;"CARRIER H" --
1265      000040 MDMRDY  = BIT5      ;"MODEM RDY H" --
1266      000020 SPEED  = BIT4      ;"BAUD RATE SWITCH - (19.2K/56K)
1267      000010 CTS    = BIT3      ;"CTS H -- CLEAR TO SEND
1268      000004 TM     = BIT2      ;"TEST MODE H" --
1269      000002 RCVDAT = BIT1      ;"RCV DATA H" --
1270      000001 UMAINT = BIT0      ; SELECT USYRT INT LOOPBACK **SELECT BIT**
1271
1272
1273      ;*****
1274      ;* DATA DIRECTION FOR PORT B -- "DORB" -- READ/WRITE
1275      ;*****
1276
1277      120002 VIADPB = 120002      ;ADDRESS OF THIS REGISTER -- HEX = A0X2
1278
1279      ; ALL BITS ARE DEFINED THE SAME:
1280      ; THE BIT SETTING DEFINED THE DIRECTION OF ITS RELATED BIT IN BIT PORT B
1281
1282      ; INITIALIZED TO 377 (HEX = FF) -- PORT B IS READ/WRITE
1283
1284
1285      ;*****
1286      ;* DATA DIRECTION FOR PORT A -- "DDRA" -- READ/WRITE
1287      ;*****
1288
1289      120003 VIADPA = 120003      ;ADDRESS OF THIS REGISTER -- HEX = A0X3
1290
1291      ; ALL BITS ARE DEFINED THE SAME:
1292      ; THE BIT SETTING DEFINED THE DIRECTION OF ITS RELATED BIT IN BIT PORT A
1293

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1294      ;      INITIALIZED TO 001 (HEX = 01) -- PORT A IS READ ONLY (EXCEPT FOR
1295      ;      BIT0 WHICH ENABLES USYRT INTERNAL LOOPBACK).
1296
1297
1298
1299      ;;*****
1300      ;* TIMER 1 LOW ORDER (LATCH & COUNTER) -- "T1L-L" & "T1C-L" -- WRITE & READ
1301      ;;*****
1302
1303      120004      VIAT1A = 120004      ;ADDRESS OF THIS REGISTER      HEX = A0X4
1304
1305      ;      WHEN WRITING, LOW ORDER LATCH IS LOADED.
1306      ;      WHEN READING, LOW ORDER COUNTER IS READ.
1307
1308
1309
1310      ;;*****
1311      ;* TIMER 1 HIGH ORDER COUNTER & TRIGGER -- "T1L-H AND TRIGGER" & "T1C-H"
1312      ;*      -- WRITE & READ
1313      ;;*****
1314
1315      120005      VIAT1B = 120005      ;ADDRESS OF THIS REGISTER -- HEX = A0X5
1316
1317      ;      WHEN WRITING; HIGH ORDER LATCH IS LOADED, BOTH LOW & HIGH ORDER LATCHES
1318      ;      ARE LOADED INTO THE COUNTER, AND THE COUNTER IS STARTED.
1319
1320      ;      WHEN READING, THE HIGH ORDER COUNTER IS READ.
1321
1322
1323
1324      ;;*****
1325      ;* TIMER 1 LOW ORDER LATCH -- "T1L-L" -- READ/WRITE
1326      ;;*****
1327
1328      120006      VIAT1C = 120006      ;ADDRESS OF THIS REGISTER -- HEX = A0X6
1329
1330      ;      THE LOW ORDER LATCH IS READ OR LOADED. THIS LATCH IS USED TO LOAD THE
1331      ;      COUNTER WHEN TIMODE (IN VIAACR) = 3
1332
1333
1334
1335      ;;*****
1336      ;* TIMER 1 HIGH ORDER LATCH -- "T1L-H" -- READ/WRITE
1337      ;;*****
1338
1339      120007      VIAT1D = 120007      ;ADDRESS OF THIS REGISTER -- HEX = A0X7
1340
1341      ;      THE HIGH ORDER LATCH IS READ OR LOADED. THIS LATCH IS USED TO LOAD THE
1342      ;      COUNTER WHEN TIMODE (IN VIAACR) = 3
1343
1344
1345
1346      ;;*****
1347      ;* TIMER 2 LOW ORDER (LATCH & COUNTER) -- "T2L-L" & "T2C-L" -- WRITE & READ
1348      ;;*****
1349
1350      120010      VIAT2A = 120010      ;ADDRESS OF THIS REGISTER -- HEX = A0X8

```


REGISTER DEFINITIONS - 6522 VIA CHIP

```

1351
1352           ; WHEN WRITING, LOW ORDER LATCH IS LOADED.
1353           ; WHEN READING, LOW ORDER COUNTER IS READ.
1354
1355
1356
1357           ;*****
1358           ;* TIMER 2 HIGH ORDER COUNTER & TRIGGER -- "T2L-H AND TRIGGER" & "T2C-H"
1359           ;*   -- WRITE & READ
1360           ;*****
1361
1362           120011      VIAT2B = 120011           ;ADDRESS OF THIS REGISTER -- HEX = A0X9
1363
1364           ; WHEN WRITING; HIGH ORDER LATCH IS LOADED, BOTH LOW & HIGH ORDER LATCHES
1365           ;           ARE LOADED INTO THE COUNTER, AND THE COUNTER IS STARTED.
1366
1367           ; WHEN READING, THE HIGH ORDER COUNTER IS READ.
1368
1369           ;*****
1370           ;* SHIFT REGISTER -- "SR" -- READ/WRITE
1371           ;*****
1372
1373           120012      VIASR  = 120012           ;ADDRESS OF THIS REGISTER -- HEX = A0XA
1374
1375           ; SHIFTING IS CONTROLLED BY THE SETTING OF VIASRC (ACR2 ---> ACR4) IN VIAACR
1376
1377
1378
1379           ;*****
1380           ;* AUXILIARY CONTROL REGISTER - "ACR" - READ/WRITE
1381           ;*****
1382
1383           120013      VIAACR = 120013           ;ADDRESS OF THIS REGISTER - HEX = A0XB
1384
1385           000300      T1MODE = BIT7!BIT6       ;CONTROL THE MODE OF TIMER # 1
1386
1387           ;BIT 7:
1388           ; 0      PB7 DISABLED -- ONLY T1TO IN VIAIFR REFLECTS TIMEOUT
1389           ; 1      PB7 & T1TO REFLECT TIMEOUT
1390
1391           ;BIT 6:
1392           ; 0      TIMER 1 IN ONE-SHOT MODE
1393           ; 1      TIMER 1 IN CONTINUOUS SQUARE WAVE MODE
1394
1395           000040      T2MODE = BIT5           ;CONTROLS THE MODE OF TIMER # 1
1396
1397           ; 0      PULSE COUNTING MODE
1398           ; 1      INTERVAL TIMER MODE
1399
1400           000034      SRMODE = BIT4!BIT3!BIT2 ;CONTROLS THE MODE OF THE SHIFT REGISTER
1401
1402           ; 0      SR DISABLED
1403           ; 1      SHIFT IN UNDER CONTROL OF T2, SHFT PULSES GEN'D ON CB1
1404           ; 2      SHIFT IN AT SYS. CLOCK RATE, SHFT PULSES GEN'D ON CB1
1405           ; 3      SHIFT IN UNDER CONTROL OF EXTERNAL INPUT PULSES
1406           ; 4      SHIFT OUT FREE RUNNING -- RATE CONTROLLED BY T2
1407           ; 5      SHIFT OUT RATE CONTROLLED BY T2 PULSES ON CB1

```

REGISTER DEFINITIONS - 6522 VIA CHIP

```

1408                                     ; 6   SHIFT OUT -- SYS. CLOCK RATE -- PULSES ON CB1
1409                                     ; 7   SHIFT OUT -- UNDER CONTROL OF PULSES APPLIED TO CB1
1410
1411      000002      PBLNB  = BIT1          ;PB LATCH CONTROL -- 1 ENABLES LATCH
1412      000001      PALENB = BIT0          ;PA LATCH CONTROL -- 1 ENABLES LATCH
1413
1414
1415
1416
1417      ;;*****
1418      ;* PERIPHERAL CONTROL REGISTER - "PCR" -- READ/WRITE
1419      ;;*****
1420
1421      120014      VIAPCR = 120014          ;ADDRESS OF THIS REGISTER -- HEX = A0XC
1422
1423      000340      CB2CTL = BIT7!BIT6!BIT5 ;CB2 MODE SELECT
1424      000020      CB1CTL = BIT4          ;CB1 MODE SELECT
1425      000016      CA2CTL = BIT3!BIT2!BIT1 ;CA2 MODE SELECT
1426      000001      CA1CTL = BIT0          ;CA1 MODE SELECT
1427
1428
1429
1430      ;;*****
1431      ;* INTERRUPT FLAG REGISTER - "IFR" -- READ ONLY
1432      ;;*****
1433
1434      120015      VIAIFR = 120015         ;ADDRESS OF THIS REGISTER   HEX = A0XD
1435
1436      000200      FLGIRQ = BIT7          ;SET WHEN A FLAG IN THIS REG. GOES HIGH AND
1437                                     ;ITS CORRESPONDING BIT IN VIAIER IS SET.
1438                                     ;(I.E. VIAIER IS THE ENABLE REGISTER FOR THE
1439                                     ;FOR THE SETTING OF IRQ AND THE ISSUANCE OF
1440                                     ;AN INTERRUPT TO THE 6502 WHEN IRQ IS SET.)
1441
1442      000100      FLGT1  = BIT6          ;TIMEOUT OF TIMER 1
1443      000040      FLGT2  = BIT5          ;TIMEOUT OF TIMER 2
1444      000020      FLGCB1 = BIT4          ;ACTIVE TRANSITION OF PIN 18 (CB1)
1445      000010      FLGCB2 = BIT3          ;ACTIVE TRANSITION OF PIN 19 (CB2)
1446      000004      FLGSR  = BIT2          ;COMPLETION OF 8 SHIFTS
1447      000002      FLGCA1 = BIT1          ;ACTIVE TRANSITION OF PIN 40 (CA1)
1448      000001      FLGCA2 = BIT0          ;ACTIVE TRANSITION OF PIN 39 (CA2)
1449
1450
1451
1452      ;;*****
1453      ;* INTERRUPT ENABLE REGISTER -- "IER" -- READ/WRITE
1454      ;;*****
1455
1456      120016      VIAIER = 120016         ;ADDRESS OF THIS REGISTER - HEX = A0XE
1457
1458      000200      INTSC  = BIT7          ;CONTROLS THE SETTING OR CLEARING OF BITS IN
1459                                     ;THE REST OF IER. IF = 0 THE OTHER BITS IN
1460                                     ;THIS REG., IF SET, WILL CLEAR THEIR RESPECTIVE
1461                                     ;BITS IN THE INT. ENAB. REG., IF = 1, THE
1462                                     ;RESPECTIVE BITS WILL BE SET.
1463
1464      ; WHEN WRITING THIS REG., THE COMMENT ABOVE HOLDS.

```

REGISTER DEFINITIONS -- 6522 VIA CHIP

```

1465
1466           ; WHEN READING THIS REG., THE CURRENT STATE OF THE INT. ENABLE REG. IS RETURNED.
1467
1468           ; THE BIT ASSIGNMENTS ARE THE SAME AS FOR VIAIFR AS DEFINED ABOVE.
1469
1470
1471
1472           ;*****
1473           ;* OUTPUT REGISTER A -- "ORA"   READ ONLY (OR READ/WRITE UNDER CONTROL OF "DDPA")
1474           ;*****
1475
1476           120017 VIAORA = 120017           ;ADDRESS OF THIS REGISTER - HEX = A0XF
1477
1478           ; THIS ADDRESS ACCESSES THE SAME DATA AS "VIAMS" EXCEPT THAT NO "HANDSHAKING"
1479           ; WILL TAKE PLACE (I.E. THERE IS NO CHANGE IN IRQ OR CA2 AS A RESULT OF
1480           ; READING ORA THROUGH THIS ADDRESS)
1481
1482           ;THE BIT ASSIGNMENTS ARE THE SAME AS FOR "VIAMS" ABOVE.
1483
1484
1485

```

REGISTER DEFINITIONS MISC

```

1487      .SBTTL REGISTER DEFINITIONS -- MISC
1488
1489      ;*****
1490      ;* SWITCH PACKS
1491      ;*****
1492
1493      121000 SWPBOT = 121000      ;"BOOT ADDRESS" SWITCH PACK [A200]
1494      121400 SWPDDCMP = 121400  ;"DDCMP ADDRESS" SWITCH PACK [A300]
1495
1496      ;MISCELLANEOUS EQUATES
1497
1498      100000 TCCHEK = BIT15      ;FLAG TO REQUEST H3254,5 CHECK
1499      001000 RAMADR = 001000    ;STARTING ADRS OF RAM PAGE 2 (ADRS 0200 HEX)
1500
1501      000002 EIAV35 = BIT1      ;SELECT V.35 OR EIA 423/232C
1502      000001 INTGRL = BIT0     ;SELECT INTEGRAL MODEM
1503
1504      040000 NORXEN = BIT14     ;KILL RXEN DURING "INITRN"
1505      001000 NOLOOP = BIT9     ;KILL TTLOOP DURING "INITRN"
1506
1507      000200 NCTBMT = BIT7      ;DISABLE INITIAL TBMT=0 CHECK IN TXCHAR
1508
1509      100000 NOCRDA = BIT15     ;DISABLE INITIAL RDA=0 CHECK IN RXCHAR
1510      040000 NFCRDA = BIT14    ;DISABLE FINAL RDA=1 CHECK IN RXCHAR
1511      020000 NCRACT = BIT13    ;DISABLE RXACT=1 CHECK AFTER CLOCKING (RXCHAR)
1512

```

GLOBAL DATA SECTION

1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783

1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816

002172
002172
002172 000000
002174 000000
002176 000000
002200 000000

002202
002202 000000
002204
002204 000000
002206
002206 000000
002210
002210 000000
002212
002212 000000
002214
002214 000000
002216
002216 000000
002220
002220 000000
002222 000000
002224 000000
002226 000000
002230 000000
002232 000000
002234 000000
002236 000000
002240 000000

002242
002262

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

                ERRTBL

                L$ERRTBL::

ERRTYP::        .WORD    0
ERRNBR::        .WORD    0
ERRMSG::        .WORD    0
ERRBLK::        .WORD    0

;*****
; * STORAGE FOR DEVICE REGISTERS
;*****
                ;STORAGE FOR DEVICE CSR REGISTERS
WSR0:
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
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.
                ;THE FIRST 7 ARE FOR THE USYRT'S ACTUAL
                ;REGISTERS. THE LAST ONE IS FOR THE STATUS
                ;REG. (USTATR).
VREGS:         .BLKW  16.
                ;STORAGE FOR VIA REGISTERS FOR PRINTOUT
```

GLOBAL DATA SECTION

```

1818 ;*****
1819 ;* MISCELLANEOUS STORAGE
1820 ;*****
1821 002322 000000 TDATA: .WORD 0 ;TEST DATA
1822 002324 000000 GDATA: .WORD 0 ;GOOD DATA
1823 002326 000000 BDATA: .WORD 0 ;BAD DATA
1824 002330 000000 XDATA: .WORD 0 ;EXCLUSIVE-OR BETWEEN GOOD AND BAD DATA
1825 002332 000000 SCRACH: .WORD 0 ;GEN'L PURPOSE SCRATCH WORD
1826 002334 000000 LOGDEV: .WORD 0 ;LOGICAL DEVICE NUMBER
1827 002336 000000 REGNUM: .WORD 0 ;CONTAINS A DEVICE REGISTER NUMBER
1828 002340 000000 PSTACK: .WORD 0 ;CONTAINS BASE LEVEL PROGRAM STACK POINTER
1829 002342 000000 PRIOR: .WORD 0 ;CPU PRIORITY FOR PRINTOUT
1830 002344 000000 SUBRPC: .WORD 0 ;PC OF SUBR CALL FOR ERROR REPORTS
1831 002346 000000 INTFLG: .WORD 0 ;INTERRUPT RECEIVED FLAGS
1832 ; BIT 0 FOR TX, BIT 1 FOR RCV
1833 002350 000000 ERRFLG: .WORD 0 ;SUBROUTINE ERROR FLAG
1834 002352 000000 TIMFLG: .WORD 0 ;EVENT TIME OUT FLAG
1835 002354 000000 RETADR: .WORD 0 ;SUBR ERROR RETURN ADDRESS
1836 002356 000000 REDBYT: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM LU REG
1837 002360 000000 WRIBYT: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO LU REG
1838 002362 000000 LOADAT: .WORD 0 ;CONTAINS TEST DATA LOADED INTO REG
1839 002364 000000 GOODAT: .WORD 0 ;STORAGE FOR EXPECTED DATA
1840 002366 000000 BADDAT: .WORD 0 ;STORAGE FOR ACTUAL DATA
1841 002370 000000 FRSTIM: .WORD 0 ;FLAG=0 IF PROGRAM JUST LOADED
1842 002372 000000 SAVE4: .WORD 0 ;SAVE LOC 4 HERE (ERROR TRAP VECTOR)
1843 002374 000000 SAVE6: .WORD 0 ;SAVE LOC 6 HERE (ERROR TRAP VECTOR)
1844 002376 000000 ERROR1: .WORD 0 ;SUBR ERR. BIT FLAGS (DEF'D IN GLOBAL EQUATES)
1845 002400 000000 CHPTYP: .WORD 0 ;USYRT CHIP TYPE, =0 FOR SMC, ELSE =1
1846 002402 000000 SAVLEN: .WORD 0 ;SAVED TX AND RCV CHAR LENGTHS
1847 002404 000000 DEVMAP: .WORD 0 ;BIT MAP OF ACTIVE DEVICES
1848 002406 000000 DEVPTR: .WORD 0 ;DEVICE MAP BIT POINTER
1849 002410 000000 UNIT: .WORD 0 ;CONTAINS UNIT NO. (1 TO N)
1850 002412 000000 STARES: .WORD 0 ;FLAG TO SHOW NO. OF PASSES SINCE STA OR RES
1851 002414 000000 TSTNUM: .WORD 0 ;NO. OF CURRENT TEST (FOR SOME TESTS)
1852

```

GLOBAL DATA SECTION

```

1854          ;***** CURRENT DEVICE PARAMETERS *****
1855 002416   BSEL0:
1856 002416   SEL0:
1857 002416   160020   MPCSR: .WORD    160020           ; POINTER TO DMV11 CSR'S
1858 002420   160021   BSEL1: .WORD    160021           ; POINTER TO BSEL1
1859 002422   BSEL2:
1860 002422   160022   SEL2: .WORD    160022           ; POINTER TO SEL2
1861 002424   160023   BSEL3: .WORD    160023           ; POINTER TO BSEL3
1862 002426   BSEL4:
1863 002426   160024   SEL4: .WORD    160024           ; POINTER TO SEL4
1864 002430   160025   BSEL5: .WORD    160025           ; POINTER TO BSEL5
1865 002432   BSEL6:
1866 002432   160026   SEL6: .WORD    160026           ; POINTER TO SEL6
1867 002434   160027   BSEL7: .WORD    160027           ; POINTER TO BSEL7
1868 002436   BSEL10:
1869 002436   160030   SEL10: .WORD   160030           ; POINTER TO SEL10
1870 002440   160031   BSEL11: .WORD   160031           ; POINTER TO BSEL11
1871 002442   BSEL12:
1872 002442   160032   SEL12: .WORD   160032           ; POINTER TO SEL12
1873 002444   160033   BSEL13: .WORD   160033           ; POINTER TO BSEL13
1874 002446   BSEL14:
1875 002446   160034   SEL14: .WORD   160034           ; POINTER TO SEL14
1876 002450   160035   BSEL15: .WORD   160035           ; POINTER TO BSEL15
1877 002452   BSEL16:
1878 002452   160036   SEL16: .WORD   160036           ; POINTER TO SEL16
1879 002454   160037   BSEL17: .WORD   160037           ; POINTER TO BSEL17
1880
1881 002456   000300   MPIVEC: .WORD    300           ; DMV11 INPUT INTERRUPT VECTOR
1882 002460   000304   MPOVEC: .WORD    304           ; DMV11 OUTPUT INTERRUPT VECTOR
1883 002462   000240   MPRIOR: .WORD    240           ; DMV11 DEVICE PRIORITY
1884 002464   000000   LUSWI1: .WORD     0           ; LINE UNIT SWITCH PACK #1
1885 002466   000000   LUSWI2: .WORD     0           ; LINE UNIT SWITCH PACK #2
1886 002470   000000   BRDTP: .WORD     0           ; 0=M8064, 1=M8053/V.35, 2=M8053/EIA
1887 002472   000000   TSTCON: .WORD     0           ; TEST CONNECTOR INDICATOR
1888 002474   000001   BDRATE: .WORD     1           ; BAUD RATE = 56 K
1889          ;           0 = 19.2 K
1890          ;           1 = 56 K

```

GLOBAL DATA SECTION

```

1892
1893 002476 120400
1894 002500 120401
1895 002502 120402
1896 002504 120403
1897 002506 120404
1898 002510 120405
1899 002512 120407
1900 002514 122000
1901
1902
1903 002516
1904
1905
1906 002526 000000
1907 002530 000000
1908 002532 000000
1909 002534 000000
1910 002536 000000
1911 002540 000000
1912 002542 000000
1913 002544 000000
1914
1915
1916 002546 000000
1917 002550 000000
1918 002552 000000
1919 002554 000000
1920 002556 000000
1921 002560 000000
1922 002562 000000
1923 002564 000000
1924
1925
1926 002566
1927 002566 377
1928 002567 000
1929 002570 000
1930 002571 360
1931 002572 000
1932 002573 000
1933 002574 347
1934
1935 002575 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

1937			.SBTTL DATA TEST PATTERNS
1938			;***** DATA PATTERN E *****
1939	002576		PATE:
1940	002576	377	.BYTE 377
1941	002577	377	.BYTE 377
1942	002600	377	.BYTE 377
1943	002601	377	.BYTE 377
1944	002602	377	.BYTE 377
1945	002603	377	.BYTE 377
1946	002604	377	.BYTE 377
1947	002605	366	.BYTE 366
1948			
1949			;***** DATA PATTERN F *****
1950	002606		PATF:
1951	002606	000	.BYTE 000
1952	002607	000	.BYTE 000
1953	002610	000	.BYTE 000
1954	002611	000	.BYTE 000
1955	002612	000	.BYTE 000
1956	002613	000	.BYTE 000
1957	002614	000	.BYTE 000
1958	002615	110	.BYTE 110
1959			
1960			;***** DATA PATTERN G *****
1961	002616		PATG:
1962	002616	000	.BYTE 000
1963	002617	001	.BYTE 001
1964	002620	003	.BYTE 003
1965	002621	004	.BYTE 004
1966	002622	005	.BYTE 005
1967	002623	007	.BYTE 007
1968	002624	100	.BYTE 100
1969	002625	101	.BYTE 101
1970	002626	103	.BYTE 103
1971	002627	104	.BYTE 104
1972	002630	105	.BYTE 105
1973	002631	107	.BYTE 107
1974	002632	000	.BYTE 000
1975	002633	017	.BYTE 017
1976	002634	027	.BYTE 027
1977	002635	041	.BYTE 041
1978	002636	200	.BYTE 200
1979	002637	277	.BYTE 277
1980	002640	103	.BYTE 103
1981	002641	144	.BYTE 144
1982	002642	115	.BYTE 115
1983	002643	157	.BYTE 157
1984	002644	000	.BYTE 000
1985			
1986			;***** DATA PATTERN X *****
1987	002645		PATX:
1988	002645	125	.BYTE 125
1989	002646	252	.BYTE 252
1990	002647	000	.BYTE 000
1991	002650	377	.BYTE 377
1992	002651	001	.BYTE 001
1993	002652	002	.BYTE 002

DATA TEST PATTERNS

1994	002653	004	.BYTE	004
1995	002654	010	.BYTE	010
1996	002655	020	.BYTE	020
1997	002656	040	.BYTE	040
1998	002657	100	.BYTE	100
1999	002660	200	.BYTE	200
2000	002661	376	.BYTE	376
2001	002662	375	.BYTE	375
2002	002663	373	.BYTE	373
2003	002664	367	.BYTE	367
2004	002665	357	.BYTE	357
2005	002666	337	.BYTE	337
2006	002667	277	.BYTE	277
2007	002670	177	.BYTE	177
2008				
2009	002671	125	.BYTE	125
2010	002672	252	.BYTE	252
2011	002673	000	.BYTE	000
2012	002674	377	.BYTE	377
2013	002675	001	.BYTE	001
2014	002676	002	.BYTE	002
2015	002677	004	.BYTE	004
2016	002700	010	.BYTE	010
2017	002701	020	.BYTE	020
2018	002702	040	.BYTE	040
2019	002703	100	.BYTE	100
2020	002704	200	.BYTE	200
2021	002705	376	.BYTE	376
2022	002706	375	.BYTE	375
2023	002707	373	.BYTE	373
2024	002710	367	.BYTE	367
2025	002711	357	.BYTE	357
2026	002712	337	.BYTE	337
2027	002713	277	.BYTE	277
2028	002714	177	.BYTE	177
2029				
2030	002715	125	.BYTE	125
2031	002716	252	.BYTE	252
2032	002717	000	.BYTE	000
2033	002720	377	.BYTE	377
2034	002721	001	.BYTE	001
2035	002722	002	.BYTE	002
2036	002723	004	.BYTE	004
2037	002724	010	.BYTE	010
2038	002725	020	.BYTE	020
2039	002726	040	.BYTE	040
2040	002727	100	.BYTE	100
2041	002730	200	.BYTE	200
2042	002731	376	.BYTE	376
2043	002732	375	.BYTE	375
2044	002733	373	.BYTE	373
2045	002734	367	.BYTE	367
2046	002735	357	.BYTE	357
2047	002736	337	.BYTE	337
2048	002737	277	.BYTE	277
2049	002740	177	.BYTE	177
2050				

EPATX: .BYTE 177

DATA TEST PATTERNS

2051			;***** DATA PATTERN I *****
2052	002741		PATI:
2053	002741	000	.BYTE 000
2054	002742	041	.BYTE 041
2055	002743	102	.BYTE 102
2056	002744	143	.BYTE 143
2057	002745	204	.BYTE 204
2058	002746	245	.BYTE 245
2059	002747	306	.BYTE 306
2060	002750	347	.BYTE 347
2061	002751	000	.BYTE 000
2062	002752	001	.BYTE 001
2063	002753	002	.BYTE 002
2064	002754	004	.BYTE 004
2065	002755	040	.BYTE 040
2066	002756	100	.BYTE 100
2067	002757	200	.BYTE 200
2068	002760	000	.BYTE 000
2069	002761	346	.BYTE 346
2070	002762	345	.BYTE 345
2071	002763	343	.BYTE 343
2072	002764	307	.BYTE 307
2073	002765	247	.BYTE 247
2074	002766	147	.BYTE 147
2075	002767	347	.BYTE 347
2076	002770	242	.BYTE 242
2077	002771	105	.BYTE 105
2078	002772	347	.BYTE 347
2079	002773	010	.BYTE 010
2080	002774	020	.BYTE 020
2081	002775	367	.BYTE 367
2082	002776	357	.BYTE 357
2083	002777	030	.BYTE 030
2084	003000	027	.BYTE 027
2085	003001	377	.BYTE 377
2086			
2087			;***** DATA PATTERN J *****
2088	003002		PATJ:
2089	003002	000	.BYTE 000
2090	003003	000	.BYTE 000
2091	003004	001	.BYTE 001
2092	003005	002	.BYTE 002
2093	003006	004	.BYTE 004
2094	003007	020	.BYTE 020
2095	003010	040	.BYTE 040
2096	003011	010	.BYTE 010
2097			
2098			;***** DATA PATTERN K *****
2099	003012		PATK:
2100	003012	000	.BYTE 000
2101	003013	377	.BYTE 377
2102	003014	376	.BYTE 376
2103	003015	375	.BYTE 375
2104	003016	373	.BYTE 373
2105	003017	376	.BYTE 376
2106	003020	177	.BYTE 177
2107	003021	377	.BYTE 377

DATA TEST PATTERNS

2108	003022	000	.BYTE	000
2109	003023	001	.BYTE	001
2110	003024	002	.BYTE	002
2111	003025	004	.BYTE	004
2112	003026	010	.BYTE	010
2113	003027	200	.BYTE	200
2114	003030	125	.BYTE	125
2115	003031	252	.BYTE	252
2116	003032	000	.BYTE	000

2117

2118

:***** DATA PATTERN L *****
PATL:

2119	003033		.BYTE	000
2120	003033	000	.BYTE	000
2121	003034	017	.BYTE	017
2122	003035	016	.BYTE	016
2123	003036	015	.BYTE	015
2124	003037	013	.BYTE	013
2125	003040	016	.BYTE	016
2126	003041	017	.BYTE	017
2127	003042	017	.BYTE	017
2128	003043	000	.BYTE	000
2129	003044	001	.BYTE	001
2130	003045	002	.BYTE	002
2131	003046	004	.BYTE	004
2132	003047	010	.BYTE	010
2133	003050	000	.BYTE	000
2134	003051	005	.BYTE	005
2135	003052	012	.BYTE	012
2136	003053	000	.BYTE	000

DATA TEST PATTERNS

2138				
2139			;***** DATA PATTERN Q *****	
2140	003054	000	PATQ: .BYTE	000
2141	003055	002	.BYTE	002
2142	003056	014	.BYTE	014
2143	003057	060	.BYTE	060
2144	003060	001	.BYTE	001
2145	003061	007	.BYTE	007
2146	003062	037	.BYTE	037
2147	003063	177	.BYTE	177
2148				
2149				
2150	003064		ENDPAT:	
2151			.EVEN	

I4

DATA TEST PATTERNS

2153
2154
2155
2156
2157 003064
2158
2159
2160
2161

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

GLOBAL TEXT SECTION

2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174

.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
; *****
; DEVTYP <M8053 OR M8064>
```

L\$DVTYP::
.ASCIZ *M8053 OR M8064*

003264
003264 115 070 060
003267 065 063 040
003272 117 122 040
003275 115 070 060
003300 066 064 000

.EVEN

2175
2176
2177
2178
2179
2180
2181

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

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

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

000012
003304
003304 104 115 126
STS - PART 3 OF 3/
003307 055 061 061
003312 040 114 111
003315 116 105 040
003320 125 116 111
003323 124 040 124
003326 105 123 124
003331 123 040 055
003334 040 120 101
003337 122 124 040
003342 063 040 117
003345 106 040 063
003350 000

.EVEN

2182 000010
2183
2184

.RADIX 8.

GLOBAL SUBROUTINE SECTION

2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224

003414
003422
003430
003436
2225
2226
2227
2228
2229
2230
2231
2232

.SBTTL GLOBAL SUBROUTINE SECTION

.SBTTL ...M-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
; MOV R3,-(SP)
; MOV @24.,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 @RUN!MCLR!MREQ,GDATA ;IDENTIFY REQUESTED FUNCTION
; GTDF EM3,ERR4 ;"MRDY" TIMEOUT
; ; QUEUE "DEVICE FATAL" ERROR # 1
; MOV @T.EDF,ERRTYP
; MOV @1,ERRNBR
; MOV @EM3,ERRMSG
; MOV @ERR4,ERRBLK
; SET CARRY TO INDICATE ERROR
; BR 9\$;EXIT WITH THE "ERROR" FLAG (CARRY BIT) SET
5\$: CLC ;CLEAR C BIT FOR NO ERRORS
9\$: RTS PC ;RETURN

....M LOOP READ

```

2234 .SBTTL ....M-LOOP -- READ
2235 ;*****
2236 ; READ - READ THE SPECIFIED ADDRESS WITHIN THE DMV-11 (M8053)
2237 ;
2238 ; CALLING SEQUENCE:
2239 ;
2240 ; JSR R5,READ
2241 ; .WORD <ADDRESS OF REGISTER WITHIN DMV-11>
2242 ; .WORD <DESTINATION ADDRESS WITHIN LSI-11>
2243 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2244 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2245 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2246 ;
2247 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2248 ;
2249 ;-----*****
2250
2251 003454 012577 176746 READ: MOV (R5)+, @SEL4 ;SETUP SOURCE POINTER
2252 003460 112777 000001 176734 MOVB @REDLOC, @SEL2 ;TELL M LOOP TO GIVE US THE REQUESTED DATA
2253
2254 003466 010346 MOV R3, -(SP)
2255 003470 012703 000050 MOV @40., R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2256 003474 077301 1$: SOB R3, 1$
2257 003476 012603 MOV (SP)+, R3
2258
2259 003500 132777 000200 176714 BITB @MRDY, @SEL2 ;DID THE M-LOOP FINISH
2260 003506 001023 BNE 5$ ;YES, GOOD. RETURN
2261
2262 003510 004737 004166 JSR PC, GETWSR ;GET BYTE SELECT REGISTERS
2263 003514 012737 000001 002324 MOV @REDLOC, GDATA ;IDENTIFY REQUESTED FUNCTION
2264 003522 GTDF EM4, ERR4 ;"MRDY" TIMEOUT
; QUEUE "DEVICE FATAL" ERROR # 2
;
; MOV @T.EDF, ERRTP
; MOV @2, ERRNBR
; MOV @EM4, ERRMSG
; MOV @ERR4, ERRBLK
2265 003522 012737 000001 002172
2266 003530 012737 000002 002174
2267 003536 012737 014307 002176
2268 003544 012737 020120 002200
2265 003552 000261 SEC ;INDICATE AN ERROR HAS BEEN STACKED
2266 003554 000401 BR 6$ ;RETURN WITH THAT INDICATION
2267
2268 003556 000241 5$: CLC ;INDICATE "NO ERROR"
2269 003560 117735 176646 6$: MOVB @SEL6, @ (R5)+ ;PUT DATA WHERE CALLER WANTS IT
2270 003564 000205 RTS R5 ;RETURN
2271
2272
2273
2274

```

....M LOOP READ IMMEDIATE

```

2276 .SBTTL ....M LOOP - READ IMMEDIATE
2277 ;*****
2278 ; READI READ IMMEDIATE THE SPECIFIED ADDRESS WITHIN THE DMV-11 (M8053)
2279 ;
2280 ; CALLING SEQUENCE:
2281 ;
2282 ; JSR R5,READI
2283 ; .WORD <ADDRESS OF REGISTER WITHIN DMV-11>
2284 ; .WORD <DESTINATION -- CONTENTS OF REG. IS PUT HERE>
2285 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2286 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2287 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2288 ;
2289 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2290 ;
2291 ; --*****
2292 ;
2293 READI:
2294 003566 012577 176634 MOV (R5)+,@SEL4 ;SETUP SOURCE POINTER
2295 003572 112777 000001 176622 MOVB @REDLOC,@SEL2 ;TFL M LOOP TO GIVE US THE REQUESTED DATA
2296 ;
2297 003600 010346 MOV R3,-(SP)
2298 003602 012703 000050 MOV @40.,R3 ;WAIT FOR THE M-LOOP TO FINISH THE OPERATION
2299 003606 077301 1$: SOB R3,1$
2300 003610 012603 MOV (SP)+,R3
2301 ;
2302 003612 132777 000200 176602 BITB @MRDY,@SEL2 ;DID THE M-LOOP FINISH
2303 003620 001023 BNE 5$ ;YES, GOOD. RETURN
2304 ;
2305 003622 004737 004166 JSR PC,GETWSR ;GET BYTE SELECT REGISTERS
2306 003626 012737 000001 002324 MOV @REDLOC,GDATA ;IDENTIFY REQUESTED FUNCTION
2307 003634 GTDF EM4,ERR4 ;"MRDY" TIMEOUT
; QUEUE "DEVICL FATAL" ERROR # 3
; MOV @T.EDF,ERRTYP
; MOV @3,ERRNBR
; MOV @EM4,ERRMSG
; MOV @ERR4,ERRBL<
003634 012737 000001 002172 SEC ;INDICATE AN ERROR HAS BEEN STACKED
003642 012737 000003 002174 BR 6$ ;RETURN WITH THAT INDICATION
003650 012737 014307 002176
003656 012737 020120 002200
2308 003664 000261 5$: CLC ;INDICATE "NO ERROR"
2309 003666 000401 6$: MOV @SEL6,(R5)+ ;PUT DATA WHERE CALLER WANTS IT
2310 ;
2311 003670 000241 RTS R5 ;RETURN
2312 003672 017725 176534
2313 003676 000205
2314
2315
2316
2317

```

....M LOOP - WRITE

2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336 003700 012577 176522
2337 003704 113577 176522
2338 003710 000404
2339
2340
2341
2342

```

.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
MOV @R5+, @SEL6 ;MAKE DATA AVAILABLE TO M-LOOP
BR MLWRI ;THE REST OF THIS ROUTINE IS THE SAME AS "WRITEI"

```


....GETBSR GET BYTE SELECT REGISTERS

2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431

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

```
GETBSR: MOV  BBSSEL0,BSR0      ;PUT THE CURRENT CSR VALUES INTO THE PRINT-OUT
        MOV  BBSSEL1,BSR1      ;TABLE
        MOV  BBSSEL2,BSR2
        MOV  BBSSEL3,BSR3
        MOV  BBSSEL4,BSR4
        MOV  BBSSEL5,BSR5
        MOV  BBSSEL6,BSR6
        MOV  BBSSEL7,BSR7
        MOV  BBSSEL10,BSR10
        MOV  BBSSEL11,BSR11
        MOV  BBSSEL12,BSR12
        MOV  BBSSEL13,BSR13
        MOV  BBSSEL14,BSR14
        MOV  BBSSEL15,BSR15
        MOV  BBSSEL16,BSR16
        MOV  BBSSEL17,BSR17
        RTS  PC                ;RETURN TO CALLER
```

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

```
GETWSR: MOV  BSEL0,WSR0      ;MOVE THE 4 WORD REGISTERS TO THE OTHERWISE
        MOV  BSEL2,WSR2      ;BYTE TABLE
        MOV  BSEL4,WSR4
        MOV  BSEL6,WSR6
        MOV  BSEL10,WSR10
        MOV  BSEL12,WSR12
        MOV  BSEL14,WSR14
        MOV  BSEL16,WSR16
        RTS  PC                ;RETURN TO CALLER
```

....STUREG STATIC TEST OF SPECIFIED USYRT REGISTER

```

2433 .SBTTL ....STUREG - STATIC TEST OF SPECIFIED USYRT REGISTER
2434 ;*****
2435 ; STUREG -- PERFORM A STATIC TEST OF THE SPECIFIED USYRT REGISTER
2436 ;
2437 ; CALLING SEQUENCE:
2438 ;
2439 ; <R0 CONTAINS THE ADDRESS OF THE REGISTER TO BE TESTED>
2440 ; <"TDATA" CONTAINS THE TEST BYTE>
2441 ; <"GDATA" CONTAINS THE EXPECTED DATA>
2442 ; <"REGNUM" CONTAINS REG INDEX FOR POSSIBLE ERRORS>
2443 ;
2444 ; JSR PC,STUREG
2445 ; BCC N$ ;IF NO ERROR OCCURED, PROCEED WITH ROUTINE
2446 ; ERROR ;AN ERROR MESSAGE HAS BEEN STACKED: PRINT IT
2447 ; <ANY OTHER SPECIAL ERROR PROCESSING MAY BE DONE HERE (I.E. CKLOOP)>
2448 ;
2449 ; N$: <RESUMPTION OF NORMAL PROCESSING>
2450 ;
2451 ;-----
2452 ;
2453 004250 010037 004264 STUREG: MOV R0,2$ ;PUT SPECIFIED REGISTER'S ADDRESS IN I/O CALLS
2454 004254 010037 004302 MOV R0,4$
2455 ;
2456 004260 004537 003700 JSR R5,WR,TE ;WRITE IT
2457 004264 000000 2$: .WORD 0 ;*** MODIFIED FROM ABOVE ***
2458 004266 002322 .WORD TDATA ;
2459 004270 103431 BCS 10$ ;ON ERROR, EXIT
2460 ;
2461 004272 005037 002326 CLR BDATA ;CLEAR BOTH BYTES -- JUST IN CASE....
2462 004276 004537 003454 JSR R5,READ ;READ IT BACK AGAIN
2463 004302 000000 4$: .WORD 0 ;*** MODIFIED FROM ABOVE ***
2464 004304 002326 .WORD BDATA ;
2465 004306 103422 BCS 10$ ;ON ERROR, EXIT
2466 ;
2467 004310 123737 002324 002326 CMPB GDATA,BDATA ;DID WE READ WHAT WE WROTE?
2468 004316 000241 CLC ; (THIS ISN'T NEEDED FOR THE ERROR TEST BUT
2469 ; MUST BE CLEARED ON EXIT IF NO ERROR OCCURED)
2470 004320 001415 BEQ 10$ ;YES, EXIT FROM SUBTEST
2471 004322 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
004322 012737 000001 002172
004330 012737 000005 002174
004336 012737 014360 002176
004344 012737 020244 002200
2472 004352 000261 SEC ;INDICATE THAT AN ERROR WAS DETECTED
2473 004354 000207 10$: RTS PC
2474 ;
2475 ;
2476 .SBTTL ....STALL -- DELAY FOR 10.5 MICRO SEC'S (ON LSI 11)
2477 ;*****
2478 ; STALL THIS SUBROUTINE STALLS FOR ABOUT 10.5 MICRO SECONDS
2479 ;-----
2480 ;
2481 004356 000207 STALL: RTS PC
2482 ;
2483 ;
2484 ;

```

```

2486          . .SBTTL
2487
2488          ;*****
2489          ;* GETURS - LOAD INTO THE 8 WORD STORAGE AREA (UREGS) THE CONTENTS OF THE
2490          ;*   VARIOUS USYRT REGISTERS
2491          ;*
2492          ;*   CALLING SEQUENCE:
2493          ;*
2494          ;*****
2495 004360 012737 002242 004422 GETURS: MOV   @UREGS,5$ ;INIT POINTER TO REG STORAGE TABLE
2496 004366 012737 120400 004420      MOV   @USYRT,4$ ;INIT POINTER TO REGISTER ADDRESSES
2497
2498 004374 005037 002260          CLR   UREGS+14. ;CLEAR STORAGE WORD
2499 004400 004537 003454          JSR   R5,READ ;READ THE USYRT STATUS REGISTER
2500 004404 122000          .WORD USTATR ;STATUS REGISTER'S ADDRESS WITHIN DMV-11
2501 004406 002260          .WORD UREGS+14. ;ADDRESS ALLOCATED TO THAT REG. W/IN "UREGS"
2502
2503 004410 005077 000006 3$:   CLR   @5$ ;CLEAR STORAGE WORD
2504 004414 004537 003454          JSR   R5,READ ;READ A LINE UNIT REG
2505 004420 000000          4$:   .WORD 0 ;REGISTER ADDRESS GOES HERE
2506 004422 000000          5$:   .WORD 0 ;STORAGE ADRS IN TABLE GOES HERE
2507
2508 004424 005237 004420 6$:   INC   4$ ;INCREMENT REG NO.
2509 004430 023727 004420 120406      CMP   4$,@USYRT+6 ;THIS IS NOT A VALID REGISTER ADDRESS
2510 004436 001772          BEQ   6$ ;SO IT MUST BE BYPASSED
2511
2512 004440 062737 000002 004422      ADD   @2,5$ ;ADVANCE ADDRESS OF STORAGE ANEA POINTER
2513 004446 023727 004420 120410      CMP   4$,@USYRT+10 ;SEE IF ALL REGS READ YET
2514 004454 001355          BNE   3$ ;BR IF NOT
2515
2516 004456 000207          RTS   PC ;RETURN
2517
2518
2519
2520          ;*****
2521          ;* GETVRS: - LOAD INTO THE 16 WORD STORAGE AREA (VREGS) THE CONTENTS OF THE
2522          ;*   VARIOUS VIA REGISTERS.
2523          ;*
2524          ;*   CALLING SEQUENCE :
2525          ;*
2526          ;*****
2526 004460 012737 002262 004506 GETVRS: MOV   @VREGS,5$ ;INIT POINTER TO REG STORAGE TABLE
2527 004466 012737 120000 004504      MOV   @VIA,4$ ;INIT POINTER TO REGISTER ADDRESSES
2528 004474 005077 000006 3$:   CLR   @5$ ;CLEAR STORAGE WORD
2529 004500 004537 003454          JSR   R5,READ ;READ A VIA REG
2530 004504 000000          4$:   .WORD 0 ;REGISTER ADDRESS GOES HERE
2531 004506 000000          5$:   .WORD 0 ;STORAGE ADRS IN TABLE GOES HERE
2532 004510 005237 004504 6$:   INC   4$ ;INCREMENT REG NO.
2533 004514 062737 000002 004506      ADD   @2,5$ ;INCREMENT STORAGE ADRS
2534 004522 023727 004504 120020      CMP   4$,@VIA+16. ;SEE IF ALL VIA REGS READ YET
2535 004530 001361          BNE   3$ ;BR IF NOT
2536 004532 000207          RTS   PC ;RETURN

