

KD11-Z  
DL11-W

DL11-W/1144 MFM SLU  
CZDLDF0

AH-8529F-MC  
FICHE 1 OF 1

FEB 1981  
COPYRIGHT © 75-80  
MADE IN USA





1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37

.REM    -

IDENTIFICATION

PRODUCT CODE:    AC-8528F-MC  
PRODUCT NAME:    CZDLDF0 DL11-W/1144 MFM SLU  
DATE CREATED:    OCTOBER, 1980  
MAINTAINER:      DIAGNOSTIC ENGINEERING  
AUTHOR:          DAN CASALETTO  
REVISED BY:      DAN MILLEVILLE

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

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

COPYRIGHT (C) 1975, 1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95

### HISTORY SECTION

-----

CZDLDDO WAS RELEASED OCT 79

THE FOLLOWING CHANGES WERE MADE. ALL CHANGES ARE INDICATED BY ;\*\* IN THE COMMENT FIELD:

1. USE THE MFPT INSTRUCTION, AND IF THE CPU IS A 11/44:
  - A. DO NOT PERFORM READER ENABLE AND RECEIVER ACTIVE TESTS.
  - B. IF ERROR FLAGS TESTS AND BREAK TESTS ARE ENABLED, PERFORM THESE TESTS ONLY FOR THE SLU AT 17650G. PERFORM THESE TESTS FOR THE CONSOLE SLU IF BIT03 OF SWR IS ADDITIONALLY SET.
  - C. WHILE IN MAINTENANCE MODE DO NOT WRITE AND READ A \*P OR AN ASCII 220. THIS WILL FORCE THE CONSOLE INTO 'CONSOLE MODE'.
  - D. IN THE CLOCK REPEATABILITY TEST, ALLOW FOR A TOLERANCE OF 2 BETWEEN CLOCK COUNTS. THIS IS TO ALLOW ENOUGH TOLERANCE WHEN MOS MEMORY IS USED AND REFRESH CYCLES ARE OCCURRING.
2. BECAUSE 11/44 SLU INTERRUPT REQUESTS ARE DEPENDANT ON A MFM CLOCK ENOUGH TIME MUST BE GIVEN FOR THEM TO OCCUR. THEREFORE, ALL TESTS ASSOCIATED WITH XMIT & RECEIVE INTERRUPTS SHOULD HAVE A MINIMUM OF 4 NOP'S ACTING AS WAIT FOR INTERRUPT.
3. IT WAS FOUND THAT AN ATTACHED TUSB WOULD BE ACTIVATED BY THE DIAGNOSTIC, AND, AS A RESULT, CHARACTERS WOULD BE SENT TO THE RECEIVER BUFFER. THIS WOULD CAUSE SOME TESTS TO FAIL. THEREFORE TO INHIBIT ANY COMMUNICATION TO THE TUSB FROM THE DIAGNOSTIC:
  - A. ENABLE MAINTENANCE MODE IN ALL TESTS THAT WRITE TO THE XMITTER.
  - B. IN ALL TESTS THAT WITE TO XMITTER AND BEFORE LEAVING TEST: DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS THAT MIGHT BE IN THE PROCESS OF BEING XMITTED TO FINISH.
  - C. DISABLE MAINTENANCE MODE FOR PURPOSE OF ERROR PRINTING ONLY TO THE SLU WHICH THE TERMINAL IS ATTACHED.
4. IT WAS FOUND THAT UPON POW-UP AND WITH THE TUSB ATTACHED, CHARACTERS WOULD BE SENT TO THE RECEIVER FROM THE TUSB. SOME RECEIVER TESTS WOULD FAIL DUE TO THIS. THEREFORE, ON ALL TESTS THAT PERFORM RECEIVER TESTS AND FOLLOWING MAINTENANCE MODE BEING ENABLED, DELAY ENOUGH TIME TO ALLOW TO ALLOW ANY CHARACTERS THAT MIGHT BE IN THE PROCESS OF BEING RECEIVED TO FINISH.
5. ADD SOFTWARE THAT WILL IMPLEMENT AUTO INITIATION OF 11/44 T/A CONSOLE TEST VIA WRAP CABLE FROM TUSB TO CONSOLE

96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124

PORTS. IT IS SELECTED BY SWR BIT02 AND IS PERFORMED ONLY AFTER ALL SLUS ARE TESTED.

CZDLDE0 WAS RELEASED JUL 80

1. PROGRAM WAS CHANGED TO WORK WITH VECTORS GREATER THAN 377.

CZDLDF0 WAS RELEASED OCT 80

1. IN TESTS 17 THROUGH 22 THE CLOCK DONE FLAG CAN GET SET BETWEEN THE TIME THAT IT IS CLEARED (BIC) AND THE TIME THAT INTERRUPTING IS ENABLED (BIS). THE (BIC-BIS) COMBINATION WAS CHANGED TO A (MOV) INSTRUCTION TO SET INTERRUPT ENABLE AND CLEAR DONE FLAG ALL WITH ONE INSTRUCTION.
2. IN TEST 20, 3 (NOP)'S WERE ADDED BEFORE ERROR +47 TO ALLOW SUFFICIENT TIME BETWEEN CLOCK FLAG AND CLOCK INTERRUPT.
3. IN TEST 23, THE COUNT DIFFERENCE ALLOWED WAS CHANGED ON 11/44'S ONLY FROM 2 TO 3.
4. TEST 45 HAD SEVERAL PROBLEMS.
  - A. PROGRAM CAN KICK OUT OF DELAY LOOP PREMATURELY. A FIXED DELAY WAS ADDED.
  - B. ERROR +113 DID NOT PRINT. A RESET WAS ADDED IMMEDIATELY BEFORE THE ERROR CALL TO BREAK THE MAINTENANCE WRAPAROUND.
  - C. CLKCNT WAS NOT INITIALIZED.

126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182

1.0 GENERAL INFORMATION  
-----

1.1 ABSTRACT  
-----

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE  
THE FOLLOWING MODULES:

1. DL11-W SERIAL LINE/REAL TIME CLOCK INTERFACE
2. 11/44 MULTIFUNCTION MODULE

THE PROGRAM WILL RUN WITHOUT ANY SPECIAL TEST FIXTURES BY DEFAULT.  
HOWEVER, THE FOLLOWING OPTIONAL TESTING IS PROVIDED:

1. TEST TO VERIFY XMIT AND RECEIVE OF THE UARTS WITH  
A WRAP CABLE. THE UART UNDER TEST IS LOOPED BACK ON  
ITSELF. THIS IS SELECTED VIA OPERATIONAL SWITCH SETTING  
BIT 7, AND IS APPLICABLE TO THE DL11W AND 11/44 MFM.
2. AUTOMATIC INITIATION OF THE T/A CONSOLE TEST OF THE 11/44  
MFM. THE TUS8 PORTS ARE LOOPED TO THE CONSOLE PORTS  
THIS IS SELECTED VIA OPERATIONAL SWITCH SETTING BIT 2  
AND IS APPLICABLE TO 11/44 MFM ONLY.  
(SEE CKKFBA0 FOR T/A CONSOLE TEST EXPLANATION)

THIS DIAGNOSTIC OPERATES ON THE CONSOLE SERIAL LINE AND CLOCK INTERFACES  
AS WELL AS UP TO FIFTEEN(15) ADDITIONAL IDENTICALLY CONFIGURED  
SERIAL LINE INTERFACES. THE DEFAULT ADDRESSES ARE:

A. CONSOLE - 177560 SERIAL LINE  
177546 CLOCK

B. OTHER SERIAL LINE - 776500 FIRST SERIAL LINE ADDRESS  
OF 15 CONSECUTIVE SERIAL  
LINE ADDRESSES

THE PROGRAM IS DESIGNED TO RUN ON ANY PDP-11 WITH 8K OF MEMORY  
. IT CAN BE RUN UNDER XXDP,APT, AND ACT MONITORS,  
AND ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER,  
SOFTWARE SWITCH REGISTER = LOCATION 176

POWER FAILURE IS SUPPORTED FOR SYSTEMS WITH CORE MEMORY.

NOTE: THIS DIAGNOSTIC WITH THE SWR = 000020  
(CLOCK TESTS ONLY) SHOULD BE USED ON SWITCHLESS CPU'S  
TO TEST KW-11L LINE CLOCK MODULES.

1.2 SYSTEM REQUIREMENTS  
-----

1.2.1 EQUIPMENT  
-----

STANDARD 11 FAMILY (COMPUTER WITH A CONSOLE OUTPUT DEVICE

183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194

AND 8K OF MEMORY.

OPTIONAL:

1. LOOP CABLE FOR UART LOOP BACK ON ITSELF
2. WRAP CABLE FOR 11/44 MFM T/E TESTING  
(SEE DWG. #7016942 WRAP AROUND CABLE  
AND SECTION 5.0 OF THIS DOCUMENT)

1.2.2    STORAGE

THE PROGRAM USES 5K WORDS OF MEMORY

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  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252

### 1.3 ASSUMPTIONS

- 
- A. IF THE UNIT UNDER TEST (UUT) IS THE CONSOLE, THE PROGRAM WILL ASSUME THE REAL TIME CLOCK (RTC) IS ENABLED AND WILL TEST IT UNLESS THE TESTS ARE DISABLED BY BIT6 OF THE SWR.
- B. IF THE UUT IS NOT THE CONSOLE, THE RTC IS NOT TESTED FOR THAT DEVICE.
- C. FOR THE DL11-W:

THE PROGRAM WILL ASSUME THE ERROR FLAG BITS AND THE BREAK FUNCTION OF THE DL11-W ARE DISABLED AND WILL NOT TEST THESE FUNCTIONS UNLESS ENABLED BY BIT10 (FOR ERROR FLAGS) AND BIT8 (FOR BREAK) OF THE SWR. THE DEFAULT CHARACTER SIZE IS 8 BITS (SEE PARA 2.3.2).

FOR THE 11/44:

THE PROGRAM ASSUMES THAT THE ERROR FLAG BITS AND THE BREAK FUNCTION ARE DISABLED, AND WILL NOT TEST THESE FUNCTIONS HOWEVER, IF BIT 10 (FOR ERROR FLAGS) AND BIT 08 (FOR BREAK) OF SWR ARE SET, THEN ERROR FLAGS AND BREAK TESTS ARE PERFORMED FOR THE TUS8 SLU ONLY. IF BIT03 OF SWR IS ALSO SET THEN THESE TESTS WILL BE ENABLED FOR THE CONSOLE.

READER ENABLE AND RECEIVER ACTIVE TESTS ARE NOT PERFORMED SINCE THE 11/44 MFM DOES NOT IMPLEMENT THESE FUNCTIONS.

### 2.0 OPERATING INSTRUCTIONS

#### 2.1 LOADING PROCEDURE

-----

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.

#### 2.2 STARTING PROCEDURE

-----

LOAD THE SWITCH REGISTER WITH SETTING

NOTE: IF USING A CPU WITHOUT HARDWARE SWITCH REGISTER SOFTWARE SWITCH REGISTER LOCATION = 176.  
(FOR A 11/44 CPU USE MFM CONSOLE FOR DEPOSITING SWITCH REGISTER. TYPE ^P TO ENTER CONSOLE)

- A. START AT 200.  
AFTER CHECKING THE TRANSMITTER, THE PROGRAM WILL PRINT ITS IDENTIFICATION AND REPORT THE NUMBER OF DEVICES UNDER TEST (NUMBER IS OCTAL).  
'END PASS' IS PRINTED AFTER A FULL PASS HAS BEEN MADE ON ALL DEVICES UNDER TEST.
- B. START AT 204. \*\*\*\*\*NOTE\*\*\*\*\*  
THE 'ECHO' TEST WILL BE EXECUTED. AN '\*' IS PRINTED AT THE BEGINNING OF THE TEST. THE ECHO TEST READS A CHARACTER FROM

253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279

THE TERMINAL WRITES THAT CHARACTER TO THE TERMINAL AND  
REPORTS ANY ERROR FLAGS SET IN THE RECEIVER BUFFER.  
A CONTROL-C HALTS THE TEST AND PRINTS "STOP" AT THE TERMINAL  
CONTINUING RESTARTS THE ECHO TEST.

C. START AT 210.      \*\*\*\*\*NOTE\*\*\*\*\*  
THE TERMINAL OUTPUT TEST WILL BE EXECUTED. DEPRESSING ANY CHARACTER  
AT THE TERMINAL, HALTS THE TEST. CONTINUING RESTARTS THE TEST.  
THE TEST OUTPUTS 32 CHARACTERS ON A LINE AND REPEATS THE  
PATTERN EVERY THREE LINES. THE PATTERN IS AS FOLLOWS  
(OCTAL CODE 040 --> 377):

!'"\$%&'()*+,-./0123456789:;<=>?	(OCTAL CODE)
@ABCDEFGHIJKLMN0PQRSTUVWXYZ[\]^_	(040 --> 077)
'ABCDEFGHIJKLMN0PQRSTUVWXYZ	(100 --> 137)
	(140 --> 177) [LOWER CASE ALPHA]

THIS BOTTOM LINE COULD BE THE FOLLOWING IF THE TERMINAL  
DOES NOT HAVE LOWER CASE:

@ABCDEFGHIJKLMN0PQRSTUVWXYZ[\]      [UPPER CASE ALPHA]

\*\*\*\*\*NOTE\*\*\*\*\*

IF THE TESTING ON TERMINALS OTHER THAN THE CONSOLE IS DESIRED  
FOR TESTS B OR C, SEE SECTION 2.3.4. AND 2.3.5. OF THIS DOCUMENT.