```

....INITT1 INITIALIZE TIMER #1

2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594

```

.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
        MOVB     (R5),R1   ;GET & PROCESS BITS FOR ACR 6 & 7
        BICB     077,R1
        MOV      R1,4$     ;SETUP CALL SET ACR'S BITS 6 & 7
        MOVB     (R5)+,R1 ;NOW, GET THE BIT TO BE USED IN SETTING OR
                           ;CLEARING BIT 6 OF "IER"
                           ;THE PASSED BIT IS IN THE WRONG POSITION
                           ;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).
        BICB     177,R1    ;FIRST, MAKE SURE ALL UNWANTED BITS ARE CLEARED
        BISB     100,R1    ;THEN SET BIT 6
        MOV      R1,2$    ;THE CALL WILL NOW WRITE THE APPROPRIATE VALUE

        JSR      R5,WRITEI ;WRITE TO
        VIAIER   0        ;THE VIA'S IER
2$:      .WORD    0        ;INTERRUPT ENABLE/DISABLE INFORMATION

        JSR      R5,READI  ;READ THE CURRENT SETTING OF
        VIAACR   0        ;THE VIA'S ACR
3$:      .WORD    0        ;INTO "3$"

        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

```

2595
2596 004642 004537 003712      JSR    R5,WRITEI      ;WRITE TO
2597 004646 120013              VIAACR                ;THE VIA'S ACR
2598 004650 000000      4$:  .WORD    0      ;THE NEW REGISTER SETTING
2599
2600 004652 004537 003712      JSR    R5,WRITEI      ;WRITE TO
2601 004656 120006              VIAT1C                ;LOW ORDER LATCH REGISTER (T1L-L)
2602 004660 000000      7$:  .WORD    0      ;THE VALUE PASSED
2603
2604 004662 113737 004661 004676  MOVB   7$+1,8$        ;SETUP FOR AND
2605 004670 004537 003712      JSR    R5,WRITEI      ;WRITE TO
2606 004674 120007              VIAT1D                ;HIGH ORDER LATCH REGISTER (T1L-H)
2607 004676 000000      8$:  .WORD    0      ;THE VALUE PASSED
2608
2609 004700 004537 003712      JSR    R5,WRITEI      ;WRITE TO
2610 004704 120004              VIAT1A                ;LOW ORDER LATCH & COUNTER (T1L-L & T1C-L)
2611 004706 000000      10$: .WORD    0      ;THE VALUE PASSED
2612
2613 004710 113737 004707 004724  MOVB   10$+1,11$     ;SETUP FOR AND
2614 004716 004537 003712      JSR    R5,WRITEI      ;WRITE TO
2615 004722 120005              VIAT1B                ;HIGH ORDER COUNTER (T1C-H) <ALSO STARTS CTR>
2616 004724 000000      11$: .WORD    0      ;THE VALUE PASSED
2617
2618      ; DON'T WAIT AROUND FOR ANYTHING TO HAPPEN - JUST (JEST) RETURN!
2619
2620 004726 012601              MOV    (SP)+,R1      ;BUT FIRST RESTORE R1
2621 004730 005205              INC    R5             ;AND PUT R5 BACK ON A WORD BOUNDARY (THE LAST
2622                                     ;PASSED PARAM. WAS A BYTE, NOT A WORD!)
2623
2624 004732 000205              RTS    R5             ;NOW, RETURN
2625
2626

```

....INITT2 - INITIALIZE TIMER #2

2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684

004734 010146
004736 012537 005056
004742 111501
004744 143701 000337
004750 010137 005046
004754 112501
004756 106301
004760 106301
004762 106301
004764 143701 000177
004770 153701 000040
004774 010137 005006
005000 004537 003712
005004 120016
005006 000000
005010 004537 003566
005014 120013
005016 000000
005020 013701 005016
005024 143701 000040
005030 057701 005046
005034 010137 005046
005040 004537 003712

```

.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
        MOVB     (R5),R1       ;GET & PROCESS BIT FOR ACR 5
        BICB     337,R1
        MOV      R1,4$        ;SETUP CALL TO SET OR CLEAR ACR'S BIT 5
        MOVB     (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
        ASLB     R1            ;OPERATION.
        ;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
        VIAIER   0            ;THE VIA'S IER
        2$: .WORD    0          ;INTERRUPT ENABLE/DISABLE INFORMATION
        JSR      R5,READI      ;READ THE CURRENT SETTING OF
        VIAACR   0            ;THE VIA S ACR
        3$: .WORD    0          ;INTO "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

```

2685 005044 120013
2686 005046 000000      4$: VIAACR      ;THE VIA'S ACR
                          .WORD 0      ;THE NEW REGISTER SETTING
2687
2688 005050 004537 003712      JSR R5,WRITEI ;WRITE TO
2689 005054 120010      VIAT2A      ;LOW ORDER LATCH & COUNTER (T2L-L & T2C L)
2690 005056 000000      10$: .WORD 0      ;THE VALUE PASSED
2691
2692 005060 113737 005057 005074      MOVB 10$+1,11$ ;SETUP FOR AND
2693 005066 004537 003712      JSR R5,WRITEI ;WRITE TO
2694 005072 120011      VIAT2B      ;HIGH ORDER COUNTER (T2C H) <ALSO STARTS CTR>
2695 005074 000000      11$: .WORD 0      ;THE VALUE PASSED
2696
2697      ; DON'T WAIT AROUND FOR ANYTHING TO HAPPEN - JUST (JEST) RETURN!
2698
2699 005076 012601      MOV (SP)+,R1 ;BUT FIRST RESTORE R1
2700 005100 005205      INC R5      ;AND PUT R5 BACK ON A WORD BOUNDARY (THE LAST
2701      ;PASSED PARAM. WAS A BYTE, NOT A WORD!)
2702
2703 005102 000205      RTS R5      ;THEN RETURN
2704

```

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

```

2706 .SBTTL ....RSTCHK -- RESET USYRT/VERIFY ALL USYRT REGS @ RESET STATE
2707 ;*****
2708 ; RSTCHK MANUALLY RESET THE USYRT AND VERIFY THAT ALL USYRT REGISTERS
2709 ; ARE IN THEIR RESET STATE. AN ERROR MESSAGE IDENTIFYING THE
2710 ; FAILING REGISTER IS STACKED IF ONE IS ENCOUNTERED.
2711 ;
2712 ; CALLING SEQUENCE:
2713 ; JSR R5,RSTCHK
2714 ;*****
2715
2716 RSTCHK:
2717 005104 MOV R1,-(SP) ;SAVE R1
2718 005106 MOV R2,-(SP) ;SAVE R2
2719
2720 005110 JSR R5,WRITEI ;SET PROGRAM RESET BIT IN VIA ORB REG
2721 005114 VIAORB
2722 005116 000031 DTR!RTSND!PRESET
2723 005120 004537 003712 JSR R5,WRITEI ;CLEAR PROGRAM RESET BIT IN VIA ORB REG
2724 005124 120000 VIAORB
2725 005126 000030 DTR!RTSND
2726
2727 005130 005001 CLR R1 ;INIT USYRT REG ADRS PTR
2728 005132 012702 002606 MOV #PATF,R2 ;INIT DATA PATTERN POINTER
2729 005136 016137 002476 005150 6$: MOV USYREG(R1),7$ ;SET USYRT READ ADDRESS
2730 005144 004537 003566 JSR R5,READI ;READ A USYRT REG
2731 005150 000000 7$: .WORD 0 ;USYRT REG ADRS GOES HERE
2732 005152 000000 8$: .WORD 0 ;DATA READ IS RETURNED HERE
2733 005154 123722 005152 CMPB 8$,(R2)+ ;SEE IF REG CONTAINS EXPECTED DATA
2734 005160 001432 BEQ 9$ ;BR IF MATCH
2735
2736 005162 010137 002336 MOV R1,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
2737 005166 006237 002336 ASR REGNUM ;GET WORD OFFSET
2738 005172 005037 002324 CLR GDATA ;GET EXPECTED DATA
2739 005176 116237 177777 002324 MOVB -1(R2),GDATA
2740 005204 013737 005152 002326 MOV 8$,BDATA ;GET ACTUAL DATA
2741 ;STACK "USYRT NOT CLEARED BY PROGRAM RESET" MSG
2742 005212 GTDF EM2,ERR10
;
; QUEUE "DEVICE FATAL" ERROR @ 6
;
; MOV #T.EDF,ERRTYP
; MOV #6,ERRNBR
; MOV #EM2,ERRMSG
; MOV #ERR10,ERRBLK
;
2743 005212 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
2744 005244 000406 BR 10$ ;TAKE ERROR EXIT
2745
2746 005246 062701 000002 9$: ADD #2,R1 ;INCR USYRT REG ADRS PTR
2747 005252 020127 000020 CMP R1,#16. ;SEE IF ALL REGS READ YET
2748 005256 002727 BLT 6$ ;BR IF NOT
2749 005260 000241 CLC ;** CLEAR C BIT FOR NO ERRORS
2750 005262 012602 10$: MOV (SP)+,R2 ;RESTORE R2
2751 005264 012601 MOV (SP)+,R1 ;RESTORE R1
2752 005266 000205 RTS R5 ;** RETURN
2753
2754

```

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

```

2756 ;*****
2757 ;* WAIT50 - THIS SUBROUTINE STALLS FOR AT LEAST 50 MICRO SEC, AND THEN RETURNS.
2758 ;*****
2759 005270 010146 000005 WAIT50: MOV R1,-(SP) ;SAVE R1
2760 005272 012701 000005 MOV #5.,R1 ;INIT COUNTER
2761 005276 077101 3$: SOB R1,3$ ;DELAY HERE FOR 23.8 MICRO-SEC'S
2762 005300 012601 MOV (SP)+,R1 ;RESTORE R1
2763 005302 000207 RTS PC ;RETURN
2764
2765 ; OVERHEAD (JSR, MOV, MOV, MOV, & RTS) ADD UP TO 25.25 MICRO-SEC'S
2766
2767 ; THEREFORE, ACTUAL TOTAL DELAY IS 49.35 MICRO SECONDS
2768
2769
2770
2771
2772 .SBTTL ....SETVIA -- SET UP VIA REGISTERS
2773 ;*****
2774 ;* SETVIA - SET UP THE VIA REGISTERS
2775 ;*
2776 ;* THIS SUBROUTINE PROGRAMS THE VIA REGISTERS FOR NORMAL OPERATION, BY
2777 ;* LOADING THE DDRB, DDRA, ORB, ACR, PCR, IER.
2778 ;*
2779 ;* CALLING SEQUENCE :
2780 ;* JSR PC,SETVIA
2781 ;*****
2782 005304 SETVIA: JSR R5,WRITEI ;SET PORT B FOR OUTPUT MODE
2783 005304 004537 003712 VIADPB
2784 005310 120002 377
2785 005312 000377 JSR R5,WRITEI ;SET PORT A FOR INPUT MODE
2786 005314 004537 003712 VIADPA ; (BIT0 IS ONLY OUTPUT BIT)
2787 005320 120003 001
2788 005322 000001 JSR R5,WRITEI ;DISABLE USYRT INTERNAL LOOPBACK
2789 005324 004537 003712 VIAORA
2790 005330 120017 000
2791 005332 000000 JSR R5,WRITEI ;INIT PORT B
2792 005334 004537 003712 VIAORB
2793 005340 120000 DTR!RTSND
2794 005342 000030 JSR R5,WRITEI ;SET ACR FOR : T1 SQUARE WAVE OUTPUT MODE,
2795 005344 004537 003712 VIAACR ; T2 ONE-SHOT OUTPUT MODE,
2796 005350 120013 350 ; SR AT SYS CLOCK RATE ON CB1
2797 005352 000350 JSR R5,WRITEI ;SET PCR FOR : CB1 NEG TRANS INPUT MODE,
2798 005354 004537 003712 VIAPCR ; CA2 NEG TRANS INPUT MODE,
2799 005360 120014 022 ; CA1 NEG TRANS INPUT MODE
2800 005362 000022 JSR R5,WRITEI ;DISABLE ALL MICRO INTRPTS
2801 005364 004537 003712 VIAIER
2802 005370 120016 177
2803 005372 000177 RTS PC ;RETURN
2804 005374 000207
2805
2806

```

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

```

2808 .SBTTL ....INIDMV -- INIT DMV (MCLR, VIA SETUP)
2809 ;*****
2810 ;* INIDMV THIS SUBROUTINE INITIALIZES THE DMV-11, BY DOING A MASTER CLEAR,
2811 ;* ENTERING THE M-LOOP, AND PROGRAMMING THE VIA REGS FOR DEFAULT
2812 ;* OPERATION.
2813 ;*
2814 ;* CALLING SEQUENCE :
2815 ;* JSR PC,INIDMV
2816 ;*****
2817 005376 004737 003352 INIDMV: JSR PC,MSTCLR ;MASTER CLR, M-LOOP
2818 005402 004737 005304 JSR PC,SETVIA ;PROGRAM VIA
2819 005406 000207 RTS PC ;RETURN
2820
2821
2822
2823

```

```

2824 .SBTTL ....CKUSTS -- CHECK USYRT STATUS REGISTERS
2825 ;*****
2826 ;* CKUSTS - THIS SUBROUTINE CHECKS THE USYRT STATUS BY READING THE USYRT
2827 ;* STATUS REGISTER AND COMPARING IT TO THE LOW BYTE OF THE WORD FOLLOWING
2828 ;* THE CALL. IF THERE IS A MISMATCH, THE SUBROUTINE STACKS THE ERROR
2829 ;* INFORMATION, AND SETS THE "C" BIT AND RETURNS.
2830 ;*****

```

```

2831 005410 CKUSTS:
2832 005410 004537 003566 JSR R5,READI ;READ USYRT STATUS REGISTER
2833 005414 122000 USTATR
2834 005416 000000 .WORD 0
2835 005420 122537 005416 1$: CMPB (R5)+,1$ ;SEE IF STATUS MATCHES EXPECTED
2836 005424 000241 CLC ;CLEAR C BIT
2837 005426 001430 BEQ 2$ ;BR IF STATUS OK
2838 005430 012737 000007 002336 MOV #7,REGNUM ;SET USYRT REG NO. FOR PRINTOUT
2839 005436 016537 177777 002324 MOV -1(R5),GDATA ;GET EXPECTED DATA
2840 005444 005037 002326 CLR BDATA ;GET ACTUAL DATA
2841 005450 113737 005416 002326 MOVB 1$,BDATA
2842 ;STACK "USYRT STATUS INCORRECT" ERROR
2843 005456 GTDF EM68,ERR10
; QUEUE "DEVICE FATAL" ERROR # 7
MOV #T.EDF,ERRTYP
MOV #7,ERRNBR
MOV #EM68,ERRMSG
MOV #ERR10,ERRBLK
2844 005506 000261 SEC ;SET C BIT FOR ERROR
2845 005510 005205 2$: INC R5 ;INCREMENT R5 PAST ARGUMENT
2846 005512 000205 RTS R5 ;RETURN
2847
2848
2849
2850

```

....CKTACT CHECK TRANSMITTER ACTIVE (TXACT)

```

2852 .SBTTL ....CKTACT -- CHECK TRANSMITTER ACTIVE (TXACT)
2853 ;*****
2854 ;* CKTACT THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TXACT IN THE USYRT
2855 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2856 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2857 ;*
2858 ;* CALLING SEQUENCE :
2859 ;* JSR R5,CKTACT
2860 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TXACT>
2861 ;*****
2862 005514 CKTACT:
2863 005514 012737 000007 002336 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2864 005522 004537 003566 JSR R5,READI ;READ USYRT STATUS
2865 005526 122000 USTATR
2866 005530 000000 1$: .WORD 0
2867 005532 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TXACT
2868 005536 001422 BEQ 2$ ;BR IF EXPECTED TXACT = 0
2869 005540 132737 000004 005530 BITB #TXACT,1$ ;SEE IF TXACT = 1
2870 005546 001040 BNE 3$ ;BR IF TXACT = 1
2871 ;STACK "TXACT NOT SET" MSG
2872 005550 GTDF EM69,ERR12
; QUEUE "DEVICE FATAL" ERROR # 8
MOV #T.EDF,ERRTYP
MOV #8,ERRNBR
MOV #EM69,ERRMSG
MOV #ERR12,ERRBLK
2873 005600 000261 SEC ;SET C BIT TO FLAG ERROR
2874 005602 000423 BR 4$ ;TAKE ERROR EXIT
2875 005604 132737 000004 005530 2$: BITB #TXACT,1$ ;SEE IF TXACT = 0
2876 005612 001416 BEQ 3$ ;BR IF TXACT = 0
2877 ;STACK "TXACT NOT CLEARED" MSG
2878 005614 GTDF EM70,ERR12
; QUEUE "DEVICE FATAL" ERROR # 9
MOV #T.EDF,ERRTYP
MOV #9,ERRNBR
MOV #EM70,ERRMSG
MOV #ERR12,ERRBLK
2879 005644 000261 SEC ;SET C BIT TO FLAG ERROR
2880 005646 000401 BR 4$ ;TAKE ERROR EXIT
2881 005650 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
2882 005652 000205 4$: RTS R5 ;RETURN
2883
2884
2885
2886

```

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

```

2888      .SBTTL ....CKRACT -- CHECK RECEIVER ACTIVE (RXACT)
2889      ;*****
2890      ;* CKRACT - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RXACT IN THE USYRT
2891      ;*   STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2892      ;*   STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2893      ;*
2894      ;*   CALLING SEQUENCE :
2895      ;*   JSR     R5,CKRACT
2896      ;*   .WORD  <BIT 0 IS EXPECTED VALUE OF RXACT>
2897      ;*****
2898 005654 CKRACT:
2899 005654 012737 000007 002336      MOV     #7,REGNUM      ;SET REG NO. FOR POSSIBLE ERROR REPORT
2900 005662 004537 003566      JSR     R5,READI      ;READ USYRT STATUS
2901 005666 122000      USTATR
2902 005670 000000      1$:   .WORD  0
2903 005672 032725 000001      BIT     #BIT0,(R5)+   ;GET EXPECTED STATE OF RXACT
2904 005676 001422      BEQ     2$            ;BR IF EXPECTED RXACT = 0
2905 005700 132737 000040 005670      BITB   #RXACT,1$     ;SEE IF RXACT = 1
2906 005706 001040      BNE     3$            ;BR IF RXACT = 1
2907      ;STACK "RXACT NOT SET" MSG
2908 005710      GTDF   EM71,ERR12
;
;   QUEUE "DEVICE FATAL" ERROR # 10
;
;   MOV     #T.EDF,ERRTYP
;   MOV     #10,ERRNBR
;   MOV     #EM71,ERRMSG
;   MOV     #ERR12,ERRBLK
;
2909 005710 012737 000001 002172      SEC
2910 005716 012737 000012 002174      BR     4$            ;SET C BIT TO FLAG ERROR
2911 005724 012737 014737 002176      BR     4$            ;TAKE ERROR EXIT
2912 005732 012737 020714 002200      BITB   #RXACT,1$     ;SEE IF RXACT = 0
2913      ;STACK "RXACT NOT CLEARED" MSG
2914 005754      BEQ     3$            ;BR IF RXACT = 0
;
;   QUEUE "DEVICE FATAL" ERROR # 11
;
;   MOV     #T.EDF,ERRTYP
;   MOV     #11,ERRNBR
;   MOV     #EM72,ERRMSG
;   MOV     #ERR12,ERRBLK
;
2915 006004 000261      SEC
2916 006006 000401      BR     4$            ;SET C BIT TO FLAG ERROR
2917 006010 000241      3$:   CLC
2918 006012 000205      4$:   RTS     R5      ;TAKE ERROR EXIT
;
;   CLEAR C BIT FOR NO ERRORS
;   RETURN
2919
2920
2921
2922

```


....CKTBMT CHECK TRANSMIT BUFFER EMPTY

```

2924 .SBTTL ....CKTBMT -- CHECK TRANSMIT BUFFER EMPTY
2925 ;*****
2926 ;* CKTBMT THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TBMT IN THE USYRT
2927 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2928 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2929 ;*
2930 ;* CALLING SEQUENCE :
2931 ;* JSR R5,CKTBMT
2932 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TBMT>
2933 ;*****
2934 006014 CKTBMT:
2935 006014 012737 000007 002336 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2936 006022 004537 003566 JSR R5,READI ;READ USYRT STATUS
2937 006026 122000 USTATR
2938 006030 000000 1$: .WORD 0
2939 006032 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TBMT
2940 006036 001422 BEQ 2$ ;BR IF EXPECTED TBMT = 0
2941 006040 132737 000100 006030 BITB #TBMT,1$ ;SEE IF TBMT = 1
2942 006046 001040 BNE 3$ ;BR IF TBMT = 1
2943 ;STACK "TBMT NOT SET" MSG
2944 006050 GTDF EM73,ERR12
; QUEUE "DEVICE FATAL" ERROR # 12
MOV #T.EDF,ERR1P
MOV #12,ERRNBR
MOV #EM73,ERRMSG
MOV #ERR12,ERRBLK
006050 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
006056 012737 000014 002174 BR 4$ ;TAKE ERROR EXIT
006064 012737 014777 002176 2$: BITB #TBMT,1$ ;SEE IF TBMT = 0
006072 012737 020714 002200 BEQ 3$ ;BR IF TBMT = 0
2945 006100 000261 ;STACK "TBMT NOT CLEARED" MSG
2946 006102 000423 GTDF EM74,ERR12
2947 006104 132737 000100 006030 3$: SEC ;SET C BIT TO FLAG ERROR
2948 006112 001416 BR 4$ ;TAKE ERROR EXIT
2949 006114 ;STACK "TBMT NOT CLEARED" MSG
2950 006114 GTDF EM74,ERR12
; QUEUE "DEVICE FATAL" ERROR # 13
MOV #T.EDF,ERR1P
MOV #13,ERRNBR
MOV #EM74,ERRMSG
MOV #ERR12,ERRBLK
006114 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
006122 012737 000015 002174 BR 4$ ;TAKE ERROR EXIT
006130 012737 015014 002176 3$: CLC ;CLEAR C BIT FOR NO ERRORS
006136 012737 020714 002200 4$: RTS R5 ;RETURN
2951 006144 000261
2952 006146 000401
2953 006150 000241
2954 006152 000205
2955
2956
2957
2958

```

....CKRDA CHECK RECEIVE DATA AVAILABLE

```

2960 .SBTTL ....CKRDA -- CHECK RECEIVE DATA AVAILABLE
2961 ;*****
2962 ;* CKRDA THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RDA IN THE USYRT
2963 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
2964 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
2965 ;*
2966 ;* CALLING SEQUENCE :
2967 ;* JSR R5,CKRDA
2968 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RDA>
2969 ;*****
2970 006154 CKRDA:
2971 006154 012737 000007 002336 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
2972 006162 004537 003566 JSR R5,READI ;READ USYRT STATUS
2973 006166 122000 USTATR
2974 006170 000000 1$: .WORD 0
2975 006172 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF RDA
2976 006176 001422 BEQ 2$ ;BR IF EXPECTED RDA = 0
2977 006200 132737 000200 006170 BITB #RDA,1$ ;SEE IF RDA = 1
2978 006206 001040 BNE 3$ ;BR IF RDA = 1
2979 ;STACK "RDA NOT SET" MSG
2980 006210 GTDF EM75,ERR12
; QUEUE "DEVICE FATAL" ERROR # 14
MOV #T.EDF,ERRTYP
MOV #14,ERRNBR
MOV #EM75,ERRMSG
MOV #ERR12,ERRBLK
006210 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
006216 012737 000016 002174 BR 4$ ;TAKE ERROR EXIT
006224 012737 015035 002176 2$: BITB #RDA,1$ ;SEE IF RDA = 0
006232 012737 020714 002200 BEQ 3$ ;BR IF RDA = 0
2981 006240 000261 ;STACK "RDA NOT CLEARED" MSG
2982 006242 000423 GTDF EM76,ERR12
2983 006244 132737 000200 006170 3$: SEC ;SET C BIT TO FLAG ERROR
2984 006252 001416 BR 4$ ;TAKE ERROR EXIT
2985 ;STACK "RDA NOT CLEARED" MSG
2986 006254 GTDF EM76,ERR12
; QUEUE "DEVICE FATAL" ERROR # 15
MOV #T.EDF,ERRTYP
MOV #15,ERRNBR
MOV #EM76,ERRMSG
MOV #ERR12,ERRBLK
006254 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
006262 012737 000017 002174 BR 4$ ;TAKE ERROR EXIT
006270 012737 015051 002176 3$: CLC ;CLEAR C BIT FOR NO ERRORS
006276 012737 020714 002200 4$: RTS R5 ;RETURN
2987 006304 000261
2988 006306 000401
2989 006310 000241
2990 006312 000205
2991
2992
2993
2994

```

....CKRSA - CHECK RECEIVER STATUS AVAILABLE

```

2996 .SBTTL ....CKRSA -- CHECK RECEIVER STATUS AVAILABLE
2997 ;*****
2998 ;* CKRSA - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF RSA IN THE USYRT
2999 ;* STATUS REGISTER, AND REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE
3000 ;* STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3001 ;*
3002 ;* CALLING SEQUENCE :
3003 ;* JSR R5,CKRSA
3004 ;* .WORD <BIT 0 IS EXPECTED VALUE OF RSA>
3005 ;*****
3006 006314 CKRSA:
3007 006314 012737 000007 002336 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3008 006322 004537 003566 JSR R5,READI ;READ USYRT STATUS
3009 006326 122000 USTATR
3010 006330 000000 1$: .WORD 0
3011 006332 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF RSA
3012 006336 001422 BEQ 2$ ;BR IF EXPECTED RSA = 0
3013 006340 132737 000020 006330 BITB #RSA,1$ ;SEE IF RSA = 1
3014 006346 001040 BNE 3$ ;BR IF RSA = 1
3015 ;STACK "RSA NOT SET" MSG
3016 006350 GTDF EM77,ERR12
; QUEUE "DEVICE FATAL" ERROR # 16
MOV #T.EDF,ERRTYP
MOV #16,ERRNBR
MOV #EM77,ERRMSG
MOV #ERR12,ERRBLK
006350 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
006356 012737 000020 002174 BR 4$ ;TAKE ERROR EXIT
006364 012737 015071 002176 BITB #RSA,1$ ;SEE IF RSA = 0
006372 012737 020714 002200 BEQ 3$ ;BR IF RSA = 0
3017 006400 000261 ;STACK "RSA NOT CLEARED" MSG
3018 006402 000423 GTDF EM78,ERR12
3019 006404 132737 000020 006330 2$: BITB #RSA,1$
3020 006412 001416 BEQ 3$
3021 ;STACK "RSA NOT CLEARED" MSG
3022 006414 GTDF EM78,ERR12
; QUEUE "DEVICE FATAL" ERROR # 17
MOV #T.EDF,ERRTYP
MOV #17,ERRNBR
MOV #EM78,ERRMSG
MOV #ERR12,ERRBLK
006414 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
006422 012737 000021 002174 BR 4$ ;TAKE ERROR EXIT
006430 012737 015105 002176 BITB #RSA,1$ ;SEE IF RSA = 0
006436 012737 020714 002200 BEQ 3$ ;BR IF RSA = 0
3023 006444 000261 ;STACK "RSA NOT CLEARED" MSG
3024 006446 000401 GTDF EM78,ERR12
3025 006450 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3026 006452 000205 4$: RTS R5 ;RETURN
3027
3028

```

....CKROR CHECK RECEIVER OVERRUN

```

3030 .SBTTL ....CKROR -- CHECK RECEIVER OVERRUN
3031 ;*****
3032 ;* CKROR - THIS SUBROUTINE CHECKS FOR THE OCCURANCE OF RECEIVER OVERRUN IN THE
3033 ;* USYRT RECEIVER STATUS REGISTER (RDSRH), AND REPORTS AN ERROR IF IT IS
3034 ;* NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3035 ;*
3036 ;* CALLING SEQUENCE :
3037 ;* JSR R5,CKROR
3038 ;* .WORD <BIT 0 IS EXPECTED VALUE OF ROR>
3039 ;*****
3040 006454 CKROR: MOV #1,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3041 006454 012737 000001 002336 JSR R5,READI ;READ RECEIVER STATUS
3042 006462 004537 003566 RDSRH
3043 006466 120401 .WORD 0
3044 006470 000000 1$: BIT #BIT0,(R5) ;GET EXPECTED STATE OF ROR
3045 006472 032725 000001 BEQ 2$ ;BR IF EXPECTED ROR = 0
3046 006476 001422 BITB #ROR,1$ ;SEE IF ROR = 1
3047 006500 132737 000010 006470 BNE 3$ ;BR IF ROR = 1
3048 006506 001040 ;STACK "RECEIVER OVRN NOT SET" MSG
3049 ;GTFD EM90,ERR12
3050 006510 ; QUEUE "DEVICE FATAL" ERROR # 18
3051 006510 012737 000001 002172 MOV #T.EDF,ERRTYP
3052 006516 012737 000022 002174 MOV #18,ERRNBR
3053 006524 012737 015450 002176 MOV #EM90,ERRMSG
3054 006532 012737 020714 002200 MOV #ERR12,ERRBLK
3055 006540 000261 SEC ;SET C BIT TO FLAG ERROR
3056 006542 C00423 BR 4$ ;TAKE ERROR EXIT
3057 006544 132737 000010 006470 2$: BITB #ROR,1$ ;SEE IF ROR = 0
3058 006552 001416 BEQ 3$ ;BR IF ROR = 0
3059 ;STACK "ROR NOT CLEARED" MSG
3060 ;GTFD EM91,ERR12
3061 ; QUEUE "DEVICE FATAL" ERROR # 19
3062 006554 012737 000001 002172 MOV #T.EDF,ERRTYP
3063 006562 012737 000023 002174 MOV #19,ERRNBR
3064 006570 012737 015501 002176 MOV #EM91,ERRMSG
3065 006576 012737 020714 002200 MOV #ERR12,ERRBLK
3066 006604 000261 SEC ;SET C BIT TO FLAG ERROR
3067 006606 000401 BR 4$ ;TAKE ERROR EXIT
3068 006610 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3069 006612 000205 4$: RTS R5 ;RETURN
3070
3071
3072
3073

```

....CKSEOM CHECK RSOM, REOM

```

3065 .SBTTL ....CKSEOM -- CHECK RSOM, REOM
3066 ;*****
3067 ;* CKSEOM - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF RSOM, REOM IN THE
3068 ;* USYRT RECEIVER STATUS REG (RDSRH) AND REPORTS AN ERROR IF THEY ARE NOT
3069 ;* PROPERLY SET TO THE STATES OF BITS 0,1 IN THE WORD FOLLOWING THE CALL.
3070 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3071 ;* IS STACKED, AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3072 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3073 ;*
3074 ;* CALLING SEQUENCE :
3075 ;* JSR R5,CKSEOM
3076 ;* <BIT 0 IS EXPECTED VALUE OF RSOM, BIT 1 IS VALUE OF REOM>
3077 ;*****
3078 006614 CKSEOM:
3079 006614 012737 000007 002336 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3080 006622 004537 003566 JSR R5,READI ;READ USYRT RECEIVER STATUS
3081 006626 120401 RDSRH
3082 006630 000000 .WORD 0
3083 006632 032725 000001 1$: BIT #BIT0,(R5) ;GET EXPECTED STATE OF RSOM
3084 006636 001422 BEQ 2$ ;BR IF EXPECTED RSOM = 0
3085 006640 132737 000001 006630 BITB #RSOM,1$ ;SEE IF RSOM = 1
3086 006646 001040 BNE 3$ ;BR IF RSOM = 1
3087 ;STACK "RSOM NOT SET" MSG
3088 006650 GTDF EM29,ERR12
; QUEUE "DEVICE FATAL" ERROR # 20
MOV #T.EDF,ERRTYP
MOV #20,ERRNBR
MOV #EM29,ERRMSG
MOV #ERR12,ERRBLK
006650 012737 000001 002172
006656 012737 000024 002174
006664 012737 014427 002176
006672 012737 020714 002200
3089 006700 000261 SEC ;SET C BIT TO FLAG ERROR
3090 006702 000473 BR 6$ ;TAKE ERROR EXIT
3091 006704 132737 000001 006630 2$: BITB #RSOM,1$ ;SEE IF RSOM = 0
3092 006712 001416 BEQ 3$ ;BR IF RSOM = 0
3093 ;STACK "RSOM NOT CLEARED" MSG
3094 006714 GTDF EM28,ERR12
; QUEUE "DEVICE FATAL" ERROR # 21
MOV #T.EDF,ERRTYP
MOV #21,ERRNBR
MOV #EM28,ERRMSG
MOV #ERR12,ERRBLK
006714 012737 000001 002172
006722 012737 000025 002174
006730 012737 014406 002176
006736 012737 020714 002200
3095 006744 000261 SEC ;SET C BIT TO FLAG ERROR
3096 006746 000451 BR 6$ ;TAKE ERROR EXIT
3097 006750 032765 000002 177776 3$: BIT #BIT1, 2(R5) ;GET EXPECTED STATE OF REOM
3098 006756 001422 BEQ 4$ ;BR IF EXPECTED REOM = 0
3099 006760 132737 000002 006630 BITB #REOM,1$ ;SEE IF REOM = 1
3100 006766 001040 BNE 5$ ;BR IF REOM = 1
3101 ;STACK "REOM NOT SET" MSG
3102 006770 GTDF EM31,ERR12
; QUEUE "DEVICE FATAL" ERROR # 22
MOV #T.EDF,ERRTYP
MOV #22,ERRNBR
MOV #EM31,ERRMSG
MOV #ERR12,ERRBLK
006770 012737 000001 002172
006776 012737 000026 002174
007004 012737 014465 002176
007012 012737 020714 002200
3103 007020 000261 SEC ;SET C BIT TO FLAG ERROR
3104 007022 000423 BR 6$ ;TAKE ERROR EXIT
3105 007024 132737 000002 006630 4$: BITB #REOM,1$ ;SEE IF REOM = 0
3106 007032 001416 BEQ 5$ ;BR IF REOM = 0

```

....CKSEOM - CHECK RSOM, REOM

```

3107
3108 007034      ;STACK "REOM NOT CLEARED" MSG
                  GTDF      EM30,ERR12
007034 012737 000001 002172
007042 012737 000027 002174
007050 012737 014444 002176
007056 012737 020714 002200
3109 007064 000261
3110 007066 000401
3111 007070 000241
3112 007072 000205
3113
3114

```

```

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

```

```

SEC
BR 6$
5$: CLC
6$: RTS R5

```

....CHKTSO - CHECK TRANSMIT SERIAL OUT BIT

```

3116 .SBTTL ....CHKTSO -- CHECK TRANSMIT SERIAL OUT BIT
3117 ;*****
3118 ;* CHKTSO - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF TSO IN THE USYRT
3119 ;* STATUS REGISTER, AND SETS THE "C" BIT IF IT IS NOT SET TO THE STATE
3120 ;* OF BIT 0 IN THE WORD FOLLOWING THE CALL.
3121 ;*
3122 ;* CALLING SEQUENCE :
3123 ;* JSR R5,CHKTSO
3124 ;* .WORD <BIT 0 IS EXPECTED VALUE OF TSO>
3125 ;*****
3126 007074 CHKTSO:
3127 007074 012737 000007 002336 MOV #7,REGNUM ;SET REG NO. FOR POSSIBLE ERROR REPORT
3128 007102 004537 003566 JSR R5,READI ;READ USYRT STATUS
3129 007106 122000 USTATR
3130 007110 000000 1$: .WORD 0
3131 007112 032725 000001 BIT #BIT0,(R5) ;GET EXPECTED STATE OF TSO
3132 007116 001422 BEQ 2$ ;BR IF EXPECTED TSO = 0
3133 007120 132737 000010 007110 BITB #TSO,1$ ;SEE IF TSO = 1
3134 007126 001040 BNE 3$ ;BR IF TSO = 1
3135 ;*** STACK "TSO NOT SET" ERROR ***
3136 007130 GTDF EM100,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 24
; MOV #T.EDF,ERRTYP
; MOV #24,ERRNBR
; MOV #EM100,ERRMSG
; MOV #ERR12,ERRBLK
007130 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
007136 012737 000030 002174 BR 4$ ;TAKE ERROR EXIT
007144 012737 015536 002176
007152 012737 020714 002200
3137 007160 000261 SEC ;SET C BIT TO FLAG ERROR
3138 007162 000423 BR 4$ ;TAKE ERROR EXIT
3139
3140 007164 132737 000010 007110 2$: BITB #TSO,1$ ;SEE IF TSO = 0
3141 007172 001416 BEQ 3$ ;BR IF TSO = 0
3142 ;*** STACK "TSO NOT CLEARED" ERROR ***
3143 007174 GTDF EM101,ERR12 ;
; QUEUE "DEVICE FATAL" ERROR # 25
; MOV #T.EDF,ERRTYP
; MOV #25,ERRNBR
; MOV #EM101,ERRMSG
; MOV #ERR12,ERRBLK
007174 012737 000001 002172 SEC ;SET C BIT TO FLAG ERROR
007202 012737 000031 002174 BR 4$ ;TAKE ERROR EXIT
007210 012737 015556 002176
007216 012737 020714 002200
3144 007224 000261 SEC ;SET C BIT TO FLAG ERROR
3145 007226 000401 BR 4$ ;TAKE ERROR EXIT
3146 007230 000241 3$: CLC ;CLEAR C BIT FOR NO ERRORS
3147 007232 000205 4$: RTS R5 ;RETURN
3148

```

....INITRN INIT TRANS SSION OF A MESSAGE

```

3150 .SBTTL ....INITRN -- INIT TRANSMISSION OF A MESSAGE
3151 ;*****
3152 ;* INITRN - THIS SUBROUTINE INITIATES TRANSMISSION OF A MESSAGE, BY LOADING
3153 ;* THE USYRT PCSARL,H AND THE PCR WITH THE DATA PASSED IN THE 2 WORDS
3154 ;* FOLLOWING THE CALL ; LOADING AND CLOCKING 1 SOM UNTIL THE FIRST
3155 ;* SYNCH OR FLAG HAS BEEN SERIALIZED IN THE USYRT. THE PROGRAM MONITORS
3156 ;* ALL THE FLAGS IN THE USYRT STATUS REGISTER THROUGHOUT THE PROCESS.
3157 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION IS STACKED
3158 ;* AND THE C-BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE DISCRETION
3159 ;* OF THE CALLING ROUTINE OR SUBROUTINE.
3160 ;*
3161 ;* CALLING SEQUENCE :
3162 ;* JSR R5,INITRN
3163 ;* .WORD <VALUE TO LOAD INTO USYRT PCSARL,H>
3164 ;* .WORD <VALUE TO LOAD INTO USYRT PCR (PASSED IN LO BYTE)>
3165 ;* <SPECIAL VIAORB MASKING VALUE (PASSED IN HI BYTE)>
3166 ;*****
3167 007234 INITRN:
3168 007234 010146 MOV R1,-(SP) ;SAVE R1
3169 007236 004537 003712 JSR R5,WRITEI ;RESET THE USYRT
3170 007242 120000 VIAORB
3171 007244 000031 RTSND!DTR!PRESET
3172 007246 004537 003712 JSR R5,WRITEI ;CLEAR USYRT RESET BIT
3173 007252 120000 VIAORB
3174 007254 000030 RTSND!DTR
3175 007256 112537 007270 MOVB (R5)+,1$ ;GET VALUE TO LOAD INTO USYRT PCSARL
3176 007262 004537 003712 JSR R5,WRITEI ;LOAD USYRT PCSARL
3177 007266 120404 PCSARL
3178 007270 000000 1$: .WORD 0
3179 007272 112537 007304 MOVB (R5)+,2$ ;GET VALUE TO LOAD INTO PCSARM
3180 007276 004537 003712 JSR R5,WRITEI ;LOAD USYRT PCSARM
3181 007302 120405 PCSARM
3182 007304 000000 2$: .WORD 0
3183 007306 112537 007332 MOVB (R5)+,3$ ;GET VALUE TO LOAD INTO PCR
3184 007312 005037 002402 CLR SAVLEN
3185 007316 113737 007332 002402 MOVB 3$,SAVLEN ;SAVE CHAR LENGTH BITS
3186 007324 004537 003712 JSR R5,WRITEI ;LOAD USYRT PCR
3187 007330 120407 PCR
3188 007332 000000 3$: .WORD 0
3189 007334 004537 003712 JSR R5,WRITEI ;SET ACR FOR T1 ONE-SHOT MODE
3190 007340 120013 VIAACR
3191 007342 000200 200
3192 007344 004537 003712 JSR R5,WRITEI ;LOAD VIA T1L-L
3193 007350 120006 VIAT1C
3194 007352 000300 300
3195 007354 004537 003712 JSR R5,WRITEI ;LOAD VIA T1L H
3196 007360 120007 VIAT1D
3197 007362 000000 000
3198 007364 004537 005410 JSR R5,CKUSTS ;CHK USYRT STATUS FOR INIT'D STATE
3199 007370 000110 110 ; TBMT = 1, TSO = 1
3200 007372 103454 BCS 7$ ;IF ERROR, EXIT SUBROUTINE
3201
3202 007374 013737 007530 007414 MOV 20$,13$ ;* SET UP DEFAULT VIAORB PARAMETERS
3203 007402 142537 007414 BICB (R5)+,13$ ;* CLEAR ANY SPECIFIED VIAORB BITS.
3204
3205 007406 004537 003712 JSR R5,WRITEI ;SET UP USYRT
3206 007412 120000 VIAORB

```


....INITRN INIT TRANSMISSION OF A MESSAGE

```

3207 007414 000142          13$:  TXEN!RXEN!TTLOOP          ;* THIS VALUE MIGHT BE MODIFIED ABOVE
3208
3209 007416 004537 003712    JSR      R5,WRITEFI          ;SET TSOM IN USYRT
3210 007422 120403          TDSRH
3211 007424 000001          TSOM
3212 007426 004537 003712    JSR      R5,WRITEI          ;LOAD SYNCH CHAR INTO TX BUF
3213 007432 120402          TDSRL
3214 007434 000226          SYNCH
3215 007436 004537 006014    JSR      R5,CKTBMT          ;CHK FOR TBMT = 0
3216 007442 000000          0
3217 007444 103427          BCS      7$                ;IF ERROR, EXIT SUBROUTINE
3218 007446 005001          CLR      R1                ;INIT CYCLE COUNTER
3219 007450 004537 012072    4$:     JSR      R5,STEPLU          ;CLOCK LU FOR 1 CYCLE
3220 007454 000001          1
3221 007456 004537 003566    JSR      R5,READI          ;READ USYRT STATUS REG
3222 007462 122000          USTATR
3223 007464 000000          .WORD   0
3224 007466 132737 000100 007464 5$:     BITB    @TBMT,5$          ;SEE IF TBMT IS SET YET
3225 007474 001010          BNE     6$                ;BR IF YES
3226 007476 005201          INC     R1                ;INCR CYCLE COUNTER
3227 007500 020127 000003    CMP     R1,#3             ;SEE IF 3 CYCLES DONE YET
3228 007504 002761          BLT    4$                ;BR IF LESS THAN 3 CYCLES
3229 007506 004537 006014    JSR      R5,CKTBMT          ;GO STACK "TBMT NOT SET" MSG
3230 007512 000001          1
3231 007514 103403          BCS     7$                ;IF ERROR, EXIT SUBROUTINE
3232 007516 004537 005514    6$:     JSR      R5,CKTACT          ;CHK FOR TFACT = 1
3233 007522 000001          1
3234 007524 012601          7$:     MOV     (SP)+,R1        ;RESTORE R1
3235 007526 000205          RTS     R5                ;RETURN (IF C = 1, WE HAD AN ERROR)
3236
3237 007530 000142          20$:   TXEN!RXEN!TTLOOP          ;DEFAULT VALUE FOR VIAORB: ENABLE
3238
3239