::\*+  
::~\*+



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

## 2.3 OPERATING PROCEDURE

-----

### 2.3.1 OPERATIONAL SWITCH SETTINGS

-----

THE DIAGNOSTIC WILL CHECK FOR EXISTENCE OF SWITCH REGISTER AT 177570.

IF NO SWITCH REGISTER IS AVAILABLE THE PROGRAM WILL AUTOMATICALLY USE THE CONTENTS OF LOCATION 176 AS THE SOFTWARE SWITCH REGISTER. THE USER SHOULD SET THIS LOCATION BEFORE STARTING THE PROGRAM. IF A HARDWARE SWITCH IS AVAILABLE AND A SOFTWARE SWR(LOC. 176) IS DESIRED, LOAD ALL 1'S INTO LOCATION 177570. (ALL SWITCHES UP IF PHYSICAL SWITCHES ARE AVAILABLE)

BIT15	- HALT ON ERROR
BIT14	- LOOP ON PRESENT TEST
BIT13	- INHIBIT ERROR TIMEOUT
BIT12	- UNUSED
BIT11	- UNUSED
BIT10	- ENABLE ERROR FLAGS TESTS
BIT09	- LOOP ON ERROR
BIT08	- ENABLE BREAK FUNCTION TESTS
BIT07	- ENABLE DATA TEST THROUGH LOOP-BACK CONNECTOR
BIT06	- INHIBIT RTC TESTS (ALLOW ONLY SLU TESTS)
BIT05	- ALLOW MANUAL SETTING OF '\$DEVN' (DEVICE MAP)
BIT04	- INHIBIT SLU TESTS (ALLOW ONLY LINE CLOCK TESTS)
BIT03	- FOR 11/44 MFM: ENABLE BOTH 'BREAK TESTS' AND 'ERROR FLAG TESTS' FOR THE CONSOLE SLU. (THIS BIT IS VALID ONLY IF BIT10 OR BIT08 IS SET.)
BIT02	- 11/44 MFM OPTION: SELECT AUTO INITIATION OF T/A CONSOLE TEST VIA WRAP CABLE

FOR DL11-W:

IF THE SOFTWARE SWITCH REGISTER IS USED(LOC. 176) THEN BUILT INTO THE PROGRAM IS THE ABILITY TO DYNAMICALLY CHANGE THE CONTENTS OF SWREG DURING PROGRAM EXECUTION. BY STRIKING ^G (CNTRL G) ON CONSOLE TTY THE OPERATOR SETS A REQUEST FLAG TO CHANGE THE CONTENTS OF SWREG, WHICH IS PROCESSED IN KEY AREAS OF THE PROGRAM CODE (IE) ERROR ROUTINES, AFTER HALTS END OF PASS, AND OTHER APPLICABLE AREAS. BECAUSE THIS DIAGNOSTIC USES THE MAINTENANCE BIT OF THE SERIAL LINE, THE CONTROL-G SHOULD BE ISSUED DURING PROGRAM TYPEOUTS AT THAT TIME THE MAINTENANCE BIT IS SURE TO BE CLEAR.

IF A CONTROL-G IS DETECTED, UPON EXECUTION THE CONTENTS OF THE SWREG ARE DUMPED IN OCTAL ON THE CONSOLE TTY AND ANY CHANGES ARE REQUESTED

(IE) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

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

1. <CR> IF NO CHANGES ARE TO BE MADE
2. 6 DIGITS 0-7 TO REPRESENT IN OCTAL THE NEW SWITCH REGISTER VALUE ;LAST DIGIT FOLLOWED BY <CR>.
3. ^U TO ALLOW REENTERING VALUE IF ERROR IS COMMITTED KEYING IN SWREG VALUE.

FOR 11/44 CPU:

SINCE THE 11/44 HAS A HARDWARE SWITCH REGISTER LOADED BY THE MFM CONSOLE THEN THE DIAGNOSTIC SHOULD ALWAYS FIND EXISTENCE OF 177570. WHEN OPERATING ON THE 11/44 CPU, DYNAMIC CHANGING OF SWREG(177570) DURING PROGRAM EXECUTION CAN BE ACCOMPLISHED BY USING THE MFM CONSOLE. TYPING ^P<CR> ON THE CONSOLE TTY WILL ENTER THE CONSOLE. THIS IS CONSIDERED "CONSOLE MODE". TO EXAMINE SWREG TYPE ' E SW<CR> ' TO THE CONSOLE PROMPT. TO LOAD THE SWREG TYPE ' D SW DATA<CR> ' WHERE 'DATA' IS AN OCTAL NUMBER. IN ORDER FOR THE DIAGNOSTIC TO TYPE TO THE TTY IT IS NECESSARY TO HAVE THE 11/44 MFM IN 'PROGRAM I/O MODE'. THIS CAN BE ACCOMPLISHED BY TYPING ' C<CR> ' TO THE CONSOLE PROMPT. THEREFORE, WHEN CONSOLE USE IS COMPLETED, PLACE THE MFM IN 'PROGRAM I/O MODE'. BECAUSE THIS DIAGNOSTIC USES THE MAINTENANCE BIT, ^P WILL NOT BE ACKNOWLEDGED DURING THESE TESTS. THEREFORE, ISSUE ^P DURING PROGRAM TYPEOUTS. AT THIS TIME THE MAINTENANCE BIT WILL BE CLEARED.

384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440

### 2.3.2 SETTING BITS PER CHARACTER

THIS PROGRAM DEFAULTS TO TESTING 8 BITS PER CHARACTER. IF THE SERIAL LINE IS SET FOR 5-->7 BITS PER CHARACTER, SET THE MEMORY LOCATION '\$USWR' AS FOLLOWS:

CHAR. SIZE (# OF BITS)	'\$USWR' CONTENTS	
	(BINARY)	(OCTAL)
8	10000000	400
7	01000000	200
6	00100000	100
5	00010000	40

'\$USWR' IS USED IN THE DATA PATH TESTS TO LIMIT THE BINARY COUNT TEST PATTERN TO THE NUMBER OF BITS SELECTED ON THE SERIAL LINE.

### 2.3.3 RUNNING UNDER APT

THE APT MAILBOX IS LOCATED AT LOCATION 500, TO ALLOW ADDITIONAL SERIAL LINE VECTOR ASSIGNMENTS TO THE 400 AREA OF MEMORY.

FOR DL11W:

THE DEFAULT EXECUTION TIMES PROVIDED(\$TSTM,\$PASTM) ARE FOR EXECUTION WITH AN 11/34 PROCESSOR, CORE MEMORY, AND 110 BAUD.

FOR 11/44:

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS AND IN SECT. 2.4 ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE, 16K CORE MEMORY, AND 300 BAUD.

THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

1. E TABLE 'A' IS USED FOR APT DUMP MODE AND RUN TIME MODES.
  - A. TWO SLU'S ARE TESTED.(\$SWREG BIT05=1 AND \$DEVN=3)  
A SLU AT 177560(DEFAULT CONSOLE SLU) AND AT 176500 (BASE ADDRESS CODE).
  - B. THE ERROR FLAG AND BREAK TESTS ARE SELECTED FOR THE SLU AT 176500(TU58 SLU IN MFG.)  
(\$SWREG BIT 10 AND 8 =1)
  - C. THE 'ENABLE DATA TEST THROUGH LOOP-BACK CONNECTOR' TEST IS ENABLED TO ALLOW TEST 44 TO EXECUTE (\$SWREG BIT 7 =1).
2. E TABLE 'B' IS USED FOR APT QV. IT ACCOMPLISHES WHAT E TABLE 'A' DOES, WITHOUT THE ENABLE DATA TEST THROUGH THE LOOPBACK CONNECTOR, BUT ADDITIONALLY IT SUPPRESSES ALL TYPEOUTS TO THE TERMINAL (\$ENVN=240) AND SELECTS AUTO TESTING OF T/A CONSOLE TEST VIA WRAP CABLE (\$SWREG BIT02=1).

1ST PASS  
RUN TIME

LONGEST  
TEST TIME

ADDITIONAL  
RUN TIME

441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469

60

50

45

.....

E TABLES

.....

E-MODE/S-MODE  
(\$ENVM/\$ENV)

A  
200/000

B  
240/001

SWITCH REGISTER 1  
(\$SWREG)

002640

002404

SWITCH REGISTER 2  
CPU TYPE/OPTIONS

000400  
00/0000

000400  
00/0000

MEMORY MAP CODE 1

000/00000000

000/00000000

MEMORY MAP CODE 2

000/00000000

000/00000000

MEMORY MAP CODE 3

000/00000000

000/00000000

MEMORY MAP CODE 4

000/00000000

000/00000000

BUS PRIORITY/INTERRUPT 1

0000

0000

BUS PRIORITY/INTERRUPT 2

0000

0000

BUS ADDRESS CODE

176500

176500

DEVICE MAP CODE

000003

000003

(\$DEVN)

CTLR. SPECIFIC WORD 1

000000

000000

CTLR. SPECIFIC WORD 2

000000

000000





525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557

STARTING AT LOCATION 200 AND HAVING BITS OF SWR CLEAR, THE PROGRAM WILL SELF  
SIZE THE NUMBER OF DEVICES (STARTING AT THE BAS ADDRESS) AND  
STORE A BIT MAP AT '\$DEVN' (DEVICE MAP) TO INDICATE WHICH UNIT NUMBERS  
ARE PRESENT AND WILL BE TESTED:

```

-----
| UNIT | UNIT | .....| UNIT | UNIT | CONSOLE|
| 15  | 14  | .....| 2   | 1   |        |
-----

```

A BIT MAP CAN BE ENTERED AT '\$DEVN' PRIOR TO STARTING THE PROGRAM  
SETTING BITS OF THE SWR INHIBITS THE SELF-SIZING AND DEVICE MAP  
GENERATION, AND USES THE VALUE STORED BY THE OPERATOR.

EXAMPLE:

```

SWR = 000040                    [BINARY 0 000 000 000 100 000]
$BASE: 776500
$VECT1: 300

$DEVN: 13                    [BINARY - 0 000 000 000 001 011]

```

```

THE PROGRAM WILL TEST -
UNIT# 0 = CONSOLE
UNIT# 1 = 776500 ; 300
UNIT# 3 = 776520 ; 320

```

559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602

## 2.4 EXECUTION TIMES -

FOR DL11-W: (110 BAUD)  
LONGEST SUBTEST TIME = 50 SECONDS  
PASS TIME = 60 SECONDS  
ADDITIONAL DEVICES = 55 SECONDS/DEVICE

FOR 11/44: (300 BAUD)  
LONGEST SUBTEST TIME = 50 SECONDS  
PASS TIME = 60 SECONDS  
ADDITIONAL DEVICES = 55 SECONDS/DEVICE

## 3.0 ERROR REPORTING

IF A ROUTINE FAILS AND THE INHIBIT ERROR TIMEOUT (BIT13) OF THE SWR IS NOT SET, A PRINTOUT RESULTS IN THE FORM:

```
''(SOME ASCII MESSAGE)''  
TEST#  ERR PC  RCSR  [ANY APPLICABLE DAT HEADINGS]  
XXXXXX XXXXXX XXXXXX [ANY APPLICABLE DATA]
```

NOTE: 'RCSR' IS DEPENDENT ON THE FAILURE  
& THEREFORE COULD BE TCSR,RBUF,TBUF,OR LKS

WHERE 'XXXXXX' IS AN OCTAL NUMBER.  
THIS ERROR PRINTOUT OCCURS PROVIDED THE ERROR THAT EXISTS  
WOULD NOT HINDER THE TIMEOUT. IN CASES WHERE IT IS NOT POSSIBLE  
TO PRINT AN ERROR MESSAGE (I.E. FATAL CONSOLE TRANSMITTER  
FAILURES), A HALT OCCURS AND THE PC CAN BE EXAMINED BY THE OPERATOR  
TO FIND THE ERROR INFORMATION IN THE PROGRAM LISTING.

NOTE: FOR SOFTWARE SWITCH OPERATION, THE SWITCH REGISTER CAN  
BE CHANGED BY TYPING A CONTROL-G AT THE CONSOLE DURING ERROR PRINTOUTS.  
AFTER CONTINUING FROM THE ERROR HALT THE OLD SWR CONTENTS  
IS DISPLAYED AND THE NEW CONTENTS CAN BE ENTERED.  
IF ERROR HALTS ARE DISABLED, THE CONTROL-G RESPONSE OCCURS  
IMMEDIATELY FOLLOWING THE TIMEOUT.

604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640

4.0    SUBROUTINE ABSTRACTS  
-----

4.1    TRAPCATCHER  
-----

THIS IS A SERIES OF INSTRUCTIONS STARTING AT LOCATION 0 DESIGNED TO DETECT AND ISOLATE UNEXPECTED TRAPS TO THE TRAP AND INTERRUPT VECTOR AREA OF MEMORY.

EACH VECTOR ENTRANCE ADDRESS IS LOADED WITH THE ADDRESS OF THE NEXT LOCATION. THE NEXT LOCATION IS LOADED WITH A BREAK POINT TRAP (000003). THUS AN ILLEGAL TRAP OR INTERRUPT CAUSES A TRAP THROUGH THE BPT VECTOR(14) WHICH POINTS TO THE "CATCH" ROUTINE.

THE "CATCH" ROUTINE REPORTS THE PC THAT CAUSED THE ORIGINAL TRAP AND THE LOCATION OF THE TRAP VECTOR (IF UNDER APT, AN ERROR IS INDICATED TO APT). AFTER REPORTING THE ERROR THE PROGRAM HALTS. THE PROGRAM MUST BE RESTARTED AT THIS POINT.

4.2    WRPSW  
-----

THIS ROUTINE IS USED TO WRITE THE PSW BY POPPING VALUES FROM THE STACK. THIS METHOD IS USED TO BE COMPATIBLE WITH ALL 11 FAMILY PROCESSORS.

4.3    SCOPE  
-----

THIS ROUTINE CALL IS PLACED BETWEEN EACH SUBTEST. IT RECORDS THE STARTING ADDRESS OF EACH SUBTEST AS IT IS BEING ENTERED AND UPDATES THE TEST NUMBER. IF A SCOPE LOOP IS REQUESTED, IT WILL JUMP TO THE START OF THE SUBTEST AT WHICH THE SCOPE LOOP IS REQUESTED.



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  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698

## 4.4 ERROR

-----  
THIS ROUTINE CALL IS PLACED WHEREEVER AN ERROR REPORT IS DESIRED. THE LOWER BYTE OF THIS CALL IS USED AS THE ERROR NUMBER AND AS A POINTER INTO THE ERROR TABLE. THIS ROUTINE REPORTS ERRORS TO APT USING '\$APTYPE' AND TYPES ERROR REPORTS TO THE CONSOLE USING '\$ERRTYPE'.

## 4.5 \$POWER

-----  
THIS ROUTINE SAVES ALL GENERAL REGISTERS DURING POWER-DOWN AND RESTORES THEM AT POWER-UP. IF A POWER FAILURE OCCURS 'POWER' IS PRINTED AT THE CONSOLE AFTER POWER IS RESTORED.

## 4.6 CKSWR

-----  
THIS ROUTINE CALL IS USED TO DETECT THE RECEPTION OF A CONTROL-G FROM THE CONSOLE. THE CALL USES '\$READ' TO PERFORM THE CONTROL-G SEQUENCE OF DISPLAYING THE CONTENTS OF THE SOFTWARE SWITCH REGISTER AND THE ENTERING THE NEW CONTENTS FROM THE TERMINAL.

## 5.0 AUTOMATIC INITIATION OF 11/44 T/A CONSOLE TEST VIA WRAP CABLE

-----  
PURPOSE: THE T/E(OR T/A) CONSOLE TEST CAN BE IMPLEMENTED BY MANUALLY TYPING T/E(OR T/A) ON THE KEYBOARD OF THE TERMINAL WHEN IN CONSOLE MODE. IN ORDER TO IMPLEMENT THIS TEST IN AN AUTOMATIC WAY IN MANUFACTURING WHILE UNDER APT, THE USE OF A WRAP CABLE FROM THE TUSB TO THE CONSOLE CAN BE USED ALLOWING THIS DIAGNOSTIC TO ISSUE THE 'T/A' COMMAND TO THE CONSOLE. THE DIAGNOSTIC WILL ISSUE THE APPROPRIATE SEQUENCE OF COMMANDS TO THE CONSOLE VIA THE WRAP CABLE WHEN BIT02 OF SWR =1. THE DIAGNOSTIC WILL THEN MONITOR THE EXPECTED RESPONSE FROM THE CONSOLE VIA THE WRAP CABLE AND HALT IF THERE IS AN ERROR. THE T/A TEST IS DONE ONLY AFTER ALL SLUS ARE TESTED. IF T/A IS SUCCESSFUL 'END PASS' IS PRINTED AND THE DIAGNOSTIC STARTS AGAIN.

FIGURE 1 SHOWS THE PROPER WRAP CABLE SETUP AND REQUIREMENTS FOR AUTOMATICALLY INITIATING T/A FROM THE THE DIAGNOSTIC. NOTICE THAT THIS ARRANGEMENT ALLOWS FOR THE TERMINAL TO MONITOR ALL COMMUNICATION FROM THE CONSOLE OUTPUT DURING EXECUTION OF THE DIAGNOSTIC IN 'WRAP MODE'.

## TYPEOUT EXAMPLES

-----  
1. WITH THE CONFIGURATION OF FIGURE 1 AND 'WRAP MODE' SELECTED

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  
740

THE PROGRAM MAY BE LOADED BY APT. IF 'E TABLE B' OF SECTION 2.3.3 WERE USED WITH THE EXCEPTION OF ALLOWING TYPEOUTS(\$ENV=200) THE FOLLOWING WOULD BE SEEN ON THE LOCAL TERMINAL:

```
CZDLDF0 DL11-W/1144 MFM SLU
02 DEVICES UNDER TEST ^P
CONSOLE
>>>T/A
```

```
CONSOLE-TESTB
END PASS ^P
CONSOLE
>>>T/A
```

```
CONSOLE-TESTB
END PASS ^P
CONSOLE
>>>T/A
```

```
CONSOLE-TESTB
END PASS ^P .....ETC.
```

DESCRIPTION: REFERING TO THE ABOVE PRINTOUTS:

- A. THE DIAGNOSTIC IS LOADED WITH THE TITLE BEING PRINTED.
- B. TWO SLU DEVICES(CONSOLE,TU58) ARE SPECIFIED
- C. BOTH SLUS ARE TESTED COMPLETELY. AT THIS POINT THE THE DIAGNOSTIC ISSUES ^P TO THE WRAP CABLE. THE CONSOLE ENTERS CONSOLE MODE AND THE ^P TYPED ON THE TERMINAL IS THE MFM CONSOLE ECHO.
- D. THE CONSOLE PERFORMS ITS OWN SELF TEST BY TYPING 'CONSOLE' FOLLOWED BY >>>, THE CONSOLE PROMPT.
- E. THE DIAGNOSTIC THEN ISSUES T/A AND THE TERMINAL SHOWS THE CONSOLE ECHO OF THIS.
- F. THE MFM THEN PERFORMS THE T/A TEST SHOWN BY THE TERMINAL TYPING 'CONSOLE-TESTB'.THE DIAGNOSTIC LOOKS FOR THE 'B' IN THIS TYPEOUT,AND WHEN FOUND , CONSIDERS THE TEST A SUCCESS. THE DIAGNOSTIC WILL TYPE END PASS INDICATING A SUCCESSFUL PASS OF THE DIAGNOSTIC.
- G. THE DIAGNOSTIC CONTINUES WITH FURTHER PASSES OF THE DIAGNOSTIC.

742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760

2. IF 'E TABLE B' OF 2.3.3 WERE USED WITH TYPEOUTS SUPPRESSED (\$ENVM-240) AS STATED, THEN THE FOLLOWING TYPEOUTS WOULD BE NOTICED:

```
^P
CONSOLE
>>>T/A

CONSOLE-TESTB^P
CONSOLE
>>>T/A

CONSOLE-TESTB^P
CONSOLE
>>>T/A

CONSOLE-TESTB^P .....FTC.
```





822  
829  
833  
837  
841  
842  
849  
850  
856  
866  
878  
879  
880  
881  
882  
883  
884  
885

```

.MCALL .STYPE, .STYPOC, .STRAP
.MCALL .SETUP, STARS, PUSH, POP, SETUP, .EQUIV, .SERRTYP
.MCALL .$APTHDR, .$APTBL, .$ACT11, .SCOPE
.MCALL .SCMTAG, .SEOP, .SREAD
.MCALL .EQUAT
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
        ERROR=EMT
        SCOPE=IOT

;*MISCELLANEOUS DEFINITIONS
HT= 11          ;;CODE FOR HORIZONTAL TAB
LF= 12          ;;CODE FOR LINE FEED
CR= 15          ;;CODE FOR CARRIAGE RETURN
CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776     ;;PROCESSOR STATUS WORD
        PSW=PS
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570   ;;HARDWARE SWITCH REGISTER
DDISP= 177570  ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= X0         ;;GENERAL REGISTER
R1= X1         ;;GENERAL REGISTER
R2= X2         ;;GENERAL REGISTER
R3= X3         ;;GENERAL REGISTER
R4= X4         ;;GENERAL REGISTER
R5= X5         ;;GENERAL REGISTER
R6= X6         ;;GENERAL REGISTER
R7= X7         ;;GENERAL REGISTER
SP= X6         ;;STACK POINTER
PC= X7         ;;PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS
PR0= 0         ;;PRIORITY LEVEL 0
PR1= 40        ;;PRIORITY LEVEL 1
PR2= 100       ;;PRIORITY LEVEL 2
PR3= 140       ;;PRIORITY LEVEL 3
PR4= 200       ;;PRIORITY LEVEL 4
PR5= 240       ;;PRIORITY LEVEL 5
PR6= 300       ;;PRIORITY LEVEL 6
PR7= 340       ;;PRIORITY LEVEL 7

;*SWITCH REGISTER SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
    
```

001100  
104000  
000004  
  
000011  
000012  
000015  
000200  
177776  
177776  
177774  
177772  
177570  
177570  
  
000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000006  
000007  
  
000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340  
  
100000  
040000  
020000

```
010000      SW12= 10000
004000      SW11=  4000
002000      SW10=  2000
001000      SW09=  1000
000400      SW08=   400
000200      SW07=   200
000100      SW06=   100
000040      SW05=    40
000020      SW04=    20
000010      SW03=    10
000004      SW02=     4
000002      SW01=     2
000001      SW00=     1
001000      SW9=SW09
000400      SW8=SW08
000200      SW7=SW07
000100      SW6=SW06
000040      SW5=SW05
000020      SW4=SW04
000010      SW3=SW03
000004      SW2=SW02
000002      SW1=SW01
000001      SW0=SW00

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
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

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
000010      RESVEC= 10        ;; RESERVED AND ILLEGAL INSTRUCTIONS
000014      TBITVEC=14        ;; "T" BIT
000014      TRTVEC= 14        ;; TRACE TRAP
```

```
000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCC?E**
000024 PWRVEC= 24 ;:POWER FAIL
000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC=34 ;:TRAP TRAP
000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
000064 TPVEC= 64 ;:TTY PRINTER VECTOR
000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
886 000007 MFPT=7
887 176500 ABASE= 176500
888 000300 AVECT1= 300
889 000400 AUSWR= 400
890 000001 $TN= 1
891 161000 $SWR= 161000
892 000003 BPT= 000003 ;THIS IS THE COMMAND FOR A TRAP
893 ; THROUGH 14 (BPT TRAP)
894
895 000000 . = 0
896 ;:*****
897 ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A '.+2,BPT'
898 ;*SEQUENCE TO CATCH ILLEGAL TRAPS & INTERRUPTS
899 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
900
908
909 000014 . = 14 ;THE BPT TRAP VECTOR POINTS TO THE
910 000014 014412 .WORD CATCH ; ILLEGAL TRAP HANDLER 'CATCH'
911 000016 000340 .WORD 340
912
913 000042 . = 42
914 000042 000000 .WORD 0
915
916
918
919
920 000174 . = 174
921 000174 000000 DISPREG: .WORD 0
922 000176 000000 SWREG .WORD 0
923
924 . = 200
925 000200 000137 003046 JMP START ;DO INTERFACE TEST
926 000204 000137 017170 JMP ECHO ;DO ECHO TEST
927 000210 000137 017410 JMP OUTST ;DO OUTPUT TEST TO TERMINAL
```

929 000500  
930

000046 000500  
000046 000046  
000046 014274  
000052 000052  
000052 000000  
000052 000500

```
. = 500  
.SBTTL ACT11 HOOKS  
:*****  
:HOOKS REQUIRED BY ACT11  
  $SVPC= . ;SAVE PC  
  . = 46  
  $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP  
  . = 52  
  .WORD 0 ;:2)SET LOC.52 TO ZERO  
  . = $SVPC ;: RESTORE PC
```



932

000500  
000024 000024  
000024 000200  
000044 000044  
000044 000500  
000500  
000500 000000  
000502 001066  
000504 000050  
000506 000060  
000510 000055  
000512 000030

```
.SBTTL APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      .SX=      ;;SAVE CURRENT LOCATION
      =24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      200      ;;FOR APT START UP
      =44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR ;;POINT TO APT HEADER BLOCK
      =.SX     ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.
$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 50     ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 60     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 55     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

934

001000 001000  
001000 000000  
001002 000  
001003 000  
001004 000000  
001006 000000  
001010 000000  
001012 000000  
001014 000  
001015 001  
001016 000000  
001020 000000  
001022 000000  
001024 000000  
001026 000000  
001030 000000  
001032 000000  
001034 000  
001035 000  
001036 000000  
001040 177570  
001042 177570  
001044 177560  
001046 177562  
001050 177564  
001052 177566  
001054 000  
001055 002  
001056 012  
001057 000  
001060 000000  
001062 077  
001063 015  
001064 012

000

```
.SBTTL COMMON TAGS
:*****
:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
:*USED IN THE PROGRAM.
.=1000
SCMTAG:                ;;START OF COMMON TAGS
$STNM: .WORD 0          ;;CONTAINS THE TEST NUMBER
$ERFLG: .BYTE 0         ;;CONTAINS ERROR FLAG
$ICNT: .WORD 0          ;;CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD 0         ;;CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD 0         ;;CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD 0         ;;CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE 0         ;;CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE 1         ;;CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD 0         ;;CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0         ;;CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0         ;;CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT: .WORD 0         ;;CONTAINS 'GOOD' DATA
$BDDAT: .WORD 0         ;;CONTAINS 'BAD' DATA
                .WORD 0 ;;RESERVED--NOT TO BE USED
                .WORD 0
$AUTOB: .BYTE 0        ;;AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 0        ;;INTERRUPT MODE INDICATOR
                .WORD 0
SWR: .WORD DSWR        ;;ADDRESS OF SWITCH REGISTER
DISPLAY: .WORD DDISP  ;;ADDRESS OF DISPLAY REGISTER
$TKS: 177560           ;;TTY KBD STATUS
$TKB: 177562           ;;TTY KBD BUFFER
$TPS: 177564           ;;TTY PRINTER STATUS REG. ADDRESS
$TPB: 177566           ;;TTY PRINTER BUFFER REG. ADDRESS
$NULL: .BYTE 0         ;;CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE 2        ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE 12       ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
$TPFLG: .BYTE 0        ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
$ESCAPE: 0            ;;ESCAPE ON ERROR ADDRESS
$QUES: .ASCII /?/     ;;QUESTION MARK
$CRLF: .ASCII <15>    ;;CARRIAGE RETURN
$LF: .ASCII <12>      ;;LINE FEED
:*****
```