```

6

....CKLPBK -- DETERMINE IF TEST CAN BE RUN

3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262 007532
3263 007532 032725 100000
3264 007536 001407
3265 007540 005737 002472
3266 007544 001002
3267 007546 000137 010072
3268 007552 000137 010076
3269
3270 007556 023727 002472 000004
3271 007564 001002
3272 007566 000137 010076
3273
3274
3275
3276 007572 026527 177776 000001
3277 007600 001406
3278 007602 026527 177776 000002
3279 007610 001422
3280 007612 000137 007742
3281
3282
3283 007616 005737 002470
3284 007622 001402
3285 007624 000137 010040
3286 007630 005737 002472
3287 007634 001406
3288 007636 023727 002472 000001
3289 007644 001402
3290 007646 000137 010006
3291 007652 000137 010072
3292
3293
3294 007656 005737 002470
3295 007662 001002
3296 007664 000137 010040
3297 007670 005737 002472

```

.SBTTL ....CKLPBK -- DETERMINE IF TEST CAN BE RUN
;*****
;* CKLPBK - THIS SUBROUTINE DETERMINES IF THE TEST CALLING IT CAN BE RUN. THE
;* TEST PASSES THE DESIRED MODEM INTERFACE TYPE IN THE WORD FOLLOWING THE
;* CALL, AND IF A PROPER EXTERNAL LOOPBACK HAS BEEN PROVIDED BY THE
;* OPERATOR FOR THAT INTERFACE, AND IF THE BAUD RATE IS CORRECT, A RETURN
;* IS MADE WITH THE C BIT CLEARED, TO RUN THE TEST. IF NOT, A RETURN IS
;* MADE WITH THE C BIT SET TO 1, SO THAT THE TEST CAN BE SKIPPED.
;*
;* IF BIT 15 IS SET IN THE WORD FOLLOWING THE CALL, THE TEST WILL NOT
;* BE RUN UNLESS THE H3254 AND H3255 TEST CONNECTORS ARE INSTALLED.
;*
;* IF THE PROGRAM PASSES '0' IN THE WORD FOLLOWING THE CALL, THE SUBRTN
;* WILL ATTEMPT TO RUN WHICHEVER MODEM INTERFACE IS SELECTED BY CABLE
;* OR TEST CONNECTOR.
;*
;* CALLING SEQUENCE :
;*   JSR R5,CKLPBK
;*   .WORD <DESIRED MODEM INTERFACE INFO>
;*****
CKLPBK:
      BIT      @TCCHEK,(R5)  ;SEE IF H3254,5 CHECK IS DESIRED
      BEQ      2$           ;BR IF NOT
      TST      TSTCON       ;SEE IF H3254,5 INSTALLED
      BNE      1$           ;BR IF NOT
      JMP      46$          ;BR TO RUN TEST
1$:    JMP      48$          ;GO TO SKIP TEST
;IF NO EXTERNAL LPBK, SKIP TEST
2$:    CMP      TSTCON,#4    ;SEE IF NO LPBK
      BNE      3$           ;BR IF LOOPBACK
      JMP      48$          ;GO TO SKIP TEST

;*** SEE IF AN INTERFACE IS REQUESTED ***
3$:    CMP      -2(R5),#INTGRL ;SEE IF INTEGRAL MODEM REQUESTED
      BEQ      8$           ;BR IF INTGRL MODEM REQUESTED
      CMP      -2(R5),#EIAV35 ;SEE IF V.35 OR EIA REQUESTED
      BEQ      16$          ;BR IF V.35 REQUESTED
      JMP      32$          ;NONE REQUESTED, FIND AN INTERFACE TO TEST

;SEE IF INTEGRAL MODEM CAN BE RUN
8$:    TST      BRDTYP       ;SEE IF M8064
      BEQ      10$          ;BR IF M8064
      JMP      42$          ;WRONG OPTION, GO TO SKIP TEST
10$:   TST      TSTCON       ;SEE IF H3254, H3255 USED
      BEQ      12$          ;BR IF YES
      CMP      TSTCON,#1    ;SEE IF OPERATOR SPEC'D INTEGRAL MODEM
      BEQ      12$          ;BR IF YES, TO RUN TEST
      JMP      40$          ;WRONG INTERFACE, GO SKIP TEST
12$:   JMP      46$          ;GO TO RUN TEST

;SEE IF V.35 OR EIA CAN BE RUN
16$:   TST      BRDTYP       ;SEE IF M8053 BOARD
      BNE      18$          ;BR IF M8053
      JMP      42$          ;WRONG OPTION, GO TO SKIP TEST
18$:   TST      TSTCON       ;SEE IF H3254, H3255 USED

```

....CKLPBK -- DETERMINE IF TEST CAN BE RUN

```

3298 007674 001002          BNE      23$          ;BR IF NOT
3299 007676 000137 010072    20$:      JMP      46$          ;GO RUN THE TEST
3300 007702 023727 002472 000003 23$:      CMP      TSTCON,#3    ;SEE IF OPERATOR SPEC'D V.35
3301 007710 001006          BNE      28$          ;BR IF NO
3302 007712 023727 002470 000001      CMP      BRDTYP,#1    ;TSTCOM MATCH BRDTYP?
3303 007720 001766          BEQ      20$          ;YES: RUN TEST
3304 007722 000137 010006          JMP      40$          ;WRONG INTERFACE, GO SKIP TEST
3305
3306 007726 023727 002472 000002 28$:      CMP      TSTCON,#2    ;SEE IF OPERATOR SPEC'D EIA
3307 007734 001760          BEQ      20$          ;BR IF YES, TO RUN EIA
3308 007736 000137 010006          JMP      40$          ;WRONG INTERFACE, GO SKIP TEST
3309
3310          ;*** NO INTERFACE REQUESTED  FIND ONE TO TEST ***
3311
3312 007742 005737 002470    32$:      TST      BRDTYP      ;SEE IF INTEGRAL MODEM SELECTED
3313 007746 001343          BNE      16$          ;BR IF NOT (TEST FOR V35/EIA)
3314 007750 000137 007652          JMP      12$          ;SEE IF INTEGRAL MODEM CAN BE RUN
3315
3316          ;PRINT "FOR BAUD RATE SPECIFIED,"
3317 007754    38$:      CMP      STARES,#1    ;SEE IF THIS IS FIRST PASS SINCE STA OR RES
3318 007754 023727 002412 000001      BNE      50$          ;BR IF NOT, TO SKIP PRINTING
3319 007762 001063          PRINTF  #FMT30
3320 007764 012746 013762          MOV      #FMT30,(SP)
3321 010004 000434          MOV      #1,(SP)
3322          ;PRINT "IMPROPER CONNECTOR TYPE SPECIFIED"
3323 010006    40$:      MOV      SP,R0
3324 010006 023727 002412 000001      TRAP    C:PNTF
3325 010014 001046          ADD     #4,SP
3326 010016 012746 014017          BR      48$          ;GO TO PRINT "TEST NOT RUN"
3327 010022 012746 000001          ;PRINT "FOR OPTION SPECIFIED,"
3328 010026 010600          MOV      #FMT31,(SP)
3329 010030 104417          MOV      #1,(SP)
3330 010032 062706 000004          MOV      SP,R0
3331 010036 000417          TRAP    C:PNTF
3332          ;PRINT "FOR OPTION SPECIFIED,"
3333 010040 023727 002412 000001 42$:      CMP      STARES,#1    ;SEE IF THIS IS FIRST PASS SINCE STA OR RES
3334 010046 001031          BNE      50$          ;BR IF NOT TO SKIP PRINTING
3335 010050          PRINTF  #FMT32
3336 010050 012746 014065          MOV      #FMT32,(SP)
3337 010054 012746 000001          MOV      #1,(SP)
3338 010060 010600          MOV      SP,R0
3339 010062 104417          TRAP    C:PNTF
3340 010064 062706 000004          ADD     #4,SP
3341 010070 000402          BR      48$          ;GO TO PRINT "TEST NOT RUN"
3342
3343          ;*** BRANCH HERE TO RUN TEST ***
3344 010072 000241    46$:      CLC
3345 010074 000417          BR      52$          ;CLEAR C BIT TO RUN TEST
3346
3347          ;*** BRANCH HERE TO SKIP TEST ***

```


....TXCHAR - TRANSMIT A CHARACTER

```

3351 .SBTTL ....TXCHAR -- TRANSMIT A CHARACTER
3352 ;*****
3353 ;* TXCHAR - THIS SUBROUTINE INITIATES TRANSMISSION OF A CHAR BY LOADING
3354 ;* THE USYRT TDSRL WITH THE DATA PASSED IN THE LO BYTE OF THE WORD
3355 ;* FOLLOWING THE CALL, AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES
3356 ;* PASSED IN THE SECOND WORD FOLLOWING THE CALL. THE PROGRAM CONTINUALLY
3357 ;* MONITORS TBMT AND TXACT THROUGHOUT THE PROCESS.
3358 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3359 ;* IS STACKED, AND THE C BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3360 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3361 ;*
3362 ;* CALLING SEQUENCE :
3363 ;* .ISR      R5,TXCHAR
3364 ;* .WORD    <DATA FOR TDSRL IN LO BYTE>
3365 ;* .WORD    <NUMBER OF CYCLES TO CLOCK (IN LO BYTE)>
3366 ;*          <SWITCH TO DISABLE INITIAL TBMT=0 CHECK (MSB IN HI BYTE)>
3367 ;*****
3368 TXCHAR:
3369 010136 010146      MOV     R1,-(SP)      ;SAVE R1
3370 010140 010246      MOV     R2,-(SP)      ;SAVE R2
3371 010142 012537 010154  MOV     (R5)+,1$     ;GET DATA FOR TDSRL
3372 010146 004537 003712  JSR     R5,WRITEI    ;LOAD DATA INTO TDSRL
3373 010152 120402      TDSRL
3374 010154 000000      1$: .WORD    0
3375 010156 005001      CLR     R1           ;INIT CYCLE COUNT AND CLEAR C BIT
3376 010160 005002      CLR     R2           ;CLEAR REQ'D CYCLE COUNT
3377 010162 112502      MOVB   (R5)+,R2     ;GET DESIRED NO. OF CYCLES
3378 010164 001425      BEQ    6$           ;BR IF NO CLOCKING DONE
3379 010166 004537 005514  3$: JSR     R5,CKTACT   ;CHECK TXACT = 1
3380 010172 000001      1
3381 010174 103421      BCS    6$           ;BR TO EXIT IF ERROR
3382 010176 020102      CMP    R1,R2       ;SEE IF REQUIRED CYCLES DONE YET
3383 010200 001414      BEQ    5$           ;BR IF YES
3384
3385 010202 131527 000200  BITB   (R5),#NCTBMT ;* CHECK FOR "TBMT=0 CHECK" DISABLE
3386 010206 001004      BNE    7$           ;* BR IF MSB IS NOT SET
3387
3388 010210 004537 006014  JSR     R5,CKTBMT   ;CHECK FOR TBMT = 0
3389 010214 000000      0
3390 010216 103410      BCS    6$           ;BR TO EXIT IF ERROR
3391 010220 004537 012072  7$: JSR     R5,STEPLU   ;CLOCK LU FOR 1 CYCLE
3392 010224 000001      1
3393 010226 005201      INC    R1           ;INCR CYCLE COUNT
3394 010230 000756      BR     3$           ;KEEP CLOCKING
3395 010232 004537 006014  5$: JSR     R5,CKTBMT   ;CHK TBMT = 1
3396 010236 000001      1
3397 010240 012602      6$: MOV    (SP)+,R2     ;RESTORE R2
3398 010242 012601      MOV    (SP)+,R1     ;RESTORE R1
3399 010244 005205      INC    R5           ;ADJUST R5 FOR SANE RETURN
3400 010246 000205      RTS    R5           ;RETURN (WITH C BIT = 1 IF ERROR)
3401
3402
3403
3404

```

....TXCTRL CONTROL MESSAGE TRANSMISSION (TDSRH)

```

3406 .SBTTL ....TXCTRL -- CONTROL MESSAGE TRANSMISSION (TDSRH)
3407 ;*****
3408 ;* TXCTRL - THIS SUBROUTINE ALLOWS CONTROL OF MESSAGE TRANSMISSION BY LOADING
3409 ;* THE USYRT TDSRH WITH THE DATA PASSED IN THE LO BYTE OF THE WORD
3410 ;* FOLLOWING THE CALL, AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES
3411 ;* PASSED IN THE SECOND WORD FOLLOWING THE CALL. THE PROGRAM CONTINUALLY
3412 ;* MONITORS TBMT AND TXACT THROUGHOUT THE PROCESS.
3413 ;* IF THE SUBROUTINE DETECTS AN ERROR, THE ERROR INFORMATION
3414 ;* IS STACKED, AND THE C BIT SET, WHICH LEAVES THE ERROR REPORTING AT THE
3415 ;* DISCRETION OF THE CALLING ROUTINE OR SUBROUTINE.
3416 ;*
3417 ;* CALLING SEQUENCE :
3418 ;* JSR R5,TXCTRL
3419 ;* .WORD <DATA FOR TDSRH IN LO BYTE>
3420 ;* .WORD <NUMBER OF CYCLES TO CLOCK>
3421 ;*****
3422 TXCTRL:
3423 010250 MOV R1,-(SP) ;SAVE R1
3424 010252 MOV R2,-(SP) ;SAVE R2
3425 010254 012537 010266 MOV (R5),R2 ;GET DATA FOR TDSRH
3426 010260 004537 003712 JSR R5,WRITEI ;LOAD DATA INTO TDSRH
3427 010264 120403 TDSRH
3428 010266 000000 2: .WORD 0
3429 010270 005001 CLR R1 ;INIT CYCLE COUNT AND CLEAR C BIT
3430 010272 012502 MOV (R5),R2 ;GET DESIRED NO. OF CYCLES
3431 010274 001422 BEQ 6: ;BR IF NO CLOCKING DONE
3432 010276 004537 005514 3: JSR R5,CKTACT ;CHECK TXACT = 1
3433 010302 000001 1
3434 010304 103416 BCS 6: ;BR TO EXIT IF ERROR
3435 010306 020102 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
3436 010310 001411 BEQ 5: ;BR IF YES
3437 010312 004537 006014 JSR R5,CKTBMT ;CHECK FOR TBMT = 0
3438 010316 000000 0
3439 010320 103410 BCS 6: ;BR TO EXIT IF ERROR
3440 010322 004537 012072 JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
3441 010326 000001 1
3442 010330 005201 INC R1 ;INCR CYCLE COUNT
3443 010332 000761 BR 3: ;KEEP CLOCKING
3444 010334 004537 006014 5: JSR R5,CKTBMT ;CHK TBMT = 1
3445 010340 000001 1
3446 010342 012602 6: MOV (SP),R2 ;RESTORE R2
3447 010344 012601 MOV (SP),R1 ;RESTORE R1
3448 010346 000205 RTS R5 ;RETURN (WITH C BIT = 1 IF ERROR)
3449
3450

```

....RXCHAR - RECEIVE A CHARACTER

3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471 010350
3472 010350 010146
3473 010352 010246
3474 010354 004537 003566
3475 010360 120401
3476 010362 000000
3477 010364 004537 003566
3478 010370 120400
3479 010372 000000
3480 010374 111501
3481 010376 042701 177400
3482 010402 023727 002402 000347
3483 010410 001005
3484 010412 142737 000200 010372
3485 010420 142701 000200
3486 010424 123701 010372
3487 010430 001462
3488 010432 004537 003566
3489 010436 122000
3490 010440 000000
3491 010442 132737 000002 010440
3492 010450 001421
3493 010452 012737 000007 002336
3494
3495 010460

010460 012737 000001 002172
010466 012737 000032 002174
010474 012737 014626 002176
010502 012737 020714 002200
3496 010510 000137 011610
3497 010514 005037 002336
3498 010520 005037 002324
3499 010524 110137 002324
3500 010530 005037 002326
3501 010534 113737 010372 002326
3502
3503 010542

```

.SBTTL ....RXCHAR -- RECEIVE A CHARACTER
;*****
; * RXCHAR  THIS SUBROUTINE READS THE USYRT RDSR AND CHECKS THE CONTENTS
; *        AGAINST THE DATA PASSED IN THE WORD FOLLOWING THE CALL.
; *        IF BIT0 = 0 IN THE SECOND WORD FOLLOWING THE CALL, THE RERR BIT IS
; *        NOT CHECKED AGAINST THE EXPECTED VALUE. THEN, IT CLOCKS
; *        THE LINE UNIT FOR THE NO. OF CYCLES PASSED IN THE THIRD WORD
; *        FOLLOWING THE CALL. THE PROGRAM CONTINUALLY MONITORS RDA AND RXACT.
; *        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,RXCHAR
; *        .WORD   <EXPECTED RDSRL IN LO BYTE, RDSRH IN HI BYTE>
; *        .WORD   <=0 FOR NO RERR CHK, =1 FOR RERR CHK>
; *        .WORD   <NUMBER OF CYCLES TO CLOCK (IN LO BYTE)>
; *        .WORD   <SPECIAL DISABLE SWITCHES: NOCRDA,NFCRDA,NCRACK(IN HI BYTE)>
;*****
RXCHAR:
MOV      R1,-(SP)      ;SAVE R1
MOV      R2,-(SP)      ;SAVE R2
JSR      R5,READI      ;READ RDSRH
RDSRH
2$:      .WORD      0
JSR      R5,READI      ;READ RDSRL
RDSRL
1$:      .WORD      0
MOVB    (R5),R1        ;GET EXPECTED RDSRL
BIC     @177400,R1     ;MASK OFF UNUSED BITS
CMP     SAVLEN,@TXDL!RXDL ;SEE IF 7 BIT CHARS BEING USED
BNE     3$            ;BR IF NOT 7-BIT CHARS
BICB   @BIT7,1$      ;CLEAR 8TH BIT FOR COMPARE
BICB   @BIT7,R1
3$:      CMPB     1$,R1      ;COMPARE RCV'D CHAR TO EXPECTED
BEQ     6$            ;BR IF MATCH
JSR     R5,READI      ;READ USYRT STATUS REG
USTATR
4$:      .WORD      0
BITB   @TXU,4$       ;SEE IF TX UNDERRUN OCCURRED
BEQ     5$            ;BR IF NOT
MOV     @7,REGNUM     ;SET USYRT REG NO. FOR STATUS REG
;STACK "TX UNDERRUN" ERROR
GDF     EM54,ERR12
;
;        QUEUE "DEVICE FATAL ERROR # 26
MOV     @T.EDF,ERR1P
MOV     @26,ERRNBR
MOV     @EM54,ERRMSG
MOV     @ERR12,ERRBLK
5$:      JMP      20$
CLR     REGNUM        ;SET USYRT REG NO. FOR RDSRL
CLR     GDATA         ;SET EXPECTED DATA
MOVB   R1,GDATA
CLR     BDATA         ;SET ACTUAL DATA
MOVB   1$,BDATA
;STACK "RCV'D DATA MISCOMPARE" ERROR
GDF     EM34,ERR10

```

...RXCHAR RECEIVE A CHARACTER

```

010542 012737 000001 002172
010550 012737 000033 002174
010556 012737 014502 002176
010564 012737 020364 002200
3504 010572 000137 011610
3505 010576 116501 000001
3506 010602 042701 177400
3507 010606 123701 010362
3508 010612 001016
3509 010614 000137 011474
3510 010620 012737 000001 002336
3511 010626 005037 002324
3512 010632 110137 002324
3513 010636 005037 002326
3514 010642 113737 010362 002326
3515 010650 012737 000001 002336
3516 010656 032765 000001 000002
3517 010664 001447
3518
3519 010666 132701 000200
3520 010672 001022
3521 010674 132737 000200 010362
3522 010702 001440
3523
3524 010704

010704 012737 000001 002172
010712 012737 000034 002174
010720 012737 014530 002176
010726 012737 020714 002200
3525 010734 000137 011610
3526 010740 132737 000200 010362
3527 010746 001016
3528
3529 010750

010750 012737 000001 002172
010756 012737 000035 002174
010764 012737 014551 002176
010772 012737 020714 002200
3530 011000 000137 011610
3531
3532 011004 132701 000010
3533 011010 001022
3534 011012 132737 000010 010362
3535 011020 001440
3536
3537 011022

011022 012737 000001 002172
011030 012737 000036 002174
011036 012737 014340 002176
011044 012737 020714 002200
3538 011052 000137 011610
3539 011056 132737 000010 010362
3540 011064 001016

```

```

6$: JMP 20$
MOV 1(R5),R1
BIC #177400,R1
CMPB 2$,R1
BNE 7$
JMP 17$
MOV #1,REGNUM
CLR GDATA
MOV 2$,R1
CLR BDATA
MOV 2$,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
GTDF EM35,ERR12

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

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

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

```

```

; QUEUE "DEVICE FATAL" ERROR # 27
MOV #T.EDF,ERRTYP
MOV #27,ERRNBR
MOV #EM34,ERRMSG
MOV #ERR10,ERRBLK

;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 # 28
MOV #T.EDF,ERRTYP
MOV #28,ERRNBR
MOV #EM35,ERRMSG
MOV #ERR12,ERRBLK

;TAKE ERROR EXIT
;SEE IF ACTUAL BIT = 1
;BR IF YES

; QUEUE "DEVICE FATAL" ERROR # 29
MOV #T.EDF,ERRTYP
MOV #29,ERRNBR
MOV #EM36,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 # 30
MOV #T.EDF,ERRTYP
MOV #30,ERRNBR
MOV #EM16,ERRMSG
MOV #ERR12,ERRBLK

;TAKE ERROR EXIT
;SEE IF ACTUAL BIT = 1
;BR IF YES

```


....RXCHAR RECEIVE A CHARACTER

```

3541          ;STACK "ROR NOT SET" MSG
3542 011066      GTDF      EM14,ERR12
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 31
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #31,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
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 32
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #32,ERRNBR
                ;   MOV      #EM39,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF ACTUAL BIT = 1
                ;   BR IF YES
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 33
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #33,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      #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      #EM31,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF EXPECTED BIT = 1
                ;   BR IF YES
                ;
011066 012737 000001 002172
011074 012737 000037 002174
011102 012737 014324 002176
011110 012737 020714 002200
3543 011116 000137 011610
3544          ;CHECK RABGA BIT
3545 011122 132701 000004      11$:  JMP      20$
3546 011126 001022          ;BITB  #RABGA,R1
3547 011130 132737 000004 010362  ;BNE  12$
3548 011136 001440          ;BITB  #RABGA,2$
3549          ;BEQ  13$
3550 011140      ;STACK "RABGA NOT CLEARED" MSG
                GTDF      EM39,ERR12
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 31
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #31,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
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 32
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #32,ERRNBR
                ;   MOV      #EM39,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF ACTUAL BIT = 1
                ;   BR IF YES
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 33
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #33,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      #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      #EM31,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF EXPECTED BIT = 1
                ;   BR IF YES
                ;
011140 012737 000001 002172
011146 012737 000040 002174
011154 012737 014566 002176
011162 012737 020714 002200
3551 011170 000137 011610
3552 011174 132737 000004 010362 12$:  JMP      20$
3553 011202 001016          ;BITB  #RABGA,2$
3554          ;BNE  13$
3555 011204      ;STACK "RABGA NOT SET" MSG
                GTDF      EM40,ERR12
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 33
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #33,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      #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      #EM31,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF EXPECTED BIT = 1
                ;   BR IF YES
                ;
011204 012737 000001 002172
011212 012737 000041 002174
011220 012737 014610 002176
011226 012737 020714 002200
3556 011234 000137 011610
3557          ;CHECK REOM BIT
3558 011240 132701 000002      13$:  JMP      20$
3559 011244 001022          ;BITB  #REOM,R1
3560 011246 132737 000002 010362  ;BNE  14$
3561 011254 001440          ;BITB  #REOM,2$
3562          ;BEQ  15$
3563 011256      ;STACK "REOM NOT CLEARED" MSG
                GTDF      EM30,ERR12
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 34
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #34,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      #EM31,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF EXPECTED BIT = 1
                ;   BR IF YES
                ;
011256 012737 000001 002172
011264 012737 000042 002174
011272 012737 014444 002176
011300 012737 020714 002200
3564 011306 000137 011610
3565 011312 132737 000002 010362 14$:  JMP      20$
3566 011320 001016          ;BITB  #REOM,2$
3567          ;BNE  15$
3568 011322      ;STACK "REOM NOT SET" MSG
                GTDF      EM31,ERR12
                ;
                ;   QUEUE "DEVICE FATAL" ERROR # 34
                ;
                ;   MOV      #T.EDF,ERRTYP
                ;   MOV      #34,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      #EM31,ERRMSG
                ;   MOV      #ERR12,ERRBLK
                ;
                ;   TAKE ERROR EXIT
                ;   SEE IF EXPECTED BIT = 1
                ;   BR IF YES
                ;
011322 012737 000001 002172
011330 012737 000043 002174
011336 012737 014465 002176
011344 012737 020714 002200
3569 011352 000137 011610
3570          ;CHECK RSOM BIT
3571 011356 132701 000001      15$:  JMP      20$
3572 011362 001022          ;BITB  #RSOM,R1
                ;BNE  16$

```

....RXCHAR - RECEIVE A CHARACTER

```

3573 011364 132737 000001 010362      BITB   #RSOM,2#      ;SEE IF ACTUAL BIT = 0
3574 011372 001440                      BEQ    17#           ;BR IF YES
3575                                     ;STACK "RSOM NOT CLEARED" MSG
3576 011374                      GTDF   EM28,ERR12
                                     ;
                                     ;   QUEUE "DEVICE FATAL" ERROR # 36
                                     ;   MOV   #T.EDF,ERRTYP
011374 012737 000001 002172                      MOV   #36,ERRNBR
011402 012737 000044 002174                      MOV   #EM28,ERRMSG
011410 012737 014406 002176                      MOV   #ERR12,ERRBLK
011416 012737 020714 002200
3577 011424 000137 011610      JMP    20#           ;TAKE ERROR EXIT
3578 011430 132737 000001 010362 16# :  BITB   #RSOM,2#      ;SEE IF ACTUAL BIT = 1
3579 011436 001016                      BNE   17#           ;BR IF YES
3580                                     ;STACK "RSOM NOT SET" MSG
3581 011440                      GTDF   EM29,ERR12
                                     ;
                                     ;   QUEUE "DEVICE FATAL" ERROR # 37
                                     ;   MOV   #T.EDF,ERRTYP
011440 012737 000001 002172                      MOV   #37,ERRNBR
011446 012737 000045 002174                      MOV   #EM29,ERRMSG
011454 012737 014427 002176                      MOV   #ERR12,ERRBLK
011462 012737 020714 002200
3582 011470 000137 011610      JMP    20#           ;TAKE ERROR EXIT
3583
3584 011474 116502 000004      17# :  MOVB   4(R5),R2      ;GET DESIRED NO. OF CYCLES
3585 011500 005001                      CLR   R1           ;INIT CYCLE COUNT
3586
3587 011502 136527 000005 000040 18# :  BITB   5(R5),#BIT5    ;* IS RXACT CHECK TO BE DISABLED ?
3588 011510 001004                      BNE   31#           ;* BR IF YES
3589 011512 004537 005654      JSR   R5,CKRACT    ;CHK FOR RACT = 1
3590 011516 000001                      1
3591 011520 103433                      BCS   20#           ;BR TO EXIT IF ERROR
3592
3593 011522 020102      31# :  CMP    R1,R2          ;SEE IF REQUIRED CYCLES DONE YET
3594 011524 001415                      BEQ   19#           ;BR IF YES
3595
3596 011526 136527 000005 000200      BITB   5(R5),#BIT7    ;* SEE IF INITIAL RDA CHECK DESIRED
3597 011534 001004                      BNE   22#           ;* BR IF NO
3598 011536 004537 006154      JSR   R5,CKRDA     ;CHK FOR RDA = 0
3599 011542 000000                      0
3600 011544 103421                      BCS   20#           ;BR TO EXIT IF ERROR
3601
3602 011546 004537 012072      22# :  JSR   R5,STEPLU     ;CLOCK LU FOR 1 CYCLE
3603 011552 000001                      1
3604 011554 005201                      INC   R1           ;INCR CYCLE COUNT
3605 011556 000751                      BR    18#          ;CONTINUE CLOCKING
3606
3607 011560 136527 000005 000100 19# :  BITB   5(R5),#BIT6    ;* IS FINAL RDA CHECK TO BE SKIPPED ?
3608 011566 001004                      BNE   30#           ;* BR IF YES
3609 011570 004537 006154      JSR   R5,CKRDA     ;CHK RDA = 1
3610 011574 000001                      1
3611 011576 103404                      BCS   20#           ;BR IF ERROR
3612
3613 011600 062705 000006      30# :  ADD    #6,R5         ;FIX UP RETURN ADRS
3614 011604 000241                      CLC
3615 011606 000403                      BR
3616 011610 062705 000006      20# :  ADD    #6,R5         ;TAKE ERROR-FREE EXIT
3617 011614 000261                      SEC          ;FIX UP RETURN ADDRESS
3618 011616 012602      21# :  MOV    (SP),R2      ;SET C BIT FOR ERROR
3619 011620 012601                      MOV    (SP),R1     ;RESTORE R2
                                     ;RESTORE R1

```

G7

....RXCHAR RECEIVE A CHARACTER

3620 011622 000205
3621

RTS R5

;RETURN

....RCV1ST - RECEIVE FIRST CHARACTER OF MESSAGE

3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638 011624
3639 011624 010146
3640 011626 010246
3641 011630 005001
3642 011632 012502
3643 011634 062702 000003
3644 011640 004537 005654
3645 011644 000000
3646 011646 103446
3647 011650 004537 006154
3648 011654 000000
3649 011656 103442
3650 011660 004537 006614
3651 011664 000000
3652 011666 103436
3653 011670 004537 012072
3654 011674 000001
3655 011676 005201
3656 011700 004537 003566
3657 011704 122000
3658 011706 000000
3659 011710 132737 000200 011706
3660 011716 001006
3661 011720 020102
3662 011722 002762
3663 011724 004537 006154
3664 011730 000001
3665 011732 103414
3666 011734 020165 177776
3667 011740 002004
3668 011742 004537 006154
3669 011746 000000
3670 011750 103405
3671 011752 004537 005654
3672 011756 000001
3673 011760 103401
3674 011762 000241
3675 011764 012602
3676 011766 012601
3677 011770 000205
3678
3679

```

.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$: JSR R5,STEPLU ;CLOCK LU FOR 1 CYCLE
1
INC R1 ;INCREMENT CYCLE COUNT
JSR R5,READI ;READ USYRT STATUS REG
USTATR
2$: .WORD 0
BITB #RDA,2$ ;SEE IF RDA SET YET
3$: BNE 3$ ;BR IF YES
CMP R1,R2 ;SEE IF LIMIT EXCEEDED
BLT 1$ ;BR IF NOT YET
JSR R5,CKRDA ;GO STACK "RDA NOT SET" MSG
1
BCS 6$ ;BR TO EXIT IF ERROR
3$: CMP R1,2(R5) ;SEE IF LESS THAN REQUIRED CYCLES
BGE 4$ ;BR IF NOT
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)

```

....ENTRAN - SHUT DOWN TRANSMITTER/RECEIVER

3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695 011772
3696 011772 012737 000002 012042
3697 012000 112537 012050
3698 012004 105725
3699 012006 100002
3700 012010 005037 012042
3701 012014 004537 005514 4\$:
3702 012020 000001
3703 012022 103422
3704 012024 004537 005654
3705 012030 000001
3706 012032 103416
3707 012034 004537 003712
3708 012040 120000
3709 012042 000002 1\$:
3710 012044 004537 012072
3711 012050 000000 2\$:
3712 012052 004537 005514
3713 012056 000000
3714 012060 103403
3715 012062 004537 005654
3716 012066 000000
3717 012070 000205 6\$:
3718
3719

```

.SBTTL ....ENTRAN -- SHUT DOWN TRANSMITTER/RECEIVER
;*****
;* ENTRAN - 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.
;*
;* CALLING SEQUENCE :
;* JSR R5,ENTRAN
;* MSB SET=NO TTLOOP ! LOWER BYTE = <NO. OF CYCLES TO CLOCK>
;*****
ENTRAN:
MOV #TTLOOP,1$ ;INIT DEFAULT VIAORB (TTLOOP=1)
MOV# (R5),2$ ;GET DESIRED # OF TICKS (LOWER BYTE)
TST# (R5), ;SEE IF MSB SET (TTLOOP DISABLE BIT)
BPL 4$ ;IS IT?
CLR 1$ ;IF YES: CLEAR VIAORB VALUE
4$: JSR R5,CKTACT ;CHK FOR TXACT = 1
1
BCS 6$ ;BR IF ERROR
JSR R5,CKRACT ;CHK FOR RXACT = 1
1
BCS 6$
JSR R5,WRITEI ;CLEAR TXEN AND RXEN IN USYRT
VIAORB ;
1$: TTLOOP ;** HOLE FOR ACTUAL VIAORB WORD **
JSR R5,STEPLU ;CLOCK LU FOR DESIRED NO. OF CYCLES
2$: .WORD 0
JSR R5,CKTACT ;CHK FOR TXACT = 0
0
BCS 6$ ;BR IF ERROR
JSR R5,CKRACT ;CHK FOR RXACT = 0
0
6$: RTS R5

```

....STEPLU CLOCK THE USYRT N TIMES

```

3721 .SBTTL ....STEPLU -- CLOCK THE USYRT N TIMES
3722 ;*****
3723 ;* STEPLU THIS SUBROUTINE CLOCKS THE LINE UNIT FOR THE NUMBER OF CYCLES
3724 ;* PASSED IN THE WORD FOLLOWING THE CALL. THE VIA ACR MUST BE PREVIOUSLY
3725 ;* SET UP FOR T1 ONE-SHOT MODE, AND THE T1 LATCHES MUST BE PREVIOUSLY SET
3726 ;* TO CONTROL THE WIDTH OF THE CLOCK PULSE. ALL THAT THIS SUBROUTINE
3727 ;* DOES IS TO LOAD 000 INTO THE HI BYTE OF THE T1 COUNTER, FOR THE
3728 ;* DESIRED NUMBER OF TIMES.
3729 ;*
3730 ;* CALLING SEQUENCE :
3731 ;* JSR R5,STEPLU
3732 ;* .WORD <NUMBER OF CYCLES TO CLOCK>
3733 ;*****
3734 012072 STEPLU:
3735 012072 010146 MOV R1,-(SP) ;SAVE R1
3736 012074 012501 MOV (R5)+,R1 ;INIT CYCLE COUNTER
3737 012076 004537 003712 JSR R5,WRITEI ;LOAD TIC-M, START COUNTER, CLOCK 1 CYCLE
3738 012102 120005 VIAT18
3739 012104 000000 000
3740 012106 005301 DEC R1 ;DECR CYCLE COUNTER
3741 012110 001372 BNE 1$ ;BR IF ALL CYCLES NOT DONE YET
3742 012112 012601 MOV (SP)+,R1 ;RESTORE R1
3743 012114 000205 RTS R5 ;RETURN
3744
3745
3746
3747
3748
3749

```

GLOBAL ERROR REPORT SECTION

```

3751          .SBTTL GLOBAL ERROR REPORT SECTION
3752
3753          ;////////////////////
3754          ;/          THE GLOBAL ERROR REPORT SECTION CONTAINS ERROR MESSAGES
3755          ;/          THAT ARE USED IN MORE THAN ONE TEST.
3756          ;////////////////////
3757
3758          .NLIST BEX
3759 012116      045      116      045 ENDEMB: .ASCIZ /#N#N/
3760 012123      045      116      000 NEWLIN: .ASCIZ /#N/          ;USED TO TERMINATE ERROR MESSAGES
3761
3762 012126      045      116      045 FMT2:  .ASCIZ /#N#AFAILING REG = #T#ASEL#01/
3763 012163      045      116      045 FMT3:  .ASCIZ /#N#A EXPECTED: #03#A ACTUAL: #03#A XOR: #03/
3764 012247      045      116      045 FMT4:  .ASCIZ /#N#ATHE CONTENTS OF ALL#T#N#T/
3765 012305      045      116      045 FMT4A: .ASCIZ /#N#S1#03#S5#03#S5#03#S5#03/
3766 012340      045      116      045 FMT4B: .ASCIZ /#N#T/
3767 012345      045      116      045 FMT4C: .ASCIZ /#N#S5#03#S5#03#S5#03#S5#03/
3768 012400      045      116      045 FMT5:  .ASCIZ /#N#A WHEN #03#A LOADED INTO BSEL1/
3769 012443      045      116      045 FMT5A: .ASCIZ /#N#A ATTEMPTING "M-LOOP" FUNCTION CODE #02#A (#T#A)/
3770 012530      045      116      045 FMT7:  .ASCIZ /#N#AMDIAG #03#A FAILED/
3771
3772 012560      045      116      045 FMT10: .ASCIZ /#N#A EXPECTED:#08#A ACTUAL:#08#A XOR:#08/
3773 012634      045      101      040 FMT10A: .ASCIZ /#A LSI ADDR:#08/
3774 012655      045      116      045 FMT11: .ASCIZ /#N#08#08#08#08/
3775 012674      045      116      045 FMT12: .ASCIZ /#N#N#T/
3776 012703      045      116      045 FMT13: .ASCIZ /#N#T#03#S2#03#S2#03#S2#03#S2#03#S2#03/
3777 012751      045      123      062 FMT14: .ASCIZ /#S2#03#S2#03/
3778 012766      045      101      040 FMT15: .ASCIZ /#A DETECTED IN #T#T#A -/
3779 013020      045      101      040 FMT15A: .ASCIZ /#A DETECTED @ TEST PATTERN ELEMENT # #02/
3780 013072      045      116      045 FMT16: .ASCIZ /#N#T#03#S4#03#S#03/
3781 013115      045      116      045 FMT16A: .ASCIZ /#N#T#03#S#03#S#03#S4#03#S#03#S#03/
3782 013157      045      101      040 FMT17: .ASCIZ /#A VALUE SENT TO NPR CONTROL REGISTER: #03/
3783 013236      045      116      045 FMT17A: .ASCIZ /#N#A VALUE READ FROM CONTROL REGISTER: #03/
3784 013317      045      116      045 FMT17B: .ASCIZ /#N#A LSI-11 MEMORY ADDRESS ACCESSED:#08/
3785 013372      045      116      045 FMT17C: .ASCIZ /#N#A INFORMATION ON THE FIRST OF #05#A ERRORS:/
3786
3787 013452      045      116      045 FMT19: .ASCIZ /#N#ATEST #02#A NOT RUN#N/
3788 013503      045      124      045 FMT21: .ASCIZ /#T#06#N/
3789 013513      045      116      045 FMT22: .ASCIZ /#N#AFAILING REG: /
3790 013535      045      101      105 FMT23: .ASCIZ /#AEXPECTED: #03#S5#A ACTUAL: #03#S5#A XOR: #03#N/
3791 013614      045      116      045 FMT24: .ASCIZ /#N#T#N#T#N/
3792 013627      045      117      063 FMT25: .ASCIZ /#03#S5#03#S5#03#S5#03#N/
3793 013657      045      123      064 FMT26: .ASCIZ /#S4#03#S5#03#S5#03#S5#03#N/
3794 013712      045      124      045 FMT27: .ASCIZ /#T#T#N/
3795 013721      045      101      105 FMT28: .ASCIZ /#AEXTENDED REG AX#01#A-#T#N/
3796 013755      045      124      045 FMT29: .ASCIZ /#T#N/
3797 013762      045      116      045 FMT30: .ASCIZ /#N#AFOR BAUD RATE SPECIFIED,/
3798 014017      045      116      045 FMT31: .ASCIZ /#N#AIMPROPER CONNECTOR TYPE SPECIFIED/
3799 014065      045      116      045 FMT32: .ASCIZ /#N#AFOR OPTION SPECIFIED,/
3800 014117      045      116      045 FMT39: .ASCIZ /#N#ATEST #02#A NOT RUN#N/
3801 014150      045      116      045 FMT40: .ASCIZ /#N#AFAILING RAM ADRS: #06#A (OCT)#N/
3802
3803 014214      125      123      131 EM2:  .ASCIZ /USYRT NOT INITIALIZED BY PROGRAM RESET/
3804 014263      115      111      103 EM3:  .ASCIZ /MICRO-DIAG. FAILURE/
3805 014307      115      122      104 EM4:  .ASCIZ /MRDY TIMEOUT/
3806 014324      122      117      122 EM14: .ASCIZ /ROR NOT SET/
3807 014340      122      117      122 EM16: .ASCIZ /ROR NOT CLEARED/

```

GLOBAL ERROR REPORT SECTION

3808	014360	122	105	101	EM25:	.ASCIZ	'READ/WRITE DATA ERROR'
3809	014406	122	123	117	EM28:	.ASCIZ	/RSOM NOT CLEARED/
3810	014427	122	123	117	EM29:	.ASCIZ	/RSOM NOT SET/
3811	014444	122	105	117	EM30:	.ASCIZ	/REOM NOT CLEARED/
3812	014465	122	105	117	EM31:	.ASCIZ	/REOM NOT SET/
3813	014502	122	103	126	EM34:	.ASCIZ	/RCV'D DATA MISCOMPARE/
3814	014530	122	105	122	EM35:	.ASCIZ	/RERR NOT CLEARED/
3815	014551	122	105	122	EM36:	.ASCIZ	/RERR NOT SET/
3816	014566	122	101	102	EM39:	.ASCIZ	/RABGA NOT CLEARED/
3817	014610	122	101	102	EM40:	.ASCIZ	/RABGA NOT SET/
3818	014626	124	130	040	EM54:	.ASCIZ	/TX UNDERRUN ERROR/
3819	014650	125	123	131	EM68:	.ASCIZ	/USYRT STATUS INCORRECT/
3820	014677	124	130	101	EM69:	.ASCIZ	/TXACT NOT SET/
3821	014715	124	130	101	EM70:	.ASCIZ	/TXACT NOT CLEARED/
3822	014737	122	130	101	EM71:	.ASCIZ	/RXACT NOT SET/
3823	014755	122	130	101	EM72:	.ASCIZ	/RXACT NOT CLEARED/
3824	014777	124	102	115	EM73:	.ASCIZ	/TBMT NOT SET/
3825	015014	124	102	115	EM74:	.ASCIZ	/TBMT NOT CLEARED/
3826	015035	122	104	101	EM75:	.ASCIZ	/RDA NOT SET/
3827	015051	122	104	101	EM76:	.ASCIZ	/RDA NOT CLEARED/
3828	015071	122	123	101	EM77:	.ASCIZ	/RSA NOT SET/
3829	015105	122	123	101	EM78:	.ASCIZ	/RSA NOT CLEARED/
3830	015125	122	101	115	EM79:	.ASCIZ	/RAM ERROR LOADING MICROCODE/
3831	015161	103	101	122	EM80:	.ASCIZ	/CARRIER NOT SET/
3832	015201	103	101	122	EM81:	.ASCIZ	/CARRIER NOT CLEARED/
3833	015225	111	116	126	EM82:	.ASCIZ	/INVALID ERROR CODE FROM 6502/
3834	015262	115	117	104	EM83:	.ASCIZ	/MODEM STATUS INCORRECT/
3835	015311	103	124	123	EM84:	.ASCIZ	/CTS NOT CLRD/
3836	015326	103	124	123	EM85:	.ASCIZ	/CTS NOT SET/
3837	015342	103	101	122	EM86:	.ASCIZ	/CARRIER NOT CLRD/
3838	015363	103	101	122	EM87:	.ASCIZ	/CARRIER NOT SET/
3839	015403	115	117	104	EM88:	.ASCIZ	/MODEM RDY NOT CLRD/
3840	015426	115	117	104	EM89:	.ASCIZ	/MODEM RDY NOT SET/
3841	015450	122	105	103	EM90:	.ASCIZ	/RECEIVER OVERRUN NOT SET/
3842	015501	122	105	103	EM91:	.ASCIZ	/RECEIVER OVERRUN NOT CLEARED/
3843	015536	124	123	117	EM100:	.ASCIZ	/TSO BIT NOT SET/
3844	015556	124	123	117	EM101:	.ASCIZ	/TSO BIT NOT CLEARED/

```

3845
3846
3847          .SBTTL ....TEXT STRINGS FOR ERROR HANDLERS -- "TXT_ _"
3848          ;-----
3849          ;----- TEXT USED BY ERROR HANDLERS -----
3850          ;-----
3851

```

3852	015602	102	123	105	TXT1:	.ASCIZ	/BSEL0 BSEL1 BSEL2 BSEL3/
3853	015640	040	040	040	TXT2:	.ASCIZ	/ BSEL4 BSEL5 BSEL6 BSEL7/
3854	015702	102	123	105	TXT2A:	.ASCIZ	/BSEL10 BSEL11 BSEL12 BSEL13/
3855	015741	040	040	040	TXT2B:	.ASCIZ	/ BSEL14 BSEL15 BSEL16 BSEL17/
3856	016004	040	102	131	TXT3:	.ASCIZ	/ BYTE SELECT REG'S ARE:/
3857	016034	040	040	040	TXT4:	.ASCIZ	/ SEL0 SEL2 SEL4 SEL6/
3858	016074	040	040	040	TXT4A:	.ASCIZ	/ SEL10 SEL12 SEL14 SEL16/
3859	016135	102	000		TXT5:	.ASCIZ	/B/
3860	016137	040	123	105	TXT6:	.ASCIZ	/ SELECT REG'S ARE:/
3861	016162	040	122	105	TXT7:	.ASCIZ	/ REGISTERS ORB ORA DDRB DDRA T1CL T1CH T1LL T1LH /
3862	016252	040	040	040	TXT7A:	.ASCIZ	/ T2CL T2CH SR ACR PCR IFR IER ORA /
3863	016342	040	105	130	TXT8:	.ASCIZ	/ EXPECTED: /
3864	016362	040	101	103	TXT9:	.ASCIZ	/ ACTUAL: /

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

```

3865 016402      040      130      117 TXT10: .ASCIZ / XOR: /
3866 016422      040      040      116 TXT11: .ASCIZ / N P R R E G I S T E R S:/
3867 016474      040      040      040 TXT11A: .ASCIZ / CONTROL DATA/
3868 016532      040      040      040 TXT11B: .ASCIZ / OUT ADDR. IN ADDR./
3869 016602      104      105      126 TXT12: .ASCIZ /DEVICE CSR ADDRESS : /
3870 016630      125      123      131 TXT13: .ASCIZ /USYRT REGS :/
3871 016645      122      104      123 TXT14: .ASCIZ /RDSRL RDSRH TDSRL TDSRH/
3872 016703      040      040      040 TXT15: .ASCIZ / PCSARL PCSARH PCR USTAT/
3873 016745      126      111      101 TXT16: .ASCIZ /VIA REGS :/
3874 016760      117      122      102 TXT17: .ASCIZ /ORB ORA DDRB DDRA/
3875 017015      040      040      040 TXT18: .ASCIZ / T1CL T1CH T1LL T1LH/
3876 017056      124      062      103 TXT19: .ASCIZ /T2CL T2CH SR ACR/
3877 017112      040      040      040 TXT20: .ASCIZ / PCR IFR IER ORA/
3878
3879 017152      021      000          TXTNUL: .BYTE 21,0 ;CTL-Q - THIS (WE HOPE) IS HARMLESS
3880
3881 017154      116      117      120 TXTML0: .ASCIZ /NOP/
3882 017160      122      105      101 TXTML1: .ASCIZ /READ 1 BYTE/
3883 017174      127      122      111 TXTML2: .ASCIZ /WRITE 1 BYTE/
3884 017211      116      120      122 TXTML3: .ASCIZ /NPR-OUT 256 BYTES/
3885 017233      116      120      122 TXTML4: .ASCIZ /NPR-IN 256 BYTES/
3886 017254      123      105      124 TXTML5: .ASCIZ /SET MICROPROCESSOR'S PC/
3887 017304      125      116      104 TXTML6: .ASCIZ /UNDEFINED/
3888 017316      101      114      114 TXTML7: .ASCIZ /ALLOW U-PROCESSOR INTERRUPTS/
3889
3890 017353      126      111      101 TXTVR: .ASCIZ /VIA REGISTER /
3891 017371      117      122      102 TXTVR0: .ASCIZ /ORB/
3892 017375      117      122      101 TXTVR1: .ASCIZ /ORA/
3893 017401      104      104      122 TXTVR2: .ASCIZ /DDRB/
3894 017406      104      104      122 TXTVR3: .ASCIZ /DDRA/
3895 017413      124      061      103 TXTVR4: .ASCIZ /T1CL/
3896 017420      124      061      103 TXTVR5: .ASCIZ /T1CH/
3897 017425      124      061      114 TXTVR6: .ASCIZ /T1LL/
3898 017432      124      061      114 TXTVR7: .ASCIZ /T1LH/
3899 017437      124      062      103 TXTVR8: .ASCIZ /T2CL/
3900 017444      124      062      103 TXTVR9: .ASCIZ /T2CH/
3901 017451      123      122      000 TXTVRA: .ASCIZ /SR/
3902 017454      101      103      122 TXTVRB: .ASCIZ /ACR/
3903 017460      120      103      122 TXTVRC: .ASCIZ /PCR/
3904 017464      111      106      122 TXTVRD: .ASCIZ /IFR/
3905 017470      111      105      122 TXTVRE: .ASCIZ /IER/
3906 017474      117      122      101 TXTVRF: .ASCIZ /ORA/
3907
3908 017500      116      120      122 TXTNP: .ASCIZ /NPR /
3909 017505      103      117      116 TXTNP0: .ASCIZ /CONTROL/
3910 017515      104      101      124 TXTNP1: .ASCIZ /DATA HI/
3911 017525      104      101      124 TXTNP2: .ASCIZ /DATA LO/
3912 017535      101      104      104 TXTNP3: .ASCIZ /ADDR. OUT EX/
3913 017552      101      104      104 TXTNP4: .ASCIZ /ADDR. OUT HI/
3914 017567      101      104      104 TXTNP5: .ASCIZ /ADDR. OUT LO/
3915 017604      101      104      104 TXTNP6: .ASCIZ /ADDR. IN EX/
3916 017620      101      104      104 TXTNP7: .ASCIZ /ADDR. IN HI/
3917 017634      101      104      104 TXTNP8: .ASCIZ /ADDR. IN LO/
3918
3919 017650      125      123      131 TXTUR: .ASCIZ /USYRT REG /
3920 017663      122      104      123 TXTUR0: .ASCIZ /RDSRL/
3921 017671      122      104      123 TXTUR1: .ASCIZ /RDSRH/

```

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

3922	017677	124	104	123	TXTUR2:	.ASCIZ	/TDSRL/
3923	017705	124	104	123	TXTUR3:	.ASCIZ	/TDSRH/
3924	017713	120	103	123	TXTUR4:	.ASCIZ	/PCSAR1/
3925	017722	120	103	123	TXTUR5:	.ASCIZ	/PCSARM/
3926	017731	120	103	122	TXTUR6:	.ASCIZ	/PCR/
3927	017735	125	123	124	TXTUR7:	.ASCIZ	/USTAT/
3928					.LIST	BEX	
3929					.EVEN		

3930
3931

3932
3933

3934
3935

3936

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

```

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

```

3937	017744	017154	017160	017174	TXTMLT:	.WORD	TXTML0,TXTML1,TXTML2,TXTML3,TXTML4,TXTML5,TXTML6,TXTML7
	017752	017211	017233	017254			
	017760	017304	017316				

3938
3939

3940	017764	017353			TXTVRT:	.WORD	TXTVR
	017766	017371	017375	017401			TXTVR0,TXTVR1,TXTVR2,TXTVR3,TXTVR4,TXTVR5,TXTVR6,TXTVR7
	017774	017406	017413	017420			

3941	020002	017425	017432			.WORD	TXTVR8,TXTVR9,TXTVRA,TXTVRB,TXTVRC,TXTVRD,TXTVRE,TXTVRF
	020006	017437	017444	017451			
	020014	017454	017460	017464			
	020022	017470	017474				

3942
3943

3944	020026	017500			TXTNPT:	.WORD	TXTNP
	020030	017505	017515	017525			TXTNPO,TXTNP1,TXTNP2,TXTNP3,TXTNP4,TXTNP5,TXTNP6,TXTNP7,TXTNP8
	020036	017535	017552	017567			
	020044	017604	017620	017634			

3945	020052	017663	017671	017677	TXTURT:	.WORD	TXTURO,TXTUR1,TXTUR2,TXTUR3,TXTUR4,TXTUR5,TXTUR6,TXTUR7
	020060	017705	017713	017722			
	020066	017731	017735				

3946
3947
3948

....TEXT ADDRESS TABLES FOR ERROR HANDLERS "TXT T"

```

3950
3951      ; SBTTL ....ERROR HANDLER      ERR3      DUMP THE BYTE SELECT REGISTERS
3952      ;
3953      BGNMSG  ERR3
3954      020072 004737 021306      JSR      PC,ERR4$
3955      020076      PRINTB  @ENDEMB
3956      020076 012746 012116      MOV      @ENDEMB, (SP)
3957      020102 012746 000001      MOV      @1, (SP)
3958      020106 010600      MOV      SP,RO
3959      020110 104414      TRAP    C$PNTB
3960      020112 062706 000004      ADD      @4,SP
3961      020116      ENDMSG
3962      020116      L10002: TRAP    C$MSG
3963      020116 104423
3964      ; SBTTL ....ERROR HANDLER      ERR4      M LOOP TIMEOUT ERROR HANDLING
3965      ;
3966      BGNMSG  ERR4
3967      020120      CLR      GDATA,1
3968      020120      MOV      R1, -(SP)
3969      020120 105037 002325      ; MAKE SURE BIT 8 DOESN'T PRINT!
3970      020124 010146      MOV      GDATA,R1
3971      020126 013701 002324      ; SAVE THE WORKING REGISTER
3972      020132 022701 000017      ; SAVE THIS FOR LATER
3973      020136 002012      CMP      @17,R1
3974      020140      BGE      5$
3975      020140      PRINTX @FMT5,R1
3976      020140 010146      ; WAS THIS AN M-LOOP REQUEST?
3977      020142 012746 012400      ; YES, THEN REPORT THE FUNCTION CODE
3978      020146 012746 000002      ; NO, THEN IT MUST BE A BSEL1 SETTING
3979      020152 010600      MOV      R1, (SP)
3980      020154 104415      MOV      @FMT5, (SP)
3981      020156 062706 000006      MOV      @2, -(SP)
3982      020162 000424      BR       20$
3983      020164 001001      5$: BNE      6$
3984      020166 005001      CLR      R1
3985      020170 022701 000007      ; IF IT WAS A 17, THIS IS A "NOP" AND
3986      020174 002002      6$: CMP      @7,R1
3987      020176 012701 000006      ; THE TEXT POINTER MUST SO REFLECT.
3988      020202 006301      BGE      7$
3989      020204      MOV      @6,R1
3990      020204 016146 017744      ; IS FUNCTION CODE > 7?
3991      020210 013746 002324      ; NO, THEN WE CAN HANDLE IT
3992      020214 012746 012443      ; YES, THEN IT'S UNDEFINED - SAY SO
3993      020220 012746 000003      ; CONVERT TO A WORD OFFSET
3994      020224 010600      PRINTX @FMT5A,GDATA,IXTMLT(R1)
3995      020226 104415      ; REPORT THE FAILING FUNCTION
3996      020230 062706 000010      MOV      TXTMLT(R1), -(SP)
3997      020234 012601      MOV      GDATA, (SP)
3998      020236 004737 021634      20$: MOV      (SP),R1
3999      020242      JSR      PC,ERR5$
4000      020242      ENDMSG
4001      020242 104423      ; RESTORE THE WORKING REGISTER
4002      ; DUMP THE SELECT REGISTERS
4003      L10003: TRAP    C$MSG
4004      3980      ; SBTTL ....ERROR HANDLER      ERR7A      USYRT REGISTER ERRORS
4005      3981      ;
4006      3982      ;

```

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

```

3983 020244          BGNMSG  ERR7A
3984 020244          113701 002336          ERR7A::
3985 020250          006301          MOV  REGNUM,R1
3986 020252          PRINTB  @FMT15,@TXTUR,TXTURT(R1)          ;AS PASSED, THIS WAS A BYTE OFFSET
                                MOV  TXTURT(R1), (SP)
                                MOV  @TXTUR, -(SP)
                                MOV  @FMT15, (SP)
                                MOV  @3, -(SP)
                                MOV  SP,R0
                                TRAP  C$PNTB
                                ADD   @10,SP
020252 016146 020052
020256 012746 017650
020262 012746 012766
020266 012746 000003
020272 010600
020274 104414
020276 062706 000010
3987 020302          JSR    PC,XORGB
3988 020306          PRINTB  @FMT3,GDATA,BDATA,XDATA
                                MOV  XDATA, (SP)
                                MOV  BDATA, (SP)
                                MOV  GDATA, -(SP)
                                MOV  @FMT3, (SP)
                                MOV  @4, -(SP)
                                MOV  SP,R0
                                TRAP  C$PNTB
                                ADD   @12,SP
020306 013746 002330
020312 013746 002326
020316 013746 002324
020322 012746 012163
020326 012746 000004
020332 010600
020334 104414
020336 062706 000012
3989 020342          PRINTB  @ENDEMB
                                MOV  @ENDEMB, -(SP)
                                MOV  @1, -(SP)
                                MOV  SP,R0
                                TRAP  C$PNTB
                                ADD   @4,SP
020342 012746 012116
020346 012746 000001
020352 010600
020354 104414
020356 062706 000004
3990 020362          ENDMSG
                                I 10004:
                                TRAP  C$MSG
020362 104423
3991
3992
3993
3994
3995 020364          BGNMSG  ERR10
3996 020364          PRINTB  @FMT21,@TXT12,MPCSR          ERR10::
                                MOV  MPCSR, -(SP)
                                MOV  @TXT12, -(SP)
                                MOV  @FMT21, -(SP)
                                MOV  @3, -(SP)
                                MOV  SP,R0
                                TRAP  C$PNTB
                                ADD   @10,SP
020364 013746 002416
020370 012746 016602
020374 012746 013503
020400 012746 000003
020404 010600
020406 104414
020410 062706 000010
3997 020414          PRINTB  @FMT22
                                MOV  @FMT22, (SP)
                                MOV  @1, -(SP)
                                MOV  SP,R0
                                TRAP  C$PNTB
                                ADD   @4,SP
020414 012746 013513
020420 012746 000001
020424 010600
020426 104414
020430 062706 000004
3998 020434          MOV  REGNUM,R1
3999 020440          ASL   R1          ;GET PTR TO USYRT REG ASCII
4000 020442          PRINTB  @FMT27,@TXTUR,TXTURT(R1)
                                MOV  TXTURT(R1), (SP)
                                MOV  @TXTUR, -(SP)
                                MOV  @FMT27, (SP)
020442 016146 020052
020446 012746 017650
020452 012746 013712

```

```

-----
;SBTTL ....ERROR HANDLER -- ERR10 -- USYRT REG ERROR (XOR, REG PRINTOUT)
-----

```

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

```

020456 012746 000003
020462 010600
020464 104414
020466 062706 000010
4001 020472 004737 021262 JSR PC,XORGB ;COMPUTE XOR OF GOOD AND BAD DATA
4002 020476 PRINTB @FMT23,GDATA,BDATA,XDATA
020476 013746 002330 MOV XDATA,-(SP)
020502 013746 002326 MOV BDATA,-(SP)
020506 013746 002324 MOV GDATA,-(SP)
020512 012746 013535 MOV @FMT23,(SP)
020516 012746 000004 MOV @4,(SP)
020522 010600 MOV SP,RO
020524 104414 TRAP C$PNTB
020526 062706 J00012 ADD @10,SP
4003 020532 004737 022364 JSR PC,ERR12; ;GET & PRINT USYRT REGISTERS
4004 020536 ENDMMSG
020536 104423 L10005: TRAP C$MSG
4005
4006
4007 ;-----
4008 ;SBTTL ....ERROR HANDLER -- ERR11 - VIA REG ERROR (XOR, REG PRINTOUT)
4009 ;-----
4010 020540 BGNMSG ERR11
4011 020540 PRINTB @FMT21,@TXT12,MPCSR ERR11::
020540 013746 002416 MOV MPCSR,-(SP)
020544 012746 016602 MOV @TXT12,-(SP)
020550 012746 013503 MOV @FMT21,-(SP)
020554 012746 000003 MOV @3,(SP)
020560 010600 MOV SP,RO
020562 104414 TRAP C$PNTB
020564 062706 000010 ADD @10,SP
4012 020570 PRINTB @FMT22
020570 012746 013513 MOV @FMT22,(SP)
020574 012746 000001 MOV @1,-(SP)
020600 010600 MOV SP,RO
020602 104414 TRAP C$PNTB
020604 062706 000004 ADD @4,SP
4013 020610 013701 002336 MOV REGNUM,R1
4014 020614 006301 ASL R1 ;GET PTR TO VIA REG ASCII
4015 020616 PRINTB @FMT27,@TXTVR,XTVRT(R1)
020616 016146 017766 MOV TXTVRT(R1),(SP)
020622 012746 017353 MOV @TXTVR,-(SP)
020626 012746 013712 MOV @FMT27,-(SP)
020632 012746 000003 MOV @3,-(SP)
020636 010600 MOV SP,RO
020640 104414 TRAP C$PNTB
020642 062706 000010 ADD @10,SP
4016 020646 004737 021262 JSR PC,XORGB ;COMPUTE XOR OF GOOD AND BAD DATA
4017 020652 PRINTB @FMT23,GDATA,BDATA,XDATA
020652 013746 002330 MOV XDATA,-(SP)
020656 013746 002326 MOV BDATA,-(SP)
020662 013746 002324 MOV GDATA,-(SP)
020666 012746 013535 MOV @FMT23,-(SP)
020672 012746 000004 MOV @4,-(SP)
020676 010600 MOV SP,RO

```

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

020700	104414				TRAP	C\$PNTB
020702	062706	000012			ADD	#12,SP
4018 020706	004737	022032		JSR PC,ERR11	;GET & PRINT VIA REGISTERS	
4019 020712				ENDMSG		
020712					L10006:	
020712	104423				TRAP	C\$MSG

4020
4021
4022
4023
4024

;SBTTLERROR HANDLER -- ERR12 -- USYRT REG ERROR (USYRT PRINTOUT)

4025 020714				BGNMSG ERR12		ERR12::
020714						
4026 020714				PRINTB #FMT21,#TXT12,MPCSR		
020714	013746	002416			MOV	MPCSR,-(SP)
020720	012746	016602			MOV	#TXT12,(SP)
020724	012746	013503			MOV	#FMT21,(SP)
020730	012746	000003			MOV	#3,-(SP)
020734	010600				MOV	SP,R0
020736	104414				TRAP	C\$PNTB
020740	062706	000010			ADD	#10,SP
4027 020744				PRINTB #FMT22		
020744	012746	013513			MOV	#FMT22,-(SP)
020750	012746	000001			MOV	#1,(SP)
020754	010600				MOV	SP,R0
020756	104414				TRAP	C\$PNTB
020760	062706	000004			ADD	#4,SP
4028 020764	013701	002336		MOV REGNUM,R1		
4029 020770	006301			ASL R1	;GET PTR TO USYRT REG ASCII	
4030 020772				PRINTB #FMT27,#TXTUR,TXTURT(R1)		
020772	016146	020052			MOV	TXTURT(R1),-(SP)
020776	012746	017650			MOV	#TXTUR,-(SP)
021002	012746	013712			MOV	#FMT27,-(SP)
021006	012746	000003			MOV	#3,-(SP)
021012	010600				MOV	SP,R0
021014	104414				TRAP	C\$PNTB
021016	062706	000010			ADD	#10,SP
4031 021022	004737	022364		JSR PC,ERR12	;GET & PRINT USYRT REGISTERS	
4032 021026				ENDMSG		
021026					L10007:	
021026	104423				TRAP	C\$MSG

4033
4034
4035
4036
4037

;SBTTLERROR HANDLER - ERR13 -- RAM ADDRESS ERRORS

4038 021030				BGNMSG ERR1		ERR13::
021030						
4039 021030				PRINTB #FMT21,#TXT12,MPCSR		
021030	013746	002416			MOV	MPCSR,-(SP)
021034	012746	016602			MOV	#TXT12,(SP)
021040	012746	013503			MOV	#FMT21,-(SP)
021044	012746	000003			MOV	#3,(SP)
021050	010600				MOV	SP,R0
021052	104414				TRAP	C\$PNTB
021054	062706	000010			ADD	#10,SP
4040 021060				PRINTB #FMT40,REGNUM		

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

```

021060 013746 002336          MOV      REGNUM,-(SP)
021064 012746 014150          MOV      #FMT40,-(SP)
021070 012746 000002          MOV      #2,(SP)
021074 010600                  MOV      SP,R0
021076 104414                  TRAP    C$PNTB
021100 062706 000006          ADD      #6,SP
4041 021104 004737 021262      JSR      PC,XORGB          ;COMPUTE XOR OF GOOD AND BAD DATA
4042 021110          PRINTB  #FMT23,GDATA,BDATA,XDATA
021110 013746 002330          MOV      XDATA,-(SP)
021114 013746 002326          MOV      BDATA,(SP)
021120 013746 002324          MOV      GDATA,(SP)
021124 012746 013535          MOV      #FMT23,-(SP)
021130 012746 000004          MOV      #4,-(SP)
021134 010600                  MOV      SP,R0
021136 104414                  TRAP    C$PNTB
021140 062706 000012          ADD      #12,SP
4043 021144          ENDMSG
021144          L10010:
021144 104423          TRAP    C$MSG
4044
4045
4046
4047
4048
4049 021146          BGNMSG  ERR14
021146          ERR14::
4050 021146          PRINTB  #FMT21,#TXT12,MPCSR
021146 013746 002416          MOV      MPCSR,(SP)
021152 012746 016602          MOV      #TXT12,(SP)
021156 012746 013503          MOV      #FMT21,-(SP)
021162 012746 000003          MOV      #3,-(SP)
021166 010600                  MOV      SP,R0
021170 104414                  TRAP    C$PNTB
021172 062706 000010          ADD      #10,SP
4051 021176          PRINTB  #FMT22
021176 012746 013513          MOV      #FMT22,(SP)
021202 012746 000001          MOV      #1,(SP)
021206 010600                  MOV      SP,R0
021210 104414                  TRAP    C$PNTB
021212 062706 000004          ADD      #4,SP
4052 021216 013701 002336      MOV      REGNUM,R1
4053 021222 006301          ASL     R1          ;GET PTR TO VIA REG ASCII
4054 021224          PRINTB  #FMT27,#TXTVR,TXTVRT(R1)
021224 016146 017766          MOV      TXTVRT(R1),-(SP)
021230 012746 017353          MOV      #TXTVR,(SP)
021234 012746 013712          MOV      #FMT27,(SP)
021240 012746 000003          MOV      #3,-(SP)
021244 010600                  MOV      SP,R0
021246 104414                  TRAP    C$PNTB
021250 062706 000010          ADD      #10,SP
4055 021254 004737 022032      JSR      PC,ERR11$      ;GET & PRINT VIA REGISTERS
4056 021260          ENDMSG
021260          L10011:
021260 104423          TRAP    C$MSG
4057
4058
4059

```

....ERROR HANDLER ERR14 - VIA REG ERRORS (VIA PRINTOUT)

```

4060
4061
4062      .SBTTL ....ERROR HANDLER SUBROUTINES
4063      :-----
4064      :----- SUBROUTINES USED ONLY BY ERROR HANDLERS -----
4065      :-----
4066
4067      .SBTTL .....ERROR HANDLER SUBROUTINE - XORGB
4068
4069      :      PERFORM EXCLUSIVE OR BETWEEN "GDATA" & "BDATA" PUTTING
4070      :      THE RESULT IN 'XDATA'
4071
4072 021262 010146      XORGB:  MOV      R1,-(SP)      ;PRESERVE WORKING REGISTER
4073 021264 013701 002324      MOV      GDATA,R1      ;GET "GOOD" DATA
4074 021270 013737 002326 002330      MOV      BDATA,XDATA  ;AND "BAD" DATA
4075 021276 074137 002330      XOR      R1,XDATA     ;PERFORM EXCLUSIVE OR
4076 021302 012601      MOV      (SP)+,R1     ;RESTORE R1
4077 021304 000207      RTS      PC          ;RETURN
4078
4079      :-----
4080      .SBTTL .....ERROR HANDLER SUBROUTINE -- ERR4$
4081      :-----
4082      :      IDENTIFY & DUMP THE BYTE SELECT REGISTERS
4083
4084 021306      ERR4$:  PRINTX  #FMT4,#TXT3,#TXT1
4085 021306 012746 015602      MOV      #TXT1,-(SP)
4086 021312 012746 016004      MOV      #TXT3,(SP)
4087 021316 012746 012247      MOV      #FMT4,-(SP)
4088 021322 012746 000003      MOV      #3,-(SP)
4089 021326 010600      MOV      SP,R0
4090 021330 104415      TRAP    C#PNTX
4091 021332 062706 000010      ADD     #10,SP
4092 021336      PRINTX  #FMT4A,BSR0,BSR1,BSR2,BSR3
4093 021336 013746 002210      MOV     BSR3,-(SP)
4094 021342 013746 002206      MOV     BSR2,-(SP)
4095 021346 013746 002204      MOV     BSR1,-(SP)
4096 021352 013746 002202      MOV     BSR0,-(SP)
4097 021356 012746 012305      MOV     #FMT4A,(SP)
4098 021362 012746 000005      MOV     #5,-(SP)
4099 021366 010600      MOV     SP,R0
4100 021370 104415      TRAP    C#PNTX
4101 021372 062706 000014      ADD     #14,SP
4102 021376      PRINTX  #FMT4B,#TXT2
4103 021376 012746 015640      MOV     #TXT2,(SP)
4104 021402 012746 012340      MOV     #FMT4B,(SP)
4105 021406 012746 000002      MOV     #2,-(SP)
4106 021412 010600      MOV     SP,R0
4107 021414 104415      TRAP    C#PNTX
4108 021416 062706 000006      ADD     #6,SP
4109 021422      PRINTX  #FMT4C,BSR4,BSR5,BSR6,BSR7
4110 021422 013746 002220      MOV     BSR7,-(SP)
4111 021426 013746 002216      MOV     BSR6,-(SP)
4112 021432 013746 002214      MOV     BSR5,-(SP)
4113 021436 013746 002212      MOV     BSR4,(SP)
4114 021442 012746 012345      MOV     #FMT4C,-(SP)
4115 021446 012746 000005      MOV     #5,(SP)
4116 021452 010600      MOV     SP,R0

```


.....ERROR HANDLER SUBROUTINE ERR4\$

```

021454 104415
021456 062706 000014
4088 021462 PRINTX #FMT4B,#TXT2A
021462 012746 015702
021466 012746 012340
021472 012746 000002
021476 010600
021500 104415
021502 062706 000006
4089 021506 PRINTX #FMT4A,BSR10,BSR11,BSR12,BSR13
021506 013746 002230
021512 013746 002226
021516 013746 002224
021522 013746 002222
021526 012746 012305
021532 012746 000005
021536 010600
021540 104415
021542 062706 000014
4090 021546 PRINTX #FMT4B,#TXT2B
021546 012746 015741
021552 012746 012340
021556 012746 000002
021562 010600
021564 104415
021566 062706 000006
4091 021572 PRINTX #FMT4C,BSR14,BSR15,BSR16,BSR17
021572 013746 002240
021576 013746 002236
021602 013746 002234
021606 013746 002232
021612 012746 012345
021616 012746 000005
021622 010600
021624 104415
021626 062706 000014
4092 021632 RTS PC
4093
4094
4095
4096
4097
4098
4099

```

```

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

```

```

021634 012746 016034
021640 012746 016137
021644 012746 012247
021650 012746 000003
021654 010600
021656 104415
021660 062706 000010
4100 021664 PRINTX #FMT11,WSR0,WSR2,WSR4,WSR6 ;DUMP THE SELECT REGISTERS
021664 013746 002210
021670 013746 002206
021674 013746 002204
021700 013746 002202
021704 012746 012655

```

.....ERROR HANDLER SUBROUTINE ERR5:

```

021710 012746 000005      MOV      #5, (SP)
021714 010600             MOV      SP,R0
021716 104415             TRAP    C$PNTX
021720 062706 000014      ADD      #14,SP
4101 021724             PRINTX  #FMT4B,#TXT4A
021724 012746 016074             MOV      #TXT4A, (SP)
021730 012746 012340             MOV      #FMT4B, -(SP)
021734 012746 000002             MOV      #2, -(SP)
021740 010600             MOV      SP,R0
021742 104415             TRAP    C$PNTX
021744 062706 000006      ADD      #6,SP
4102 021750             PRINTX  #FMT11,WSR10,WSR12,WSR14,JSR16 ;DUMP THE SELECT REGISTERS
021750 013746 002220             MOV      WSR16, -(SP)
021754 013746 002216             MOV      WSR14, -(SP)
021760 013746 002214             MOV      WSR12, -(SP)
021764 013746 002212             MOV      WSR10, (SP)
021770 012746 012655             MOV      #FMT11, -(SP)
021774 012746 000005             MOV      #5, -(SP)
022000 010600             MOV      SP,R0
022002 104415             TRAP    C$PNTX
022004 062706 000014      ADD      #14,SP
4103 022010             PRINTB  #ENDEMB
022010 012746 012116             MOV      #ENDEMB, -(SP)
022014 012746 000001             MOV      #1, (SP)
022020 010600             MOV      SP,R0
022022 104414             TRAP    C$PNTB
022024 062706 000004      ADD      #4,SP
4104 022030             RTS     PC
4105
4106
4107
4108
4109
4110
4111 022032 004737 004460      ERR11$: JSR     PC,GETVRS ;GET VIA REGS FOR PRINTOUT
4112 022036             PRINTX  #FMT24,#TXT16,#TXT17
022036 012746 016760             MOV      #TXT17, (SP)
022042 012746 016745             MOV      #TXT16, (SP)
022046 012746 013614             MOV      #FMT24, -(SP)
022052 012746 000003             MOV      #3, (SP)
022056 010600             MOV      SP,R0
022060 104415             TRAP    C$PNTX
022062 062706 000010      ADD      #10,SP
4113 022066             PRINTX  #FMT25,VREGS+0,VREGS+2,VREGS+4,VREGS+6
022066 013746 002270             MOV      VREGS+6, -(SP)
022072 013746 002266             MOV      VREGS+4, -(SP)
022076 013746 002264             MOV      VREGS+2, -(SP)
022102 013746 002262             MOV      VREGS+0, -(SP)
022106 012746 013627             MOV      #FMT25, (SP)
022112 012746 000005             MOV      #5, (SP)
022116 010600             MOV      SP,R0
022120 104415             TRAP    C$PNTX
022122 062706 000014      ADD      #14,SP
4114 022126             PRINTX  #FMT29 #TXT18
022126 012746 017015             MOV      #TXT18, (SP)
022132 012746 013755             MOV      #FMT29, (SP)
022136 012746 000002             MOV      #2, (SP)

```

```

;-----
;SBTTL .....ERROR HANDLER SUBROUTINE - ERR11$
;-----
; COMMON ERROR SUBROUTINE TO GET/PRINT VIA REGISTERS

```

.....ERROR HANDLER SUBROUTINE ERR11:

```

022142 010600
022144 104415
4115 022146 062706 000006          PRINTX  #FMT26,VREGS+8.,VREGS+10.,VREGS+12.,VREGS+14.
022152 013746 002300
022156 013746 002276
022162 013746 002274
022166 013746 002272
022172 012746 013657
022176 012745 000005
022202 010600
022204 104415
4116 022206 062706 000014          PRINTX  #FMT29,#TXT19
022212 012746 017056
022216 012746 013755
022222 012746 000002
022226 010600
022230 104415
022232 062706 000006
4117 022236 000006          PRINTX  #FMT25,VREGS+16.,VREGS+18.,VREGS+20.,VREGS+22.
022236 013746 002310
022242 013746 002306
022246 013746 002304
022252 013746 002302
022256 012746 013627
022262 012746 000005
022266 010600
022270 104415
022272 062706 000014
4118 022276 000014          PRINTX  #FMT29,#TXT20
022276 012746 017112
022302 012746 013755
022306 012746 000002
022312 010600
022314 104415
022316 062706 000006
4119 022322 000006          PRINTX  #FMT26,VREGS+24.,VREGS+26.,VREGS+28.,VREGS+30.
022322 013746 002320
022326 013746 002316
022332 013746 002314
022336 013746 002312
022342 012746 013657
022346 012746 000005
022352 010600
022354 104415
022356 062706 000014
4120 022362 000207          RTS      PC
4121
4122
4123 ;-----
4124 ;.SBTTL .....ERROR HANDLER SUBROUTINE ERR12:
4125 ;
4126 ; COMMON ERROR ROUTINE TO GET AND PRINTOUT USYRT REGISTERS
4127 ERR12: JSR      PC,GETURS          ;GET USYRT REGS FOR PRINTOUT
4128 PRINTX #FMT24,#TXT13,#TXT14
022370 012746 016645          MOV      #TXT14,(SP)

```

.....ERROR HANDLER SUBROUTINE

ERR12:

022374 012746 016630
 022400 012746 013614
 022404 012746 000003
 022410 010600
 022412 104415
 022414 062706 000010
 4129 022420
 022420 013746 002250
 022424 013746 002246
 022430 013746 002244
 022434 013746 002242
 022440 012746 013627
 022444 012746 000005
 022450 010600
 022452 104415
 022454 062706 000014
 4130 022460
 022460 012746 016703
 022464 012746 013755
 022470 012746 000002
 022474 010600
 022476 104415
 022500 062706 000006
 4131 022504
 022504 013746 002260
 022510 013746 002256
 022514 013746 002254
 022520 013746 002252
 022524 012746 013657
 022530 012746 000005
 022534 010600
 022536 104415
 022540 062706 000014
 4132 022544 000207

PRINTX #FMT25,UREGS+0,UREGS+2,UREGS+4,UREGS+6

PRINTX #FMT29,#TXT15

PRINTX #FMT26,UREGS+10,UREGS+12,UREGS+14,UREGS+16

RTS PC

.EVEN

MOV #TXT13,-(SP)
 MOV #FMT24,-(SP)
 MOV #3,(SP)
 MOV SP,R0
 TRAP C#PNTX
 ADD #10,SP

MOV UREGS+6,-(SP)
 MOV UREGS+4,-(SP)
 MOV UREGS+2,(SP)
 MOV UREGS+0,(SP)
 MOV #FMT25,(SP)
 MOV #5,-(SP)
 MOV SP,R0
 TRAP C#PNTX
 ADD #14,SP

MOV #TXT15,(SP)
 MOV #FMT29,-(SP)
 MOV #2,(SP)
 MOV SP,R0
 TRAP C#PNTX
 ADD #6,SP

MOV UREGS+16,-(SP)
 MOV UREGS+14,-(SP)
 MOV UREGS+12,-(SP)
 MOV UREGS+10,-(SP)
 MOV #FMT26,-(SP)
 MOV #5,-(SP)
 MOV SP,R0
 TRAP C#PNTX
 ADD #14,SP

LOAD DEVICE PROTECTION TABLE

4136
 4137
 4138
 4139
 4140
 4141
 4142
 4143 022546
 022546
 4144 022546 177777
 4145 022550 177777
 4146 022552 177777
 4147 022554

.SBTTL LOAD DEVICE PROTECTION TABLE

```

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

```

BGNPROT

```

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

```

L\$PROT::

INITIALIZE SECTION

```

4149          .SBTTL  INITIALIZE SECTION
4150
4151          ;////////////////////////////////////
4152          ;/ THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
4153          ;/ AT THE BEGINNING OF THE TEST SEQUENCE ON THE NEXT UNIT.
4154          ;////////////////////////////////////
4155
4156 022554          BGNINIT
4157          L$INIT::
4158 022554          SETVEC  #140,#170000,#340          ;ODT ROM ADDRESS
4159          022554 012746 000340          MOV          #340,-(SP)
4160          022560 012746 170000          MOV          #170000,-(SP)
4161          022564 012746 000140          MOV          #140,(SP)
4162          022570 012746 000003          MOV          #3,-(SP)
4163          022574 104437          TRAP          C$SVEC
4164          022576 062706 000010          ADD          #10,SP
4165
4166 022602 010637 002340          MOV          SP,PSTACK          ;SAVE BASE-LEVEL STACK POINTER
4167 022606 005037 002344          CLR          SUBRPC          ;CLEAR SUBR CALL PC
4168 022612 005037 002400          CLR          CHPTYP          ;CLEAR USYRT CHIP TYPE INDICATOR
4169 022616 005037 002376          CLR          ERROR1          ;CLEAR ERROR FLAG
4170 022622 005037 002402          CLR          SAVLEN          ;CLEAR CHAR LENGTH FROM SETUP
4171 022626 005737 002370          TST          FRSTIM          ;SEE IF FIRST TIME THROUGH AFTER LOAD
4172 022632 001007          BNE          6$          ;BR IF NOT
4173 022634 013737 000004 002372          MOV          @#4,SAVE4          ;SAVE ERROR TRAP VECTOR
4174 022642 013737 000006 002374          MOV          @#6,SAVE6
4175 022650 000406          BR          9$
4176
4177 022652 013737 002372 000004 6$: MOV          SAVE4,@#4          ;RESTORE ERROR TRAP VECTOR
4178 022660 013737 002374 000006          MOV          SAVE6,@#6
4179
4180 022666 012737 000001 002370 9$: MOV          #1,FRSTIM          ;MARK FLAG FOR NEXT TIME THROUGH
4181
4182          ;SEE IF PROGRAM JUST STARTED, BR IF YES
4183 022674          READEF  #EF.START
4184          022674 012700 000040          MOV          #EF.START,RO
4185          022700 104447          TRAP          C$REFG
4186
4187 022702          BCOMPLETE          STARST
4188          022702 103415          BCS          STARST
4189
4190          ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
4191 022704          READEF  #EF.RESTART
4192          022704 012700 000037          MOV          #EF.RESTART,RO
4193          022710 104447          TRAP          C$REFG
4194
4195 022712          BCOMPLETE          STARST
4196          022712 103411          BCS          STARST
4197
4198          ;SEE IF THIS IS A NEW PASS, BR IF YES
4199 022714          READEF  #EF.NEW
4200          022714 012700 000035          MOV          #EF.NEW,RO
4201          022720 104447          TRAP          C$REFG
4202
4203 022722          BCOMPLETE          NEWST
4204          022722 103411          BCS          NEWST
4205
4206          ;SEE IF PROGRAM WAS JUST CONTINUED

```

INITIALIZE SECTION

```

4190 022724          READEF  #EF.CONTINUE
      022724 012700 000036          MOV  #EF.CONTINUE,RO
      022730 104447          TRAP  C$REFG
4191 022732          BCOMPLETE      ENDIT
      022732 103473          BCS   ENDIT
4192 022734 000414          BR    GETPRM
4193
4194 022736          STARST:
4195 022736 005037 002412      CLR    STARES          ;CLEAR FLAG TO SHOW JUST HAD STA OR RES
4196
4197          ;CLEAR DEVICE MAP
4198 022742 005037 002404      CLR    DEVMAP
4199 022746          NEWST:
4200 022746 012737 177777 002334      MOV    #-1,LOGDEV      ;RESET LOGICAL DEVICE TO -1
4201 022754 005237 002412          INC    STARES          ;INCREMENT NO. OF PASSES SINCE STA OR RES
4202 022760 012737 000001 002406      MOV    #BIT0,DEVPTR    ;INIT DEVICE MAP BIT POINTER
4203
4204          ; GET UNIBUS ADDRESS, VECTOR, PRIORITY LEVEL, SWITCH PACKS, TEST
4205          ; CONNECTOR INFORMATION FOR THIS LOGICAL DEVICE
4206 022766          GETPRM:
4207 022766 005237 002334          INC    LOGDEV          ;INCREMENT LOGICAL DEVICE NUMBER
4208 022772          GPHARD  LOGDEV,R1      ;GET P-TABLE POINTER INTO R1
      022772 013700 002334          MOV    LOGDEV,RO
      022776 104442          TRAP  C$GPHRD
      023000 010001          MOV    RO,R1
4209 023002          BCOMPLETE      10$      ;BR IF DEVICE AVAILABLE
      023002 103403          BCS   10$
4210 023004 006337 002406      ASL    DEVPTR          ;SHIFT DEVICE POINTER
4211 023010 000766          BR    GETPRM          ;SKIP THIS DEVICE
4212 023012 053737 002406 002404 10$:  BIS    DEVPTR,DEVMAP    ;SET BIT FOR THIS DEVICE
4213 023020 006337 002406      ASL    DEVPTR          ;SHIFT BIT POINTER
4214
4215 023024 012102          MOV    (R1)+,R2        ;R2=CSR ADDR VALUE
4216 023026 012703 002416      MOV    #MPCSR,R3      ;R3=POINTER TO CSR ADDR STORAGE AREA
4217
4218 023032 010223          11$:  MOV    R2,(R3)+        ;PUT CSR ADDRESSES IN 'BSEL' AREA
4219 023034 005202          INC    R2             ;BUMP BSEL ADDR
4220 023036 022703 002456      CMP    #BSEL17+2,R3   ;ALL 16 ADDRESSES MOVED ?
4221 023042 001373          BNE    11$           ;NO: DO ANOTHER ADDRESS
4222          ;YES: CONTINUE
4223
4224 023044 011137 002456      MOV    (R1),MPIVEC     ;GET DMV11 INPUT INTRPT VECTOR
4225 023050 012137 002460      MOV    (R1)+,MPOVEC    ;GET DMV11 OUTPUT INTRPT VECTOR
4226 023054 062737 000004 002460      ADD    #4,MPOVEC
4227 023062 012137 002462      MOV    (R1)+,MPRIOR   ;GET DMV11 DEVICE PRIORITY
4228 023066 012137 002464      MOV    (R1)+,LUSWI1    ;GET LU SWITCH PACK #1
4229 023072 012137 002466      MOV    (R1)+,LUSWI2    ;GET LU SWITCH PACK #2
4230 023076 012137 002470      MOV    (R1)+,BRDTYP    ;GET DMV-11 BOARD TYPE
4231 023102 012137 002472      MOV    (R1)+,TSTCON    ;GET TEST CONNECTOR INDICATOR
4232 023106 011137 002474      MOV    (R1),BDRATE     ;GET BAUD RATE FOR THIS DEVICE
4233          ;ISSUE LSI BUS RESET, TO INIT DMV11
4234 023112          BRESET
      023112 104433          TRAP  C$RESET
4235 023114 005000          CLR    RO             ;# TIME DELAY TO ALLOW COMPLETION
4236 023116 000240          15$:  NOP
      023120 077002          ;# OF DMV11 MICRODIAGNOSTICS.
4237 023120 077002          SOB    RO,15$
4238 023122          ENDIT:

```

INITIALIZE SECTION

4239 023122
023122
023122 104411

ENDINIT

L10013: TRAP C\$INIT

^

AUTO DROP UNIT SECTION

4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257 023124
023124
4258
4259 023124
023124 012746 000000
023130 012746 023242
023134 012746 000004
023140 012746 000003
023144 104437
023146 062706 000010
4260 023152 005037 002546
4261 023156 012702 000001
4262 023162 013703 002416
4263
4264 023166 105723
4265 023170 006302
4266 023172 103375
4267
4268 023174 013703 002416
4269 023200 012702 000001
4270 023204 005723
4271 023206 006302
4272 023210 006302
4273 023212 103374
4274
4275 023214
023214 012700 000004
023220 104436
4276 023222 005737 002546
4277 023226 001403
4278 023230
023230 013700 002334
023234 104451
4279
4280 023236 000240
4281
4282 023240
023240
023240 104461
4283
4284 023242 050237 002546

.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 CNOMA 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.
; *****

```

BGNAUTO

L\$AUTO::

```

SETVEC #4,#AD.HIT,#0 ;SETUP INVALID ADDRESS TRAP VECTOR
MOV #0,(SP)
MOV #AD.HIT,-(SP)
MOV #4,-(SP)
MOV #3,-(SP)
TRAP C$SVEC
ADD #10,SP

CLR TMO ;INITIALIZE TRAP FLAG REGISTER
MOV #1,R2 ;FLAG BIT
MOV BSEL0,R3 ;INIT ADDRESS POINTER

1$: TSTB (R3) ;ACCESS THE CSR S BY BYTES.
ASL R2
BCC 1$

MOV BSEL0,R3 ;RE-INIT ADDRESS POINTER
MOV #1,R2 ;RE-INIT FLAG BIT
2$: TST (R3) ;ACCESS THE CSR'S BY WORDS.
ASL R2
ASL R2
BCC 2$

CLRVEC #4 ;RESTORE THE VECTOR TO DS
MOV #4,R0
TRAP C$CVEC

TST TMO ;DID WE GET HIT WITH AN INVALID ADDRESS TRAP?
BEQ AD.OK ;NO, EXIT TEST
DODU LOGDEV ;YES, DROP THIS LOGICAL DEV.
MOV LOGDEV,R0
TRAP C$DODU

AD.OK: NOP ;(FOR PATCHING IN A HALT IF NECESSARY)

ENDAUTO

L10014: TRAP C$AUTO

AD.HIT: BIS R2,TMO ;FLAG THE HIT IF WE GET IT!

```

D9

CNDMEAO DMV11 LINE UNIT DIAGS MACRO M1200 22 FEB 84 15:48 PAGE 60 1

SEQ 0107

AUTO DROP UNIT SECTION

4285 023246 000002
4286

RTI

RETURN

CLEANUP CODING SECTION

4288
 4289
 4290
 4291
 4292
 4293
 4294
 4295 023250
 023250
 4296
 4297
 4298 023250
 023250
 023250 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

L\$CLEAN::

ENDCLN

L10015: TRAP C\$CLEAN

DROP UNIT SECTION

```

4300          .SBTTL  DROP UNIT SECTION
4301
4302          ;//////////
4303          ;/ THE DROP UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
4304          ;/ TO NO LONGER BE TESTED.
4305          ;//////////
4306
4307 023252          BGNDU
4308          ;ISSUE UNIBUS RESET TO CLEAN UP
4309 023252          BRESET
4310 023252 104433          ENDDU
         023254
         023254 104453
                                L$DU::
                                TRAP   C$RESET
                                L10016:
                                TRAP   C$DU

```

ADD UNIT SECTION

4312
 4313
 4314
 4315
 4316
 4317
 4318
 4319
 4320 023256
 023256
 4321 023256
 023256
 023256 104452

.SBTTL ADD UNIT SECTION

```

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

```

BGNAU

ENDAU

L\$AU: :

L10017:

TRAP C\$AU

TEST 1 RX DATA FLUSHING TEST

4332

.SBTTL TEST 1 -- RX DATA FLUSHING TEST

```

;*****
;
; TEST 1 -- RX DATA FLUSHING TEST
;
; IN BCP MODE/HALF DUPLEX IT IS DESIRABLE TO HAVE THE ABILITY TO FLUSH
; THE USYRT OF ITS CRC CHARACTERS. THIS FLUSHING IS ACCOMPLISHED BY READING
; TO THE VIA SHIFT REGISTER.
; THIS TEST VERIFIES THAT WHEN THE VIA SR IS READ, 8 PULSES WILL
; BE GENERATED AT THE CB1 PIN (WHICH DIRECTLY FEEDS THE CHARACTER FIFO).
;*****

```