```
.SBTTL APT MAILBOX-ETABLE
:*****
```

```
.EVEN
$MAIL:                ;;APT MAILBOX
$MSGTY: .WORD AMSGTY   ;;MESSAGE TYPE CODE
$FATAL: .WORD AFATAL   ;;FATAL ERROR NUMBER
$TESTN: .WORD ATESTN   ;;TEST NUMBER
$PASS: .WORD APASS     ;;PASS COUNT
$DEVCT: .WORD ADEVCT   ;;DEVICE COUNT
$UNIT: .WORD AUNIT     ;;I/O UNIT NUMBER
$MSGAD: .WORD AMSGAD   ;;MESSAGE ADDRESS
$MSGLG: .WORD AMSGLG   ;;MESSAGE LENGTH
$ETABLE:              ;;APT ENVIRONMENT TABLE
$ENV: .BYTE AENV       ;;ENVIRONMENT BYTE
$ENVM: .BYTE AENVM     ;;ENVIRONMENT MODE BITS
$SWREG: .WORD ASWREG   ;;APT SWITCH REGISTER
```

001112 000400  
001114 000000

\$USWR: .WORD AUSWR  
\$CPUOP: .WORD ACPUOP

:::USER SWITCHES  
:::CPU TYPE,OPTIONS  
BITS 15-11=CPU TYPE  
11/04=01,11/05=02,11/20=03,11/40=04,11/45=05  
11/70=06,PDQ=07,Q=10

001116 000  
001117 000

\$MAMS1: .BYTE AMAMS1  
\$MTYP1: .BYTE AMTYP1

BIT 10=REAL TIME CLOCK  
BIT 9=FLOATING POINT PROCESSOR  
BIT 8=MEMORY MANAGEMENT  
:::HIGH ADDRESS,M.S. BYTE  
:::MEM. TYPE,BLK#1  
MEM.TYPE BYTE -- (HIGH BYTE)

001120 000000

\$MADR1: .WORD AMADR1

900 NSEC CORE=001  
300 NSEC BIPOLAR=002  
500 NSEC MOS=003  
:::HIGH ADDRESS,BLK#1  
MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE

001122 000  
001123 000  
001124 000000  
001126 000  
001127 000  
001130 000000  
001132 000  
001133 000  
001134 000000  
001136 000300  
001140 000000  
001142 176500  
001144 000000  
001146

\$MAMS2: .BYTE AMAMS2  
\$MTYP2: .BYTE AMTYP2  
\$MADR2: .WORD AMADR2  
\$MAMS3: .BYTE AMAMS3  
\$MTYP3: .BYTE AMTYP3  
\$MADR3: .WORD AMADR3  
\$MAMS4: .BYTE AMAMS4  
\$MTYP4: .BYTE AMTYP4  
\$MADR4: .WORD AMADR4  
\$VECT1: .WORD AVECT1  
\$VECT2: .WORD AVECT2  
\$BASE: .WORD ABASE  
\$DEVM: .WORD ADEVM  
\$ETEND:  
.MEXIT

:::HIGH ADDRESS,M.S. BYTE  
:::MEM.TYPE,BLK#2  
:::MEM.LAST ADDRESS,BLK#2  
:::HIGH ADDRESS,M.S.BYTE  
:::MEM.TYPE,BLK#3  
:::MEM.LAST ADDRESS,BLK#3  
:::HIGH ADDRESS,M.S.BYTE  
:::MEM.TYPE,BLK#4  
:::MEM.LAST ADDRESS,BLK#4  
:::INTERRUPT VECTOR#1,BUS PRIORITY#1  
:::INTERRUPT VECTOR#2BUS PRIORITY#2  
:::BASE ADDRESS OF EQUIPMENT UNDER TEST  
:::DEVICE MAP