```

;
; BGNTST
;
; T1::
4333 023260
4334 023260 004737 005376 JSR PC,INIDMV ;INIT DMV 11, ENTER M LOOP
4335
4336 023264 004537 007234 JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
4337 023270 063626 DDCMP:NOCHK!SYNCH!STRIPS ;SET DDCMP,NO CHECK,SYNCH=226
4338 023272 000000 0 ;USE 8 BIT CHARS
4339 023274 103003 BCC .+8. ;BR IF NO ERROR
4340 023276 ERROR ;REPORT STACKED ERROR
4341 023300 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
023300 104410 TRAP C$ESCAPE
023302 000254 .WORD L10020-
4342
4343 023304 004537 003712 JSR R5,WRITEI ;SET SHIFT REGISTER TO
4344 023310 120013 VIAACR ; "SYSTEM CLOCK RATE" MODE (CB1=CLK)
4345 023312 000210 210 ;(BIT 7 PREVIOUSLY SET)
4346 023314 103003 BCC .+8. ;BR IF NO ERROR
4347 023316 ERROR ; REPORT STACKED ERROR
4348 023320 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
023320 104410 TRAP C$ESCAPE
023322 000234 .WORD L10020 .
4349
4350 023324 004537 010250 JSR R5,TXCTRL ;OUTPUT 1ST SYNC CHARACTER
4351 023330 000001 TSOM ;AND KNOCK DOWN TBMT
4352 023332 000007 7.
4353
4354 023334 004537 010250 JSR R5,TXCTRL ;OUTPUT 2ND SYNC CHARACTER
4355 023340 000001 TSOM ;AND KNOCK DOWN TBMT
4356 023342 000010 8.
4357
4358 023344 004537 010250 JSR R5,TXCTRL ;CLEAR TSOM (GET READY TO SEND DATA)
4359 023350 000000 000
4360 023352 000000 0
4361
4362 023354 004537 010136 JSR R5,TXCHAR ;LOAD 125, TX 3RD SYNCH
4363 023360 000125 125
4364 023362 000010 8.
4365 023364 103003 BCC .+8. ;BR IF NO ERROR
4366 023366 ERROR ;REPORT STACKED ERROR

```

TEST 1 - RX DATA FLUSHING TEST

```

4367 023366 104460
023370          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
023370 104410          ;                               TRAP C$ESCAPE
023372 000164          ;                               .WORD L10020 .
4368
4369 023374 012702 000004      MOV    #4,R2          ;** TRANSFER 4 CHARACTERS **
4370
4371 023400 004537 003712      1$: JSR    R5,WRITEI    ;SET RTS & FULL DUPLEX (SO STEPLU WORKS)
4372 023404 120000          VIAORB
4373 023406 000142          TXEN!RXEN!TTLOOP
4374 023410 103003          BCC    .+8.          ;BR IF NO ERROR
4375 023412          ERROR          ; REPORT STACKED ERROR
023412 104460          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
4376 023414          ;                               TRAP C$ESCAPE
023414 104410          ;                               .WORD L10020 .
023416 000140
4377
4378 023420 004537 012072      JSR    R5,STEPLU    ;FLIP TSO BIT VALUE(WILL BE SHIFTED INTO
4379 023424 000001          1          ; FIFO DURING FLUSHING).
4380
4381 023426 004537 003712      JSR    R5,WRITEI    ;CLEAR RTS, SET MDX (SO THAT SR CLOCK WORKS)
4382 023432 120000          VIAORB
4383 023434 000152          TXEN!RXEN!RTSND!TTLOOP ;#
4384 023436 103003          BCC    .+8.          ;BR IF NO ERROR
4385 023440          ERROR          ; REPORT STACKED ERROR
023440 104460          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
4386 023442          ;                               TRAP C$ESCAPE
023442 104410          ;                               .WORD L10020-.
023444 000112
4387
4388 023446 004537 003566      JSR    R5,READI    ;READ VIA SHIFT REGISTER (SHOULD CAUSE
4389 023452 120012          VIASR          ; 8 CLOCKS FROM CB1 LEAD => FIFO)
4390 023454 000000          000
4391 023456 103003          BCC    .+8.          ;BR IF NO ERROR
4392 023460          ERROR          ; REPORT STACKED ERROR
023460 104460          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
4393 023462          ;                               TRAP C$ESCAPE
023462 104410          ;                               .WORD L10020 .
023464 000072
4394
4395 023466 077234          SOB    R2,1$        ;** LOOP UNTIL ALL 4 SENT (VIA CB1) **
4396
4397 023470 004537 010350      JSR    R5,RXCHAR    ;READ AND CHECK FOR 377
4398 023474 000377          377          ;* ERROR HERE INDICATES HI SPEED SR CLOCK
4399 023476 000000          0          ;* DIDN'T WORK.
4400 023500 100000          NOCRDA
4401 023502 103003          BCC    .+8.          ;BR IF NO ERROR
4402 023504          ERROR          ;REPORT STACKED ERROR
023504 104460          ESCAPE TST          ;SKIP TO END OF TEST          TRAP C$ERROR
4403 023506          ;                               TRAP C$ESCAPE
023506 104410          ;                               .WORD L10020 .
023510 000046
4404
4405 023512 004537 010350      JSR    R5,RXCHAR    ;READ AND CHECK FOR 000
4406 023516 000003          003          ;* ERROR HERE INDICATES HI SPEED SR CLOCK
4407 023520 000000          0          ;* DIDN'T WORK.
4408 023522 100000          NOCRDA

```

TEST 1 RX DATA FLUSHING TEST

```

4409 023524 103003      BCC      .+8.      ;BR IF NO ERROR
4410 023526      ERROR      ;REPORT STACKED ERROR
      023526 104460
4411 023530      ESCAPE  TST      ;SKIP TO END OF TEST      TRAP      C$ERROR
      023530 104410      ;                                TRAP      C$ESCAPE
      023532 000024      ;                                .WORD     L10020 .
4412
4413 023534 004537 010350 JSR      R5,RXCHAR  ;READ AND CHECK FOR 377
4414 023540 000360      360      ;* ERROR HERE INDICATES HI SPEED SR CLOCK
4415 023542 000000      0        ;* DIDN'T WORK
4416 023544 100000      NOCRDA
4417 023546 103003      BCC      .+8.      ;BR IF NO ERROR
4418 023550      ERROR      ;REPORT STACKED ERROR
      023550 104460      ;                                TRAP      C$ERROR
4419 023552      ESCAPE  TST      ;SKIP TO END OF TEST      TRAP      C$ESCAPE
      023552 104410      ;                                .WORD     L10020 .
      023554 000002
4420 023556      ENDTST
      023556      ;                                L10020:
      023556 104401      ;                                TRAP      C$ETST

```


TEST 2 INTEGRAL MODEM INTERFACE TEST

4434

.SBTTL TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

*****
;*
;* TEST 2 -- INTEGRAL MODEM INTERFACE TEST
;*
;* THE INTEGRAL MODEM IS SELECTED BY THE PROGRAM AND A MESSAGE IS
;* TRANSMITTED, RECEIVED, AND CHECKED USING A TURNAROUND CONNECTOR ON
;* THE BOARD OR AT THE END OF A CABLE. THE FOLLOWING MESSAGE WILL BE
;* SENT IN BCP MODE WITH CRC 16 SPECIFIED:
;*
;* SYNC SYNC 000 125 252 377 000 CRC1 CRC2 SYNC
;*
;* IF THE P-TABLE FOR THE CURRENT UNIT INDICATES THAT NO EXTERNAL
;* TURNAROUND IS PROVIDED, THE TEST WILL BE SKIPPED FOR THAT UNIT.
*****

```

BGNTST

```

4435 023560 012737 000002 002414      MOV    #2.,TSTNUM      ;SET TEST NO. FOR POSSIBLE PRINTOUT
4436 023566 004737 005376                JSR    PC,INIDMV      ;INIT DMV-11, ENTER MAINT LOOP
4437 023572 004537 007532                JSR    R5,CKLPBK     ;SEE IF THIS INTERFACE CAN BE RUN
4438 023576 000001                INTGRL
4439 023600 103002                BCC    2$            ;BR IF YES
4440 023602                EXIT    TST          ;WRONG INTERFACE - SKIP TEST
                                TRAP    C$EXIT
                                .WORD   L10021-.

4441
4442      ;LOAD 6502 MICROCODE FOR INTEGRAL MODEM TEST INTO RAM PAGE 2
4443 023606 012701 025436      2$:    MOV    #MCODE,R1      ;GET STARTING ADRS OF DMV MICROCODE
4444 023612 012702 001000      MOV    #RAMADR,R2      ;GET STARTING ADRS OF RAM PAGE 2
4445 023616 112137 023634      3$:    MOVB   (R1)+,6$     ;SET DATA BYTE TO BE WRITTEN
4446 023622 010237 023632      MOV    R2,4$          ;SET RAM WRITE ADRS
4447 023626 004537 003712      JSR    R5,WRITEI     ;WRITE A DATA BYTE INTO RAM
4448 023632 000000      4$:    .WORD   0
4449 023634 000000      6$:    .WORD   0
4450 023636 005202      INC    R2            ;INCR RAM ADRS
4451 023640 020127 026263      CMP    R1,#ENDCOD    ;SEE IF ALL CODE LOADED YET
4452 023644 103764      BLO    3$            ;BR IF NOT

4453
4454      ;READ AND VERIFY 6502 MICROCODE IN RAM
4455 023646 012701 025436      MOV    #MCODE,R1      ;GET STARTING ADRS OF DMV MICROCODE TO CHECK
4456 023652 012702 001000      MOV    #RAMADR,R2      ;GET STARTING ADRS OF RAM PAGE 2
4457 023656 010237 023666      8$:    MOV    R2,10$     ;SET RAM READ ADRS
4458 023662 004537 003566      JSR    R5,READI     ;READ A RAM BYTE
4459 023666 000000      10$:   .WORD   0
4460 023670 000000      12$:   .WORD   0
4461 023672 122137 023670      CMPB   (R1)+,12$     ;SEE IF BYTE IS CORRECT
4462 023676 001422      BEQ    16$          ;BR IF CORRECT
4463 023700 010237 002336      MOV    R2,REGNUM     ;SET RAM ADRS FOR ERROR REPORT
4464 023704 005037 002324      CLR    GDATA        ;SET EXPECTED RAM DATA
4465 023710 116137 177777 002324      MOVB   -1(R1),GDATA
4466 023716 005037 002326      CLR    BDATA        ;SET ACTUAL RAM DATA
4467 023722 113737 023670 002326      MOVB   12$,BDATA
4468
4469 023730      ;REPORT RAM ERROR LOADING MICROCODE
                                GEDF   EM79,ERR13

```

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

                                ; "DEVICE FATAL" ERROR # 38
                                TRAP      C$ERDF
                                .WORD    38
                                .WORD    EM79
                                .WORD    ERR13
023730 104455
023732 000046
023734 015125
023736 021030
4470 023740          ESCAPE  TST
                                TRAP      C$ESCAPE
                                .WORD    L10021-.
                                023740 104410
                                023742 001472
4471 023744 005202
4472 023746 020127 026263
4473 023752 103741
4474
4475
                                ;SET UP VIA AND USYRT FOR OPERATION
4476 023754 004537 003712      JSR      R5,WRITEI      ;RESET THE USYRT
                                VIAORB
4477 023760 120000
                                RTSND!DTR!PRESET
4478 023762 000031
                                JSR      R5,WRITEI      ;CLEAR USYRT RESET BIT
4479 023764 004537 003712      VIAORB
4480 023770 120000
                                RTSND!DTR
4481 023772 000030
                                JSR      R5,WRITEI      ;SET SYNCH CHAR = 226
4482 023774 004537 003712      PCSARL
4483 024000 120404
                                226
4484 024002 000226
                                MOV      #DDCMP!STRIPS!IDLES!CRC16,18$ ;SET DDCMP,STRIP,IDLE, CRC16
4485 024004 012737 065400 024024 SWAB      18$           ;GET DATA INTO LO BYTE
4486 024012 000337 024024
                                JSR      R5,WRITEI      ;PROGRAM THE PCSARH
4487 024016 004537 003712      PCSARH
4488 024022 120405
                                18$: .WORD    0
4489 024024 000000
                                JSR      R5,CKUSTS      ;CHK USYRT STATUS FOR INITIALIZED STATE
4490 024026 004537 005410
                                110
4491 024032 000110
                                BCC      .+8.           ; IF TBMT=1, TSO=1
4492 024034 103003
                                ERROR
4493 024036
                                TRAP      C$ERROR
                                024036 104460
4494 024040          ESCAPE  TST
                                TRAP      C$ESCAPE
                                .WORD    L10021-.
                                024040 104410
                                024042 001372
4495 024044 004537 003712      JSR      R5,WRITEI      ;SET TSOM IN USYRT
4496 024050 120403
                                TDSRH
4497 024052 000001
                                TSOM
4498 024054 004537 003712      JSR      R5,WRITEI      ;LOAD 237 CHAR FOR INTGRL MODEM SYNCHRONIZATION
4499 024060 120402
                                TDSRL
4500 024062 000237
                                237
4501 024064 004537 006014      JSR      R5,CKTBMT
4502 024070 000000
                                0
4503 024072 103003
                                BCC      .+8.           ; IF TBMT = 0
4504 024074
                                ERROR
                                TRAP      C$ERROR
                                024074 104460
4505 024076          ESCAPE  TST
                                TRAP      C$ESCAPE
                                .WORD    L10021 .
                                024076 104410
                                024100 001334
4506
4507
                                ;INITIATE 6502 TEST OF INTEGRAL MODEM
4508 024102 005077 156320      CLR      @SEL4          ;CLEAR SEL4
4509 024106 012777 001000 156312 MOV      #RAMADR,@SEL4  ;SET START ADRS OF RAM CODE IN SEL4
4510 024114 112777 000005 156300 MOVB     #EXECUT,@SEL2  ;ISSUE M-LOOP CMND TO EXECUTE AT PC IN SEL4
4511
                                ;WAIT SEVERAL MILLI-SEC FOR COMPLETION OF TEST
4512 024122 012701 001750      MOV      #1000.,R1     ;INIT WAIT LOOP COUNTER
4513 024126 005301
                                22$: DEC      R1           ;DECREMENT COUNTER

```

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

4514 024130 000240      NOP
4515 024132 001375      BNE      22$           ;BR IF NOT DONE COUNTING YET
4516 024134 132777 000200 156260  BITB     @MRDY,@BSEL2 ;SEE IF M-LOOP FINISHED PROPERLY
4517 024142 001013      BNE      24$           ;BR IF YES
4518 024144 004737 004166  JSR      PC,GETWSR    ;GET CSR'S FOR PRINTOUT
4519 024150 012737 000005 002324  MOV      @EXECUT,GDATA ;IDENTIFY REQUESTED FUNCTION
4520                                ;REPORT MRDY TIMEOUT ERROR
4521 024156                                GEDF     EM4,ERR4
;
;           "DEVICE FATAL" ERROR # 39
;           TRAP      C$ERDF
;           .WORD     39
;           .WORD     EM4
;           .WORD     ERR4
      024156 104455
      024160 000047
      024162 014307
      024164 020120
4522 024166                                ESCAPE  TST
      024166 104410                                TRAP      C$ESCAPE
      024170 001244                                .WORD     L10021-.
4523
4524 024172 105777 156234 24$:  TSTB     @BSEL6    ;SEE IF ANY ERRORS OCCURRED IN TEST
4525 024176 001002      BNE      26$           ;BR IF ERROR DETECTED
4526 024200 000137 025434  JMP      90$           ;SUCCESSFUL COMPLETION OF TEST
4527 024204 012737 000007 002336 26$:  MOV      @7,REGNUM   ;SET REG NO. FOR PRINTOUT
4528 024212 005037 002324  CLR      GDATA        ;CLEAR EXPECTED DATA AREA
4529 024216 005037 002326  CLR      BDATA        ;CLEAR ACTUAL DATA AREA
4530                                ;CHK FOR ERROR 1
4531 024222 127727 156204 000001  CMPB     @BSEL6,#1    ;CHK FOR ERROR 1
4532 024230 001006      BNE      28$           ;BR IF NOT
4533                                ;REPORT TBMT NOT SET
4534 024232                                GEDF     EM73,ERR12
;
;           "DEVICE FATAL" ERROR # 40
;           TRAP      C$ERDF
;           .WORD     40
;           .WORD     EM73
;           .WORD     ERR12
      024232 104455
      024234 000050
      024236 014777
      024240 020714
4535 024242                                ESCAPE  TST
      024242 104410                                TRAP      C$ESCAPE
      024244 001170                                .WORD     L10021-.
4536
4537 024246 127727 156160 000002  ;CHK FOR ERROR 2
4538 024254 001006 28$:  CMPB     @BSEL6,#2    ;CHK FOR ERROR 2
4539                                BNE      30$           ;BR IF NOT
4540                                ;REPORT TBMT NOT SET
      024256 104455                                GEDF     EM73,ERR12
      024260 000051
      024262 014777
      024264 020714
4541 024266                                ESCAPE  TST
      024266 104410                                TRAP      C$ESCAPE
      024270 001144                                .WORD     L10021-.
4542
4543 024272 127727 156134 000003  ;CHK FOR ERROR 3
4544 024300 001006 30$:  CMPB     @BSEL6,#3    ;CHK FOR ERROR 3
4545                                BNE      31$           ;BR IF NOT
4546 024302                                ;REPORT TBMT NOT SET
      024302 104455                                GEDF     EM73,ERR12
      024304 000052
;
;           "DEVICE FATAL" ERROR # 42
;           TRAP      C$ERDF
;           .WORD     42

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

	024306	014777										.WORD	EM73
	024310	020714										.WORD	ERR12
4547	024312					ESCAPE	TST						
	024312	104410										TRAP	C\$ESCAPE
	024314	001120										.WORD	L10021 .
4548													
4549	024316	127727	156110	000004		;CHK FOR ERROR 4							
4550	024324	001011				31\$:	CMPB	08SEL6,04		;CHK FOR ERROR 4			
4551							BNE	32\$;BR IF NOT			
4552	024326	012737	000001	002336		;REPORT	CARRIER	NOT SET					
4553	024334						MOV	01,REGNUM		;SET REG NO. FOR PRINTOUT			
							GEDF	EM80,ERR14					
	024334	104455								; "DEVICE FATAL" ERROR # 43			
	024336	000053										TRAP	C\$ERDF
	024340	015161										.WORD	43
	024342	021146										.WORD	EM80
4554	024344					ESCAPE	TST					.WORD	ERR14
	024344	104410										TRAP	C\$ESCAPE
	024346	001066										.WORD	L10021-.
4555													
4556	024350	127727	156056	000005		;CHK FOR ERROR 5							
4557	024356	001006				32\$:	CMPB	08SEL6,05		;CHK FOR ERROR 5			
4558							BNE	34\$;BR IF NOT			
4559	024360					;REPORT	TBMT	NOT SET					
							GEDF	EM73,ERR12					
	024360	104455								; "DEVICE FATAL" ERROR # 44			
	024362	000054										TRAP	C\$ERDF
	024364	014777										.WORD	44
	024366	020714										.WORD	EM73
4560	024370					ESCAPE	TST					.WORD	ERR12
	024370	104410										TRAP	C\$ESCAPE
	024372	001042										.WORD	L10021 .
4561													
4562	024374	127727	156032	000006		;CHK FOR ERROR 6							
4563	024402	001006				34\$:	CMPB	08SEL6,06		;CHK FOR ERROR 6			
4564							BNE	36\$;BR IF NOT			
4565	024404					;REPORT	TBMT	NOT SET					
							GEDF	EM73,ERR12					
	024404	104455								; "DEVICE FATAL" ERROR # 45			
	024406	000055										TRAP	C\$ERDF
	024410	014777										.WORD	45
	024412	020714										.WORD	EM73
4566	024414					ESCAPE	TST					.WORD	ERR12
	024414	104410										TRAP	C\$ESCAPE
	024416	001016										.WORD	L10021-.
4567													
4568	024420	127727	156006	000007		;CHK FOR ERROR 7							
4569	024426	001006				36\$:	CMPB	08SEL6,07		;CHK FOR ERROR 7			
4570							BNE	38\$;BR IF NOT			
4571	024430					;REPORT	TBMT	NOT SET					
							GEDF	EM73,ERR12					
	024430	104455								; "DEVICE FATAL" ERROR # 46			
	024432	000056										TRAP	C\$ERDF
	024434	014777										.WORD	46
	024436	020714										.WORD	EM73
4572	024440					ESCAPE	TST					.WORD	ERR12
	024440	104410										TRAP	C\$ESCAPE

TEST 2 INTEGRAL MODEM INTERFACE TEST

```

024442 000772 .WORD L10021..
4573
4574 024444 127727 155762 000010 ;CHK FOR ERROR 8
381: CMPB 88SEL6,08. ;CHK FOR ERROR 8
4575 024452 001006 BNE 401 ;BR IF NOT
4576 ;REPORT TBMT NOT SET
4577 024454 GEDF EM73,ERR12
; "DEVICE FATAL" ERROR # 47
TRAP C$ERDF
024454 104455 .WORD 47
024456 000057 .WORD EM73
024460 014777 .WORD ERR12
024462 020714
4578 024464 ESCAPE TST
TRAP C$ESCAPE
024464 104410 .WORD L10021.
024466 000746
4579 ;CHK FOR ERROR 9
4580 024470 127727 155736 000011 401: CMPB 88SEL6,09. ;CHK FOR ERROR 9
4581 024476 001006 BNE 421 ;BR IF NOT
4582 ;REPORT RDA NOT SET
4583 024500 GEDF EM75,ERR12
; "DEVICE FATAL" ERROR # 48
TRAP C$ERDF
024500 104455 .WORD 48
024502 000060 .WORD EM75
024504 015035 .WORD ERR12
024506 020714
4584 024510 ESCAPE TST
TRAP C$ESCAPE
024510 104410 .WORD L10021.
024512 000722
4585 ;CHK FOR ERROR 10
4586 024514 127727 155712 000012 421: CMPB 88SEL6,010. ;CHK FOR ERROR 10
4587 024522 001017 BNE 441 ;BR IF NOT
4588 ;REPORT RCV'D DATA MISCOMPARE ERROR
4589 024524 012737 000000 002336 MOV 00,REGNUM ;SET REG NO. FOR PRINTOUT
4590 024532 112737 000000 002324 MOV 0000,GDATA ;SET EXPECTED DATA
4591 024540 117737 155670 002326 MOV 88SEL7,BDATA ;SET ACTUAL DATA
4592 024546 GEDF EM34,ERR10
; "DEVICE FATAL" ERROR # 49
TRAP C$ERDF
024546 104455 .WORD 49
024550 000061 .WORD EM34
024552 014502 .WORD ERR10
024554 020364
4593 024556 ESCAPE TST
TRAP C$ESCAPE
024556 104410 .WORD L10021.
024560 000654
4594 ;CHK FOR ERROR 11
4595 024562 127727 155644 000013 441: CMPB 88SEL6,011. ;CHK FOR ERROR 11
4596 024570 001006 BNE 461 ;BR IF NOT
4597 ;REPORT RDA NOT SET
4598 024572 GEDF EM75,ERR12
; "DEVICE FATAL" ERROR # 50
TRAP C$ERDF
024572 104455 .WORD 50
024574 000062 .WORD EM75
024576 015035 .WORD ERR12
024600 020714
4599 024602 ESCAPE TST
TRAP C$ESCAPE
024602 104410 .WORD L10021.
024604 000630
4600 ;CHK FOR ERROR 12

```

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

4601 024606 127727 155620 000014 461:  CMPB  @SEL6,@12.  ;CHK FOR ERROR 12
4602 024614 001017  BNE  481      ;BR IF NOT
4603                ;REPORT RCV'D DATA MISCOMPARE ERROR
4604 024616 012737 000000 002336  MOV  @0,REGNUM ;SET REG NO. FOR PRINTOUT
4605 024624 112737 000125 002324  MOVB @125,GDATA ;SET EXPECTED DATA
4606 024632 117737 155576 002326  MOVB @SEL7,BDATA ;SET ACTUAL DATA
4607 024640  GDF  EM34,ERR10

                ; "DEVICE FATAL" ERROR # 51
                TRAP  C$ERDF
                .WORD 51
                .WORD EM34
                .WORD ERR10
4608 024650                ESCAPE TST
                TRAP  C$ESCAPE
                .WORD L10021
4609                ;CHK FOR ERROR 13
4610 024654 127727 155552 000015 481:  CMPB  @SEL6,@13.  ;CHK FOR ERROR 13
4611 024662 001006  BNE  501      ;BR IF NOT
4612                ;REPORT RDA NOT SET
4613 024664  GDF  EM75,ERR12

                ; "DEVICE FATAL" ERROR # 52
                TRAP  C$ERDF
                .WORD 52
                .WORD EM75
                .WORD ERR12
4614 024674                ESCAPE TST
                TRAP  C$ESCAPE
                .WORD L10021
4615                ;CHK FOR ERROR 14
4616 024700 127727 155526 000016 501:  CMPB  @SEL6,@14.  ;CHK FOR ERROR 14
4617 024706 001017  BNE  521      ;BR IF NOT
4618                ;REPORT RCV'D DATA MISCOMPARE ERROR
4619 024710 012737 000000 002336  MOV  @0,REGNUM ;SET REG NO. FOR PRINTOUT
4620 024716 112737 000252 002324  MOVB @252,GDATA ;SET EXPECTED DATA
4621 024724 117737 155504 002326  MOVB @SEL7,BDATA ;SET ACTUAL DATA
4622 024732  GDF  EM34,ERR10

                ; "DEVICE FATAL" ERROR # 53
                TRAP  C$ERDF
                .WORD 53
                .WORD EM34
                .WORD ERR10
4623 024742                ESCAPE TST
                TRAP  C$ESCAPE
                .WORD L10021
4624                ;CHK FOR ERROR 15
4625 024746 127727 155460 000017 521:  CMPB  @SEL6,@15.  ;CHK FOR ERROR 15
4626 024754 001006  BNE  541      ;BR IF NOT
4627                ;REPORT RDA NOT SET
4628 024756  GDF  EM75,ERR12

                ; "DEVICE FATAL" ERROR # 54
                TRAP  C$ERDF
                .WORD 54
                .WORD EM75
                .WORD ERR12
4629 024766                ESCAPE TST
                TRAP  C$ESCAPE
                .WORD L10021
024640 104455
024642 000063
024644 014502
024646 020364
024664 104455
024666 000064
024670 015035
024672 020714
024674 104410
024676 000536
024732 104455
024734 000065
024736 014502
024740 020364
024756 104455
024760 000066
024762 015035
024764 020714
024766 104410
024770 000444

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

```

4630 ;CHK FOR ERROR 16
4631 024772 127727 155434 000020 54$: CMPB 8BSEL6,016. ;CHK FOR ERROR 16
4632 025000 001017 BNE 56$ ;BR IF NOT
4633 ;REPORT RCV'D DATA MISCOMPARE ERROR
4634 025002 012737 000000 002336 MOV 00,REGNUM ;SET REG NO. FOR PRINTOUT
4635 025010 112737 000377 002324 MOVB 0377,GDATA ;SET EXPECTED DATA
4636 025016 117737 155412 002326 MOVB 8BSEL7,BDATA ;SET ACTUAL DATA
4637 025024 GEDF EM34,ERR10
; "DEVICE FATAL" ERROR # 55
TRAP C$ERDF
. WORD 55
. WORD EM34
. WORD ERR10
4638 025024 104455
025026 000067
025030 014502
025032 020364
025034 ESCAPE TST
025034 104410 TRAP C$ESCAPE
025036 000376 . WORD L10021 .
4639 ;CHK FOR ERROR 17
4640 025040 127727 155366 000021 56$: CMPB 8BSEL6,017. ;CHK FOR ERROR 17
4641 025046 001006 BNE 58$ ;BR IF NOT
4642 ;REPORT RDA NOT SET
4643 025050 GEDF EM75,ERR12
; "DEVICE FATAL" ERROR # 56
TRAP C$ERDF
. WORD 56
. WORD EM75
. WORD ERR12
4644 025050 104455
025052 000070
025054 015035
025056 020714
025060 ESCAPE TST
025060 104410 TRAP C$ESCAPE
025062 000352 . WORD L10021 .
4645 ;CHK FOR ERROR 18
4646 025064 127727 155342 000022 58$: CMPB 8BSEL6,018. ;CHK FOR ERROR 18
4647 025072 001017 BNE 60$ ;BR IF NOT
4648 ;REPORT RCV'D DATA MISCOMPARE ERROR
4649 025074 012737 000000 002336 MOV 00,REGNUM ;SET REG NO. FOR PRINTOUT
4650 025102 112737 000000 002324 MOVB 0000,GDATA ;SET EXPECTED DATA
4651 025110 117737 155320 002326 MOVB 8BSEL7,BDATA ;SET ACTUAL DATA
4652 025116 GEDF EM34,ERR10
; "DEVICE FATAL" ERROR # 57
TRAP C$ERDF
. WORD 57
. WORD EM34
. WORD ERR10
4653 025116 104455
025120 000071
025122 014502
025124 020364
025126 ESCAPE TST
025126 104410 TRAP C$ESCAPE
025130 000304 . WORD L10021 .
4654 ;CHK FOR ERROR 19
4655 025132 127727 155274 000023 60$: CMPB 8BSEL6,019. ;CHK FOR ERROR 19
4656 025140 001006 BNE 62$ ;BR IF NOT
4657 ;REPORT RERR NOT SET
4658 025142 GEDF EM36,ERR10
; "DEVICE FATAL" ERROR # 58
TRAP C$ERDF
. WORD 58
. WORD EM36
. WORD ERR10
4659 025142 104455
025144 000072
025146 014551
025150 020364
025152 ESCAPE TST
025152 104410 TRAP C$ESCAPE

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

```

025154 000260                                     .WORD L10021-.
4660                                     ;CHK FOR ERROR 20
4661 025156 127727 155250 000024 62$:  CMPB  88SEL6,#20.  ;CHK FOR ERROR 20
4662 025164 001006                                     ;BR IF NOT
4663                                     ;REPORT RDA NOT SET
4664 025166                                     GEDF  EM75,ERR12
; "DEVICE FATAL" ERROR # 59
025166 104455                                     TRAP  C$ERDF
025170 000073                                     .WORD 59
025172 015035                                     .WORD EM75
025174 020714                                     .WORD ERR12
4665 025176                                     ESCAPE TST
025176 104410                                     TRAP  C$ESCAPE
025200 000234                                     .WORD L10021-.
4666                                     ;CHK FOR ERROR 21
4667 025202 127727 155224 000025 64$:  CMPB  88SEL6,#21.  ;CHK FOR ERROR 21
4668 025210 001017                                     ;BR IF NOT
4669                                     ;REPORT RCV'D DATA MISCOMPARE ERROR
4670 025212 012737 000000 002336  MOV  #0,REGNUM  ;SET REG NO. FOR PRINTOUT
4671 025220 112737 000160 002324  MOVB  #160,GDATA ;SET EXPECTED DATA
4672 025226 117737 155202 002326  MOVB  88SEL7,BDATA ;SET ACTUAL DATA
4673 025234                                     GEDF  EM34,ERR10
; "DEVICE FATAL" ERROR # 60
025234 104455                                     TRAP  C$ERDF
025236 000074                                     .WORD 60
025240 014502                                     .WORD EM34
025242 020364                                     .WORD ERR10
4674 025244                                     ESCAPE TST
025244 104410                                     TRAP  C$ESCAPE
025246 000166                                     .WORD L10021-.
4675                                     ;CHK FOR ERROR 22
4676 025250 127727 155156 000026 66$:  CMPB  88SEL6,#22.  ;CHK FOR ERROR 22
4677 025256 001006                                     ;BR IF NOT
4678                                     ;REPORT RDA NOT SET
4679 025260                                     GEDF  EM75,ERR12
; "DEVICE FATAL" ERROR # 61
025260 104455                                     TRAP  C$ERDF
025262 000075                                     .WORD 61
025264 015035                                     .WORD EM75
025266 020714                                     .WORD ERR12
4680 025270                                     ESCAPE TST
025270 104410                                     TRAP  C$ESCAPE
025272 000142                                     .WORD L10021-.
4681                                     ;CHK FOR ERROR 23
4682 025274 127727 155132 000027 68$:  CMPB  88SEL6,#23.  ;CHK FOR ERROR 23
4683 025302 001017                                     ;BR IF NOT
4684                                     ;REPORT RCV'D DATA MISCOMPARE ERROR
4685 025304 012737 000000 002336  MOV  #0,REGNUM  ;SET REG NO. FOR PRINTOUT
4686 025312 112737 000034 002324  MOVB  #034,GDATA ;SET EXPECTED DATA
4687 025320 117737 155110 002326  MOVB  88SEL7,BDATA ;SET ACTUAL DATA
4688 025326                                     GEDF  EM34,ERR10
; "DEVICE FATAL" ERROR # 62
025326 104455                                     TRAP  C$ERDF
025330 000076                                     .WORD 62
025332 014502                                     .WORD EM34
025334 020364                                     .WORD ERR10
4689 025336                                     ESCAPE TST

```


TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

025336 104410
025340 000074
4690
4691 025342 127727 155064 000030 ;CHK FOR ERROR 24
4692 025350 001006 70$: CMPB 88SEL6,024. ;CHK FOR ERROR 24
4693 ;REPORT RDA NOT SET ;BR IF NOT
4694 025352 GEDF EM75,ERR12
; "DEVICE FATAL" ERROR # 63
025352 104455 TRAP C$ERDF
025354 000077 .WORD 63
025356 015035 .WORD EM75
025360 020714 .WORD ERR12
4695 025362 ESCAPE TST
025362 104410 TRAP C$ESCAPE
025364 000050 .WORD L10021-.
4696
4697 025366 127727 155040 000031 ;CHK FOR ERROR 25
4698 025374 001011 72$: CMPB 88SEL6,025. ;CHK FOR ERROR 25
4699 ;REPORT CARRIER NOT CLEARED ;BR IF NOT
4700 025376 012737 000001 002336 ;REPORT MOV 01,REGNUM ;SET REG NO. FOR PRINTOUT
4701 025404 GEDF EM81,ERR14
; "DEVICE FATAL" ERROR # 64
025404 104455 TRAP C$ERDF
025406 000100 .WORD 64
025410 015201 .WORD EM81
025412 021146 .WORD ERR14
4702 025414 ESCAPE TST
025414 104410 TRAP C$ESCAPE
025416 000016 .WORD L10021-.
4703 025420 004737 004166 74$: JSR PC,GETWSR ;GET CSR'S FOR PRINTOUT
4704 ;REPORT INVALID ERROR CODE FROM 6502
4705 025424 GEDF EM82,ERR3
; "DEVICE FATAL" ERROR # 65
025424 104455 TRAP C$ERDF
025426 000101 .WORD 65
025430 015225 .WORD EM82
025432 020072 .WORD ERR3
4706
4707 025434 90$:
4708 025434 ENDTST
025434 104401 L10021: TRAP C$ETST

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

```

4710 025436          MCODE:
4711                ;
4712                ;LINE# LOC  CODE      LINE
4713                ;
4714                ;0001  0000          **$0200          ;START OF MICROCODE FOR INTEGRAL MOD
EM TEST
4715                ;0002  0200
4716                ;0003  0200
4717                ;0005  0200
4718                ;0006  0200 ;*****
4719                ;0007  0200 ; THIS IS THE 6502 MICROCODE WHICH IS LOADED INTO RAM AND EXECUTED FOR THE
4720                ;0008  0200 ; PURPOSE OF TESTING THE INTEGRAL MODEM ON THE M8064, AT 56K BAUD. AFTER THE
4721                ;0009  0200 ; LSI-11 PROGRAM DOES SOME INITIAL SETUP, IT TRANSFERS CONTROL TO THIS CODE
4722                ;0010  0200 ; IN RAM, AND WAITS FOR COMPLETION OF THE TEST, AS INDICATED BY MRDY SET.
4723                ;0011  0200 ; THIS CODE TRANSMITS, RECEIVES, AND CHECKS THE FOLLOWING CHARACTERS :
4724                ;0012  0200 ; 2 SYNCH CHARACTERS, 5 DATA CHARACTERS 000, 125, 252, 377, 000, 2 CRC-16
4725                ;0013  0200 ; CHARACTERS 160 AND 034, AND 2 TERMINATING SYNCHS. THE MESSAGE IS SENT USING
4726                ;0014  0200 ; CHARACTER (DDCMP) MODE, THE SYNCH CHARACTER USED IS 226, STRIP SYNCH AND
4727                ;0015  0200 ; IDLE MODES ARE SET, AND THE DATA CLOCK IS PROVIDED BY THE INTEGRAL MODEM.
4728                ;0016  0200 ; ALL DATA AND CRC CHARACTERS ARE CHECKED AS THEY ARE RECEIVED, AND THE CRC
4729                ;0017  0200 ; ERROR CHECK BIT IS CHECKED TO BE SET WITH RECEPTION OF THE LAST DATA
4730                ;0018  0200 ; CHARACTER (000).
4731                ;0019  0200 ;*****
4732                ;0020  0200
4733                ;0021  0200          ;EQUATES FOR BIT DEFINITIONS
4734                ;0022  0200          BIT0   =@1
4735                ;0023  0200          BIT1   =@2
4736                ;0024  0200          BIT2   =@4
4737                ;0025  0200          BIT3   =@10
4738                ;0026  0200          BIT4   =@20
4739                ;0027  0200          BIT5   =@40
4740                ;0028  0200          BIT6   =@100
4741                ;0029  0200          BIT7   =@200
4742                ;0030  0200          BIT8   =@400
4743                ;0031  0200          BIT9   =@1000
4744                ;0032  0200          BIT10  =@2000
4745                ;0033  0200          BIT11  =@4000
4746                ;0034  0200          BIT12  =@10000
4747                ;0035  0200          BIT13  =@20000
4748                ;0036  0200          BIT14  =@40000
4749                ;0037  0200          BIT15  =@100000
4750                ;0038  0200
4751                ;0039  0200
4752                ;0040  0200
4753                ;0041  0200          ;ADDRESS EQUATES FOR CSR REGISTERS
4754                ;0042  0200          SEL0   =$10
4755                ;0043  0200          BSEL0  =SEL0
4756                ;0044  0200          BSEL1  =SEL0.1
4757                ;0045  0200          SEL2   =SEL0.2
4758                ;0046  0200          BSEL2  =SEL0.2
4759                ;0047  0200          BSEL3  =SEL0.3
4760                ;0048  0200          SEL4   =SEL0.4
4761                ;0049  0200          BSEL4  =SEL0.4
4762                ;0050  0200          BSEL5  =SEL0.5
4763                ;0051  0200          SEL6   =SEL0.6
4764                ;0052  0200          BSEL6  =SEL0.6
4765                ;0053  0200          BSEL7  =SEL0.7
4766                ;0054  0200

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

```

4767 ;0055 0200
4768 ;0056 0200
4769 ;0057 0200
4770 ;0058 0200
4771 ;0059 0200
4772 ;0060 0200
4773 ;0061 0200
4774 ;0062 0200
4775 ;0063 0200
4776 ;0064 0200
4777 ;0065 0200
4778 ;0066 0200
4779 ;0067 0200
4780 ;0068 0200
4781 ;0069 0200
4782 ;0070 0200
4783 ;0071 0200
4784 ;0072 0200
4785 ;0073 0200
4786 ;0074 0200
4787 ;0075 0200
4788 ;0076 0200
4789 ;0077 0200
4790 ;0078 0200
4791 ;0079 0200
4792 ;0080 0200
4793 ;0081 0200
4794 ;0082 0200
4795 ;0083 0200
4796 ;0084 0200
4797 ;0085 0200
4798 ;0086 0200
4799 ;0087 0200
4800 ;0088 0200
4801 ;0089 0200
4802 ;0090 0200
4803 ;0091 0200
4804 ;0092 0200
4805 ;0093 0200
4806 ;0094 0200
4807 ;0095 0200
4808 ;0096 0200
4809 ;0097 0200
4810 ;0098 0200
4811 ;0099 0200
4812 ;0100 0200
4813 ;0101 0200
4814 ;0102 0200
4815 ;0103 0200
4816 ;0104 0200
4817 ;0105 0200
4818 ;0106 0200
4819 ;0107 0200
4820 ;0108 0200
4821 ;0109 0200
4822 ;0110 0200
4823 ;0111 0200

```

```

;VERSATILE INTERFACE ADAPTER REGISTER EQUATES
OREGB = $A000 ;OUTPUT REGISTER B
OREGA = OREGB+1 ;OUTPUT REGISTER A
DDR8 = OREGB+2 ;DATA DIRECTION REGISTER B
DDRA = OREGB+3 ;DATA DIRECTION REGISTER A
T1LL = OREGB+6 ;TIMER 1 LATCH LOW BITS
T1LH = OREGB+7 ;TIMER 1 LATCH HIGH BITS
ACR = OREGB+$B ;AUXILIARY CONTROL REGISTER
PCR = OREGB+$C ;PERIPHERAL CONTROL REGISTER

```

```

;VIA OUTPUT REGISTER B BIT EQUATES
NULCLK = BIT7
RXEN = BIT6
TXEN = BIT5
DTR = BIT4
RTSND = BIT3
MDX = BIT2
TTLOOP = BIT1
PRESET = BIT0

```

```

;VIA OUTPUT REGISTER A BIT EQUATES
RING = BIT7
CARRIER = BIT6
MDMRDY = BIT5
BDRATE = BIT4
CTS = BIT3
TM = BIT2
RCVDAT = BIT1
UMAINT = BIT0

```

```

;USYRT REGISTER ADDRESS EQUATES
RXDB = $A100
RDSR = RXDB+1
TXDB = RXDB+2
TDSR = RXDB+3
SAR = RXDB+4
PCSAR = RXDB+5
PCTLR = RXDB+7

```

```

;USYRT TDSR REGISTER BIT EQUATES
TEOM = BIT1
TSOM = BIT0

```

```

;USYRT RDSR BIT EQUATES
RERR = BIT7

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

```

4824 ;0112 0200
4825 ;0113 0200
4826 ;0114 0200
4827 ;0115 0200
4828 ;0116 0200
4829 ;0117 0200
4830 ;0118 0200
4831 ;0119 0200
4832 ;0120 0200
4833 ;0121 0200
4834 ;0122 0200
4835 ;0123 0200
4836 ;0124 0200
4837 ;0125 0200
4838 ;0126 0200
4839 ;0127 0200
4840 ;0128 0200
4841 ;0129 0200
4842 ;0130 0200
4843 ;0131 0200 A0 00
4844 025436 240 000 .BYTE 240,000
4845 ;0132 0202 84 16 STY BSEL6 ;CLEAR BSEL6
4846 025440 204 026 .BYTE 204,026
4847 ;0133 0204 84 17 STY BSEL7 ;CLEAR BSEL7
4848 025442 204 027 .BYTE 204,027
4849 ;0134 0206 ;TURN ON THE USYRT, CLOCK
4850 ;0135 0206 A2 60 LDX #TXEN!RXEN ;ASSERT TXEN,RXEN,RTS,DTR
4851 025444 242 140 .BYTE 242,140
4852 ;0136 0208 8E 00 A0 STX OREGB ; AND RELEASE INT MODEM RESET
4853 025446 216 000 240 .BYTE 216,000,240
4854 ;0137 0208 A2 00 LDX #0 ;INIT TBMT TIME OUT COUNTER
4855 025451 242 000 .BYTE 242,000
4856 ;0138 0200 2C 00 A4 BIT USTATR ;SEE IF TBMT SET
4857 025453 054 000 244 .BYTE 054,000,244
4858 ;0139 0210 70 08 BVS *.10 ;BR IF TBMT SET
4859 025456 160 010 .BYTE 160,010
4860 ;0140 0212 E8 INX ;INCREMENT TIME-OUT COUNTER
4861 025460 350 .BYTE 350
4862 ;0141 0213 D0 F8 BNE *.6 ;BR IF NO TIME OUT
4863 025461 320 370 .BYTE 320,370
4864 ;0142 0215 ; *** ERROR 1 ***
4865 ;0143 0215 A0 01 LDY #1 ;SET CODE FOR TBMT TIME-OUT ERROR
4866 025463 240 001 .BYTE 240,001
4867 ;0144 0217 4C 90 03 JMP A100 ;GO TAKE ERROR EXIT
4868 025465 114 220 003 .BYTE 114,220,003
4869 ;0145 021A ;LOAD FIRST SYNCH CHAR INTO TRANSMITTER
4870 ;0146 021A A2 96 LDX #SYNCH ;LOAD FIRST SYNCH CHAR
4871 025470 242 226 .BYTE 242,226
4872 ;0147 021C 8E 02 A1 STX TXDB
4873 025472 216 002 241 .BYTE 216,002,241
4874 ;0148 021F A2 00 LDX #0 ;INIT TBMT TIME OUT COUNTER
4875 025475 242 000 .BYTE 242,000
4876 ;0149 0221 2C 00 A4 BIT USTATR ;SEE IF TBMT SET
4877 025477 054 000 244 .BYTE 054,000,244
4878 ;0150 0224 70 08 BVS *.10 ;BR IF TBMT SET
4879 025502 160 010 .BYTE 160,010
4880 ;0151 0226 E8 INX ;INCREMENT TIME OUT COUNTER

```

;USYRT STATUS REGISTER EQUATES

```

USTATR = $A400
RDA = BIT7
TBMT = BIT6
RXACT = BIT5
RSA = BIT4
TSO = BIT3
TXACT = BIT2
TXUERR = BIT1
SYNFLG = BIT0