```
.SBTTL ERROR POINTER TABLE
:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
:*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
:*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
:*      EM      ;;POINTS TO THE ERROR MESSAGE
:*      DH      ;;POINTS TO THE DATA HEADER
:*      DT      ;;POINTS TO THE DATA
:*      DF      ;;POINTS TO THE DATA FORMAT
```

935	001146				
936	001146	017474	EM1	;	'CAN NOT ACCESS TCSR'
937	001150	023655	DH1	;	'TEST# ERR PC TCSR'
938	001152	024434	DT1	;	\$TESTN,\$ERRPC,TCSR
939	001154	000000	0		
940					
941	001156	017520	EM2	;	'CAN NOT ACCESS TBUF'
942	001160	023702	DH2	;	'TEST# ERR PC TBUF'
943	001162	024444	DT2	;	\$TESTN,\$ERRPC,TBUF
944	001164	000000	0		
945					
946	001166	017544	EM3	;	'TCSR DONE NOT CLEARED WITH TBUF FULL'
947	001170	023655	DH1	;	'TEST# ERR PC TCSR'
948	001172	024434	DT1	;	\$TESTN,\$ERRPC,TCSR
949	001174	000000	0		
950					
951	001176	017611	EM4	;	'TCSR DONE NOT SET'
952	001200	023655	DH1	;	'TEST# ERR PC TCSR'
953	001202	024434	DT1	;	\$TESTN,\$ERRPC,TCSR
954	001204	000000	0		
955					
956	001206	017633	EM5	;	TCSR DONE NOT SET WITH RESET
957	001210	023655	DH1	;	'TEST# ERR PC TCSR'
958	001212	0244 4	DT1	;	\$TESTN,\$ERRPC,TCSR
959	001214	000000	0		
960					
961	001216	017670	EM6	;	'CAN NOT ACCESS RCSR'
962	001220	02372 7	DH6	;	'TEST# ERR PC RCSR'
963	001222	02445 4	DT6	;	\$TESTN,\$ERRPC,RCSR
964	001224	000000	0		

965	001226	017714	EM7	;'CAN NOT ACCESS RBUF''
966	001230	023754	DH7	;'TEST# ERR PC RBUF''
967	001232	024464	DT7	;\$TESTN,\$ERRPC,RBUF
968	001234	000000	0	
969				
970	001236	017740	EM10	;' CAN NOT ACCESS LKS''
971	001240	024001	DH10	;'TEST# ERR PC LKS''
972	001242	024474	DT10	;\$TESTN,\$ERRPC,LKS
973	001244	000000	0	
974				
975	001246	017763	EM11	;'BIT0 OF TCSR NOT CLEAR AFTER RESET''
976	001250	023655	DH1	;'TEST# ERR PC TCSR
977	001252	024434	DT1	;\$TESTN,\$ERRPC,TCSR
978	001254	000000	0	
979				
980	001256	020026	EM12	;'CAN NOT SET BIT0 OF TCSR''
981	001260	023655	DH1	;'TEST# ERR PC TCSR''
982	001262	024434	DT1	;\$TESTN,\$ERRPC,TCSR
983	001264	000000	0	
984				
985	001266	020057	EM13	;'CAN NOT CLEAR BIT0 OF TCSR''
986	001270	023655	DH1	;'TEST# ERR PC TCSR''
987	001272	024434	DT1	;\$TESTN,\$ERRPC,TCSR
988	001274	000000	0	
989				
990	001276	020112	EM14	;'RESET DID NOT CLEAR BIT0 OF TCSR''
991	001300	023655	DH1	;'TEST# ERR PC TCSR''
992	001302	024434	DT1	;\$TESTN,\$ERRPC,TCSR
993	001304	000000	0	
994				
995	001306	020153	EM15	;'BIT2 OF TCSR NOT CLEAR AFTER RESET''
996	001310	023655	DH1	;'TEST# ERR PC TCSR''
997	001312	024434	DT1	;\$TESTN,\$ERRPC,TCSR
998	001314	000000	0	
999				
1000	001316	020216	EM16	;'CAN NOT SET BIT2 OF TCSR''
1001	001320	023655	DH1	;'TEST# ERR PC TCSR''
1002	001322	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1003	001324	000000	0	
1004				
1005	001326	020247	EM17	;'CAN NOT CLEAR BIT2 OF TCSR''
1006	001330	023655	DH1	;'TEST# ERR PC TCSR''
1007	001332	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1008	001334	000000	0	

1009	001336	020302	EM20	;'RESET DID NOT CLEAR BIT2 OF TCSR''
1010	001340	023655	DH1	;'TEST# ERR PC TCSR''
1011	001342	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1012	001344	000000	0	
1013				
1014	001346	020343	EM21	;'BIT6 OF TCSR NOT CLEAR AFTER RESET2
1015	001350	023655	DH1	;'TEST# ERR PC TCSR''
1016	001352	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1017	001354	000000	0	
1018				
1019	001356	020406	EM22	;'XMIT INTERRUPT WITH PRIORITY 7''
1020	001360	023655	DH1	;'TEST# ERR PC TCSR''
1021	001362	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1022	001364	000000	0	
1023				
1024	001366	020443	EM23	;'CAN NOT SET BIT6 OF TCSR''
1025	001370	023655	DH1	;'TEST# ERR PC TCSR''
1026	001372	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1027	001374	000000	0	
1028				
1029	001376	020474	EM24	;'CAN NOT CLEAR BIT6 OF TCSR''
1030	001400	023655	DH1	;'TEST# ERR PC TCSR''
1031	001402	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1032	001404	000000	0	
1033				
1034	001406	020527	EM25	;'RESET DID NOT CLEAR BIT6 OF TCSR''
1035	001410	023655	DH1	;'TEST# ERR PC TCSR''
1036	001412	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1037	001414	000000	0	
1038				
1039	001416	020570	EM26	;'BIT6 OF RCSR NOT CLEAR AFTER RESET''
1040	001420	023727	DH6	;'TEST# ERR PC RCSR''
1041	001422	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1042	001424	000000	0	
1043				
1044	001426	020633	EM27	;'RCVR INTERRUPT WITH PRIORITY 7''
1045	001430	023727	DH6	;'TEST# ERR PC RCSR''
1046	001432	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1047	001434	000000	0	
1048				
1049	001436	020672	EM30	;'CAN NOT SET BIT6 OF RCSR''
1050	001440	023727	DH6	;'TEST# ERR PC RCSR''
1051	001442	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1052	001444	000000	0	

1053	001446	020723	EM31	:'CAN NOT CLEAR BIT6 OF RCSR'
1054	001450	023727	DH6	:'TEST# ERR PC RCSR'
1055	001452	024454	DT6	:'\$TESTN,\$ERRPC,RCSR'
1056	001454	000000	0	
1057				
1058	001456	020756	EM32	:'CAN NOT CLEAR BIT6 OF RCSR WITH RESET2'
1059	001460	023727	DH6	:'TEST# ERR PC RCSR'
1060	001462	024454	DT6	:'\$TESTN,\$ERRPC,RCSR'
1061	001464	000000	0	
1062				
1063	001466	021024	EM33	:'BIT6 OF LKS NOT CLEAR AFTER RESET'
1064	001470	024001	DH10	:'TEST# ERR PC LKS'
1065	001472	024474	DT10	:'\$TESTN,\$ERRPC,LKS'
1066	001474	000000	0	
1067				
1068	001476	021066	EM34	:'LKS INTERRUPT WITH PRIORITY 7'
1069	001500	024001	DH10	:'TEST# ERR PC LKS'
1070	001502	024474	DT10	:'\$TESTN,\$ERRPC,LKS'
1071	001504	000000	0	
1072				
1073	001506	021124	EM35	:'CAN NOT SET BIT6 OF LKS'
1074	001510	024001	DH10	:'TEST# ERR PC LKS'
1075	001512	024474	DT10	:'\$TESTN,\$ERRPC,LKS'
1076	001514	000000	0	
1077				
1078	001516	021154	EM36	:'CAN NOT CLEAR BIT6 OF LKS'
1079	001520	024001	DH10	:'TEST# ERR PC LKS'
1080	001522	024474	DT10	:'\$TESTN,\$ERRPC,LKS'
1081	001524	000000	0	
1082				
1083	001526	021206	EM37	:'RESET DID NOT CLEAR BIT6 OF LKS'
1084	001530	024001	DH10	:'TEST# ERR PC LKS'
1085	001532	024474	DT10	:'\$TESTN,\$ERRPC,LKS'
1086	001534	000000	0	
1087				
1088	001536	021246	EM40	:'DUAL ADDRESSING ERROR'
1089	001540	024025	DH40	:'TEST# ERR PC GOOD BAD'
1090	001542	024504	DT40	:'\$TESTN,\$ERRPC,\$GDADR,\$BDCSR'
1091	001544	000000	0	
1092				
1093	001546	021274	EM41	:'BIT7 OF LKS NOT SET AFTER RESET2'
1094	001550	024001	DH10	:'TEST# ERR PC LKS'
1095	001552	024474	DT10	:'\$TESTN,\$ERRPC,LKS'
1096	001554	000000	0	

1097	001556	021334	EM42	:'CAN NOT CLEAR BIT7 OF LKS''
1098	001560	024001	DH10	:'TEST# ERR PC LKS''
1099	001562	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1100	001564	000000	0	
1101				
1102	001566	021366	EM43	:'BIT7 OF LKS DOES NOT SET''
1103	001570	024001	DH10	:'TEST# ERR PC LKS''
1104	001572	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1105	001574	000000	0	
1106				
1107	001576	021417	EM44	:'RTC INTERRUPT AT PRIORITY 7''
1108	001600	024001	DH10	:'TEST# ERR PC LKS''
1109	001602	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1110	001604	000000	0	
1111				
1112	001606	021453	EM45	:'RTC INTERRUPTS WHEN DISABLED''
1113	001610	024001	DH10	:'TEST# ERR PC LKS''
1114	001612	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1115	001614	000000	0	
1116				
1117	001616	021510	EM46	:'RTC INTERRUPT DID NOT OCCUR''
1118	001620	024001	DH10	:'TEST# ERR PC LKS''
1119	001622	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1120	001624	000000	0	
1121				
1122	001626	021510	EM47	:'RTC INTERRUPT DID NOT OCCUR''
1123	001630	024001	DH10	:'TEST# ERR PC LKS''
1124	001632	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1125	001634	000000	0	
1126				
1127	001636	021544	EM50	:'RTC DOUBLE INTERRUPT''
1128	001640	024001	DH10	:'TEST# ERR PC LKS''
1129	001642	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1130	001644	000000	0	
1131				
1132	001646	021571	EM51	:'RESET DID NOT CLEAR RTC INTERRUPT''
1133	001650	024001	DH10	:'TEST# ERR PC LKS''
1134	001652	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1135	001654	000000	0	
1136				
1137	001656	021621	EM52	:'RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS''
1138	001660	024001	DH10	:'TEST# ERR PC LKS''
1139	001662	024474	DT10	:'\$TESTN,\$ERRPC,LKS
1140	001664	000000	0	



1141	001666	021676	EM53	
1142	001670	024061	DH53	;'TEST# ERR PC LKS CNT1 CNT2''
1143	001672	024516	DT53	;\$TESTN,\$ERRPC,LKS,FIRST,SECND
1144	001674	000000	0	
1145				
1146	001676	021722	EM54	;'XMIT INTERRUPTS WHEN DISABLED''
1147	001700	023655	DH1	;'TEST# ERR PC TCSR''
1148	001702	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1149	001704	000000	0	
1150				
1151	001706	022060	EM55	;'XMIT DID NOT INTERRUPT''
1152	001710	023655	DH1	;'TEST# ERR PC TCSR''
1153	001712	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1154	001714	000000	0	
1155				
1156	001716	021760	EM56	;'XMIT INTERRUPT AT PRIORITY 7''
1157	001720	023655	DH1	;'TEST# ERR PC TCSR''
1158	001722	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1159	001724	000000	0	
1160				
1161	001726	022016	EM57	;'XMIT INTERRUPTS WITH ENABLE CLEAR''
1162	001730	023655	DH1	;'TEST# ERR PC TCSR''
1163	001732	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1164	001734	000000	0	
1165				
1166	001736	022060	EM60	;'XMIT DID NOT INTERRUPT''
1167	001740	023655	DH1	;'TEST# ERR PC TCSR''
1168	001742	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1169	001744	000000	0	
1170				
1171	001746	022107	EM61	;'XMIT RE-INTERRUPTED''
1172	001750	023655	DH1	;'TEST# ERR PC TCSR''
1173	001752	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1174	001754	000000	0	
1175				
1176	001756	022133	EM62	;'LOADING TBUF DID NOT CLEAR INTERRUPT''
1177	001760	023655	DH1	;'TEST# ERR PC TCSR''
1178	001762	024434	DT1	;\$TESTN,\$ERRPC,TCSR
1179	001764	000000	0	
1180				
1181	001766	022200	EM63	;'RCVR ACTIVE NOT SET''
1182	001770	023727	DH6	;'TEST# ERR PC RCSR''
1183	001772	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1184	001774	000000	0	

1185	001776	022224	EM64	;'RECEIVER DONE NEVER SET'
1186	002000	023727	DH6	;'TEST# ERR PC RCSR'
1187	002002	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1188	002004	000000	0	
1189				
1190	002006	022250	EM65	;'RCVR ACTIVE NOT CLEARED WITH DONE SET2
1191	002010	023727	DH6	;'TEST# ERR PC RCSR'
1192	002012	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1193	002014	000000	0	
1194				
1195	002016	022316	EM66	;'RESET DID NOT CLEAR RCVR DONE'
1196	002020	023727	DH6	;'TEST# ERR PC RCSR'
1197	002022	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1198	002024	000000	0	
1199				
1200	002026	022354	EM67	;'RDR ENABLE SET DID NOT CLEAR RCVR DONE'
1201	002030	023727	DH6	;'TEST# ERR PC RCSR'
1202	002032	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1203	002034	000000	0	
1204				
1205	002036	022417	EM70	;'READING RBUF DID NOT CLEAR RCVR DONE'
1206	002040	023727	DH6	;'TEST# ERR PC RCSR'
1207	002042	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1208	002044	000000	0	
1209				
1210	002046	022464	EM71	;'RCVR INTERRUPTS WITH ENABLE CLEAR'
1211	002050	023727	DH6	;'TEST# ERR PC RCSR'
1212	002052	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1213	002054	000000	0	
1214				
1215	002056	022633	EM72	;'RCVR DID NOT INTERRUPT'
1216	002060	023727	DH6	;'TEST# ERR PC RCSR'
1217	002062	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1218	002064	000000	0	
1219				
1220	002066	022526	EM73	;'RCVR INTERRUPTS AT PRIORITY 7'
1221	002070	023727	DH6	;'TEST# ERR PC RCSR'
1222	002072	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1223	002074	000000	0	
1224				
1225	002076	022564	EM74	;'RCVR INTERRUPT REQUEST PASSED WITH ENABLE CLEAR'
1226	002100	023727	DH6	;'TEST# ERR PC RCSR'
1227	002102	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1228	002104	000000	0	

1229	002106	022633	EM75	;'RCVR DID NOT INTERRUPT''
1230	002110	023727	DH6	;'TEST# ERR PC RCSR''
1231	002112	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1232	002114	000000	0	
1233	002116	022662	EM76	;'RECEIVER RE-INTERRUPTED''
1234	002120	023727	DH6	;'TEST# ERR PC RCSR''
1235	002122	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1236	002124	000000	0	
1237				
1238	002126	022706	EM77	;'READING RBUF DID NOT CLEAR INTERRUPT''
1239	002130	023727	DH6	;'TEST# ERR PC RCSR''
1240	002132	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1241	002134	000000	0	
1242				
1243	002136	022753	EM100	;'RSET DID NOT CLEAR RCVR INTERRUPT''
1244	002140	023727	DH6	;'TEST# ERR PC RCSR''
1245	002142	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1246	002144	000000	0	
1247				
1248				
1249	002146	023016	EM101	;'OR' FLAG DID NOT SET''
1250	002150	023727	DH6	
1251	002152	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1252	002154	000000	0	
1253				
1254	002156	023044	EM102	;'ERROR' NOT SET WITH 'OR' FLAG''
1255	002160	023727	DH6	;'TEST# ERR PC RCSR''
1256	002162	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1257	002164	000000	0	
1258	002166	023103	EM103	;'BREAK DID NOT TRANSMIT ALL ZEROES''
1259	002170	024126	DH103	;'TEST# ERR PC RCSR DATA''
1260	002172	024532	DT103	;\$TESTN,\$ERRPC,RCSR,\$BDDAT
1261	002174	000000	0	
1262				
1263	002176	023141	EM104	;'BREAK DID NOT SET FRAMING ERROR''
1264	002200	023727	DH6	;'TEST# ERR PC RCSR''
1265	002202	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1266	002204	000000	0	
1267				
1268	002206	023176	EM105	;'DATA COMPARE ERROR''
1269	002210	024163	DH105	;'TEST# ERR PC RCSR GOOD BAD''
1270	002212	024544	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1271	002214	000000	0	

1272	002216	023221	EM106	;'LOOP-BACK DATA COMPARE ERROR''
1273	002220	024163	DH105	;'TEST# ERR PC RCSR GOOD BAD''
1274	002222	024544	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1275	002224	000000	0	
1276				
1277	002226	023256	EM107	;'TIMEOUT IN EXERCISER TEST''
1278	002230	023727	DH6	;'TEST# ERR PC RCSR''
1279	002232	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1280	002234	000000	0	
1281				
1282	002236	023310	EM110	;'INCORRECT RECEIVE COUNT
1283	002240	024227	DH110	;'TEST# ERR PC RCSR TRANS RCV''
1284	002242	024560	DT110	;\$TESTN,\$ERRPC,RCSR,XMTCNT,RCVCNT
1285	002244	000000	0	
1286				
1287	002246	023340	EM111	;'DATA COMPARE ERROR IN EXERCISER''
1288	002250	024163	DH105	;'TEST# ERR PC RCSR GOOD BAD''
1289	002252	024544	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1290	002254	000000	0	
1291				
1292	002256	023400	EM112	;'TRAP CATCHER''
1293	002260	024273	DH112	;'TEST# ERR PC RCSR OLDPC TRAP ADR''
1294	002262	024574	DT112	;\$TESTN,\$ERRPC,RCSR,OLDPC,BDVECT
1295	002264	000000	0	
1296				
1297	002266	023415	EM113	;'NO CLK INTERRUPT IN EXERCISER''
1298	002270	024001	DH10	;'TEST# ERR PC LKS''
1299	002272	024474	DT10	;\$TESTN,\$ERRPC,LKS
1300	002274	000000	0	
1301				
1302	002276	023453	EM114	;'ERROR NOT SET WITH 'FR' FLAG''
1303	002300	023727	DH6	;'TEST# ERR PC RCSR''
1304	002302	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1305	002304	000000	0	
1306				
1307	002306	023512	EM115	;'RCV ACTVIE NOT CLEAR WITH INIT
1308	002310	023727	DH6	;'TEST# ERR PC RCSR''
1309	002312	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1310	002314	000000	0	
1311				
1312	002316	023551	EM116	;'RCV ACTIVE WITHOUT "START" BIT
1313	002320	023727	DH6	;'TEST# ERR PC RCSR''
1314	002322	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1315	002324	000000	0	
1316				
1317	002326	023610	EM117	;'RDR ENABLE NOT CLEAR WITH RCV ACTIVE
1318	002330	023727	DH6	;'TEST# ERR PC RCSR''
1319	002332	024454	DT6	;\$TESTN,\$ERRPC,RCSR
1320	002334	000000	0	
1321				
1322				
1323	002336	000000		;'STORAGE LOCATIONS
1324	002340	000000		FIRST: .WORD 0
1325	002342	000000		LOC1: .WORD 0
1326	002344	000000		LOC2: .WORD 0
1327	002346			SAVE0: .WORD 0
1328	002412			SAVLOC: .BLKW 22
				ENDSTK: .BLKW 10
				;'TIMER LOOP COUNTER
				;'TIMER LOOP COUNTER
				;'STORAGE FOR LOCATIONS
				;'THAT T/E USES
				;'JIMS SPECIAL STACK (WRAP ARGUND)

Line	Address	Value	Label	Unit	Comment
1329	002432	000000	JIMSTK:	.WORD	0
1330	002434	000000	SAVEPS:	.WORD	0
1331	002436	000000	OLDSUM:	.WORD	0
1332	002440	000000	RCVCNT:	.WORD	0
1333	002442	000000	XMTCNT:	.WORD	0
1334	002444	000000	CLKCNT:	.WORD	0
1335	002446	000000	STPSW:	.WORD	0
1336	002450	000000	SRPSW:	.WORD	0
1337	002452	000000	SCPSW:	.WORD	0
1338	002454		BUF:	.BLKW	44
1339	002564	020	CNTLP:	.ASCIZ	<20>
1340	002566	136	PROMPT:	.ASCII	/*P/<0><CR><LF>/CONSOLE/<CR><LF>/>>>/<377>
1341	002610	124	TA:	.ASCIZ	\$T/AS<CR><LF>
1342					
1343					
1344	002616	000000	SECND:	.WORD	0
1345	002620	000000	OLDPC:	.WORD	0
1346	002622	000000	BDVECT:	.WORD	0
1347					
1348					
1349					
1350					
1351					
1352	002624	000000	CTSTFL:	.WORD	0
1353	002626	000000	TMP1:	.WORD	0
1354	002630	000000	TMP2:	.WORD	0
1355	002632	000000	TMP3:	.WORD	0
1356			;REGISTER AND VECTOR ADDRESSES FOR THE DL-11W UNDER TEST		
1357					
1358	002634	000000	RCSR:	.WORD	0
1359	002636	000000	RBUF:	.WORD	0
1360	002640	000000	TCSR:	.WORD	0
1361	002642	000000	TBUF:	.WORD	0
1362	002644	000000	RVECT:	.WORD	0
1363	002646	000000	RPSW:	.WORD	0
1364	002650	000000	TVECT:	.WORD	0
1365	002652	000000	TPSW:	.WORD	0
1366					
1367			;CONSOLE REGISTER AND VECTOR ADDRESSES FOR THE DL-11W		
1368					
1369	002654	177560	CRCSR:	177560	;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
1370	002656	177562	CRBUF:	177562	;ADDRESS OF RECEIVER BUFFER
1371	002660	177564	CTCSR:	177564	;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
1372	002662	177566	CTBUF:	177566	;ADDRESS OF TRANSMITTER BUFFER
1373	002664	000060	CRVECT:	60	;RECEIVER INTERRUPT VECTOR
1374	002666	000062	CRPSW:	62	
1375	002670	000064	CTVECT:	64	;TRANSMITTER INTERRUPT VECTOR
1376	002672	000066	CTPSW:	66	
1377					
1378			;REAL TIME CLOCK REGISTER AND VECTOR ADDRESSES		
1379	002674	177546	LKS:	.WORD	177546
1380	002676	000100	RTCVT:	.WORD	100
1381	002700	000102	RTCPSW:	.WORD	102
1382					
1383	002702		ADRTBL:	.BLKW	20
1384	002742		VCTTBL:	.BLKW	20
1385	003002	000000	FLAG44:	.WORD	0

ERROR POINTER TABLE  
 1386 003004 176500  
 1387 003006 176502  
 1388 003010 176504  
 1389 003012 176506  
 1390  
 1391  
 1392  
 1393  
 1394 003014 012702 002702  
 1395 003020 013700 001142  
 1396 003024 010001  
 1397 003026 062701 000170  
 1398 003032 010022  
 1399 003034 062700 000010  
 1400 003040 020001  
 1401 003042 003773  
 1402 003044 000207  
 1403  
 1404  
 1405  
 1406 003046 005037 001070  
 1407 003052 005037 001066  
 1408 003056 005037 001072  
 1409 003062 005037 002624  
 1410 003066 005037 001076  
 1411 003072 005037 001100  
 1412 003076 005037 003002  
 1413 003102 005737 001112  
 1414 003106 001003  
 1415 003110 012737 000400 001112  
 1416 003116 012737 000006 000004  
 1417 003124 012737 000003 000006  
 1418  
 1419

TURCSR: .WORD 176500  
 TURBUF: .WCRD 176502  
 TUTCSR: .WCRD 176504  
 TUTBUF: .WCRD 176506

;SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE

```

DEVADR: MOV #ADRTBL,R2 ;POINT R2 TO THE DEVICE ADDRESS TABLE
        MOV $BASE,R0 ;LOAD BASE DEVICE ADDRESS IN R0
        MOV R0,R1 ;
        ADD #170,R1 ;POINT R1 TO LAST DEVICE ADDRESS
1$: MOV RO,(R2)+ ;MOVE DEVICE ADDRESS TO TABL.E
    ADD #10,RO ;POINT RO TO NEXT DEVICE ADDRESS
    CMP RO,R1 ;FINISHED GENERATING TABLE?
    BLE 1$ ;BR, IF LAST DEVICE ADDRESS NOT LOADED
    RTS PC
    
```

```

START: CLR $FATAL ;CLEAR ERROR NO.
        CLR $MSGTYP ;CLEAR MESSAGE TYPE
        CLR $TESTN ;CLEAR TEST NO.
        CLR CTSTFL ;CLEAR CONSOLE UNDER TEST FLAG
        CLR $DEVCT ;CLEAR DEVICE COUNT
        CLR $UNIT ;CLEAR UNIT NUMBER
        CLR FLAG44 ;** CLEAR 11/44 CPU FLAG
        TST $USWR ;IS $USWR LOADED?
        BNE 1$ ;BR IF YES
1$: MOV #400,$USWR ;ELSE, DEFAULT TO $USWR=400
    MOV #6,@#4 ;INITIALIZE TIMEOUT VECTORS TO TRAP
    MOV #3,@#6 ; CATCHER ROUTINE
    
```

.SBTTL INITIALIZE THE COMMON TAGS

```

;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #1000,SP ;;SETUP THE STACK POINTER

;;INITIALIZE A FEW VECTORS
MOV #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@#IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@#EMTVEC+2 ;;LEVEL 7
MOV #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@#TRAPVEC+2;LEVEL 7
MOV #PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@#PWRVEC+2 ;;LEVEL 7
MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
MOV #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS

;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
    
```

003266 013746 000004

```

CZDLDF0 DL11-W/1144 MFM SLU      MACRO M1113 05-NOV-80 12:32 PAGE 28-3 M 3 SEQUENCE 38
INITIALIZE THE COMMON TAGS
003272 012737 003326 000004      MOV      #64$,@#ERRVEC      ;;SET UP ERROR VECTOR
003300 012737 177570 001040      MOV      #DSWR,SWR          ;;SETUP FOR A HARDWARE SWICH REGISTER
003306 012737 177570 001042      MOV      #DDISP,DISPLAY     ;;AND A HARDWARE DISPLAY REGISTER
003314 022777 177777 175516      CMP      #-1,@SWR           ;;TRY TO REFERENCE HARDWARE SWR
003322 001012                                BNE      66$                ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                ;;AND THE HARDWARE SWR IS NOT = -1
003324 000403                                BR       65$                ;;BRANCH IF NO TIMEOUT
003326 012716 003334 54$:      MOV      #65$, (SP)        ;;SET UP FOR TRAP RETURN
003332 000002                                RTI
003334 012737 000176 001040 65$:  MOV      #SWREG,SWR         ;;POINT TO SOFTWARE SWR
003342 012737 000174 001042      MOV      #DISPREG,DISPLAY
003350 012637 000004 66$:      MOV      (SP)+,@#ERRVEC     ;;RESTORE ERROR VECTOR
003354 005037 001074      CLR      $PASS              ;;CLEAR PASS COUNT
003360 132737 000200 001107      BITB    #APTSIZE,$ENVM     ;;TEST USER SIZE UNDER APT
003366 001403                                BEQ      67$                ;;YES,USE NON-APT SWITCH
003370 012737 001110 001040      MOV      #$$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
003376                                67$:

1420
1421                                .SIZE FOR 11/44 CPU
1422
1423 003376 013746 000010      MOV      @#10,-(SP)         ;;** SAVE VECTOR
1424 003402 012737 003430 000010      MOV      #10$,@#10         ;;** SET UP FOR TRAP
1425 003410 000007                                MFPT        ;;** WHAT CPU??
1426 003412 122700 000001      CMPB    #1,R0              ;;** ARE WE A 11/44
1427 003416 001007                                BNE      11$                ;;** NO GO RESET VECTOR
1428 003420 052737 000001 003002      BIS      #1,@#FLAG44      ;;** YES SET FLAG
1429 003426 000403                                BR       11$                ;;** SKIP RTI
1430 003430 012716 003436 10$:      MOV      #11$, (SP)        ;;** SET STACK RETURN
1431 003434 000002                                RTI        ;;** RETURN
1432 003436 012637 000010 11$:      MOV      (SP)+,@#10        ;;** RESTORE VECTOR
1433 003442 032777 000020 175370      BIT      #BIT4,@SWR        ;;TEST CLOCK ONLY?
1434 003450 001404                                BEQ      INIT              ;;BR IF NOT
1435 003452 005237 002624      INC      CTSTFL            ;;ELSE, SET CONSOLE TEST FLAG TO ENABLE CLOCK TESTS
1436 003456 000137 004620      JMP      ID                ;; AND JUMP TO TYPE PROGRAM ID
1437 003462 132737 000001 001106 INIT:  BITB    #BIT0,$ENV         ;;CHECK IF ON APT
1438 003470 001404                                BEQ      MANL              ;;BR IF NOT APT
1439 003472 132737 000200 001107      BITB    #BIT7,$ENVM        ;;DID APT SIZE
1440 003500 001056                                BNE      APTSZD            ;;BR, IF APT SIZED
1441 003502 032777 000040 175330 MANL:  BIT      #BIT5,@SWR        ;;WAS '$DEVN' MANUALLY SET?
1442 003510 001052                                BNE      APTSZD            ;;IF YES, SKIP SELF-SIZING
1443
1444 003512 004737 003014      SIZE:  JSR      PC,DEVADR    ;;GENERATE DEVICE ADDRESS TABLE
1445 003516 005037 002630      CLR      TMP2              ;;CLR TEMP LOCATION TO KEEP DEVICE COUNT
1446 003522 005037 001144      CLR      $DEVN            ;;CLEAR DEVICE MAP
1447 003526 013703 000004      MOV      @#4,R3           ;;SAVE TIMEOUT VECTOR
1448 003532 012737 003562 000004      MOV      #4$,@#4         ;;SET TIMEOUT POINTER
1449 003540 013700 001142      MOV      $BASE,R0         ;;LOAD BASE ADDRESS
1450 003544 062700 000160      ADD      #160,R0          ;;POINT R0 TO UNIT #15 (UNIT#0=CONSOLE)
1451 003550 005710 3$:      TST      (R0)             ;;CHECK FOR DEVICE EXISTANCE
1452 003552 005237 001144      INC      $DEVN            ;;INDICATE DEVICE EXISTANCE IN DEVICE MAP
1453 003556 005237 002630      INC      TMP2              ;;INCREMENT DEVICE COUNT
1454 003562 012706 001000 4$:      MOV      #1000,SP         ;;RESET STACK POINTER
1455 003566 006337 001144      ASL      $DEVN            ;;ADJUST DEVICE MAP FOR NEXT UNIT CHECK
1456 003572 162700 000010      SUB      #10,R0           ;;POINT R0 TO NEXT DEVICE NUMBER
1457 003576 023700 001142      CMP      $BASE,R0         ;;FINISHED SIZING?
1458 003602 003762                                BLE      3$                ;;BR, IF BASE ADDRESS HAS NOT BEEN CHECKED
1459 003604 013700 002654      MOV      CRCSR,R0         ;;LOAD CONSOLE DEVICE ADDRESS

```

```

CZDLDF0 DL11-W/1144 MFM SLU      MACRO M1113 05-NOV-80 12:32 PAGE 28-4 N 3 SEQUENCE 39
INITIALIZE THE COMMON TAGS
1460 003610 012737 003630 000004      MOV      #5$,@#4      ;SET UP TIMEOUT POINTER
1461 003616 005710              TST      (R0)         ;TEST FOR CONSOLE EXISTANCE
1462 003620 005237 001144          INC      $DEVMM      ;INDICATE CONSOLE EXISTANCE IN DEVICE MAP
1463 003624 005237 002630          INC      TMP2        ;INCREMENT DEVICE COUNT
1464 003630 010337 000004          5$:     MOV      R3,@#4      ;RESTORE TIMEOUT VECTOR
1465
1466 003634 000415              BR       VCTADR      ;BR TO GENERATE VECTOR ADDRESS TABLE
1467
1468 003636 005037 002630          APTSZD: CLR     TMP2        ;CLEAR TEMP LOCATION TO KEEP DEVICE CNT
1469 003642 013702 001144          MOV     $DEVMM,R2    ;MOVE DEVICE MAP TO R2
1470 003646 005702              TSTDVM: TST     R2      ;TEST MSB OF DEVICE MAP
1471 003650 100002              BPL     1$          ;BR, IF MSB IS ZERO
1472 003652 005237 002630          INC     TMP2        ;INCREMENT DEVICE COUNT, IF MSB=1
1473 003656 006302              1$:     ASL     R2      ;SHIFT NEXT BIT INTO MSB POSITION
1474 003660 001401              BEQ     DVADT       ;BR, IF NO OTHER BITS ARE SET IN $DEVMM
1475 003662 000771              BR      TSTDVM      ;CONTINUE CHECKING $DEVMM, IF MORE BITS SET
1476 003664 004737 003014          DVADT: JSR     PC,DEVADR ;GENERATE DEVICE ADDRESS TABLE
1477
1478          ;GENERATE VECTOR ADDRESS TABLE
1479
1480 003670 012702 002742          VCTADR: MOV     #VCTTBL,R2 ;GET LOCATION OF VECTOR TABLE
1481 003674 013700 001136          MOV     @#$VECT1,R0  ;COPY BASE VECTOR
1482 003700 042700 177000          BIC     #177000,R0   ;CLEAR UPPER BITS
1483 003704 010001              MOV     R0,R1
1484 003706 062701 000170          ADD     #170,R1     ;POINT R1 TO LAST DEVICE VECTOR
1485 003712 010022              1$:     MOV     R0,(R2)+ ;PUT VECTOR ADDRESS IN TABLE
1486 003714 062700 000010          ADD     #10,R0     ;POINT R0 TO NEXT VECTOR ADDRESS
1487 003720 020001              CMP     R0,R1
1488 003722 003773              BLE     1$          ;FINISHED GENERATING VECTOR TABLE?
1489
1490          ;MOVE DEVICE COUNT INTO DEVICE COUNT MESSAGE
1491
1492 003724 013700 002630          MOV     TMP2,R0     ;COPY DEVICE COUNT INTO R0
1493 003730 005001              CLR     R1          ;CLEAR AUXILARY REGISTER
1494 003732 000300              SWAB   R0          ;PUT DEVICE COUNT IN UPPER BYTE OF R0
1495 003734 006300              ASL     R0          ;MOVE MSB OF COUNT INTO
1496 003736 006300              ASL     R0          ;MSB OF R0
1497 003740 006300          SHIFT: ASL     R0          ;PUT MSB OF COUNT INTO CARRY
1498 003742 106101              ROLB   R1          ;MOVE MSB OF COUNT INTO R1
1499 003744 006300              ASL     R0          ;MOVE NEXT BIT TO CARRY
1500 003746 106101              ROLB   R1          ;MOVE INTO R1
1501 003750 006300              ASL     R0          ;MOVE LAST BIT OF DIGIT
1502 003752 106101              ROLB   R1          ;INTO R1
1503 003754 062701 000060          ADD     #60,R1     ;CONVERT DIGIT TO ASCII
1504 003760 000301              SWAB   R1          ;MOVE DIGIT TO UPPER BYTE
1505 003762 032701 000020          BIT     #BIT4,R1   ;HAVE BOTH DIGITS BEEN MOVED TO R1?
1506 003766 001764              BEQ     SHIFT
1507 003770 010137 024404          MOV     R1,M2A     ;MOVE DEVICE COUNT TO OUTPUT MESSAGE
1508
1509
1510 003774 052737 000002 002632 BEGIN: BIS     #BIT1,TMP3 ;SET UP BIT MASK TO TEST $DEVMM FOR DEVICES EXCEPT CONSOLE
1511 004002 005037 002626          CLR     TMP1        ;CLEAR LOCATION TO STORE TABLE OFFSETS
1512 004006 032737 000001 001144          BIT     #BIT0,$DEVMM ;IS CONSOLE TO BE TESTED?
1513 004014 001001              BNE     TCONS      ;BR, IF CONSOLE IS TO BE TESTED
1514 004016 000414              BR      TSTDVM      ;BR, TO TEST OTHER DEVICES
1515 004020 005237 002624          TCONS: INC     CTSTFL ;INDICATE CONSOLE UNDER TEST
1516 004024 012700 002654          MOV     #CRCSR,R0  ;SET UP CONSOLE DEVICE ADDRESSES

```