```

;MISCELLANEOUS EQUATES

```

SYNCH = @226

```

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

4881 025504      350      .BYTE      350
4882              ;0152 0227 D0 F8      BNE      *-6      ;BR IF NO TIME-OUT
4883 025505      320      370      .BYTE      320,370
4884              ;0153 0229      ; *** ERROR 2 ***
4885              ;0154 0229 A0 02      LDY      #2      ;SET CODE FOR TBMT TIME OUT ERROR
4886 025507      240      002      .BYTE      240,002
4887              ;0155 022B 4C 90 03      JMP      A100      ;GO TAKE ERROR EXIT
4888 025511      114      220      003      .BYTE      114,220,003
4889              ;0156 022E      ;LOAD SECOND SYNCH CHAR INTO TRANSMITTER
4890              ;0157 022E A2 96      LDX      #SYNCH      ;LOAD SECOND SYNCH CHAR
4891 025514      242      226      .BYTE      242,226
4892              ;0158 0230 8E 02 A1      STX      TXDB
4893 025516      216      002      241      .BYTE      216,002,241
4894              ;0159 0233 A2 00      LDX      #0      ;INIT TBMT TIME OUT COUNTER
4895 025521      242      000      .BYTE      242,000
4896              ;0160 0235 2C 00 A4      BIT      USTATR      ;SEE IF TBMT SET
4897 025523      054      000      244      .BYTE      054,000,244
4898              ;0161 0238 70 08      BVS      *-10      ;BR IF TBMT SET
4899 025526      160      010      .BYTE      160,010
4900              ;0162 023A E8      INX
4901 025530      350      .BYTE      350
4902              ;0163 023B D0 F8      BNE      *-6      ;BR IF NO TIME OUT
4903 025531      320      370      .BYTE      320,370
4904              ;0164 023D      ; *** ERROR 3 ***
4905              ;0165 023D A0 03      LDY      #3      ;SET CODE FOR TBMT TIME OUT ERROR
4906 025533      240      003      .BYTE      240,003
4907              ;0166 023F 4C 90 03      JMP      A100      ;GO TAKE ERROR EXIT
4908 025535      114      220      003      .BYTE      114,220,003
4909              ;0167 0242      ;CHECK FOR CARRIER SET
4910              ;0168 0242 2C 01 A0      BIT      OREGA      ;SEE IF CARRIER SET YET
4911 025540      054      001      240      .BYTE      054,001,240
4912              ;0169 0245 70 05      BVS      *-7      ;BR IF CARRIER SET
4913 025543      160      005      .BYTE      160,005
4914              ;0170 0247      ; *** ERROR 4 ***
4915              ;0171 0247 A0 04      LDY      #4      ;SET CODE FOR CARRIER NOT SET ERROR
4916 025545      240      004      .BYTE      240,004
4917              ;0172 0249 4C 90 03      JMP      A100      ;GO TAKE ERROR EXIT
4918 025547      114      220      003      .BYTE      114,220,003
4919              ;0173 024C      ;LOAD TRANSMITTER WITH 000 CHAR
4920              ;0174 024C A2 00      LDX      #000      ;CLEAR TSOM
4921 025552      242      000      .BYTE      242,000
4922              ;0175 024E 8E 03 A1      STX      TDSR
4923 025554      216      003      241      .BYTE      216,003,241
4924              ;0176 0251 8E 02 A1      STX      TXDB      ;LOAD 000 CHAR
4925 025557      216      002      241      .BYTE      216,002,241
4926              ;0177 0254 A2 00      LDX      #0      ;INIT TBMT TIME OUT COUNTER
4927 025562      242      000      .BYTE      242,000
4928              ;0178 0256 2C 00 A4      BIT      USTATR      ;SEE IF TBMT SET
4929 025564      054      000      244      .BYTE      054,000,244
4930              ;0179 0259 70 08      BVS      *-10      ;BR IF TBMT SET
4931 025567      160      010      .BYTE      160,010
4932              ;0180 025B E8      INX      ;INCREMENT TIME OUT COUNTER
4933 025571      350      .BYTE      350
4934              ;0181 025C D0 F8      BNE      * 6      ;BR IF NO TIME OUT
4935 025572      320      370      .BYTE      320,370
4936              ;0182 025E      ; *** ERROR 5 ***
4937              ;0183 025E A0 05      LDY      #5      ;SET CODE FOR TBMT TIME OUT ERROR

```

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

4938 025574 240 005 .BYTE 240,005
4939 ;0184 0260 4C 90 03 JMP A100 ;GO TAKE ERROR EXIT
4940 025576 114 220 007 .BYTE 114,220,003 ;LOAD TRANSMITTER WITH 125 CHAR
4941 ;0185 0263 ;LOAD TRANSMITTER WITH 125 CHAR
4942 ;0186 0263 A2 55 LDX #0125 ;LOAD 125 CHAR
4943 025601 242 125 .BYTE 242,125
4944 ;0187 0265 8E 02 A1 STX TXDB
4945 025603 216 002 241 .BYTE 216,002,241
4946 ;0188 0268 A2 00 LDX #0 ;INIT TBMT TIME OUT COUNTER
4947 025606 242 000 .BYTE 242,000
4948 ;0189 026A 2C 00 A4 BIT USTATR ;SEE IF TBMT SET
4949 025610 054 000 244 .BYTE 054,000,244
4950 ;0190 026D 70 08 BVS *-10 ;BR IF TBMT SET
4951 025613 160 010 .BYTE 160,010
4952 ;0191 026F E8 INX ;INCREMENT TIME-OUT COUNTER
4953 025615 350 .BYTE 350
4954 ;0192 0270 D0 F8 BNE * 6 ;BR IF NO TIME-OUT
4955 025616 320 370 .BYTE 320,370
4956 ;0193 0272 ; *** ERROR 6 ***
4957 ;0194 0272 A0 06 LDY #6 ;SET CODE FOR TBMT TIME OUT ERROR
4958 025620 240 006 .BYTE 240,006
4959 ;0195 0274 4C 90 03 JMP A100 ;GO TAKE ERROR EXIT
4960 025622 114 220 003 .BYTE 114,220,003 ;LOAD TRANSMITTER WITH 252 CHAR
4961 ;0196 0277 ;LOAD TRANSMITTER WITH 252 CHAR
4962 ;0197 0277 A2 AA LDX #0252 ;LOAD 252 CHAR
4963 025625 242 252 .BYTE 242,252
4964 ;0198 0279 8E 02 A1 STX TXDB
4965 025627 216 002 241 .BYTE 216,002,241
4966 ;0199 027C A2 00 LDX #0 ;INIT TBMT TIME OUT COUNTER
4967 025632 242 000 .BYTE 242,000
4968 ;0200 027E 2C 00 A4 BIT USTATR ;SEE IF TBMT SET
4969 025634 054 000 244 .BYTE 054,000,244
4970 ;0201 0281 70 08 BVS *-10 ;BR IF TBMT SET
4971 025637 160 010 .BYTE 160,010
4972 ;0202 0283 E8 INX ;INCREMENT TIME-OUT COUNTER
4973 025641 350 .BYTE 350
4974 ;0203 0284 D0 F8 BNE * 6 ;BR IF NO TIME OUT
4975 025642 320 370 .BYTE 320,370
4976 ;0204 0286 ; *** ERROR 7 ***
4977 ;0205 0286 A0 07 LDY #7 ;SET CODE FOR TBMT TIME-OUT ERROR
4978 025644 240 007 .BYTE 240,007
4979 ;0206 0288 4C 90 03 JMP A100 ;GO TAKE ERROR EXIT
4980 025646 114 220 003 .BYTE 114,220,003 ;LOAD TRANSMITTER WITH 377 CHAR AND END OF MESSAGE
4981 ;0207 028B ;LOAD TRANSMITTER WITH 377 CHAR AND END OF MESSAGE
4982 ;0208 028B A2 FF LDX #0377 ;LOAD 377 CHAR
4983 025651 242 377 .BYTE 242,377
4984 ;0209 028D 8E 02 A1 STX TXDB
4985 025653 216 002 241 .BYTE 216,002,241
4986 ;0210 0290 A2 00 LDX #0 ;INIT TBMT TIME OUT COUNTER
4987 025656 242 000 .BYTE 242,000
4988 ;0211 0292 2C 00 A4 BIT USTATR ;SEE IF TBMT SET
4989 025660 054 000 244 .BYTE 054,000,244
4990 ;0212 0295 70 08 BVS *-10 ;BR IF TBMT SET
4991 025663 160 010 .BYTE 160,010
4992 ;0213 0297 E8 INX ;INCREMENT TIME OUT COUNTER
4993 025665 350 .BYTE 350
4994 ;0214 0298 D0 F8 BNE * 6 ;BR IF NO TIME OUT

```

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

4995 025666      320      370
4996             ;0215 .BYTE      320,370
4997             ;0216 029A A0 08      ; *** LRROR 8 ***
4998 025670      240      010             LDY      #8             ;SET CODE FOR TBMT TIME-OUT ERROR
4999             .BYTE      240,010
5000 025672      114      220      003     ;0217 029C 4C 90 03             JMP      A100             ;GO TAKE ERROR EXIT
5001             .BYTE      114,220,003
5002             ;0218 029F             ;LOAD TRANSMITTER WITH 000 CHAR
5003 025675      242      000             ;0219 029F A2 00             LDX      #000             ;LOAD 000 CHAR
5004             .BYTE      242,000
5005 025677      216      002      241     ;0220 02A1 8E 02 A1             STX      TXDB
5006             .BYTE      216,002,241
5007 025702      242      000             ;0221 02A4 A2 00             LDX      #0             ;INIT RDA TIME OUT COUNTER
5008             .BYTE      242,000
5009 025704      054      000      244     ;0222 02A6 2C 00 A4             BIT      USTATR             ;SEE IF RDA SET
5010             .BYTE      054,000,244
5011 025707      060      010             ;0223 02A9 30 08             BMI      *-10             ;BR IF RDA SET
5012             .BYTE      060,010
5013 025711      350
5014             ;0224 02AB E8             INX
5015 025712      320      370             ;0225 02AC D0 F8             BNE      *-6             ;BR IF NO TIME OUT
5016             .BYTE      320,370
5017             ;0226 02AE             ; *** ERROR 9 ***
5018 025714      240      011             ;0227 02AE A0 09             LDY      #9             ;SET CODE FOR RDA TIME OUT ERROR
5019             .BYTE      240,011
5020 025716      114      220      003     ;0228 02B0 4C 90 03             JMP      A100             ;GO TAKE ERROR EXIT
5021             .BYTE      114,220,003
5022             ;0229 02B3             ;READ AND CHECK 000 CHAR
5023 025721      255      000      241     ;0230 02B3 AD 00 A1             LDA      RXDB             ;READ RECEIVER BUFFER
5024             .BYTE      255,000,241
5025 025724      311      000             ;0231 02B6 C9 00             CMP      #0000             ;CHK FOR 000
5026             .BYTE      311,000
5027 025726      360      005             ;0232 02B8 F0 05             BEQ      *-7             ;BR IF 000
5028             .BYTE      360,005
5029             ;0233 02BA             ; *** ERROR 10 ***
5030 025730      240      012             ;0234 02BA A0 0A             LDY      #10             ;SET CODE FOR DATA MISCOMPARE ERROR
5031             .BYTE      240,012
5032 025732      114      220      003     ;0235 02BC 4C 90 03             JMP      A100             ;GO TAKE ERROR EXIT
5033             .BYTE      114,220,003
5034 025735      242      000             ;0236 02BF A2 00             LDX      #0             ;INIT RDA TIME OUT COUNTER
5035             .BYTE      242,000
5036 025737      054      000      244     ;0237 02C1 2C 00 A4             BIT      USTATR             ;SEE IF RDA SET
5037             .BYTE      054,000,244
5038 025742      060      010             ;0238 02C4 30 08             BMI      *-10             ;BR IF RDA SET
5039             .BYTE      060,010
5040 025744      350
5041             ;0239 02C6 E8             INX
5042 025745      320      370             ;0240 02C7 D0 F8             BNE      *-6             ;BR IF NO TIME OUT
5043             .BYTE      320,370
5044             ;0241 02C9             ; *** ERROR 11 ***
5045 025747      240      013             ;0242 02C9 A0 0B             LDY      #11             ;SET CODE FOR RDA TIME-OUT ERROR FOR
5046             .BYTE      240,013
5047 025751      114      220      003     ;0243 02CB 4C 90 03             JMP      A100             ;GO TAKE ERROR EXIT
5048             .BYTE      114,220,003
5049 025754      242      002             ;0244 02CE A2 02             LDX      #TEOM             ;SET TEOM TO TERMINATE MSG
5050             .BYTE      242,002
5051 025756      216      003      241     ;0245 02D0 8E 03 A1             STX      TD5R
5051             .BYTE      216,003,241

```

TEST 2 INTEGRAL MODEM INTERFACE TEST

5052					;0246 02D3	;READ AND CHECK 125 CHAR	
5053					;0247 02D3 AD 00 A1	LDA RXDB	;READ RECEIVER BUFFER
5054	025761	255	000	241	.BYTE 255,000,241		
5055					;0248 02D6 C9 55	CMP #0125	;CHK FOR 125
5056	025764	311	125		.BYTE 311,125		
5057					;0249 02D8 F0 05	BEQ *-7	;BR IF 125
5058	025766	360	005		.BYTE 360,005		
5059					;0250 02DA	; *** ERROR 12 ***	
5060					;0251 02DA A0 0C	LDY #12	;SET CODE FOR DATA MISCOMPARE ERROR
5061	025770	240	014		.BYTE 240,014		
5062					;0252 02DC 4C 90 03	JMP A100	;GO TAKE ERROR EXIT
5063	025772	114	220	003	.BYTE 114,220,003		
5064					;0253 02DF A2 00	LDX #0	;INIT RDA TIME-OUT COUNTER
5065	025775	242	000		.BYTE 242,000		
5066					;0254 02E1 2C 00 A4	BIT USTATR	;SEE IF RDA SET
5067	025777	054	000	244	.BYTE 054,000,244		
5068					;0255 02E4 30 08	BMI *-10	;BR IF RDA SET
5069	026002	060	010		.BYTE 060,010		
5070					;0256 02E6 E8	INX	;INCREMENT TIME-OUT COUNTER
5071	026004	350			.BYTE 350		
5072					;0257 02E7 D0 F8	BNE *-6	;BR IF NO TIME OUT
5073	026005	320	370		.BYTE 320,370		
5074					;0258 02E9	; *** ERROR 13 ***	
5075					;0259 02E9 A0 0D	LDY #13	;SET CODE FOR RDA TIME OUT ERROR
5076	026007	240	015		.BYTE 240,015		
5077					;0260 02EB 4C 90 03	JMP A100	;GO TAKE ERROR EXIT
5078	026011	114	220	003	.BYTE 114,220,003		
5079					;0261 02EE	;READ AND CHECK 252 CHAR	
5080					;0262 02EE AD 00 A1	LDA RXDB	;READ RECEIVER BUFFER
5081	026014	255	000	241	.BYTE 255,000,241		
5082					;0263 02F1 C9 AA	CMP #0252	;CHK FOR 252
5083	026017	311	252		.BYTE 311,252		
5084					;0264 02F3 F0 05	BEQ *-7	;BR IF 252
5085	026021	360	005		.BYTE 360,005		
5086					;0265 02F5	; *** ERROR 14 ***	
5087					;0266 02F5 A0 0E	LDY #14	;SET CODE FOR DATA MISCOMPARE ERROR
5088	026023	240	016		.BYTE 240,016		
5089					;0267 02F7 4C 90 03	JMP A100	;GO TAKE ERROR EXIT
5090	026025	114	220	003	.BYTE 114,220,003		
5091					;0268 02FA A2 00	LDX #0	;INIT RDA TIME-OUT COUNTER
5092	026030	242	000		.BYTE 242,000		
5093					;0269 02FC 2C 00 A4	BIT USTATR	;SEE IF RDA SET
5094	026032	054	000	244	.BYTE 054,000,244		
5095					;0270 02FF 30 08	BMI *-10	;BR IF RDA SET
5096	026035	060	010		.BYTE 060,010		
5097					;0271 0301 E8	INX	;INCREMENT TIME-OUT COUNTER
5098	026037	350			.BYTE 350		
5099					;0272 0302 D0 F8	BNE *-6	;BR IF NO TIME-OUT
5100	026040	320	370		.BYTE 320,370		
5101					;0273 0304	; *** ERROR 15 ***	
5102					;0274 0304 A0 0F	LDI #15	;SET CODE FOR RDA TIME-OUT ERROR FOR
5103	026042	240	017		.BYTE 240,017		
5104					;0275 0306 4C 90 03	JMP A100	;GO TAKE ERROR EXIT
5105	026044	114	220	003	.BYTE 114,220,003		
5106					;0276 0309	;READ AND CHECK 377 CHAR	
5107					;0277 0309 AD 00 A1	LDA RXDB	;READ RECEIVER BUFFER
5108	026047	255	000	241	.BYTE 255,000,241		

TEST 2 INTEGRAL MODEM INTERFACE TEST

5109					:0278	030C C9 FF		CMP	#0377		;CHK FOR 377
5110	026052	311	377			.BYTE 311,377					
5111					:0279	030E F0 05		BEQ	*+7		;BR IF 377
5112	026054	360	005			.BYTE 360,005					
5113					:0280	0310			*** ERROR 16 ***		
5114					:0281	0310 A0 10		LDY	#16		;SET CODE FOR DATA MISCOMPARE ERROR
5115	026056	240	020			.BYTE 240,020					
5116					:0282	0312 4C 90 03		JMP	A100		;GO TAKE ERROR EXIT
5117	026060	114	220	003		.BYTE 114,220,003					
5118					:0283	0315 A2 00		LDX	#0		;INIT RDA TIME-OUT COUNTER
5119	026063	242	000			.BYTE 242,000					
5120					:0284	0317 2C 00 A4		BIT	USTATR		;SEE IF RDA SET
5121	026065	054	000	244		.BYTE 054,000,244					
5122					:0285	031A 30 08		BMI	*+10		;BR IF RDA SET
5123	026070	060	010			.BYTE 060,010					
5124					:0286	031C E8		INX			;INCREMENT TIME-OUT COUNTER
5125	026072	350				.BYTE 350					
5126					:0287	031D D0 F8		BNE	*-6		;BR IF NO TIME OUT
5127	026073	320	370			.BYTE 320,370					
5128					:0288	031F			*** ERROR 17 ***		
5129					:0289	031F A0 11		LDY	#17		;SET CODE FOR RDA TIME-OUT ERROR
5130	026075	240	021			.BYTE 240,021					
5131					:0290	0321 4C 90 03		JMP	A100		;GO TAKE ERROR EXIT
5132	026077	114	220	003		.BYTE 114,220,003					
5133					:0291	0324			;READ AND CHECK 000 CHAR		
5134					:0292	0324 AD 00 A1		LDA	RXDB		;READ RECEIVER BUFFER
5135	026102	255	000	241		.BYTE 255,000,241					
5136					:0293	0327 C9 00		CMP	#0000		;CHK FOR 000
5137	026105	311	000			.BYTE 311,000					
5138					:0294	0329 F0 05		BEQ	*+7		;BR IF 000
5139	026107	360	005			.BYTE 360,005					
5140					:0295	032B			*** ERROR 18 ***		
5141					:0296	032B A0 12		LDY	#18		;SET CODE FOR DATA MISCOMPARE ERROR
5142	026111	240	022			.BYTE 240,022					
5143					:0297	032D 4C 90 03		JMP	A100		;GO TAKE ERROR EXIT
5144	026113	114	220	003		.BYTE 114,220,003					
5145					:0298	0330 AE 01 A1		LDX	RDSR		;CHECK FOR RERR BIT SET
5146	026116	256	001	241		.BYTE 256,001,241					
5147					:0299	0333 30 05		BMI	*+7		;BR IF RERR BIT SET (NO CRC ERROR)
5148	026121	060	005			.BYTE 060,005					
5149					:0300	0335			*** ERROR 19 ***		
5150					:0301	0335 A0 13		LDY	#19		
5151	026123	240	023			.BYTE 240,023					
5152					:0302	0337 4C 90 03		JMP	A100		;GO TAKE ERROR EXIT
5153	026125	114	220	003		.BYTE 114,220,003					
5154					:0303	033A A2 00		LDX	#0		;INIT RDA TIME-OUT COUNTER
5155	026130	242	000			.BYTE 242,000					
5156					:0304	033C 2C 00 A4		BIT	USTATR		;SEE IF RDA SET
5157	026132	054	000	244		.BYTE 054,000,244					
5158					:0305	033F 30 08		BMI	*+10		;BR IF RDA SET
5159	026135	060	010			.BYTE 060,010					
5160					:0306	0341 E8		INX			;INCREMENT TIME-OUT COUNTER
5161	026137	350				.BYTE 350					
5162					:0307	0342 D0 F8		BNE	*-6		;BR IF NO TIME-OUT
5163	026140	320	370			.BYTE 320,370					
5164					:0308	0344			*** ERROR 20 ***		
5165					:0309	0344 A0 14		LDY	#20		;SET CODE FOR RDA TIME OUT ERROR

TEST 2 - INTEGRAL MODEM INTERFACE TEST

5166	026142	240	024			.BYTE 240,024			
5167									
5168	026144	114	220	003	:0310	0346 4C 90 03	JMP	A100	;GO TAKE ERROR EXIT
5169						.BYTE 114,220,003			
5170					:0311	0349			;READ AND CHECK FIRST CRC CHAR (160)
5171	026147	255	000	241	:0312	0349 AD 00 A1	LDA	RXDB	;READ RECEIVER BUFFER
5172						.BYTE 255,000,241			
5173	026152	311	160		:0313	034C C9 70	CMP	#B160	;CHK FOR 160
5174						.BYTE 311,160			
5175	026154	360	005		:0314	034E F0 05	BFG	*.7	;BR IF 160
5176						.BYTE 360,005			
5177					:0315	0350			; *** ERROR 21 ***
5178	026156	240	025		:0316	0350 A0 15	LDY	#21	;SE CODE FOR DATA MISCOMPARE ERROR
5179						.BYTE 240,025			
5180	026160	114	220	003	:0317	0352 4C 90 03	JMP	A100	;GO TAKE ERROR EXIT
5181						.BYTE 114,220,003			
5182	026163	242	000		:0318	0355 A2 00	LDX	#0	;INIT RDA TIME-OUT COUNTER
5183						.BYTE 242,000			
5184	026165	054	000	244	:0319	0357 2C 00 A4	BIT	USTATR	;SEE IF RDA SET
5185						.BYTE 054,000,244			
5186	026170	060	010		:0320	035A 30 08	BMI	*.10	;BR IF RDA SET
5187						.BYTE 060,010			
5188	026172	350			:0321	035C E8	INX		;INCREMENT TIME-OUT COUNTER
5189						.BYTE 350			
5190	026173	320	370		:0322	035D D0 F8	BNE	*.6	;BR IF NO TIME-OUT
5191						.BYTE 320,370			
5192					:0323	035F			; *** ERROR 22 ***
5193	026175	240	026		:0324	035F A0 16	LDY	#22	;SET CODE FOR RDA TIME OUT ERROR
5194						.BYTE 240,026			
5195	026177	114	220	003	:0325	0361 4C 90 03	JMP	A100	;GO TAKE ERROR EXIT
5196						.BYTE 114,220,003			
5197					:0326	0364			;READ AND CHECK 2ND CRC CHAR (034)
5198	026202	255	000	241	:0327	0364 AD 00 A1	LDA	RXDB	;READ RECEIVER BUFFER
5199						.BYTE 255,000,241			
5200	026205	311	034		:0328	0367 C9 1C	CMP	#B034	;CHK FOR 034
5201						.BYTE 311,034			
5202	026207	360	005		:0329	0369 F0 05	BEQ	*.7	;BR IF 034
5203						.BYTE 360,005			
5204					:0330	036B			; *** ERROR 23 ***
5205	026211	240	027		:0331	036B A0 17	LDY	#23	;SET CODE FOR DATA MISCOMPARE ERROR
5206						.BYTE 240,027			
5207	026213	114	220	003	:0332	036D 4C 90 03	JMP	A100	;GO TAKE ERROR EXIT
5208						.BYTE 114,220,003			
5209	026216	242	000		:0333	0370 A2 00	LDX	#0	;INIT RDA TIME-OUT COUNTER
5210						.BYTE 242,000			
5211	026220	054	000	244	:0334	0372 2C 00 A4	BIT	USTATR	;SEE IF RDA SET
5212						.BYTE 054,000,244			
5213	026223	060	010		:0335	0375 30 08	BMI	*.10	;BR IF RDA SET
5214						.BYTE 060,010			
5215	026225	350			:0336	0377 E8	INX		;INCREMENT TIME-OUT COUNTER
5216						.BYTE 350			
5217	026226	320	370		:0337	0378 D0 F8	BNE	*.6	;BR IF NO TIME-OUT
5218						.BYTE 320,370			
5219					:0338	037A			; *** ERROR 24 ***
5220	026230	240	030		:0339	037A A0 18	LDY	#24	;SET CODE FOR RDA TIME OUT ERROR
5221						.BYTE 240,030			
5222	026232	114	220	003	:0340	037C 4C 90 03	JMP	A100	;GO TAKE ERROR EXIT
						.BYTE 114,220,003			

TEST 2 - INTEGRAL MODEM INTERFACE TEST

```

5223 ;0341 037F ;DROP RTS, CHECK FOR CARRIER TO DROP
5224 ;0342 037F A2 68 LDX #TXEN!RXEN!RTSND ;DE ASSERT RTS
5225 026235 242 150 .BYTE 242,150
5226 ;0343 0381 8E 00 A0 STX OREGB
5227 026237 216 000 240 .BYTE 216,000,240
5228 ;0344 0384 A2 00 LDX #0 ;INIT CARRIER DROP TIME-OUT COUNTER
5229 026242 242 000 .BYTE 242,000
5230 ;0345 0386 2C 01 A0 BIT OREGA ;SEE IF CARRIER CLEARED
5231 026244 054 001 240 .BYTE 054,001,240
5232 ;0346 0389 50 05 BVC *.7 ;BR IF CARRIER CLEARED
5233 026247 120 005 .BYTE 120,005
5234 ;0347 038B E8 INX ;INCREMENT TIME-OUT COUNTER
5235 026251 350 .BYTE 350
5236 ;0348 036C D0 F8 BNE *.6 ;BR IF NO TIME OUT
5237 026252 320 370 .BYTE 320,370
5238 ;0349 038E ; *** ERROR 25 ***
5239 ;0350 038E A0 19 LDY #25 ;SET CODE FOR CARRIER DROP TIME OUT
5240 026254 240 031 .BYTE 240,031
5241 ;0351 0390 ;COME HERE FOR EXIT
5242 ;0352 0390 84 16 A100 STY BSEL6 ;PUT ERROR NO. (IF ANY) INTO BSEL6
5243 026256 204 026 .BYTE 204,026
5244 ;0353 0392 85 17 STA BSEL7 ;PUT BAD DATA (IF ANY) INTO BSEL7
5245 026260 205 027 .BYTE 205,027
5246 ;0354 0394 60 RTS ;RETURN CONTROL TO LSI-11 PROGRAM
5247 026262 140 .BYTE 140
5248 ;0355 0395
5249 ;0356 0395
5250 ;
5251 ;
5252 ;ERRORS = 0000
5253 ;
5254 026263 ENDCOD:
5255 .EVEN

```

TEST 3 DATA TEST BCP XLB CRC 16

5273

.SBTTL TEST 3 -- DATA TEST -- BCP XLB CRC 16

```

*****
;*
;* TEST 3 -- DATA TEST -- BCP XLB CRC-16
;*
;* IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
;* RECEIVE IN BCP MODE WITH CRC-16 ERROR DETECTION THE FOLLOWING
;* MESSAGE:
;*
;* 125 252 000 377 001 002 004 010 020 040 100 200 376 375 373 367
;* 357 337 277 177
;*
;* THIS MESSAGE WILL BE PRECEDED BY 3 SYNC CHARACTERS AND REPEATED
;* THREE TIMES WITH CRC'S FOLLOWING EACH ONE. THE LAST TRANSMISSION OF
;* THE CRC WILL BE FOLLOWED BY SEVERAL SYNC CHARACTERS BEFORE DROPPING
;* TXE & RXE. 8-BIT CHARACTER LENGTHS ARE ALSO UTILIZED.
;*
;* IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
;* USING INTERNAL LOOPBACK (TTLOOP=1).
*****

```

```

026264
5274 026264 012737 000003 002526
5275 026272 004737 005376
5276 026276 042737 001000 026354
5277 026304 012737 000011 027126
5278 026312 023727 002470 000000
5279 026320 001412
5280 026322 023727 002472 000004
5281 026330 001406
5282 026332 052737 001000 026354
5283 026340 052737 100000 027126
5284
5285 026346 004537 007234
5286 026352 065626
5287 026354 000000
5288 026356 103003
5289 026360
026360 104460
5290 026362
026362 104410
026364 000554
5291
5292 026366 004537 010136
5293 026372 000226
5294 026374 000007
5295 026376 103003
5296 026400
026400 104460
5297 026402
026402 104410
026404 000534
5298
5299 026406 004537 010136

```

```

: BGNTST
:
: T3::
:
: MOV #3,REGO ;INIT COUNT (TEXT TRANSMITTED 3 TIMES,
: JSR PC,INIDMV ;INIT DMV-11, ENTER MAINT LOOP
: BIC #NOLOOP,1$
: MOV #9,3$ ;*INIT ENTRAN COUNT/TTLOOP STATUS
: CMP BRDTYP,#0 ;IS THIS AN M8064?
: BEQ 2$ ; YES: USE TTLOOP (NOT XLB).
: CMP TSTCON,#4 ;IS A LOOPBACK CONNECTOR/CABLE SPECIFIED ?
: BEQ 2$ ;BR IF NO
: BIS #NOLOOP,1$ ; YES: SPECIFY NO TTLOOP (INITRN)
: BIS #BIT15,3$ ; AND SET MSB OF ENTRAN STATUS (NOLOOP)
:
: 2$: JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
: DDCMP!STRIPS!IDLES!CRC16 ;SYNCH ;SET DDCMP, STRIP, IDLE, CRC 16. SYNCH=226
: 1$: 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 L10022 .
:
: JSR R5, TXCHAR ;LOAD 2ND SYNCH, TX 1ST SYNCH
: SYNCH
: 7.
: BCC ..8. ;BR IF NO ERROR
: ERROR ;REPORT STACKED ERROR
:
: ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
:
: TRAP C$ESCAPE
: .WORD L10022 .
:
: JSR R5, TXCHAR ;LOAD 3RD SYNCH, TX 2ND SYNCH

```

TEST 3 DATA TEST BCP XLB CRC 16

5300	026412	000226		SYNCH				
5301	026414	000010		8.				
5302	026416	103003		BCC	..8.	;BR IF NO ERROR		
5303	026420			ERROR		;REPORT STACKED ERROR		
	026420	104460					TRAP	C\$ERROR
5304	026422			ESCAPE	TST	;SKIP TO END OF TEST		
	026422	104410					TRAP	C\$ESCAPE
	026424	000514					.WORD	L10022-
5305								
5306	026426	004537	010250	JSR	R5,TXCTRL	;CLEAR TSOM		
5307	026432	000000		000				
5308	026434	000000		0				
5309								
5310	026436	004537	010136	JSR	R5,TXCHAR	;LOAD 125(DATA1), TX 3RD SYNCH		
5311	026442	000125		125				
5312	026444	000010		8.				
5313	026446	103003		BCC	..8.	;BR IF NO ERROR		
5314	026450			ERROR		;REPORT STACKED ERROR		
	026450	104460					TRAP	C\$ERROR
5315	026452			ESCAPE	TST	;SKIP TO END OF TEST		
	026452	104410					TRAP	C\$ESCAPE
	026454	000464					.WORD	L10022-
5316								
5317	026456	004537	010136	JSR	R5,TXCHAR	;LOAD 252(DATA2), TX 125(DATA1)		
5318	026462	000252		252				
5319	026464	000010		8.				
5320	026466	103003		BCC	..8.	;BR IF NO ERROR		
5321	026470			ERROR		;REPORT STACKED ERROR		
	026470	104460					TRAP	C\$ERROR
5322	026472			ESCAPE	TST	;SKIP TO END OF TEST		
	026472	104410					TRAP	C\$ESCAPE
	026474	000444					.WORD	L10022-
5323								
5324	026476	004537	010136	JSR	R5,TXCHAR	;LOAD 000(DATA3), TX 252(DATA2)		
5325	026502	000000		000				
5326	026504	000010		8.				
5327	026506	103003		BCC	..8.	;BR IF NO ERROR		
5328	026510			ERROR		;REPORT STACKED ERROR		
	026510	104460					TRAP	C\$ERROR
5329	026512			ESCAPE	TST	;SKIP TO END OF TEST		
	026512	104410					TRAP	C\$ESCAPE
	026514	000424					.WORD	L10022-
5330								
5331	026516	004537	010136	JSR	R5,TXCHAR	;LOAD 377(DATA4), TX 000(DATA3)		
5332	026522	000377		377				
5333	026524	000010		8.				
5334	026526	103003		BCC	..8.	;BR IF NO ERROR		
5335	026530			ERROR		;REPORT STACKED ERROR		
	026530	104460					TRAP	C\$ERROR
5336	026532			ESCAPE	TST	;SKIP TO END OF TEST		
	026532	104410					TRAP	C\$ESCAPE
	026534	000404					.WORD	L10022-
5337								
5338	026536	004537	010136	JSR	R5,TXCHAR	;LOAD 001(DATA5)		
5339	026542	000001		001				
5340	026544	000000		0				
5341	026546	103003		BCC	..8.	;BR IF NO ERROR		

TEST 3 DATA TEST -- BCP XLB CRC 16

```

5342 026550          ERROR          ;REPORT STACKED ERROR
      026550 104460
5343 026552          ESCAPE TST      ;SKIP TO END OF TEST          TRAP   C$ERROR
      026552 104410
      026554 000364          .WORD   L10022 .
5344
5345 026556 004537 011624 JSR      R5,RCV1ST      ;CLOCK AND RCV 125
5346 026562 000000          0
5347 026564 103003          BCC     .+8.          ;BR IF NO ERROR
5348 026566          ERROR          ;REPORT STACKED ERROR          TRAP   C$ERROR
      026566 104460
5349 026570          ESCAPE TST      ;SKIP TO END OF TEST          TRAP   C$ERROR
      026570 104410
      026572 000346          .WORD   L10022 .
5350
5351 026574 004537 010350 JSR      R5,RXCHAR      ;READ & CHK 125(DATA1), RCV 252(DATA2)
5352 026600          125
5353 026602 000000          0
5354 026604 000010          8.
5355 026606 103003          BCC     .+8.          ;BR IF NO ERROR
5356 026610          ERROR          ;REPORT STACKED ERROR          TRAP   C$ERROR
      026610 104460
5357 026612          ESCAPE TST      ;SKIP TO END OF TEST          TRAP   C$ERROR
      026612 104410
      026614 000324          .WORD   L10022 .
5358
5359          ;-----
5360          ; TRANSMIT THE BULK OF DATA OUT OF TABLE "PATX"
5361 026616 012702 002646          MOV     @PATX+1,R2      ;SET UP TABLE POINTER
5362 026622 112237 026660          5$:   MOVB   (R2)+,20$      ;SET UP EXPECTED CHARACTER
5363 026626 116237 000003 026640          MOVB   3(R2),10$      ;SET UP TRANSMIT CHARACTER
5364
5365 026634 004537 010136          JSR      R5,TXCHAR      ;LOAD A CHARACTER
5366 026640          000          10$:   ;** HOLE FOR NEXT TX CHARACTER
5367 026642 000000          0
5368 026644 103003          BCC     .+8.          ;BR IF NO ERROR
5369 026646          ERROR          ;REPORT STACKED ERROR          TRAP   C$ERROR
      026646 104460
5370 026650          ESCAPE TST      ;SKIP TO END OF TEST          TRAP   C$ERROR
      026650 104410
      026652 000266          .WORD   L10022 .
5371
5372 026654 004537 010350          JSR      R5,RXCHAR      ;CLK/RECEIVE/CHECK PREVIOUS CHARACTER
5373 026660          000          20$:   ;** HOLE FOR EXPECTED CHARACTER
5374 026662 000000          0
5375 026664 000010          8.
5376 026666 103003          BCC     .+8.          ;BR IF NO ERROR
5377 026670          ERROR          ;REPORT STACKED ERROR          TRAP   C$ERROR
      026670 104460
5378 026672          ESCAPE TST      ;SKIP TO END OF TEST          TRAP   C$ERROR
      026672 104410
      026674 000244          .WORD   L10022 .
5379
5380 026676 022702 002665          CMP     @PATX+16.,R2    ;CHECK FOR 20TH CHARACTER OF TABLE
5381 026702 001347          BNE     5$            ;BR IF NOT DONE
5382
5383 026704 004537 010250          JSR      R5,TXCTRL      ;LOAD 1ST TEOM

```

TEST 3 DATA TEST BCP XLB CRC 16

5384	026710	000002		TEOM				
5385	026712	000000		0				
5386	026714	004537	010350	JSR	R5,RXCHAR	;READ/CHK 357(DATA17), RCV 337(DATA18)		
5387	026720	000357		357				
5388	026722	000000		0				
5389	026724	000010		8.				
5390	026726	103003		BCC	.+8.	;BR IF NO ERROR		
5391	026730			ERROR		;REPORT STACKED ERROR	TRAP	C\$ERROR
5392	026730	104460						
	026732			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	026732	104410					TRAP	C\$ESCAPE
	026734	000204					.WORD	L10022 .
5393								
5394	026736	004537	010250	JSR	R5,TXCTRL	;LOAD 2ND TEOM		
5395	026742	000002		TEOM				
5396	026744	000000		0				
5397	026746	004537	010350	JSR	R5,RXCHAR	;READ/CHK 337(DATA18), RCV 277(DATA19)		
5398	026752	000337		337				
5399	026754	000000		0				
5400	026756	000010		8.				
5401	026760	103003		BCC	.+8.	;BR IF NO ERROR		
5402	026762			ERROR		;REPORT STACKED ERROR	TRAP	C\$ERROR
	026762	104460						
5403	026764			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	026764	104410					TRAP	C\$ESCAPE
	026766	000152					.WORD	L10022-.
5404								
5405	026770	004537	010350	JSR	R5,RXCHAR	;READ/CHK 277(DATA19), RCV 177(DATA20)		
5406	026774	000277		277				
5407	026776	000000		0				
5408	027000	000010		8.				
5409	027002	103003		BCC	.+8.	;BR IF NO ERROR		
5410	027004			ERROR		;REPORT STACKED ERROR	TRAP	C\$ERROR
	027004	104460						
5411	027006			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	027006	104410					TRAP	C\$ESCAPE
	027010	000130					.WORD	L10022-.
5412								
5413	027012	004537	010350	JSR	R5,RXCHAR	;READ/CHK 177(DATA20), RCV FIRST CRC BYTE		
5414	027016	100177		RXERR!177				
5415	027020	000001		RERCHK				
5416	027022	000010		8.				
5417	027024	103003		BCC	.+8.	;BR IF NO ERROR		
5418	027026			ERROR		;REPORT STACKED ERROR	TRAP	C\$ERROR
	027026	104460						
5419	027030			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ESCAPE
	027030	104410					TRAP	C\$ESCAPE
	027032	000106					.WORD	L10022 .
5420								
5421	027034	004537	010350	JSR	R5,RXCHAR	;READ & CHK 1ST CRC BYTE, RCV SECOND CRC BYTE		
5422	027040	000156		156				
5423	027042	000000		0				
5424	027044	000010		8.				
5425	027046	103003		BCC	.+8.	;BR IF NO ERROR		
5426	027050			ERROR		;REPORT STACKED ERROR	TRAP	C\$ERROR
	027050	104460						
5427	027052			ESCAPE	TST	;SKIP TO END OF TEST		

TEST 3 DATA TEST BCP XLB CRC 16

```

027052 104410 TRAP C$ESCAPE
027054 000064 .WORD L10022-.
5428
5429 027056 004537 010350 JSR R5,RXCHAR ;READ & CHK 2ND CRC BYTE, RCV 1ST SYNCH
5430 027062 000236 236
5431 027064 000000 0
5432 027066 000010 8.
5433 027070 103003 BCC .+8. ;BR IF NO ERROR
5434 027072 104460 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
027072 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5435 027074 104410 .WORD L10022 .
027076 000042
5436
5437 027100 005337 002526 ;
5438 027104 001406 DEC REGO ;DECREMENT COUNT
5439 BEQ 40$ ;BR IF TRIPLE LOOP IS COMPLETED
5440 027106 004537 010250 JSR R5, TXCTRL ;CLEAR TEOM, SET TSOM
5441 027112 000001 TSOM
5442 027114 000001 1
5443 027116 000137 026346 JMP 2$ ;AND RUN TX/RX AGAIN
5444
5445 027122 004537 011772 40$: JSR R5,ENTRAN ;SHUT DOWN TRANSMITTER, RECEIVER
5446 027126 000011 3$: 9.
5447 027130 103003 BCC .+8. ;BR IF NO ERROR
5448 027132 104460 ERROR ;REPORT STACKED ERROR TRAP C$ERROR
027132 104460 ESCAPE TST ;SKIP TO END OF TEST TRAP C$ESCAPE
5449 027134 104410 .WORD L10022 .
027136 000002
5450 027140 ENDTST
027140 104401 L10022: TRAP C$ETST

```


TEST 4 DATA TEST BCP XLB ODD VRC

5468

.SRTTL TEST 4 - DATA TEST - BCP XLB ODD VRC

```

*****
;*
;* TEST 4 -- DATA TEST -- BCP XLB ODD VRC
;*
;* IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
;* RECEIVE IN BCP MODE WITH ODD VRC ERROR DETECTION THE FOLLOWING
;* MESSAGE:
;*
;* 125 252 000 377 001 002 004 010 020 040 100 200 376 375 373 367
;* 357 337 277 177
;*
;* THIS MESSAGE WILL BE PRECEDED BY 3 SYNC CHARACTERS AND REPEATED
;* THREE TIMES. AFTER THE LAST MESSAGE, SEVERAL SYNC CHARACTERS ARE
;* SENT BEFORE DROPPING TXE & RXE. 7-BIT CHARACTER LENGTHS ARE ALSO
;* UTILIZED.
;*
;* IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
;* USING INTERNAL LOOPBACK (TTLOOP=1).
;*
*****

```

```

027142
5469 027142 004737 005376          JSR    PC,INIDMV      ;INIT DMV-11. ENTER MAINT LOOP
5470 027146 042737 001000 027224    BIC    @NOLOOP,1$    ;
5471 027154 012737 000011 027670    MOV    @9.,3$        ;INIT ENTRAN COUNT/TTLOOP STATUS
5472 027162 023727 002470 000000    CMP    BRDTYP,@0     ;IS THIS AN M8064?
5473 027170 001412                    BEQ    2$             ; YES: USE TTLOOP (NOT XLB).
5474 027172 023727 002472 000004    CMP    TSTCON,@4     ;IS A LOOPBACK CONNECTOR/CABLE SPECIFIED ?
5475 027200 001406                    BEQ    2$             ;BR IF NO
5476 027202 052737 001000 027224    BIS    @NOLOOP,1$    ; YES: SPECIFY NO TTLOOP (INITRN)
5477 027210 052737 100000 027670    BIS    @BIT15,3$     ; AND SPECIFY NOLOOP IN ENTRAN
5478
5479
5480 027216 004537 007234          2$:   JSR    R5,INITRN   ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5481 027222 062226                    DDCMP!STRIPS!OVRC!SYNCH ;SET DDCMP,STRIP SYNC,ODD VRC,SYNCH=226
5482 027224 000347                    1$:   TXDL!RXDL        ;USE 7 BIT CHARS FOR RX & TX
5483 027226 103003                    BCC    ..8.          ;BR IF NO ERROR
5484 027230                    ERROR                ;REPORT STACKED ERROR
5485 027230 104460                    ESCAPE TST           ;SKIP TO END OF TEST          TRAP    C$ERROR
5485 027232 104410                    .WORD                TRAP    C$ESCAPE
5485 027234 000446                    .WORD                L10023-.
5486
5487 027236 004537 010136          JSR    R5,IXCHAR     ;LOAD 2ND SYNCH, TX 1ST SYNCH
5488 027242 000226                    SYNCH
5489 027244 000007                    7.
5490 027246 103003                    BCC    ..8.          ;BR IF NO ERROR
5491 027250                    ERROR                ;REPORT STACKED ERROR
5492 027250 104460                    ESCAPE TST           ;SKIP TO END OF TEST          TRAP    C$ERROR
5492 027252 104410                    .WORD                TRAP    C$ESCAPE
5492 027254 000426                    .WORD                L10023-.
5493
5494 027256 004537 010136          JSR    R5,IXCHAR     ;LOAD 3RD SYNCH, TX 2ND SYNCH

```

TEST 4	DATA	TEST	BCP	XLB	ODD	VRC				
5495	027262	000226					SYNCH			
5496	027264	000010					8.			
5497	027266	103003					BCC	.+8.	;BR IF NO ERROR	
5498	027270						ERROR		;REPORT STACKED ERROR	
	027270	104460								TRAP C\$ERROR
5499	027272						ESCAPE	TST	;SKIP TO END OF TEST	
	027272	104410								TRAP C\$ESCAPE
	027274	000406								.WORD L10023 .
5500										
5501	027276	004537	010250				JSR	R5, TXCTRL	;CLEAR TSOM	
5502	027302	000000					000			
5503	027304	000000					0			
5504										
5505	027306	004537	010136				JSR	R5, TXCHAR	;LOAD 125(DATA1), TX 3RD SYNCH	
5506	027312	000125					125			
5507	027314	000010					8.			
5508	027316	103003					BCC	.+8.	;BR IF NO ERROR	
5509	027320						ERROR		;REPORT STACKED ERROR	
	027320	104460								TRAP C\$ERROR
5510	027322						ESCAPE	TST	;SKIP TO END OF TEST	
	027322	104410								TRAP C\$ESCAPE
	027324	000356								.WORD L10023 .
5511										
5512	027326	004537	010136				JSR	R5, TXCHAR	;LOAD 252(DATA2), TX 125(DATA1)	
5513	027332	000252					252			
5514	027334	000010					8.			
5515	027336	103003					BCC	.+8.	;BR IF NO ERROR	
5516	027340						ERROR		;REPORT STACKED ERROR	
	027340	104460								TRAP C\$ERROR
5517	027342						ESCAPE	TST	;SKIP TO END OF TEST	
	027342	104410								TRAP C\$ESCAPE
	027344	000336								.WORD L10023 .
5518										
5519	027346	004537	010136				JSR	R5, TXCHAR	;LOAD 000(DATA3)	
5520	027352	000000					000			
5521	027354	000000					0			
5522	027356	103003					BCC	.+8.	;BR IF NO ERROR	
5523	027360						ERROR		;REPORT STACKED ERROR	
	027360	104460								TRAP C\$ERROR
5524	027362						ESCAPE	TST	;SKIP TO END OF TEST	
	027362	104410								TRAP C\$ESCAPE
	027364	000316								.WORD L10023 .
5525										
5526	027366	004537	011624				JSR	R5, RCV1ST	;CLOCK AND RCV 125(DATA1)	
5527	027372	000000					0			
5528	027374	103003					BCC	.+8.	;BR IF NO ERROR	
5529	027376						ERROR		;REPORT STACKED ERROR	
	027376	104460								TRAP C\$ERROR
5530	027400						ESCAPE	TST	;SKIP TO END OF TEST	
	027400	104410								TRAP C\$ESCAPE
	027402	000300								.WORD L10023 .
5531										
5532	027404	004537	010350				JSR	R5, RXCHAR	;READ & CHK 125(DATA1), RCV 252(DATA2)	
5533	027410	000125					125			
5534	027412	000001					RERCHK		; & CHECK RERR BIT=0 (GOOD VRC)	
5535	027414	000010					8.			
5536	027416	103003					BCC	.+8.	;BR IF NO ERROR	

TEST 4 - DATA TEST - BCP XLB ODD VRC

```

5537 027420          ERROR          ;REPORT STACKED ERROR
      027420 104460
5538 027422          ESCAPE TST     ;SKIP TO END OF TEST
      027422 104410          TRAP    C$ERROR
      027424 000256          .WORD  L10023 .
5539
5540          ;-----
5541          ; TRANSMIT THE BULK OF DATA OUT OF TABLE "PATX"
5542 027426 012702 002646          MOV    #PATX+1,R2      ;SET UP TABLE POINTER
5543 027432 112237 027470 5$:    MOVB  (R2)+,20$      ;SET UP EXPECTED CHARACTER
5544 027436 116237 000001 027450  MOVB  1(R2),10$     ;SET UP TRANSMIT CHARACTER
5545
5546 027444 004537 010136          JSR    R5,TXCHAR     ;LOAD A CHARACTER
5547 027450 000000          10$:    000          ;** HOLE FOR NEXT TX CHARACTER
5548 027452 000000          0
5549 027454 103003          BCC   .+8.          ;BR IF NO ERROR
5550 027456          ERROR          ;REPORT STACKED ERROR
      027456 104460          TRAP    C$ERROR
5551 027460          ESCAPE TST     ;SKIP TO END OF TEST
      027460 104410          TRAP    C$ESCAPE
      027462 000220          .WORD  L10023 .
5552
5553 027464 004537 010350          JSR    R5,RXCHAR     ;CLK/RECEIVE/CHECK PREVIOUS CHARACTER
5554 027470 000000          20$:    000          ;** HOLE FOR EXPECTED CHARACTER
5555 027472 000001          RERCHK ; & CHECK RERR BIT=0 (GOOD VRC)
5556 027474 000010          8.
5557 027476 103003          BCC   .+8.          ;BR IF NO ERROR
5558 027500          ERROR          ;REPORT STACKED ERROR
      027500 104460          TRAP    C$ERROR
5559 027502          ESCAPE TST     ;SKIP TO END OF TEST
      027502 104410          TRAP    C$ESCAPE
      027504 000176          .WORD  L10023-.
5560
5561 027506 022702 002737          CMP    #EPATX-1,R2
5562 027512 001347          BNE   5$
5563          ;-----
5564 027514 004537 010250          JSR    R5,TXCTRL     ;LOAD TSOM
5565 027520 000001          TSOM
5566 027522 000000          0
5567 027524 103003          BCC   .+8.          ;BR IF NO ERROR
5568 027526          ERROR          ;REPORT STACKED ERROR
      027526 104460          TRAP    C$ERROR
5569 027530          ESCAPE TST     ;SKIP TO END OF TEST
      027530 104410          TRAP    C$ESCAPE
      027532 000150          .WORD  L10023 .
5570
5571 027534 004537 010350          JSR    R5,RXCHAR     ;READ & CHK 277, RCV 177
5572 027540 000277          277
5573 027542 000001          RERCHK ; & CHECK RERR BIT=0 (GOOD VRC)
5574 027544 000010          8.
5575 027546 103003          BCC   .+8.          ;BR IF NO ERROR
5576 027550          ERROR          ;REPORT STACKED ERROR
      027550 104460          TRAP    C$ERROR
5577 027552          ESCAPE TST     ;SKIP TO END OF TEST
      027552 104410          TRAP    C$ESCAPE
      027554 000126          .WORD  L10023 .
5578

```

TEST 4 - DATA TEST -- BCP XLB ODD VRC

```

5579 027556 004537 010250      JSR    R5,TXCTRL      ;LOAD 2ND TSOM
5580 027562 000001              TSOM
5581 027564 000000              0
5582 027566 103003      BCC    .+8.          ;BR IF NO ERROR
5583 027570 104460      ERROR          ;REPORT STACKED ERROR
                                TRAP    C$ERROR
5584 027572 104410      ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP    C$ESCAPE
                                .WORD  L10023 .
                                027572 104410
                                027574 000106
5585
5586 027576 004537 010350      JSR    R5,RXCHAR      ;READ & CHK 177, RCV FIRST SYNC
5587 027602 000177              177
5588 027604 000001      RERCHK          ; & CHECK RERR BIT=0 (GOOD VRC)
5589 027606 000010              8.
5590 027610 103003      BCC    .+8.          ;BR IF NO ERROR
5591 027612 104460      ERROR          ;REPORT STACKED ERROR
                                TRAP    C$ERROR
5592 027614 104410      ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP    C$ESCAPE
                                .WORD  L10023-.
                                027614 104410
                                027616 000064
5593
5594 027620 004537 010350      JSR    R5,RXCHAR      ;READ & CHK 1ST SYNC, RCV SECOND SYNC
5595 027624 000226              SYNCH
5596 027626 000001      RERCHK          ; & CHECK RERR BIT=0 (GOOD VRC)
5597 027630 000010              8.
5598 027632 103003      BCC    .+8.          ;BR IF NO ERROR
5599 027634 104460      ERROR          ;REPORT STACKED ERROR
                                TRAP    C$ERROR
5600 027636 104410      ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP    C$ESCAPE
                                .WORD  L10023-.
                                027636 104410
                                027640 000042
5601
5602 027642 004537 010350      JSR    R5,RXCHAR      ;READ & CHK 2ND SYNC, RCV NEXT ONE
5603 027646 000226              SYNCH
5604 027650 000001      RERCHK          ; & CHECK RERR BIT=0 (GOOD VRC)
5605 027652 000010              8.
5606 027654 103003      BCC    .+8.          ;BR IF NO ERROR
5607 027656 104460      ERROR          ;REPORT STACKED ERROR
                                TRAP    C$ERROR
5608 027660 104410      ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP    C$ESCAPE
                                .WORD  L10023-.
                                027660 104410
                                027662 000020
5609
5610 027664 004537 011772      JSR    R5,ENTRAN      ;SHUT DOWN TRANSMITTER, RECEIVER
5611 027670 000011      3$: 9.
5612 027672 103003      BCC    .+8.          ;BR IF NO ERROR
5613 027674 104460      ERROR          ;REPORT STACKED ERROR
                                TRAP    C$ERROR
5614 027676 104410      ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP    C$ESCAPE
                                .WORD  L10023-.
                                027676 104410
                                027700 000002
5615 027702 104401      ENDTST          L10023: TRAP    C$ETST
                                027702
                                027702

```

TEST 5 - DATA TEST - BCP XLB EVEN VRC

5633

.SBTTL TEST 5 -- DATA TEST -- BCP XLB EVEN VRC

```

*****
;*
;* TEST 5 -- DATA TEST -- BCP XLB EVEN VRC
;*
;* IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
;* RECEIVE IN BCP MODE WITH EVEN VRC ERROR DETECTION THE FOLLOWING
;* MESSAGE:
;*
;* 125 252 000 377 001 002 004 010 020 040 100 200 376 375 373 367
;* 357 337 277 177
;*
;* THIS MESSAGE WILL BE PRECEDED BY 3 SYNC CHARACTERS AND REPEATED
;* THREE TIMES. AFTER THE LAST MESSAGE, SEVERAL SYNC CHARACTERS ARE
;* SENT BEFORE DROPPING TXE & RXE. 7-BIT CHARACTER LENGTHS ARE ALSO
;* UTILIZED.
;*
;* IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
;* USING INTERNAL LOOPBACK (ITLOOP=1).
;*
```

```

*****
;
; BGNTST
;
; JSR PC,INIDMV ;INIT DMV-11, ENTER M-LOOP
; BIC #NOLOOP,1$ ;
; MOV #9,3$ ;INIT ENTRAN COUNT/STATUS
; CMP BRDTYP,#0 ;IS BOARD TYPE M8064?
; BEQ 2$ ; YES: SPECIFY ITLOOP (NOT XLB)
; CMP TSTCON,#4 ;IS A LOOPBACK CONNECTOR/CABLE SPECIFIED ?
; BEQ 2$ ;BR IF NO
; BIS #NOLOOP,1$ ; YES: SPECIFY NO TTLOOP (INITRN)
; BIS BIT15,3$ ; AND SPECIFY NOLOOP IN ENTRAN
;
;-----
; 2$: JSR R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
; DDCMP!STRIPS!EVRC!SYNCH ;SET DDCMP,STRIP SYNCH,EVEN VRC.SYNCH=226
; 1$: TXDL!RXDL ;USE 7 BIT CHARS FOR TX & RX
; BCC .+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
;
; ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
;
; 27774 104410 TRAP C$ESCAPE
; 27776 000446 .WORD L10024-.
;
; JSR R5, TXCHAR ;LOAD 2ND SYNCH, TX 1ST SYNCH
; SYNCH
; 7.
; BCC .+8. ;BR IF NO ERROR
; ERROR ;REPORT STACKED ERROR
;
; ESCAPE TST ;SKIP TO END OF TEST TRAP C$ERROR
;
; 30014 104410 TRAP C$ESCAPE
; 30016 000426 .WORD L10024.
;
; JSR R5, TXCHAR ;LOAD 3RD SYNCH, TX 2ND SYNCH
; SYNCH
```

```

027704
5634 027704 004737 005376
5635 027710 042737 001000 027766
5636 027716 012737 000011 030432
5637 027724 023727 002470 000000
5638 027732 001412
5639 027734 023727 002472 000004
5640 027742 001406
5641 027744 052737 001000 027766
5642 027752 053737 100000 030432
5643
5644 027760 004537 007234
5645 027764 062626
5646 027766 000347
5647 027770 103003
5648 027772
027772 104460
5649 027774
027774 104410
027776 000446
5650
5651 030000 004537 010136
5652 030004 000226
5653 030006 000007
5654 030010 103003
5655 030012
030012 104460
5656 030014
030014 104410
030016 000426
5657
5658 030020 004537 010136
5659 030024 000226
```

TEST 5 - DATA TEST -- BCP XLB EVEN VRC

5660	030026	000010		8.					
5661	030030	103003		BCC	.+8.			;BR IF NO ERROR	
5662	030032			ERROR				;REPORT STACKED ERROR	
	030032	104460							TRAP C\$ERROR
5663	030034			ESCAPE	TST			;SKIP TO END OF TEST	
	030034	104410							TRAP C\$ESCAPE
	030036	000406							.WORD L10024-
5664									
5665	030040	004537	010250	JSR	R5,TXCTRL			;CLEAR TSOM	
5666	030044	000000		000					
5667	030046	000000		0					
5668									
5669	030050	004537	010136	JSR	R5,TXCHAR			;LOAD 125(DATA1), TX 3RD SYNCH	
5670	030054	000125		125					
5671	030056	000010		8.					
5672	030060	103003		BCC	.+8.			;BR IF NO ERROR	
5673	030062			ERROR				;REPORT STACKED ERROR	
	030062	104460							TRAP C\$ERROR
5674	030064			ESCAPE	TST			;SKIP TO END OF TEST	
	030064	104410							TRAP C\$ESCAPE
	030066	000356							.WORD L10024-
5675									
5676	030070	004537	010136	JSR	R5,TXCHAR			;LOAD 252(DATA2), TX 125(DATA1)	
5677	030074	000252		252					
5678	030076	000010		8.					
5679	030100	103003		BCC	.+8.			;BR IF NO ERROR	
5680	030102			ERROR				;REPORT STACKED ERROR	
	030102	104460							TRAP C\$ERROR
5681	030104			ESCAPE	TST			;SKIP TO END OF TEST	
	030104	104410							TRAP C\$ESCAPE
	030106	000336							.WORD L10024-
5682									
5683	030110	004537	010136	JSR	R5,TXCHAR			;LOAD 000(DATA3)	
5684	030114	000000		000					
5685	030116	000000		0					
5686	030120	103003		BCC	.+8.			;BR IF NO ERROR	
5687	030122			ERROR				;REPORT STACKED ERROR	
	030122	104460							TRAP C\$ERROR
5688	030124			ESCAPE	TST			;SKIP TO END OF TEST	
	030124	104410							TRAP C\$ESCAPE
	030126	000316							.WORD L10024-
5689									
5690	030130	004537	011624	JSR	R5,RCV1ST			;CLOCK AND RCV 125(DATA1)	
5691	030134	000000		0					
5692	030136	103003		BCC	.+8.			;BR IF NO ERROR	
5693	030140			ERROR				;REPORT STACKED ERROR	
	030140	104460							TRAP C\$ERROR
5694	030142			ESCAPE	TST			;SKIP TO END OF TEST	
	030142	104410							TRAP C\$ESCAPE
	030144	000300							.WORD L10024-
5695									
5696	030146	004537	010350	JSR	R5,RXCHAR			;READ & CHK 125(DATA1), RCV 252(DATA2)	
5697	030152	000125		125					
5698	030154	000001		RERCHK				; & CHECK RERR BIT=0 (GOOD VRC)	
5699	030156	000010		8.					
5700	030160	103003		BCC	.+8.			;BR IF NO ERROR	
5701	030162			ERROR				;REPORT STACKED ERROR	

TEST 5 - DATA TEST BCP XLB EVEN VRC

5702	030162	104460							TRAP	C\$ERROR
	030164			ESCAPE	TST		;SKIP TO END OF TEST			
	030164	104410							TRAP	C\$ESCAPE
	030166	000256							.WORD	L10024 .
5703										
5704							; TRANSMIT THE BULK OF DATA OUT OF TABLE "PATX"			
5705										
5706	030170	012702	002646				MOV @PATX+1,R2			;SET UP TABLE POINTER
5707	030174	112237	030232		5\$:		MOVB (R2)+,20\$;SET UP EXPECTED CHARACTER
5708	030200	116237	000001	030212			MOVB 1(R2),10\$;SET UP TRANSMIT CHARACTER
5709										
5710	030206	004537	010136				JSR R5,TXCHAR			;LOAD A CHARACTER
5711	030212	000000			10\$:		000			;** HOLE FOR NEXT TX CHARACTER
5712	030214	000000					0			
5713	030216	103003					BCC .+8.			;BR IF NO ERROR
5714	030220						ERROR			;REPORT STACKED ERROR
	030220	14460							TRAP	C\$ERROR
5715	030222						ESCAPE	TST		;SKIP TO END OF TEST
	030222	104410							TRAP	C\$ESCAPE
	030224	000.20							.WORD	L10024 .
5716										
5717	030226	004537	010350				JSR R5,RXCHAR			;CLK/RECEIVE/CHECK PREVIOUS CHARACTER
5718	030232	000000			20\$:		000			;** HOLE FOR EXPECTED CHARACTER
5719	030234	000001					RERCHK			; & CHECK RERR BIT=0 (GOOD VRC)
5720	030236	000010					8.			
5721	030240	103003					BCC .+8.			;BR IF NO ERROR
5722	030242						ERROR			;REPORT STACKED ERROR
	030242	104460							TRAP	C\$ERROR
5723	030244						ESCAPE	TST		;SKIP TO END OF TEST
	030244	104410							TRAP	C\$ESCAPE
	030246	000176							.WORD	L10024 .
5724										
5725	030250	022702	002737				CMP @EPATX 1,R2			
5726	030254	001347					BNE 5\$			
5727										
5728	030256	004537	010250				JSR R5,TXCTRL			;LOAD 1ST TSOM
5729	030262	000001					TSOM			
5730	030264	000000					0			
5731	030266	103003					BCC .+8.			;BR IF NO ERROR
5732	030270						ERROR			;REPORT STACKED ERROR
	030270	104460							TRAP	C\$ERROR
5733	030272						ESCAPE	TST		;SKIP TO END OF TEST
	030272	104410							TRAP	C\$ESCAPE
	030274	000150							.WORD	L10024 .
5734										
5735	030276	004537	010350				JSR R5,RXCHAR			;READ & CHK 277, RCV 177
5736	030302	000277					277			
5737	030304	000001					RERCHK			; & CHECK RERR BIT=0 (GOOD VRC)
5738	030306	000010					8.			
5739	030310	103003					BCC .+8.			;BR IF NO ERROR
5740	030312						ERROR			;REPORT STACKED ERROR
	030312	104460							TRAP	C\$ERROR
5741	030314						ESCAPE	TST		;SKIP TO END OF TEST
	030314	104410							TRAP	C\$ESCAPE
	030316	000126							.WORD	L10024 .
5742										
5743	030320	004537	010250				JSR R5,TXCTRL			;LOAD 2ND TSOM

TEST 5 DATA TEST BCP XLB EVEN VRC

5744	030324	000001		TSOM					
5745	030326	000000		0					
5746	030330	103003		BCC	.+8.		;BR IF NO ERROR		
5747	030332			ERROR			;REPORT STACKED ERROR		
	030332	104460						TRAP	C\$ERROR
5748	030334			ESCAPE	TST		;SKIP TO END OF TEST		
	030334	104410						TRAP	C\$ESCAPE
	030336	000106						.WORD	L10024 .
5749									
5750	030340	004537	010350	JSR	R5,RXCHAR		;READ & CHK 177. RCV FIRST SYNC		
5751	030344	000177		177					
5752	030346	000001		RERCHK			; & CHECK RERR BIT=0 (GOOD VRC)		
5753	030350	000010		8.					
5754	030352	103003		BCC	.+8.		;BR IF NO ERROR		
5755	030354			ERROR			;REPORT STACKED ERROR		
	030354	104460						TRAP	C\$ERROR
5756	030356			ESCAPE	TST		;SKIP TO END OF TEST		
	030356	104410						TRAP	C\$ESCAPE
	030360	000064						.WORD	L10024 .
5757									
5758	030362	004537	010350	JSR	R5,RXCHAR		;READ & CHK 1ST SYNC. RCV SECOND SYNC		
5759	030366	000226		SYNCH					
5760	030370	000001		RERCHK			; & CHECK RERR BIT=0 (GOOD VRC)		
5761	030372	000010		8.					
5762	030374	103003		BCC	.+8.		;BR IF NO ERROR		
5763	030376			ERROR			;REPORT STACKED ERROR		
	030376	104460						TRAP	C\$ERROR
5764	030400			ESCAPE	TST		;SKIP TO END OF TEST		
	030400	104410						TRAP	C\$ESCAPE
	030402	000042						.WORD	L10024 .
5765									
5766	030404	004537	010350	JSR	R5,RXCHAR		;READ & CHK 2ND SYNC. RCV NEXT ONE		
5767	030410	000226		SYNCH					
5768	030412	000001		RERCHK			; & CHECK RERR BIT=0 (GOOD VRC)		
5769	030414	000010		8.					
5770	030416	103003		BCC	.+8.		;BR IF NO ERROR		
5771	030420			ERROR			;REPORT STACKED ERROR		
	030420	104460						TRAP	C\$ERROR
5772	030422			ESCAPE	TST		;SKIP TO END OF TEST		
	030422	104410						TRAP	C\$ESCAPE
	030424	000020						.WORD	L10024 .
5773									
5774	030426	004537	011772	JSR	R5,ENTRAN		;SHUT DOWN TRANSMITTER, RECEIVER		
5775	030432	000011		9.					
5776	030434	103003		BCC	.+8.		;BR IF NO ERROR		
5777	030436			ERROR			;REPORT STACKED ERROR		
	030436	104460						TRAP	C\$ERROR
5778	030440			ESCAPE	TST		;SKIP TO END OF TEST		
	030440	104410						TRAP	C\$ESCAPE
	030442	000002						.WORD	L10024 .
5779	030444			ENDTST					
	030444							L10024:	
	030444	104401						TRAP	C\$ETST

TEST 6 DATA TEST BOP XLB CRC CCITT-1

5793

.SBTTL TEST 6 - DATA TEST - BOP XLB CRC CCITT 1

```

*****
;*
;* TEST 6 -- DATA TEST -- BOP XLB CRC-CCITT 1
;*
;* IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
;* RECEIVE IN BOP MODE WITH CRC-CCITT-1 ERROR DETECTION THE FOLLOWING
;* SHORT MESSAGE: 125 252 000 377 001
;*
;* THIS MESSAGE WILL BE PRECEDED BY FLAG CHARACTERS AND REPEATED
;* THREE TIMES WITH CRC AND FLAG'S FOLLOWING EACH ONE. 8-BIT CHARACTER
;* LENGTHS ARE ALSO UTILIZED.
;*
;* IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
;* USING INTERNAL LOOPBACK (TTLOOP=1).
;*
*****

```

```

;
; BGNTST
;
; T6:
5794 030446 012737 000003 002526      MOV     #3,REGO      ;INIT COUNT (TEXT TRANSMITTED 3 TIMES)
5795 030454 004737 005376              JSR     PC,INIDMV    ;INIT DMV-11, ENTER MAINT LOOP
5796 030460 042737 001000 030522      BIC     @NOLOOP,1$
5797 030466 023727 002470 000000      CMP     BRDYP,#0     ;IS BOARD TYPE = M8064 ?
5798 030474 001407              BEQ     2$           ; YES: SPECIFY TTLOOP (NO XLB)
5799 030476 023727 002472 000004      CMP     TSTCON,#4   ;IS A LOOPBACK CONNECTOR/CABLE SPECIFIED ?
5800 030504 001403              BEQ     2$           ;BR IF NO
5801 030506 052737 001000 030522      BIS     @NOLOOP,1$  ; YES: SPECIFY NO TTLOOP (INITRN)
5802
5803 030514 004537 007234      2$: JSR     R5,INITRN  ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5804 030520 000000              0           ;SET BOP MODE,CRC CCITT->1'S CHECK
5805 030522 000000      1$: 0           ;USE 8 BIT CHARS
5806 030524 103003              BCC     .+8.       ;BR IF NO ERROR
5807 030526              ERROR          ;REPORT STACKED ERROR
5808 030530              ESCAPE TST       ;SKIP TO END OF TEST          TRAP     C$ERROR
5809 030532 000374              .WORD     L10025 .
5810 030534 004537 010250      JSR     R5,TXCTRL   ;LOAD 2ND FLAG, TX 1ST FLAG
5811 030540 000001              TSOM
5812 030542 000007              7.
5813 030544 004537 010250      JSR     R5,TXCTRL   ;LOAD 3RD FLAG, TX 2ND FLAG
5814 030550 000001              TSOM
5815 030552 000010              8.
5816 030554 004537 010250      JSR     R5,TXCTRL   ;LOAD 4TH FLAG, TX 3RD FLAG
5817 030560 000001              TSOM
5818 030562 000010              8.
5819 030564 004537 010250      JSR     R5,TXCTRL   ;CLEAR TSOM
5820 030570 000000              000
5821 030572 000000              0
5822 030574 004537 010136      JSR     R5,TXCHAR   ;LOAD DATA1(125), TX 4TH FLAG
5823 030600 000125              125
5824 030602 000010              8.
5825 030604 103003              BCC     .+8.       ;BR IF NO ERROR
5826 030606              ERROR          ;REPORT STACKED ERROR

```

TEST 6 DATA TEST BOP XLB CRC CCITT 1

5827	030606	104460		ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030610						TRAP	C\$ESCAPE
	030610	104410					.WORD	L10025 .
	030612	000314						
5828								
5829	030614	004537	010136	JSR	R5, TXCHAR	;LOAD DATA2(252), TX DATA1(125)		
5830	030620	000252						
5831	030622	000010						
5832	030624	103003						
5833	030626			BCC	.+8.	;BR IF NO ERROR		
	030626	104460		ERROR		;REPORT STACKED ERROR		
5834	030630			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030630	104410					TRAP	C\$ESCAPE
	030632	000274					.WORD	L10025 .
5835								
5836	030634	004537	010136	JSR	R5, TXCHAR	;LOAD DATA3(000), TX DATA2(252)		
5837	030640	000000						
5838	030642	000010						
5839	030644	103003						
5840	030646			BCC	.+8.	;BR IF NO ERROR		
	030646	104460		ERROR		;REPORT STACKED ERROR		
5841	030650			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030650	104410					TRAP	C\$ESCAPE
	030652	000254					.WORD	L10025 .
5842								
5843	030654	004537	010136	JSR	R5, TXCHAR	;LOAD DATA4(377), TX DATA3(000)		
5844	030660	000377						
5845	030662	000010						
5846	030664	103003						
5847	030666			BCC	.+8.	;BR IF NO ERROR		
	030666	104460		ERROR		;REPORT STACKED ERROR		
5848	030670			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030670	104410					TRAP	C\$ESCAPE
	030672	000234					.WORD	L10025 .
5849								
5850	030674	004537	010136	JSR	R5, TXCHAR	;LOAD DATA5(001), TX DATA4(377)		
5851	030700	000001						
5852	030702	000011						
5853	030704	103003						
5854	030706			BCC	.+8.	;BR IF NO ERROR		
	030706	104460		ERROR		;REPORT STACKED ERROR		
5855	030710			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030710	104410					TRAP	C\$ESCAPE
	030712	000214					.WORD	L10025 .
5856								
5857	030714	004537	010250	JSR	R5, TXCTRL	;SET TEOM		
5858	030720	000002						
5859	030722	000000						
5860								
5861	030724	004537	011624	JSR	R5, RCV1ST	;CLOCK AND RCV DATA1(125)		
5862	030730	000000						
5863	030732	103003						
5864	030734			BCC	.+8.	;BR IF NO ERROR		
	030734	104460		ERROR		;REPORT STACKED ERROR		
5865	030736			ESCAPE	TST	;SKIP TO END OF TEST	TRAP	C\$ERROR
	030736	104410					TRAP	C\$ESCAPE
	030740	000166					.WORD	L10025 .

TEST 6 DATA TEST BOP XLB CRC CCITT-1

5866								
5867	030742	004537	010350	JSR	R5,RXCHAR			
5868	030746	000525		RXSOM!	125			
5869	030750	000000		0				
5870	030752	000010		8.				
5871	030754	103003		BCC	..+8.			
5872	030756			ERROR				
	030756	104460						
5873	030760			ESCAPE	TST			
	030760	104410						
	030762	000144						
5874								
5875	030764	004537	010250	JSR	R5, TXCTRL			
5876	030770	000002		TEOM				
5877	030772	000000		0				
5878								
5879	030774	004537	010350	JSR	R5,RXCHAR			
5880	031000	000252		252				
5881	031002	000000		0				
5882	031004	000010		8.				
5883	031006	103003		BCC	..+8.			
5884	031010			ERROR				
	031010	104460						
5885	031012			ESCAPE	TST			
	031012	104410						
	031014	000112						
5886								
5887	031016	004537	010350	JSR	R5,RXCHAR			
5888	031022	000000		000				
5889	031024	000000		0				
5890	031026	000010		8.				
5891	031030	103003		BCC	..+8.			
5892	031032			ERROR				
	031032	104460						
5893	031034			ESCAPE	TST			
	031034	104410						
	031036	000070						
5894								
5895	031040	004537	010350	JSR	R5,RXCHAR			
5896	031044	000377		377				
5897	031046	000000		0				
5898	031050	020010		NCRACK!	8.			
5899	031052	103003		BCC	..+8.			
5900	031054			ERROR				
	031054	104460						
5901	031056			ESCAPE	TST			
	031056	104410						
	031060	000046						
5902								
5903	031062	004537	010350	JSR	R5,RXCHAR			
5904	031066	001001		RXEOM!	001			
5905	031070	000001		RERCHK				
5906	031072	060000		NFCRDA!	NCRACK			
5907	031074	103003		BCC	..+8.			
5908	031076			ERROR				
	031076	104460						
5909	031100			ESCAPE	TST			

TEST 6 - DATA TEST - BOP XLB CRC CCITT-1

031100	104410					TRAP	C\$ESCAPE
031102	000024					.WORD	L10025-
5910							
5911	031104	005337	002526				
5912	031110	001406		DEC	REGO		
5913				BEQ	40\$		
5914	031112	004537	010250				
5915	031116	000001		JSR	R5, TXCTRL		
5916	031120	000001		TSOM			
5917	031122	000137	030514	1			
5918	031126			JMP	2\$		
5919	031126						
	031126						
	031126	104401					

40\$:
ENDTST

L10025: TRAP C\$ETST

TEST 7 DATA TEST - BOP XLB CRC-CCITT-0

5933

.SBTTL TEST 7 DATA TEST BOP XLB CRC CCITT 0

```

*****
:
:* TEST 7 -- DATA TEST - BOP XLB CRC-CCITT 0
:*
:* IF XLB IS SPECIFIED IN THE P-TABLE, THIS TEST WILL TRANSMIT &
:* RECEIVE IN BOP MODE WITH CRC CCITT-0 ERROR DETECTION THE FOLLOWING
:* SHORT MESSAGE: 125 252 000 377 001
:*
:* THIS MESSAGE WILL BE PRECEDED BY FLAG CHARACTERS AND REPEATED
:* THREE TIMES WITH CRC AND FLAG'S FOLLOWING EACH ONE. 8-BIT CHARACTER
:* LENGTHS ARE ALSO UTILIZED.
:*
:* IF XLB WAS NOT SPECIFIED (AND/OR BOARD TYPE IS M8064), THIS TEST MAY BE RUN
:* USING INTERNAL LOOPBACK (TTLOOP=1).
:*
*****

```

```

:
: BGNTST
:
5934 031130 012737 000003 002526      MOV    #3,REG0      ;INIT COUNT (TEXT TRANSMITTED 3 TIMES)
5935 031136 004737 005376              JSR    PC,INIDMV    ;INIT DMV-11, ENTER MAINT LOOP
5936 031142 042737 001000 031204      BIC    #NOLOOP,1#
5937 031150 023727 002470 000000      CMP    BRDYP,#0    ;BOARD TYPE = M8064 ?
5938 031156 001407                    BEQ    2#           ; YES: SPECIFY TTLOOP (NOT XLB)
5939 031160 023727 002472 000004      CMP    TSTCON,#4   ;IS A LOOPBACK CONNECTOR/CABLE SPECIFIED ?
5940 031166 001403                    BEQ    2#           ;BR IF NO
5941 031170 052737 001000 031204      BIS    #NOLOOP,1# ; YES: SPECIFY NO T'LOOP (INITRN)
5942
5943 031176 004537 007234      2#: JSR    R5,INITRN ;LOAD 1 SOM, CLK TX UNTIL ACTIVE
5944 031202 000400              CRCOS              ;SET BOP MODE,CRC CCITT->0'S CHECK
5945 031204 000000              0                 ;USE 8 BIT CHARS
5946 031206 103003              1#: BCC    .+8.      ;BR IF NO ERROR
5947 031210 104460              ERROR             ;REPORT STACKED ERROR
5948 031212 104410              ESCAPE TST        ;SKIP TO END OF TEST
5949 031214 000374              .WORD            TRAP C$ERROR
5950 031216 004537 010250              JSR    R5,TXCTRL  ;LOAD 2ND FLAG, TX 1ST FLAG
5951 031222 000001              TSOM              7.
5952 031224 000007              JSR    R5,TXCTRL  ;LOAD 3RD FLAG, TX 2ND FLAG
5953 031226 004537 010250              TSOM              8.
5954 031232 000001              8.
5955 031234 000010              JSR    R5,TXCTRL  ;LOAD 4TH FLAG, TX 3RD FLAG
5956 031236 004537 010250              TSOM              8.
5957 031242 000001              JSR    R5,TXCTRL  ;CLEAR TSOM
5958 031244 000010              0
5959 031246 004537 010250              0
5960 031252 000000              JSR    R5,TXCHAR  ;LOAD DATA1(125), TX 4TH FLAG
5961 031254 000000              125
5962 031256 004537 010136              8.
5963 031262 000125              BCC    .+8.      ;BR IF NO ERROR
5964 031264 000010              ERROR             ;REPORT STACKED ERROR
5965 031266 103003
5966 031270

```


TEST 7 - DATA TEST - BOP XLB CRC CCITT 0

6006	031424	004537	010350	JSR	R5,RXCHAR	;READ/CHK DATA1(125), RCV DATA2(252)		
6007	031430	000525		RXSOM!	125	; & CHECK RSOM=1		
6008	031432	000000		0				
6009	031434	000010		8.				
6010	031436	103003		BCC	+.8.	;BR IF NO ERROR		
6011	031440			ERROR		;REPORT STACKED ERROR		
	031440	104460					TRAP	C\$ERROR
6012	031442			ESCAPE	TST	;SKIP TO END OF TEST		
	031442	104410					TRAP	C\$ESCAPE
	031444	000144					.WORD	L10026 .
6013								
6014	031446	004537	010250	JSR	R5, TXCTRL	;SET TEOM		
6015	031452	000002		TEOM				
6016	031454	000000		0				
6017	031456	004537	010350	JSR	R5,RXCHAR	;READ/CHK DATA2(252), RCV DATA3(000)		
6018	031462	000252		252				
6019	031464	000000		0				
6020	031466	000010		8.				
6021	031470	103003		BCC	+.8.	;BR IF NO ERROR		
6022	031472			ERROR		;REPORT STACKED ERROR		
	031472	104460					TRAP	C\$ERROR
6023	031474			ESCAPE	TST	;SKIP TO END OF TEST		
	031474	104410					TRAP	C\$ESCAPE
	031476	000112					.WORD	L10026-.
6024								
6025	031500	004537	010350	JSR	R5,RXCHAR	;READ/CHK DATA3(000), RCV DATA4(377)		
6026	031504	000000		000				
6027	031506	000000		0				
6028	031510	000010		8.				
6029	031512	103003		BCC	+.8.	;BR IF NO ERROR		
6030	031514			ERROR		;REPORT STACKED ERROR		
	031514	104460					TRAP	C\$ERROR
6031	031516			ESCAPE	TST	;SKIP TO END OF TEST		
	031516	104410					TRAP	C\$ESCAPE
	031520	000070					.WORD	L10026-.
6032								
6033	031522	004537	010350	JSR	R5,RXCHAR	;READ/CHK DATA4(377), RCV DATA5(001)		
6034	031526	000377		377				
6035	031530	000000		0				
6036	031532	020010		NCRCT!	8.	;DON'T CHECK FOR FINAL RXACT=1		
6037	031534	103003		BCC	+.8.	;BR IF NO ERROR		
6038	031536			ERROR		;REPORT STACKED ERROR		
	031536	104460					TRAP	C\$ERROR
6039	031540			ESCAPE	TST	;SKIP TO END OF TEST		
	031540	104410					TRAP	C\$ESCAPE
	031542	000046					.WORD	L10026 .
6040								
6041	031544	004537	010350	JSR	R5,RXCHAR	;READ/CHK DATA5(001), RCV FIRST FLAG		
6042	031550	001001		RXEOM!	001	; & CHECK REOM		
6043	031552	000001		RERCHK		; & CHECK RERR BIT=0 (GOOD CRC)		
6044	031554	060000		NFCRDA!	NCRCT	;DON'T CHECK FOR FINAL RDA=RXACT=1		
6045	031556	103003		BCC	+.8.	;BR IF NO ERROR		
6046	031560			ERROR		;REPORT STACKED ERROR		
	031560	104460					TRAP	C\$ERROR
6047	031562			ESCAPE	TST	;SKIP TO END OF TEST		
	031562	104410					TRAP	C\$ESCAPE
	031564	000024					.WORD	L10026 .

TEST 8 MODEM CONTROL SIGNAL LOOPBACK TEST

6077

.SBTTL TEST 8 -- MODEM CONTROL SIGNAL LOOPBACK TEST

```

;*****
;*
;* TEST 8 -- MODEM CONTROL SIGNAL LOOPBACK TEST
;*
;* FIRST, THE DMV-11 IS INITIALIZED. THEN, TTL LOOPBACK IS SELECTED,
;* AND THE FOLLOWING CHECKS ARE PERFORMED INVOLVING THE MODEM STATUS
;* REGISTER :
;* - RING, CARRIER, MODEM READY, TEST MODE, CTS ARE CHECKED FOR 1 STATE.
;* - RTS IS DE-ASSERTED AND CTS IS CHECKED FOR 0.
;* - RTS IS ASSERTED AND CTS IS CHECKED FOR 1.
;*
;* NEXT, IF THE OPTION IS AN M8053 WITH AN H3254 TEST CONNECTOR INSTALLED,
;* THE DMV-11 IN INITIALIZED AGAIN, (TTL LOOPBACK IS CLEARED), AND
;* THE FOLLOWING CHECKS ARE PERFORMED :
;* - RING (IF EIA), CARRIER, MODEM READY, CTS ARE CHECKED FOR 1, TEST
;* MODE IS CHECKED FOR 0.
;* - RTS IS DE-ASSERTED, AND CARRIER AND CTS ARE CHECKED FOR 0.
;* - RTS IS ASSERTED, AND CARRIER AND CTS ARE CHECKED FOR 1.
;* - DTR IS DE-ASSERTED, AND MODEM READY IS CHECKED FOR 0.
;* - DTR IS ASSERTED, AND MODEM READY IS CHECKED FOR 1.
;*
;*****

```

```

;
; BGNTST
;
6078 031612 012737 000010 002414      MOV     #8.,TSTNUM      ;SET TEST NO. FOR POSSIBLE PRINTOUT
6079 031620 004537 007532              JSR     R5,CKLPBK      ;CHK FOR H3254/5 INSTALLED
6080 031624 100000                      TCCHEK
6081 031626 103002                      BCC     2$             ;BR IF YES, TO RUN TEST
6082 031630                                EXIT     TST           ;NO TEST CONNECTOR, SKIP TEST
        031630 104432
        031632 000722                                TRAP    C$EXIT
6083 031634                                .WORD   L10027-.
6084 031634 2$:
        031634                                BGNSUB
        031634 104402                                T8.1:
6085                                TRAP    C$BSUB
;INIT DMV, SET TTL LOOPBACK, CHK MODEM STATUS
6086 031636 004737 003352              JSR     PC,MSTCLR     ;PERFORM MASTER CLEAR TO INIT DMV11
6087 031642 004537 003712              JSR     R5,WRITEI    ;SET PORT B FOR OUTPUT MODE
6088 031646 120002                      VIADPB
6089 031650 000377                      377
6090 031652 004537 003712              JSR     R5,WRITEI    ;SET PORT A FOR INPUT MODE
6091 031656 120003                      VIADPA
6092 031660 000000                      000
6093 031662 004537 003712              JSR     R5,WRITEI    ;SET TTL LOOPBACK
6094 031666 120000                      VIAORB
6095 031670 000002                      TTLOOP
6096 031672 012737 000001 002336      MOV     #1,REGNUM    ;SET REG NO. FOR PRINTOUT
6097                                ;CHK FOR RING, CARRIER, MODEM RDY, CTS, TEST MODE, = 1
6098 031700 004537 003566              JSR     R5,READI     ;READ MODEM STATUS
6099 031704 120017                      VIAORA
6100 031706 000000                      .WORD   0
6101 031710 142737 000023 031706      BICB   #SPEED!RCVDAT!UMAIN,4$ ;CLEAR UNNEEDED BITS
6102 031716 123727 031706 000354      CMPB   4$,#RING!CARIER!MDMRDY!CTS!TM ;CHK FOR BITS SET
6103 031724 001414                      BEQ    8$            ;BR IF ALL BITS SET

```

TEST 8 - MODEM CONTROL SIGNAL LOOPBACK TEST

```

6104 031726 012737 000354 002324      MOV    #RING!CARRIER!MDMRDY!CTS!TM,GDATA ;SET EXPECTED DATA
6105 031734 013737 031706 002326      MOV    4$,BDATA ;SET ACTUAL DATA
6106                                     ;REPORT "MODEM STATUS INCORRECT"
6107 031742      GEDF    EM83,ERR11
;
; "DEVICE FATAL" ERROR # 66
        TRAP    C$ERDF
        .WORD   66
        .WORD   EM83
        .WORD   ERR11
        031742 104455
        031744 000102
        031746 015262
        031750 020540
6108 031752      ESCAPE  SUB
        031752 104410      TRAP    C$ESCAPE
        031754 000106      .WORD   L10030
6109                                     ;DE-ASSERT RTS, CHK FOR CTS = 0
6110 031756 004537 003712      8$:   JSR    R5,WRITEI ;DE-ASSERT RTS
        VIAORB
        RTSND!TTLOOP
6111 031762 120000
6112 031764 000012
6113 031766 004537 003566      JSR    R5,READI ;READ MODEM STATUS
        VIAORA
6114 031772 120017
6115 031774 000000      10$:  .WORD   0
6116 031776 132737 000010 031774  BITB   #CTS,10$ ;CHK FOR CTS = 0
6117 032004 001406      BEQ    12$ ;BR IF YES
6118                                     ;REPORT CTS NOT CLEARED
6119 032006      GEDF    EM84,ERR14
;
; "DEVICE FATAL" ERROR # 67
        TRAP    C$ERDF
        .WORD   67
        .WORD   EM84
        .WORD   ERR14
        032006 104455
        032010 000103
        032012 015311
        032014 021146
6120 032016      ESCAPE  SUB
        032016 104410      TRAP    C$ESCAPE
        032020 000042      .WORD   L10030-
6121                                     ;ASSERT RTS, CHK FOR CTS = 1
6122 032022 004537 003712      12$:  JSR    R5,WRITEI ;ASSERT RTS
        VIAORB
        TTLOOP
6123 032026 120000
6124 032030 000002
6125 032032 004537 003566      JSR    R5,READI ;READ MODEM STATUS
        VIAORA
6126 032036 120017
6127 032040 000000      14$:  .WORD   0
6128 032042 132737 000010 032040  BITB   #CTS,14$ ;CHK FOR CTS = 1
6129 032050 001004      BNE    15$ ;BR IF YES
6130                                     ;REPORT CTS NOT SET
6131 032052      GEDF    EM85,ERR14
;
; "DEVICE FATAL" ERROR # 68
        TRAP    C$ERDF
        .WORD   68
        .WORD   EM85
        .WORD   ERR14
        032052 104455
        032054 000104
        032056 015326
        032060 021146
6132 032062      15$:
6133 032062      ENDSUB
;
; L10030:
        TRAP    C$ESUB
        032062 104403
6134                                     ;SEE IF BOARD IS M8053 WITH H3254 INSTALLED
6135 032064 005737 002470      TST    BRDTYP ;SEE IF M8053
6136 032070 001002      BNE    17$ ;BR IF YES
6137 032072 000137 032554      16$:  JMP    A1 ;SKIP THIS SECTION OF CODE
6138 032076 023727 002472 000000  17$:  CMP    TSTCON,#0 ;SEE IF H3254 INSTALLED
6139 032104 001372      BNE    16$ ;BR IF NOT, TO SKIP CODE

```

TEST 8 - MODEM CONTROL SIGNAL LOOPBACK TEST