```

1517 004030 012701 002634      MOV      #RCSR,R1      ;POINT R1 TO UUT ADDRESS TABLE
1518 004034 012021      1$:  MOV      (R0)+,(R1)+ ;TRANSFER CONSOLE ADDRESSES
1519 004036 022701 002652      CMP      #TPSW,R1     ;FINISHED TRANSFER?
1520 004042 002374      BGE      1$           ;BR, IF NOT
1521 004044 000137 004172      JMP      TST1        ;GO TEST CONSOLE INTERFACE
1522
1523      .PREPARE ADDRESSES AND VECTORS FOR UUT
1524 004050 033737 002632 001144 1$TDEV: BIT      TMP3,$DEVM ;CHECK TO SEE IF DEVICE IS TO BE TESTED
1525 004056 001010      BNE      SETADR      ;BR, IF YES
1526 004060 006337 002632      ASL      TMP3        ;SHIFT MASK TO CHECK NEXT $DEVM BIT
1527 004064 062737 000002 002626  ADD      #2,TMP1     ;INCREMENT TABLE INDEX
1528 004072 005237 001100      INC      $UNIT       ;INCREMENT UNIT NUMBER
1529 004076 000764      BR       TSTDEV     ;GO TEST NEXT BIT OF DEVICE MAP
1530
1531 004100 005237 001100      SETADR: INC      $UNIT ;UPDATE UNIT NUMBER
1532 004104 006337 002632      ASL      TMP3        ;UPDATE DEVICE MAP TEST MASK
1533 004110 013702 002626      MOV      TMP1,R2     ;MOVE TABLE OFFSET TO R2
1534 004114 062737 000002 002626  ADD      #2,TMP1     ;UPDATE TABLE OFFSET FOR NEXT DEVICE
1535 004122 016200 002702      MOV      ADRTBL(R2),R0 ;PUT UUT ADDRESS INTO R0
1536 004126 012701 002634      MOV      #RCSR,R1   ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
1537 004132 010021      ADR:  MOV      R0,(R1)+ ;TRANSFER UUT ADDRESS
1538 004134 062700 000002      ADD      #2,R0      ;POINT TO NEXT UUT REGISTER
1539 004140 030027 000006      BIT      R0,#6      ;FINISHED TRANSFER?
1540 004144 001372      BNE      ADR        ;BR, IF NOT
1541
1542 004146 016200 002742      VECT:  MOV      VCTTBL(R2),R0 ;PUT UUT VECTOR INTO R0
1543 004152 010021      MOV      R0,(R1)+   ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
1544 004154 062700 000002      ADD      #2,R0      ;POINT TO NEXT VECTOR
1545 004160 030027 000006      BIT      R0,#6      ;FINISHED TRANSFER?
1546 004164 001372      BNE      VECT       ;BR, IF NOT
1547 004166 000137 004172      JMP      TST1        ;GO TEST DEVICE

```

1548

```
.SBTTL TEST # 1 - TEST ABILITY TO REFERENCE TCSR
:*****
:TEST 1 - TEST ABILITY TO REFERENCE TCSR
:*****
```

```
1549 004172 000004
1550 004174 013703 000004
1551 004200 012737 004214 000004
1552 004206 005777 176426
1553 004212 000412
1554 004214 022626
1555 004216 005737 002624
1556 004222 001002
1557 004224 104001
1558 004226 000404
1559 004230
      004230 004737 015326
      004234 000001
1560 004236 000000
1561 004240 010337 000004
1562
1563
1564
```

```

TST1:  SCOPE
      MOV    @#4,R3          ;SAVE TIMEOUT VECTOR
      MOV    #1$,@#4        ;SET UP TIMEOUT VECTOR
      TST    @TCSR          ;REFERENCE THE XMIT COMMAND/STATUS REG.
      BR     4$             ;GO TO END OF TEST

1$:   CMP    (SP)+,(SP)+    ;RESTORE SP AFTER TIMEOUT
      TST    CTSTFL        ;CHECK IF DEVICE IS CONSOLE
      BNE    2$            ;IF YES, SKIP ERROR TYPEOUT
      ERROR  +1            ;REPORT ERROR TO APT & TTY
      BR     4$            ;BR TO END OF TEST

2$:   JSR    PC,$ATY4      ;: ONLY REPORT A FATAL ERROR
      1      ;: THE ERROR NUMBER (FROM APT LIST)

3$:   HALT
4$:   MOV    R3,@#4        ;RESTORE TIMEOUT VECTOR
```

1565

```

.SBTTL TEST # 2 - TEST ABILITY TO REFERENCE TBUF
:*****
:*TEST 2 - TEST ABILITY TO REFERENCE TBUF
:*****
TST2:  SCOPE
      MOV    @#4,R3          ;SAVE TIMEOUT VECTOR
      MOV    #1$,@#4        ;SET UP TIMEOUT VECTOR
      TST    @TBUF          ;REFERENCE THE XMIT BUFFER
      BR     4$             ;GO TO END OF TEST

1$:   CMP    (SP)+,(SF +    ;RESTORE SP AFTER TIMEOUT
      TST    CTSTFL        ;CHECK IF DEVICE IS CONSOLE
      BNE    2$            ;IF YES, SKIP ERROR TYPEOUT
      ERROR  +2            ;REPORT ERROR TO APT & TTY
      BR     4$            ;BR TO END OF TEST

2$:   JSR    PC,$ATY4      ;;ONLY REPORT A FATAL ERROR
      2                      ;;THE ERROR NUMBER (FROM APT LIST)

3$:   HALT

4$:   MOV    R3,@#4        ;RESTORE TIMEOUT VECTOR
  
```

```

004244 000004
1566 004246 013703 000004
1567 004252 012737 004266 000004
1568 004260 005777 176356
1569 004264 000412
1570
1571 004266 022626
1572 004270 005737 002624
1573 004274 001002
1574 004276 104002
1575 004300 000404
1576 004302
      004302 004737 015326
      004306 000002
1577 004310 000000
1578 004312 010337 000004
  
```

1579

.SBTTL TEST # 3 - TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET  
 ;\*\*\*\*\*  
 ;\*TEST 3 - TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET  
 ;\*\*\*\*\*

```

1580 004316 000004
1580 004320 000005
1581 004322 032777 000004 176310
1582 004330 001411
1583
1584 004332 005737 002624
1585 004336 001002
1586 004340 104015
1587 004342 000404
1588
1589 004344
      004344 004737 015326
      004350 000015
1590 004352 000000
1591
1592 004354 052777 000004 176256
1593 004362 032777 000004 176250
1594 004370 001001
1595
1596 004372 104016
1597
1598 004374 042777 000004 176236
1599 004402 032777 000004 176230
1600 004410 001411
1601
1602 004412 005737 002624
1603 004416 001002
1604 004420 104017
1605 004422 000404
1606 004424
      004424 004737 015326
      004430 000017
1607 004432 000000
1608
1609 004434 052777 000004 176176
1610 004442 000005
1611 004444 032777 000004 176166
1612 004452 001404
1613
1614 004454 042777 000004 176156
1615 004462 104020
1616
1617
1618 004464 000240
  
```

```

TST3:  SCOPE
      RESET ;CLEAR EVERYTHING
      BIT #BIT2,@TCSR ;TEST FOR BIT2 OF TCSR CLEAR
      BEQ 3$ ;BR IF CLEAR

      TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
      BNE 1$ ;IF YES, SKIP ERROR TYPEOUT
      ERROR +15 ;BIT2 OF TCSR NOT CLEAR AFTER RESET
      BR 3$

1$: JSR PC,$ATY4 ;:ONLY REPORT A FATAL ERROR
      15 ;:THE ERROR NUMBER (FROM APT LIST)
2$: HALT

3$: BIS #BIT2,@TCSR ;SET BIT2 OF TCSR
      BIT #BIT2,@TCSR ;TEST FOR BIT2 SET
      BNE 4$ ;BR IF SET

      ERROR +16 ;BIT2 OF TCSR WILL NOT SET

4$: BIC #BIT2,@TCSR ;CLEAR BIT2 OF TCSR
      BIT #BIT2,@TCSR ;TEST BIT2 CLEAR
      BEQ 7$ ;BR IF CLEAR

      TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
      BNE 5$ ;IF YES, SKIP ERROR TYPEOUT
      FRROR +17
      BR 7$

5$: JSR PC,$ATY4 ;:ONLY REPORT A FATAL ERROR
      17 ;:THE ERROR NUMBER (FROM APT LIST)
6$: HALT ;BIT0 OF TCSR WILL NOT CLEAR

7$: BIS #BIT2,@TCSR ;SET BIT2 OF TCSR
      RESET ;CLEAR BIT2 WITH RESET
      BIT #BIT2,@TCSR ;TEST FOR BIT2 CLEAR
      BEQ 10$ ;** IF CLEAR, GO TO NEXT TEST

      BIC #BIT2,@TCSR ;CLEAR BIT2, TO PRINT ERROR
      ERROR +20 ;RESET DID NOT CLEAR BIT2 OF TCSR

10$: NOP ;**
  
```

1619

.SBTTL TEST # 4 - TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED

\*\*\*\*\*  
 \*TEST 4 - TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED  
 \*\*\*\*\*

```

TST4:  SCOPE
1620 004466 000004          BIS      #BIT2,@TCSR    ;** ENABLE MAINT. WRAP
1621 004470 052777 000004 176142    CLR      @TBUF        ;LOAD XBUF
1622 004476 005077 176140          TSTB    @TCSR        ;CHECK DONE
1623 004502 105777 176132          BPL     3$           ;BR IF CLEAR
1624                                ;FILL SECOND BUFFER, BECAUSE REFRESH COULD CAUSE
1625                                ;  FIRST TEST TO FAIL
1626 004510 005077 176126          CLR      @TBUF        ;FILL DOUBLE BUFFER
1627 004514 105777 176120          TSTB    @TCSR        ;CHECK DONE
1628 004520 100014          BPL     3$           ;BR IF CLEAR
1629
1630 004522 005737 002624          TST     CTSTFL       ;CHECK IF DEVICE IS CONSOLE
1631 004526 001005          BNE     1$           ;IF YES, SKIP ERROR TYPEOUT
1632 004530 042777 000004 176122    BIC     #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
                                ;** CONSOLE TO ALLOW FOR COMMUNICATION
                                ;** WITH TERMINAL.
                                ;DONE NOT CLEARED WITH TBUF FULL
1633 004536 104003          ERROR   +3          ;BR TO END OF TEST
1634 004540 000404          BR      3$
1635 004542          1$:
                                JSR     PC,$ATY4      ;: ONLY REPORT A FATAL ERROR
                                3          ;: THE ERROR NUMBER (FROM APT LIST)
1636 004546 000003          2$:
1637 004550 000000          3$: HALT
1638 004552 005000          4$: CLR      RO          ;TCSR 'DONE' NOT CLEARED WITH TBUF FULL
1639 004554 105777 176060          TSTB    @TCSR        ;CLEAR TIMER
1640 004560 100417          BMI    ID           ;CHECK FOR XMIT DONE
1641 004562 005200          INC     RO          ;IF DONE SETS, BR TO END OF TEST
1642 004564 001373          BNE     4$          ;INCREMENT TIMER
1643 004566 005737 002624          TST     CTSTFL       ;BR IF TIMER NOT DONE
1644 004572 001005          BNE     5$          ;CHECK IF DEVICE IS CONSOLE
1645 004574 042777 000004 176056    BIC     #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
                                ;** CONSOLE TO ALLOW FOR COMMUNICATION
                                ;** WITH TERMINAL.
                                ;TCSR 'DONE' DOES NOT SET
1646 004602 104004          ERROR   +4          ;BR TO END OF TEST
1647 004604 000405          BR      ID
1648 004606          5$:
                                JSR     PC,$ATY4      ;: ONLY REPORT A FATAL ERROR
                                4          ;: THE ERROR NUMBER (FROM APT LIST)
1649 004612 000004          HALT
1650 004614 000000          BR      ENDB7       ;** BR TO NEXT TEST,
1651                                ;** AND SKIP THE TYPEOUT THAT FOLLOWS
1652                                ;** BECAUSE OF THIS FAILURE
1653
1654 004620          ID:
1655 004620 042777 000004 176032    BIC     #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
                                ;** CONSOLE TO ALLOW FOR COMMUNICATION
                                ;** WITH TERMINAL.
1656 004626 023737 000042 000046    CMP     @#42,@#46    ;UNDER ACT11?
1657 004634 001412          BEQ     6$          ;IF YES, SKIP IDENT. TYPEOUT
1658 004636 005737 001074          TST     $PASS       ;IS THIS THE FIRST PASS?
1659 004642 001007          BNE     6$          ;IF NOT BR TO NEXT TEST & SKIP THE IDENTIFICATION TYPEOUTS
1660 004644 005737 001076          TST     $DEVCT      ;IS THIS THE FIRST SUBPASS?
  
```

1661	004650	001004			BNE	6\$			;IF NOT, BRTO NEXT TEST
1662	004652	104401			TYPE				;TYPE PROGRAM IDENTIFICATION
1663	004654	024344			M1				
1664	004656	104401			TYPE				;TYPE NUMBER OF DEVICES UNDER TEST
1665	004660	024402			M2				
1666	004662	032777	000020	174150	6\$:	BIT	#BIT4,@SWR		;CLOCK TEST ONLY?
1667	004670	001402			BEQ	ENDB7			** BR IF NOT
1668	004672	000137	005536		JMP	TCLOCK			;ELSE, JUMP TO TEST CLOCK
1669	004676				ENDB7:				
1670	004676	005000			CLR	R0			** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	004700	012701	000002		MOV	#2,R1			** THAT MIGHT BE IN THE PROCESS OF BEING
	004704	005300			40\$:	DEC	R0		** TRANSMITTED TO FINISH BEFORE MAINTENANCE
	004706	001376			BNE	40\$			** WRAP FOR THE UART UNDER TEST IS DISABLED.
	004710	005301			DEC	R1			** THIS WILL INHIBIT ANY COMMUNICATION
	004712	001374			BNE	40\$			** TO HARDWARE MEDIA SUCH AS TU58 THAT MIGHT
									** BE ATTACHED TO UART UNDER TEST.

1671