```

6140 032106          BGNSUB
      032106          T8.2: TRAP C$BSUB
      032106 104402
6141          ;INIT DMV, (TTL LOOPBACK IS CLEARED), CHK MODEM STATUS
6142 032110 004737 003352 JSR PC,MSTCLR ;PERFORM MASTER CLEAR TO INIT DMV11
6143 032114 004537 003712 JSR R5,WRITEI ;SET PORT B FOR OUTPUT MODE
6144 032120 120002 VIADPB
6145 032122 000377 377
6146 032124 004537 003712 JSR R5,WRITEI ;SET PORT A FOR INPUT MODE
6147 032130 120003 VIADPA
6148 032132 000000 000
6149 032134 004537 003712 JSR R5,WRITEI ;DISABLE TTLOOP
6150 032140 120000 VIAOR#
6151 032142 000000 000
6152 032144 012737 000001 002336 MOV #1,REGNUM ;SET REG NO. FOR PRINTOUT
6153          ;-----
6154          ;CHK FOR RING (IF EIA), CARRIER, MODEM RDY, CTS = 1, TEST MODE, = 0
6155          ;-----
6156 032152 004537 003566 JSR R5,READI ;READ MODEM STATUS
6157 032156 120017 VIAORA
6158 032160 000000 18$: .WORD 0
6159
6160 032162 023727 002470 000001 CMP BRDYP,#1 ;IS V.35 THE SELECTED I/F ?
6161 032170 001013 BNE 21$ ; NO: BR TO DO CHECK WITH RING
6162          ;YES: REMOVE RING BEFORE CHECKING
6163 032172 142737 000223 032160 BICB #RING!SPEED!RCVDAT!UMAIN,18$ ;CLEAR UNNEEDED BITS
6164 032200 123727 032160 000150 CMPB 18$,#CARRIER!MDMRDY!CTS ;CHK FOR CORRECT STATUS
6165 032206 001427 BEQ 20$ ;BR IF STATUS CORRECT
6166 032210 012737 000150 002324 MOV #CARRIER!MDMRDY!CTS,GDATA ;SET EXPECTED DATA
6167 032216 000412 BR 19$
6168          ; DO CHECK WITH RING....
6169 032220 142737 000023 032160 21$: BICB #SPEED!RCVDAT!UMAIN,18$ ;CLEAR UNNEEDED BITS
6170 032226 123727 032160 000350 CMPB 18$,#RING!CARRIER!MDMRDY!CTS ;CHK FOR CORRECT STATUS
6171 032234 001414 BEQ 20$ ;BR IF STATUS CORRECT
6172 032236 012737 000350 002324 MOV #RING!CARRIER!MDMRDY!CTS,GDATA ;SET EXPECTED DATA
6173
6174          ;REPORT "MODEM STATUS INCORRECT"
6175 032244 013737 032160 002326 19$: MOV 18$,BDATA ;SET ACTUAL DATA
6176 032252 GEDF EM83,ERR11
          ; "DEVICE FATAL" ERROR # 69
          TRAP C$ERDF
          .WORD 69
          .WORD EM83
          .WORD ERR11
6177 032262          ESCAPE SUB
          TRAP C$ESCAPE
          .WORD L10031
6178
6179 032266 004537 003712 ;DE-ASSERT RTS, CHK FOR CTS,CARRIER = 0
6180 032272 120000 20$: JSR R5,WRITEI ;DE-ASSERT RTS
6181 032274 000010 VIAORB
6182 032276 004537 003566 RTSND
6183 032302 120017 JSR R5,READI ;READ MODEM STATUS
6184 032304 000000 VIAORA
6185 032306 132737 000010 032304 22$: .WORD 0
6186 032314 001406 BITB #CTS,22$ ;CHK FOR CTS = 0
6187          BEQ 24$ ;BR IF YES
          ;REPORT CTS NOT CLEARED

```

TEST 8 MODEM CONTROL SIGNAL LOOPBACK TEST

```

6188 032316          GEDF   EM84,ERR14          ;
; "DEVICE FATAL " ERROR # 70
; TRAP   C$ERDF
; .WORD  70
; .WORD  EM84
; .WORD  ERR14
6189 032326          ESCAPE SUB
; TRAP   C$ESCAPE
; .WORD  L10031 .
6190 032332 104455 000100 032304 24$: BITB   #CARRIER,22$ ;CHK FOR CARRIER = 0
6191 032340 000106          BEQ     26$ ;BR IF YES
6192          ;REPORT CARRIER NOT CLEARED
6193 032342          GEDF   EM86,ERR14          ;
; "DFVICE FATAL " ERROR # 71
; TRAP   C$ERDF
; .WORD  71
; .WORD  EM86
; .WORD  ERR14
6194 032352          ESCAPE SUB
; TRAP   C$ESCAPE
; .WORD  L10031 .
6195          ;ASSERT RTS, CHK FOR CTS,CARRIER = 1
6196 032356 004537 003712 26$: JSR    R5,WRITEI ;ASSERT RTS
6197 032362 120000          VIAORB
6198 032364 000000          OOO
6199 032366 004537 003566 JSR    R5,READI ;READ MODEM STATUS
6200 032372 120017          VIAORA
6201 032374 000000          .WORD  0
6202 032376 132737 000010 032374 28$: BITB   #CTS,28$ ;CHK FOR CTS = 1
6203 032404 001006          BNE    30$ ;BR IF YES
6204          ;REPORT CTS NOT SET
6205 032406          GEDF   EM85,ERR14          ;
; DEVICE FATAL ERROR # 72
; TRAP   C$ERDF
; .WORD  72
; .WORD  EM85
; .WORD  ERR14
6206 032416          ESCAPE SUB
; TRAP   C$ESCAPE
; .WORD  L10031 .
6207 032422 104410 000100 032374 30$: BITB   #CARRIER,28$ ;CHK FOR CARRIER = 1
6208 032430 001006          BNE    32$ ;BR IF YES
6209          ;REPORT CARRIER NOT SET
6210 032432          GEDF   EM87,ERR14          ;
; "DEVICE FATAL " ERROR # 73
; TRAP   C$ERDF
; .WORD  73
; .WORD  EM87
; .WORD  ERR14
6211 032442          ESCAPE SUB
; TRAP   C$ESCAPE
; .WORD  L10031 .
6212          ;DE ASSERT DTR, CHK FOR MODEM READY = 0
6213 032446 004537 003712 32$: JSR    R5,WRITEI ;DE ASSERT DTR
6214 032452 120000          VIAORB
6215 032454 000020          DTR
6216 032456 004537 003566 JSR    R5,READI ;READ MODEM STATUS

```

TEST 8 MODEM CONTROL SIGNAL LOOPBACK TEST

```

6217 032462 120017
6218 032464 000000
6219 032466 132737 000040 032464
6220 032474 001406
6221
6222 032476
        032476 104455
        032500 000112
        032502 015403
        032504 021146
6223 032506          ESCAPE SUB
        032506 104410
        032510 000042
6224
6225 032512 004537 003712
6226 032516 120000
6227 032520 000000
6228 032522 004537 003566
6229 032526 120017
6230 032530 000000
6231 032532 132737 000040 032530
6232 032540 001004
6233
6234 032542
        032542 104455
        032544 000113
        032546 015426
        032550 021146
6235 032552
6236 032552
        032552
        032552 104403
6237 032554
6238 032554
        032554 104401

```

```

34$: VIAORA
      .WORD 0
      BITB @MDMRDY,34$ ;CHK FOR MODEM READY = 0
      BEQ 36$ ;BR IF YES
;REPORT MODEM READY NOT CLEARED
GEDF EM88,ERR14
; "DEVICE FATAL" ERROR # 74
TRAP C$ERDF
      .WORD 74
      .WORD EM88
      .WORD ERR14
ESCAPE SUB
TRAP C$ESCAPE
      .WORD L10031
;ASSERT DTR, CHK FOR MODEM READY = 1
36$: JSR R5,WRI*EI ;ASSERT DTR
      VIAORB
      000
      JSR R5,READI ;READ MODEM STATUS
      VIAORA
38$: .WORD 0
      BITB @MDMRDY,38$ ;CHK FOR MODEM READY = 1
      BNE 40$ ;BR IF YES
;REPORT MODEM READY NOT SET
GEDF EM89,ERR14
; "DEVICE FATAL" ERROR # 75
TRAP C$EPDF
      .WORD 75
      .WORD EM89
      .WORD ERR14
40$: ENDSUB
L10031: TRAP C$ESUB
A1:
ENDTST
L10027: TRAP C$ETST

```

TEST 9 DDCMP MESSAGE TEST

0.59

.SBTTL TEST 9 DDCMP MESSAGE TEST

```

*****
;
; TEST 9 - DDCMP MESSAGE TEST
;
; THIS TEST WILL USE XLB IF IT IS ENABLED -- OTHERWISE TTL LOOPBACK
; WILL BE UTILIZED. THIS ASSURES THAT IT CAN ALWAYS BE RUN AS A
; GENERAL "RINGOUT" OF THE M8053.
;
; INITIALIZATION: BCP MODE, CRC-16, IDLE = 0, SYNC (S/A?) = 226 OCT.
; (96 HEX.), RXCL & TXCL = 0 (CHAR. LENGTH = 8).
;
; THE FOLLOWING SAMPLE DDCMP MESSAGE IS TRANSMITTED & RECEIVED AND ALL
; DATA AND CRC CHARACTERS ARE CHECKED FOR ERRORS:
;
;          - HEADER - -----   --- DATA (PATTERN K) ---
; SYNC SYNC 201 000 075 003 002 001 CRC CRC 000 377 ... 252 000 CRC CRC
;
; THE ATTEMPT HERE IS TO PROVIDE A TEST JUST BELOW THE LEVEL OF THE
; FUNCTIONAL DIAGNOSTIC. THE USYRT WILL BE RESPONSIBLE FOR ALL CRC
; GENERATION AND VERIFICATION BUT THE CRC'S WILL ALSO BE VERIFIED BY
; SOFTWARE.
;
; *****

```

```

032556
6260 032556 012737 000003 002526
6261 032564 004737 005376
6262 032570 042737 001000 032632
6263 032576 023727 002470 000000
6264 032604 001407
6265 032606 023727 002472 000004
6266 032614 001403
6267 032616 052737 001000 032632
6268
6269 032624 004537 007234
6270 032630 065626
6271 032632 000000
6272 032634 103003
6273 032636
032636 104460
6274 032640
032640 104410
032642 001032
6275
6276 032644 004537 010250
6277 032650 000001
6278 032652 000007
6279 032654 004537 010250
6280 032660 000001
6281 032662 000010
6282 032664 004537 010250
6283 032670 000000
6284 032672 000000
6285 032674 004537 010136

```

```

;
; BGNTST
;
;          T9::
;          MOV #3,REGO ;INIT COUNT (TEXT TRANSMITTED 3 TIMES)
;          JSR PC,INIDMV ;INIT DMV-11, ENTER MAINT LOOP
;          BIC #NOLOOP,1$
;          CMP BRDTYP,#0 ;BOARD TYPE = M8064 ?
;          BEQ 2$ ; YES: SPECIFY TTLOOP (NOT XLB)
;          CMP TSTCON,#4 ;IS A LOOPBACK CONNECTOR/CABLE SPECIFIED ?
;          BEQ 2$ ;BR IF NO
;          BIS #NOLOOP,1$ ; YES: SPECIFY NO TTLOOP (INITRN)
;
;-----
;          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 L10032 .
;
;          JSR R5,TXCTRL ;SET TSOM, TX 1ST SYNCH
;          TSOM
;          7.
;          JSR R5,TXCTRL ;TX 2ND SYNCH
;          TSOM
;          8.
;          JSR R5,TXCTRL ;CLEAR TSOM
;          000
;          0
;          JSR R5,TXCHAR ;LOAD 201(HEADR1), TX 3RD SYNCH

```


TEST 9 - DDCMP MESSAGE TEST

6325									
6326	033052	004537	010350	JSR	R5,RXCHAR	;READ & CHK 201(HEADR1), RCV HEADR2			
6327	033036	000201			201				
6328	033040	000000			0				
6329	033042	000010			8.				
6330	033044	103003		BCC	.+8.	;BR IF NO ERROR			
6331	033046			ERROR		;REPORT STACKED ERROR			
	033046	104460					TRAP	C\$ERROR	
6332	033050			ESCAPE	TST	;SKIP TO END OF TEST			
	033050	104410					TRAP	C\$ESCAPE	
	033052	000622					.WORD	L10032 .	
6333									
6334	033054	004537	010136	JSR	R5, TXCHAR	;LOAD 001(HEADR6)			
6335	033060	000001			001				
6336	033062	000000			0				
6337	033064	103003		BCC	.+8.	;BR IF NO ERROR			
6338	033066			ERROR		;REPORT STACKED ERROR			
	033066	104460					TRAP	C\$ERROR	
6339	033070			ESCAPE	TST	;SKIP TO END OF TEST			
	033070	104410					TRAP	C\$ESCAPE	
	033072	000602					.WORD	L10032 .	
6340									
6341	033074	004537	010350	JSR	R5,RXCHAR	;READ & CHK 000(HEADR2), RCV HEADR3			
6342	033100	000000			000				
6343	033102	000000			0				
6344	033104	000010			8.				
6345	033106	103003		BCC	.+8.	;BR IF NO ERROR			
6346	033110			ERROR		;REPORT STACKED ERROR			
	033110	104460					TRAP	C\$ERROR	
6347	033112			ESCAPE	TST	;SKIP TO END OF TEST			
	033112	104410					TRAP	C\$ESCAPE	
	033114	000560					.WORD	L10032 .	
6348									
6349	033116	004537	010250	JSR	R5, TXCTRL	;SET TEOM			
6350	033122	000002		TEOM		;(STARTS CRC 16 CHARACTER)			
6351	033124	000000			0				
6352	033126	004537	010350	JSR	R5,RXCHAR	;READ & CHK 075(HEADR3), RCV HEADR4			
6353	033132	000075			075				
6354	033134	000000			0				
6355	033136	000010			8.				
6356	033140	103003		BCC	.+8.	;BR IF NO ERROR			
6357	033142			ERROR		;REPORT STACKED ERROR			
	033142	104460					TRAP	C\$ERROR	
6358	033144			ESCAPE	TST	;SKIP TO END OF TEST			
	033144	104410					TRAP	C\$ESCAPE	
	033146	000526					.WORD	L10032 .	
6359									
6360	033150	004537	010350	JSR	R5,RXCHAR	;READ & CHK 003(HEADR4), RCV HEADR5			
6361	033154	000003			003				
6362	033156	000000			0				
6363	033160	000010			8.				
6364	033162	103003		BCC	.+8.	;BR IF NO ERROR			
6365	033164			ERROR		;REPORT STACKED ERROR			
	033164	104460					TRAP	C\$ERROR	
6366	033166			ESCAPE	TST	;SKIP TO END OF TEST			
	033166	104410					TRAP	C\$ESCAPE	
	033170	000504					.WORD	L10032 .	

TEST 9 DDCMP MESSAGE TEST

```

6409 033326 004537 010350      JSR      R5,RXCHAR      ;READ & CHK 1ST CRC BYTE, RCV SECOND CRC BYTE
6410 033332 000043              043
6411 033334 000000              0
6412 033336 000010              8.
6413 033340 103003              BCC      .+8.          ;BR IF NO ERROR
6414 033342              ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
6415 033344              ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD    L10032 .
6416 033346 000326
6416 033350 004537 010136      JSR      R5, TXCHAR     ;LOAD 375(DATA4)
6417 033354 000375              375
6418 033356 000000              0
6419 033360 103003              BCC      .+8.          ;BR IF NO ERROR
6420 033362              ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
6421 033362 104460              ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD    L10032 .
6422 033364 104410
6423 033366 000306
6423 033370 004537 010350      JSR      R5,RXCHAR     ;READ & CHK SECOND CRC BYTE; RCV DATA1
6424 033374 000035              035
6425 033376 000000              0
6426 033400 000010              8.
6427 033402 103003              BCC      .+8.          ;BR IF NO ERROR
6428 033404              ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
6429 033404 104460              ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD    L10032 .
6430 033406 104410
6431 033410 000264
6432
6433
6434
-----
; TRANSMIT THE BULK OF DATA OUT OF TABLE "PATK"
-----
6435 033412 012702 003012      MOV      @PATK,R2      ;SET UP TABLE POINTER
6436 033416 112237 033454      5$:     MOVVB   (R2)+,20$ ;SET UP EXPECTED RX CHARACTER
6437 033422 116237 000003      MOVVB   3(R2),10$    ;SET UP TRANSMIT CHARACTER
6438
6439 033430 004537 010136      JSR      R5, TXCHAR     ;LOAD A CHARACTER
6440 033434 000000      10$:    000          ;** HOLE FOR NEXT TX CHARACTER
6441 033436 000000              0
6442 033440 103003              BCC      .+8.          ;BR IF NO ERROR
6443 033442              ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
6444 033444              ESCAPE TST          ;SKIP TO END OF TEST
                                TRAP      C$ESCAPE
                                .WORD    L10032 .
6445 033446 000226
6445 033450 004537 010350      JSR      R5,RXCHAR     ;CLK/RECEIVE/CHECK PREVIOUS CHARACTER
6446 033454 000000      20$:    000          ;** HOLE FOR EXPECTED CHARACTER
6447 033456 000000              0
6448 033460 000010              8.
6449 033462 103003              BCC      .+8.          ;BR IF NO ERROR
6450 033464              ERROR          ;REPORT STACKED ERROR
                                TRAP      C$ERROR
6451 033464 104460              ESCAPE TST          ;SKIP TO END OF TEST
6452 033466

```

TEST 9 DDCMP MESSAGE TEST

```

033466 104410
033470 000204                                TRAP      C$ESCAPE
                                              .WORD      L10032 .
6453
6454 033472 022702 003027                    CMP      #PATK+13.,R2 ;CHECK FOR 14TH CHARACTER OF TABLE
6455 033476 001347                            BNE      5$          ;BR IF NOT DONE
6456
6457 033500 004537 010250                    JSR      R5,TXCTRL  ;SET TEOM
6458 033504 000002                            TEOM
6459 033506 000000                            0
6460 033510 004537 010350                    JSR      R5,RXCHAR  ;READ/CHK 200(DATA13), RCV 125(DATA14)
6461 033514 000200                            200
6462 033516 000000                            0
6463 033520 000010                            8.
6464 033522 103003                            BCC      .+8.       ;BR IF NO ERROR
6465 033524                                ERROR          ;REPORT STACKED ERROR
033524 104460
6466 033526                                ESCAPE TST       ;SKIP TO END OF TEST                                TRAP      C$ERROR
033526 104410                                .WORD
033530 000144                                C$ESCAPE
                                              L10032-.
6467
6468 033532 004537 010250                    JSR      R5,TXCTRL  ;SET TEOM
6469 033536 000002                            TEOM
6470 033540 000000                            0
6471 033542 004537 010350                    JSR      R5,RXCHAR  ;READ/CHK 125(DATA14), RCV 252(DATA15)
6472 033546 000125                            125
6473 033550 000000                            0
6474 033552 000010                            8.
6475 033554 103003                            BCC      .+8.       ;BR IF NO ERROR
6476 033556                                ERROR          ;REPORT STACKED ERROR                                TRAP      C$ERROR
033556 104460                                .WORD
6477 033560                                ESCAPE TST       ;SKIP TO END OF TEST                                TRAP      C$ESCAPE
033560 104410                                .WORD
033562 000112                                C$ESCAPE
                                              L10032-.
6478
6479 033564 004537 010350                    JSR      R5,RXCHAR  ;READ/CHK 252(DATA15), RCV 000(DATA16)
6480 033570 000252                            252
6481 033572 000000                            0
6482 033574 000010                            8.
6483 033576 103003                            BCC      .+8.       ;BR IF NO ERROR
6484 033600                                ERROR          ;REPORT STACKED ERROR                                TRAP      C$ERROR
033600 104460                                .WORD
6485 033602                                ESCAPE TST       ;SKIP TO END OF TEST                                TRAP      C$ESCAPE
033602 104410                                .WORD
033604 000070                                C$ESCAPE
                                              L10032-.
6486
6487 033606 004537 010350                    JSR      R5,RXCHAR  ;READ/CHK 000(DATA16), RCV FIRST CRC BYTE
6488 033612 100000                            RXERR!000
6489 033614 000001                            RERCHK
6490 033616 000010                            8.
6491 033620 103003                            BCC      .+8.       ;BR IF NO ERROR
6492 033622                                ERROR          ;REPORT STACKED ERROR                                TRAP      C$ERROR
033622 104460                                .WORD
6493 033624                                ESCAPE TST       ;SKIP TO END OF TEST                                TRAP      C$ESCAPE
033624 104410                                .WORD
033626 000046                                C$ESCAPE
                                              L10032 .
6494
6495 033630 004537 010350                    JSR      R5,RXCHAR  ;READ & CHK 1ST CRC BYTE, RCV SECOND CRC BYTE

```

TEST 9 - DDCMP MESSAGE TEST

```

6496 033634 000231          231
6497 033636 000000          0
6498 033640 000010          8.
6499 033642 103003          BCC      .+8.      ;BR IF NO ERROR
6500 033644          ERROR      ;REPORT STACKED ERROR
        033644 104460          TRAP      C$ERROR
6501 033646          ESCAPE TST      ;SKIP TO END OF TEST
        033646 104410          TRAP      C$ESCAPE
        033650 000024          .WORD    L10032 .
6502
6503 033652 004537 010350    JSR      R5,RXCHAR ;READ & CHK 2ND CRC BYTE, RCV 1ST SYNCH
6504 033656 000176          176
6505 033660 000000          0
6506 033662 000010          8.
6507 033664 103003          BCC      .+8.      ;BR IF NO ERROR
6508 033666          ERROR      ;REPORT STACKED ERROR
        033666 104460          TRAP      C$ERROR
6509 033670          ESCAPE TST      ;SKIP TO END OF TEST
        033670 104410          TRAP      C$ESCAPE
        033672 000002          .WORD    L10032 .
6510 033674          ENDTST
        033674          L10032:
        033674 104401          TRAP      C$ETST

```

HARDWARE PARAMETER CODING SECTION

.SBTTL HARDWARE PARAMETER CODING SECTION

6512
6513
6514
6515
6516
6517
6518
6519
6520
6521
6522
6523
6524

:/ 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.

6525 033676 BGNHRD .WORD L10033-L\$HARD/2
033676 000027 L\$HARD: :
033700

6526 033700 GPRMA ADDRES,0,0,160020,177776,YES .WORD T\$CODE
6527 033700 000031 .WORD ADDRES
033702 033756 .WORD T\$LLOLIM
033704 160020 .WORD T\$HILIM
033706 177776

6528 033710 GPRMA VECTOR,2,0,0,674,YES .WORD T\$CODE
033710 001031 .WORD VECTOR
033712 034004 .WORD T\$LLOLIM
033714 000000 .WORD T\$HILIM
033716 000674

6529 033720 GPRMD PRIRTY,4,0,7000,4,7,YES .WORD T\$CODE
033720 002032 .WORD PRIRTY
033722 034035 .WORD 7000
033724 007000 .WORD T\$LLOLIM
033726 000004 .WORD T\$HILIM
033730 000007

6530 033732 GPRMD BDTY,M,12,0,7,0,2,YES .WORD T\$CODE
033732 005032 .WORD BDTY,M
033734 034066 .WORD 7
033736 000007 .WORD T\$LLOLIM
033740 000000 .WORD T\$HILIM
033742 000002

6531 033744 GPRMD TCON,M,14,0,7,0,4,YES .WORD T\$CODE
033744 006032 .WORD TCON,M
033746 034151 .WORD 7
033750 000007 .WORD T\$LLOLIM
033752 000000 .WORD T\$HILIM
033754 000004

6532 033756 ENDHRD .EVEN
6533 033756 L10033:

6534
6535 .NLIST BEX
6536 033756 104 105 126 ADDRES: .ASCIZ /DEVICE CSR ADDRESS : /
6537 034004 104 105 126 VECTOR: .ASCIZ /DEVICE VECTOR ADDRESS : /
6538 034035 104 105 126 PRIRTY: .ASCIZ /DEVICE PRIORITY LEVEL : /
6539 034066 102 117 101 BDTY,M: .ASCIZ /BOARD TYPE (0=M8064, 1=M8053 V.35, 2=M8053 EIA) : /
6540 034151 124 125 122 TCON,M: .ASCII /TURNAROUND CCONNECTOR TYPE /<15><12>
6541 034206 050 060 075 .ASCII /(0=H3254&H3255, 1=INTEGRAL MODEM CABLE, 2=EIA CABLE, /<15><12>

L13

HARDWARE PARAMETER CODING SECTION

6542	034274	040	063	075	.ASCIZ / 3-V.35 CABLE, 4-NONE) : /
6543				.LIST	BEX
6544				.EVEN	

SOFTWARE PARAMETER CODING SECTION

```

6546          .SBTTL  SOFTWARE PARAMETER CODING SECTION
6547
6548
6549          ;////////////////////////////////////
6550          ;// THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
6551          ;// THAT ARE USED BY THE SUPERVISOR TO BUILD P TABLES.  THE
6552          ;// MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
6553          ;// INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
6554          ;// MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
6555          ;// WITH THE OPERATOR.
6556          ;////////////////////////////////////
6557
6558 034326          BGNSFT
6559          034326 000000
6560          034330          L$SOFT:: .WORD L10034 L$SOFT/2
6561          034330          ENDSFT
6562          034330          L10034: .EVEN

```

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

6562
6563
6564 034330
6565 034530 034530
6566 034530 000240
6567 034532 000240
6568 034534 000240
6569
6570
6571
6572
6573 034536
6574
6575
6576 034536

034536 000000
034540 000000
034542
6577
6578 000001

.SBTTL ***** PATCH AREA FOR DEBUG *****
PATCH:
 .=.+200
 NOP
 NOP
 NOP
;*****
.SBTTL "ENDMOD" STATEMENT
 ENDMOD
.SBTTL "LASTAD" STATEMENT & END OF PROGRAM
 LASTAD

L\$LAST::
.END

.EVEN
.WORD 0
.WORD 0

SYMBOL TABLE

ABC	000160	BSR0	002202	C#ESEG	000005	EM34	014502	EXADD	000020
ADDRES	033756	BSR1	002204	C#ESUB	000003	EM35	014530	EXCON	000010
ADR	000020 G	BSR10	002222	C#ETST	000001	EM36	014551	EXECUT	000005
AD.HIT	023242	BSR11	002224	C#EXIT	000032	EM39	014566	E#END	002100
AD.OK	023236	BSR12	002226	C#GETB	000026	EM4	014307	E#LOAD	000035
APA	000200	BSR13	002230	C#GETW	000027	EM40	014610	FLGCA1	000002
APAD	100000	BSR14	002232	C#GMAN	000043	EM54	014626	FLGCA2	000001
ASSEMB	000010	BSR15	002234	C#GPHR	000042	EM68	014650	FLGCB1	000020
A1	032554	BSR16	002236	C#GPLD	000030	EM69	014677	FLGCB2	000010
BADDAT	002366	BSR17	002240	C#GPRI	000040	EM70	014715	FLGIRQ	000200
BADATA	002326	BSR2	002206	C#INIT	000011	EM71	014737	FLGSR	000004
BDRATE	002474	BSR3	002210	C#INLP	000020	EM72	014755	FLGT1	000100
BDTY.M	034066	BSR4	002212	C#MANI	000050	EM73	014777	FLGT2	000040
BIT0	000001 G	BSR5	002214	C#MEM	000031	EM74	015014	FMT10	012560
BIT00	000001 G	BSR6	002216	C#MSG	000023	EM75	015035	FMT10A	012634
BIT01	000002 G	BSR7	002220	C#OPEN	000034	EM76	015051	FMT11	012655
BIT02	000004 G	CARRIER	000100	C#PNTB	000014	EM77	015071	FMT12	012674
BIT03	000010 G	CA1CTL	000001	C#PNTF	000017	EM78	015105	FMT13	012703
BIT04	000020 G	CA2CTL	000016	C#PNTS	000016	EM79	015125	FMT14	012751
BIT05	000040 G	CB1CTL	000020	C#PNTX	000015	EM80	015161	FMT15	012766
BIT06	000100 G	CB2CTL	000340	C#QIO	000377	EM81	015201	FMT15A	013020
BIT07	000200 G	CHKTSO	007074	C#RDBU	000007	EM82	015225	FMT16	013072
BIT08	000400 G	CHPTYP	002400	C#REFG	000047	EM83	015262	FMT16A	013115
BIT09	001000 G	CKLRBK	007532	C#RESE	000033	EM84	015311	FMT17	013157
BIT1	000002 G	CKRACT	005654	C#REVI	000003	EM85	015326	FMT17A	013236
BIT10	002000 G	CKRDA	006154	C#RFLA	000021	EM86	015342	FMT17B	013317
BIT11	004000 G	CKROR	006454	C#RPT	000025	EM87	015363	FMT17C	013372
BIT12	010000 G	CKRSA	006314	C#SEFG	000046	EM88	015403	FMT19	013452
BIT13	020000 G	CKSEOM	006614	C#SPRI	000041	EM89	015426	FMT2	012126
BIT14	040000 G	CKTACT	005514	C#SVEC	000037	EM90	015450	FMT21	013503
BIT15	100000 G	CKTBMT	006014	C#TPRI	000013	EM91	015501	FMT22	013513
BIT2	000004 G	CKUSTS	005410	DDCMP	040000	ENDCOD	026263	FMT23	013535
BIT3	000010 G	CRCOS	000400	DEVMAP	002404	ENDEMB	012116	FMT24	013614
BIT4	000020 G	CRC16	001400	DEVPTX	002406	ENDIT	023122	FMT25	013627
BIT5	000040 G	CTS	000010	DFPTBL	002150 G	ENDPAT	003064	FMT26	013657
BIT6	000100 G	C#AU	000052	DIAGMC	000000	ENTRAN	011772	FMT27	013712
BIT7	000200 G	C#AUTO	000061	DOTBMT	000007	EPATX	002740	FMT28	013721
BIT8	000400 G	C#BRK	000022	DTR	000020	ERRBLK	002200 G	FMT29	013755
BIT9	001000 G	C#BSEG	000004	DTRL	000000	ERRFLG	002350	FMT3	012163
BOE	000400 G	C#BSUB	000002	D.BUG	000000	ERRMSG	002176 G	FMT30	013762
BRDTPP	002470	C#CEFG	000045	EF.CON	000036 G	ERRNBR	002174 G	FMT31	014017
BSELO	002416	C#CLCK	000062	EF.NEW	000035 G	ERROR1	002376	FMT32	014065
BSEL1	002420	C#CLEA	000012	EF.PWR	000034 G	ERRTPP	002172 G	FMT39	014117
BSEL10	002436	C#CLOS	000035	EF.RES	000037 G	ERR10	020364 G	FMT4	012247
BSEL11	002440	C#CLP1	000006	EF.STA	000040 G	ERR11	020540 G	FMT4A	012305
BSEL12	002442	C#CVEC	000036	EIAV35	000002	ERR11A	022032	FMT4B	012340
BSEL13	002444	C#DCLN	000044	EM100	015536	ERR12	020714 G	FMT4C	012345
BSEL14	002446	C#DODU	000051	EM101	015556	ERR12A	022364	FMT40	014150
BSEL15	002450	C#DRPT	000024	EM14	014324	ERR13	021030 G	FMT5	012400
BSEL16	002452	C#DU	000053	EM16	014340	ERR14	021146 G	FMT5A	012443
BSEL17	002454	C#EDIT	000003	EM2	014214	ERR3	020072 G	FMT7	012530
BSEL2	002422	C#ERDF	000055	EM25	014360	ERR4	020120 G	FRSTIM	002370
BSEL3	002424	C#ERHR	000056	EM28	014406	ERR4A	021306	F#AU	000015
BSEL4	002426	C#ERRO	000060	EM29	014427	ERR5A	021634	F#AUTO	000020
BSEL5	002430	C#ERSF	000054	EM3	014263	ERR7A	020244 G	F#BGN	000040
BSEL6	002432	C#ERSO	000057	EM30	014444	EVL	000004 G	F#CLEA	000007
BSEL7	002434	C#ESCA	000010	EM31	014465	EVRC	002400	F#DU	000016

SYMBOL TABLE

F\$END	=	000041	INTGRL	=	000001	L\$HIME	002120	G	MLWRI	003722	PSTACK	002340		
F\$HARD	=	000004	INTSC	=	000200	L\$HPCP	002016	G	MPCSR	002416	RABGA	=	000004	
F\$HW	=	000013	ISR	=	000100	L\$HPTP	002022	G	MPIVEC	002456	RAMADR	=	001000	
F\$INIT	=	000006	IXL	=	004000	L\$HW	002150	G	MPOVEC	002460	RCVBUF	=	003064	
F\$JMP	=	000050	I\$AU	=	000041	L\$ICP	002104	G	MPRIOR	002462	RCVDAT	=	000002	
F\$MOD	=	000000	I\$AUTO	=	000041	L\$INIT	022554	G	MREDY	=	000200	RCV1ST	=	011624
F\$MSG	=	000011	I\$CLN	=	000041	L\$LADP	002026	G	MREQ	=	000001	RDA	=	000200
F\$PROT	=	000021	I\$DU	=	000041	L\$LAST	034542	G	MSTCLR	003352	RDSRM	=	120401	
F\$PWR	=	000017	I\$HRD	=	000041	L\$LOAD	002100	G	NCRACT	=	020000	RDSRL	=	120400
F\$RPT	=	000012	I\$INIT	=	000041	L\$LUP	002074	G	NCTBMT	=	000200	READ	=	003454
F\$SEG	=	000003	I\$MOD	=	000041	L\$MREV	002050	G	NEWLIN	012123	READI	=	003566	
F\$SOFT	=	000005	I\$MSG	=	000041	L\$NAME	002000	G	NEWST	022746	REDBYT	=	002356	
F\$SRV	=	000010	I\$PROT	=	000040	L\$PRIO	002042	G	NFCRDA	=	040000	REDDAT	=	002516
F\$SUB	=	000002	I\$PTAB	=	000041	L\$PROT	022546	G	NOCHK	=	003400	REDLOC	=	000001
F\$SW	=	000014	I\$PWR	=	000041	L\$PRT	002112	G	NOCRDA	=	100000	REDPAG	=	000003
F\$TEST	=	000001	I\$RPT	=	000041	L\$REPP	002062	G	NOLOOP	=	001000	REGNUM	=	002336
GDATA	=	002324	I\$SEG	=	000041	L\$REV	002010	G	NORXEN	=	040000	REG0	=	002526
GETBSR	=	004024	I\$SETU	=	000041	L\$SOFT	034330	G	NULCLK	=	000200	REG1	=	002530
GETPRM	=	022766	I\$SFT	=	000041	L\$SPC	002056	G	OVRC	=	002000	REG2	=	002532
GETURS	=	004360	I\$SRV	=	000041	L\$SPCP	002020	G	O\$APTS	=	000000	REG3	=	002534
GETVRS	=	004460	I\$SUB	=	000041	L\$SPTP	002024	G	O\$AU	=	000001	REG4	=	002536
GETWSR	=	004166	I\$TST	=	000041	L\$STA	002030	G	O\$BGNR	=	000000	REG5	=	002540
GOODAT	=	002364	J\$JMP	=	000167	L\$SW	002172	G	O\$BGNS	=	000000	REG6	=	002542
G\$CNT0	=	000200	LOADAT	=	002362	L\$TEST	002114	G	O\$DU	=	000001	REG7	=	002544
G\$DELM	=	000372	LOE	=	040000	L\$TIML	002014	G	O\$ERRT	=	000001	REOM	=	000002
G\$DISP	=	000003	LOGDEV	=	002334	L\$UNIT	002012	G	O\$GNSW	=	000000	RERCHK	=	000001
G\$EXCP	=	000400	LOT	=	000010	L10000	002170		O\$POIN	=	000001	RERR	=	000200
G\$HILI	=	000002	LUSWI1	=	002464	L10001	002172		O\$SETU	=	000000	RETADR	=	002354
G\$LOLI	=	000001	LUSWI2	=	002466	L10002	020116		PALENB	=	000001	RING	=	000200
G\$NO	=	000000	LU1MOD	=	002000	L10003	020242		PATCH	034330	ROR	=	000010	
G\$OFFS	=	000400	L\$ACP	=	002110	L10004	020362		PATE	002576	RSA	=	000020	
G\$OFFSI	=	000376	L\$APT	=	002036	L10005	020536		PATF	002606	R\$OM	=	000001	
G\$PRMA	=	000001	L\$AU	=	023256	L10006	020712		PATG	002616	RSTCHK	=	005104	
G\$PRMD	=	000002	L\$AUT	=	002070	L10007	021026		PATI	002741	RTSND	=	000010	
G\$PRML	=	000000	L\$AUTO	=	023124	L10010	021144		PATJ	003002	RUN	=	000200	
G\$RADA	=	000140	L\$CCP	=	002106	L10011	021260		PATK	003012	RXABGA	=	002000	
G\$RADB	=	000000	L\$CLEA	=	023250	L10013	023122		PATL	003033	RXACT	=	000040	
G\$RADD	=	000040	L\$CO	=	002032	L10014	023240		PATQ	003054	RXCHAR	=	010350	
G\$RADL	=	000120	L\$DEPO	=	002011	L10015	023250		PATX	002645	RXDL	=	000007	
G\$RADO	=	000020	L\$DESC	=	003304	L10016	023254		PBLENB	=	000002	RXEN	=	000100
G\$XFER	=	000004	L\$DESP	=	002076	L10017	023256		PCR	=	120407	RXEOM	=	001000
G\$YES	=	000010	L\$DEVP	=	002060	L10020	023556		PCSARM	=	120405	RXERR	=	100000
HDX	=	000004	L\$DISP	=	002124	L10021	025434		PCSARL	=	120404	RXOR	=	004000
HELP	=	000000	L\$DLY	=	002116	L10022	027140		PNT	=	001000	RXSOM	=	000400
HOE	=	100000	L\$DTP	=	002040	L10023	027702		PRESET	=	000001	SAVE4	=	002372
IBE	=	010000	L\$DTYP	=	002034	L10024	030444		PRI	=	002000	SAVE6	=	002374
IDLE	=	000010	L\$DU	=	023252	L10025	031126		PRIOR	002342	SAVLEN	=	002402	
IDLES	=	004000	L\$DUT	=	002072	L10026	031610		PRIPTY	034035	SCRACH	=	002332	
IDU	=	000040	L\$DVTY	=	003264	L10027	032554		PRI00	=	000000	SECAD	=	000020
IEI	=	000001	L\$EF	=	002052	L10030	032062		PRI01	=	000040	SECADR	=	010000
IEO	=	000020	L\$ENVI	=	002044	L10031	032552		PRI02	=	000100	SELO	=	002416
IER	=	020000	L\$ERRT	=	002172	L10032	033674		PRI03	=	000140	SEL10	=	002436
INIDMV	=	005376	L\$ETP	=	002102	L10033	033756		PRI04	=	000200	SEL12	=	002442
INITRN	=	007234	L\$EXP1	=	002046	L10034	034330		PRI05	=	000240	SEL14	=	002446
INITT1	=	004534	L\$EXP4	=	002064	MCLR	=	000100	PRI06	=	000300	SEL16	=	002452
INITT2	=	004734	L\$EXP5	=	002066	MCODE	=	025436	PRI07	=	000340	SEL2	=	002422
INTFLG	=	002346	L\$HARD	=	033700	MDMRDr	=	000040	PROTO	=	000100	SEL4	=	002426

SYMBOL TABLE

SEL6	002432	TTLOOP=	000002	TXTVRT	017766	T\$NEST=	177777	UREGS	002242
SETVIA	005304	TxAB	= 002000	TXTVRO	017371	T\$NS0	= 000000	USTATR=	122000
SFPTBL	002172 G	TXACT	= 000004	TXTVR1	017375	T\$NS1	= 000005	USYREG	002476
SFR	= 000001	TXCHAR	010136	TXTVR2	017401	T\$NS2	= 000002	USYRT	= 120400
SPEED	= 000020	TXCTRL	010250	TXTVR3	017406	T\$PTNU=	000000	VECTOR	034004
SRMODE	= 000034	TXDL	= 000340	TXTVR4	017413	T\$SAVL=	177777	VIA	= 120000
STALL	004356	TXEN	= 000040	TXTVR5	017420	T\$SEGL=	177777	VIAACR=	120013
STARES	002412	TXEOM	= 001000	TXTVR6	017425	T\$SUBN=	000000	VIADPA=	120003
STARST	022736	TXERR	= 100000	TXTVR7	017432	T\$TAGL=	177777	VIADPB=	120002
STEPLU	012072	TXGA	= 004000	TXTVR8	017437	T\$TAGN=	010035	VIAIER=	120016
STRIP	= 000040	TXSOM	= 000400	TXTVR9	017444	T\$TEMP=	000000	VIAIFR=	120015
STRIPS=	020000	TXTMLT	017744	TXT1	015602	T\$TEST=	000011	VIAMS	= 120001
STRTML=	000301	TXTML0	017154	TXT10	016402	T\$TSTM=	177777	VIAORA=	120017
STUREG	004250	TXTML1	017160	TXT11	016422	T\$TSTS=	000001	VIAORB=	120000
SUBRPC	002344	TXTML2	017174	TXT11A	016474	T\$AU	= 010017	VIAPCR=	120014
SVCGBL=	000000	TXTML3	017211	TXT11B	016532	T\$AUT	= 010014	VIASR	= 120012
SVCINS=	000001	TXTML4	017233	TXT12	016602	T\$CLE	= 010015	VIAT1A=	120004
SVCSUB=	000001	TXTML5	017254	TXT13	016630	T\$DU	= 010016	VIAT1B=	120005
SVCTAG=	000001	TXTML6	017304	TXT14	016645	T\$HAR=	010033	VIAT1C=	120006
SVCTST=	000001	TXTML7	017316	TXT15	016703	T\$HW	= 010000	VIAT1D=	120007
SWPBOT=	121000	TXTNP	017500	TXT16	016745	T\$INI	= 010013	VIAT2A=	120010
SWPDDC=	121400	TXTNPT	020030	TXT17	016760	T\$MSG=	010011	VIAT2B=	120011
SYNCH	= 000226	TXTNPO	017505	TXT18	017015	T\$PRO=	010012	VREGS	002262
S\$LSYM=	010000	TXTNP1	017515	TXT19	017056	T\$SOF=	010034	WAIT50	005270
TAB	= 000004	TXTNP2	017525	TXT2	015640	T\$SUB=	010031	WRIBYT	002360
TBMT	= 000100	TXTNP3	017535	TXT2A	015702	T\$SW	= 010001	WRILOC=	000002
TCCHEK=	100000	TXTNP4	017552	TXT2B	015741	T\$TES=	010032	WRIPAG=	000004
TCON.M	034151	TXTNP5	017567	TXT20	017112	T.EDF	= 000001	WRITE	003700
TDATA	002322	TXTNP6	017604	TXT3	016004	T.EMRD=	000002	WRITEI	003712
TDSRH	= 120403	TXTNP7	017620	TXT4	016034	T.ESF	= 000000	WSRO	002202
TDSRL	= 120402	TXTNP8	017634	TXT4A	016074	T.ESFT=	000003	WSR10	002212
TDSRNR	002575	TXTNUL	017152	TXT5	016135	T1	023260 G	WSR12	002214
TEOM	= 000002	TXTUR	017650	TXT6	016137	T1MODE=	000300	WSR14	002216
TERR	= 000200	TXTURT	020052	TXT7	016162	T2	023560 G	WSR16	002220
TGA	= 000010	TXTURO	017663	TXT7A	016252	T2MODE=	000040	WSR2	002204
TIMFLG	002352	TXTUR1	017671	TXT8	016342	T3	026264 G	WSR4	002206
TM	= 000004	TXTUR2	017677	TXT9	016362	T4	027142 G	WSR6	002210
TMP0	002546	TXTUR3	017705	TXU	= 000002	T5	027704 G	XDATA	002330
TMP1	002550	TXTUR4	017713	T\$ARGC=	000005	T6	030446 G	XORGB	021262
TMP2	002552	TXTUR5	017722	T\$CODE=	006032	T7	031130 G	XYZ	= 000007
TMP3	002554	TXTUR6	017731	T\$ERRN=	000113	T8	031612 G	X\$ALWA=	000000
TMP4	002556	TXTUR7	017735	T\$EXCP=	000000	T8.1	031634	X\$FALS=	000040
TMP5	002560	TXTVR	017353	T\$FLAG=	000040	T8.2	032106	X\$OFFS=	000400
TMP6	002562	TXTVRA	017451	T\$GMAN=	000000	T9	032556 G	X\$TRUE=	000020
TMP7	002564	TXTVRB	017454	T\$HILI=	000004	UAM	= 000200 G	\$E	= 000113
TSO	= 000010	TXTVRC	017460	T\$LAST=	000001	UMAINT=	000001	\$LSTIN=	000001
TSOM	= 000001	TXTVRD	017464	T\$LOLI=	000000	UNIT	002410	\$LSTTA=	000001
TSTCON	002472	TXTVRE	017470	T\$LSYM=	010000	UPBITS	002566	\$T	= 000011
TSTNUM	002414	TXTVRF	017474	T\$LTNO=	000011				

. ABS. 034542 000
000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 31536 WORDS (124 PAGES)
DYNAMIC MEMORY: 19748 WORDS (75 PAGES)
ELAPSED TIME: 00:09:36
CNDMEA.BIC,CNDMEA.SEQ/CR/ -SP=SVC34.MLB/ML,CNDMEA.P11