```
.SBTTL TEST # 5 - TEST THAT TCSR 'DONE' SETS WITH RESET
:*****
:*TEST 5 - TEST THAT TCSR 'DONE' SETS WITH RESET
:*****
```

```
TST5:  SCOPE
1672 004714 000004
1673 004716 052777 000004 175714  BIS #BIT2,@TCSR ;** ENABLE MAINT. WRAP
1674 004724 005077 175712  CLR @TBUF ;LOAD TRANSMIT BUFFER
1675 004730 105777 175704 1$:  TSTB @TCSR ;WAIT FOR DONE
1676 004734 100375  BPL 1$
1677 004736 005077 175700  CLR @TBUF ;LOAD SECOND BUFFER
1678 004742 000240  NOP
1679 004744 000005  RESET ;CLEAR DONE WITH RESET
1680 004746 105777 175666  TSTB @TCSR ;CHECK FOR DONE SET
1681 004752 100401  BMI 10$ ;** BR TO NEXT TEST IF DONE SET
1682 004754 104005  ERROR +5 ;TCSR 'DONE' DOES NOT SET WITH RESET
1683 10$:  NOP ;**
1684 004756 000240
1685
1686
```

1687

.SBTTL TEST # 6 - TEST ABILITY TO ACCESS RCSR  
:.....  
: \*TEST 6 - TEST ABILITY TO ACCESS RCSR  
:.....

1688 004760 000004  
1688 004762 013703 000004  
1689 004766 012737 005002 000004  
1690 004774 005777 175634  
1691 005000 000402  
1692  
1693 005002 022626  
1694 005004 104006  
1695 005006 010337 000004

TST6:    SCOPE  
          MOV     @#4,R3           ;SAVE TIMEOUT VECTOR  
          MOV     #18,@#4       ;SET UP TIMEOUT VECTOR  
          TST     @RCSR         ;ACCESS RCSR  
          BR      2\$           ;BR TO END OF TEST  
  
1\$:     CMP     (SP)+,(SP)+     ;RESTORE SP AFTER TIMEOUT  
          ERROR   +6           ;CAN NOT ACCESS RCSR  
2\$:     MOV     R3,@#4         ;RESTORE TIMEOUT VECTOR



1696

```
.SBTTL TEST # 7 - TEST ABILITY TO ACCESS RBUF
:*****
:*TEST 7 - TEST ABILITY TO ACCESS RBUF
:*****
```

```
TST7:  SCOPE
1697 005012 000004
1697 005014 013703 000004
1698 005020 012737 005034 000004
1699 005026 005777 175604
1700 005032 000402
1701
1702 005034 022626      1$:  CMP      (SP)+,(SP)+  ;RESTORE SP AFTER TIMEOUT
1703 005036 104007      ERROR    +7           ;CAN NOT ACCESS RBUF
1704 005040 010337 000004      2$:  MOV      R3,@#4     ;RESTORE TIMEOUT VECTOR
```

1705

.SBTTL TEST # 10 - TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET  
 \*\*\*\*\*  
 ;\*TEST 10 - TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET  
 \*\*\*\*\*

```

TST10: SCOPE
1706 005044 000004 000400 173764 BIT #BIT8,@SWR ;IS BREAK FUNCTION ENABLED?
1707 005054 001475 BEQ 10$ ;** BR TO NEXT TEST, IF NOT
1708 005056 032737 000001 003002 BIT #BIT0,FLAG44 ;** IS THIS A 11/44
1709 005064 001407 BEQ 9$ ;** NO
1710 005066 005737 002624 TST CTSTFL ;** YES THIS IS 11/44. IS THIS THE CONSOLE
1711 BEQ 9$ ;** SLU
1712 005072 001404 BEQ 9$ ;** NO
1713 005074 032777 000010 173736 BIT #BIT03,@SWR ;** THIS IS THE CONSOLE SLU.SHOULD THE BREAK
1714 BIT #BIT03,@SWR ;** TEST BE PERFORMED
1715 005102 001462 BEQ 10$ ;** NO
1716
1717 005104 000005 9$: RESET ;CLEAR EVERYTHING
1718 005106 032777 000001 175524 BIT #BIT0,@TCSR ;CHECK BIT0 OF TCSR CLEAR
1719 005114 001411 BEQ 3$ ;BR IF CLEAR
1720 005116 005737 002624 TST CTSTFL
1721 005122 001002 BNE 1$
1722 005124 104011 ERROR +11 ;BIT0 WAS NOT CLEAR AFTER RESET
1723 005126 000404 BR 3$
1724 005130 1$: JSR PC,$ATY4 ;:ONLY REPORT A FATAL ERROR
005130 004737 015326 11 ;:THE ERROR NUMBER (FROM APT LIST)
005134 000011
1725 005136 000000 2$: HALT
1726
1727 005140 052777 000001 175472 3$: BIS #BIT0,@TCSR ;SET BIT0 IN TCSR
1728 005146 032777 000001 175464 BIT #BIT0,@TCSR ;TEST BIT0 OF TCSR
1729 005154 001001 BNE 4$ ;BR IF SET
1730
1731 005156 104012 ERROR +i2 ;BIT0 OF TCSR WILL NOT SET
1732
1733 005160 042777 000001 175452 4$: BIC #BIT0,@TCSR ;CLEAR BIT0 OF TCSR
1734 005166 032777 000001 175444 BIT #BIT0,@TCSR ;TEST BIT0 OF TCSR
1735 005174 001411 BEQ 7$
1736 005176 005737 002624 TST CTSTFL
1737 005202 001002 BNE 5$
1738 005204 104013 ERROR +13 ;BIT0 OF TCSR WILL NOT CLEAR
1739 005206 000404 BR 7$
1740 005210 5$: JSR PC,$ATY4 ;:ONLY REPORT A FATAL ERROR
005210 004737 015326 13 ;:THE ERROR NUMBER (FROM APT LIST)
005214 000013
1741 005216 000000 6$: HALT
1742
1743 005220 052777 000001 175412 7$: BIS #BIT0,@TCSR ;SET BIT0 IN TCSR
1744 005226 000005 RESET ;CLEAR BIT0 WITH RESET
1745 005230 032777 000001 175402 BIT #BIT0,@TCSR ;TEST BIT0 CLEAR
1746 005236 001404 BEQ 10$ ;** BR IF CLEAR
1747 005240 042777 000001 175372 BIC #BIT0,@TCSR ;CLEAR BIT0, TO PRINT ERROR
1748 005246 104014 ERROR +14 ;RESET DID NOT CLEAR BIT0 OF TCSR
1749 005250 000240 10$: NOP ;**
    
```

1750

.SBTT. TEST # 11 - TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET  
 :\*\*\*\*\*  
 :\*TEST 11 - TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET  
 :\*\*\*\*\*

```

1751 005252 000004
1751 005254 000005
1752 005256 017703 175366
1753 005262 012777 005312 175360
1754 005270 004737 014400
1755 005274 000340
1756 005276 032777 000100 175334
1757 005304 001404
1758 005306 104021
1759
1760 005310 000402
1761
1762 005312 022626 1$:
1763 005314 104022 ERROR
1764
1765
1766 005316 052777 000100 175314 2$:
1767 005324 032777 000100 175306
1768 005332 001001
1769
1770 005334 104023 ERROR
1771
1772
1773 005336 042777 000100 175274 3$:
1774 005344 032777 000100 175266
1775 005352 001401
1776 005354 104024 ERPOR
1777
1778
1779 005356 052777 000100 175254 4$:
1780 005364 000005
1781 005366 032777 000100 175244
1782 005374 001401
1783
1784 005376 104025 ERROR
1785
1786 005400 010377 175244 5$:
    
```

```

TST11: SCOPE
RESET ;CLEAR EVERYTHING
MOV @TVECT,R3 ;SAVE XMIT VECTOR
MOV #1$,@TVECT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
JSR PC,WRPSW ;SET PSW TO PRIORITY=7
;WORD 340
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BEQ 2$ ;BR IF ZERO
ERROR +21 ;BIT6 IN TCSR NOT CLEAR AFTER RESET
BR 2$
1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR +22 ;XMIT INTERRUPT OCCURRED PRIO=7
2$: BIS #BIT6,@TCSR ;SET BIT6 OF TCSR
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BNE 3$ ;BR, IF SET
ERROR +23 ;CANNOT SET BIT6 OF TCSR
3$: BIC #BIT6,@TCSR ;CLEAR BIT6 OF TCSR
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BEQ 4$ ;BR IF CLEAR
ERROR +24 ;CANNOT CLEAR BIT6 OF TCSR
4$: BIS #BIT6,@TCSR ;SET BIT6 OF TCSR
RESET ;CLEAR BIT6 WITH RESET
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BEQ 5$ ;BR IF CLEAR
ERROR +25 ;CANNOT CLEAR BIT6 OF TCSR WITH RESET
5$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
    
```

1787

.SBTTL TEST # 12 - TEST THAT BIT6 OF RCSR CAN BE SET & RESET  
 :\*\*\*\*\*  
 :\*TEST 12 - TEST THAT BIT6 OF RCSR CAN BE SET & RESET  
 :\*\*\*\*\*  
 TST12:

1788	005404	000004				SCOPE				
	005406	000005				RESET				:CLEAR EVERYTHING
1789	005410	017703	175230			MOV	@RVECT,R3			:SAVE RECEIVE VECTOR
1790	005414	012777	005444	175222		MOV	#1\$,@RVECT			:SET UP INTERRUPT VECTOR FOR ERROR REPORT
1791	005422	004737	014400			JSR	PC,WRPSW			:SET PSW TO PRIORITY=7
1792	005426	000340					.WORD 340			
1793	005430	032777	000100	175176		BIT	#BIT6,@RCSR			:TEST BIT6 OF RCSR
1794	005436	001404				BEQ	2\$			
1795	005440	104026				ERROR	+26			
1796										:BIT6 OF RCSR NOT CLEAR AFTER RESET
1797	005442	000402				BR	2\$			
1798										
1799	005444	022626			1\$:	CMP	(SP)+,(SP)+			:RESTORE SP AFTER INTERRUPT
1800	005446	104027				ERROR	+27			
1801										:RCVR INTERRUPT WITH PRIORITY=7
1802										
1803	005450	052777	000100	175156	2\$:	BIS	#BIT6,@RCSR			:SET BIT6 OF RCSR
1804	005456	032777	000100	175150		BIT	#BIT6,@RCSR			:TEST BIT6 OF RCSR
1805	005464	001001				BNE	3\$			:BR, IF SET
1806										
1807	005466	104030				ERROR	+30			
1808										:CANNOT SET BIT6 OF RCSR
1809										
1810	005470	042777	000100	175136	3\$:	BIC	#BIT6,@RCSR			:CLEAR BIT6 OF RCSR
1811	005476	032777	000100	175130		BIT	#BIT6,@RCSR			:TEST BIT6 OF RCSR
1812	005504	001401				BEQ	4\$			:BR, IF CLEAR
1813										
1814	005506	104031				ERROR	+31			
1815										:CANNOT CLEAR BIT6 OF RCSR
1816										
1817	005510	052777	000100	175116	4\$:	BIS	#BIT6,@RCSR			:SET BIT6 OF RCSR
1818	005516	000005				RESET				:CLEAR BIT6 OF RCSR WITH RESET
1819	005520	032777	000100	175106		BIT	#BIT6,@RCSR			:TEST BIT6 OF RCSR
1820	005526	001401				BEQ	5\$			:BR, IF CLEAR
1821										
1822	005530	104032				ERROR	+32			
1823										:CANNOT CLEAR BIT6 OF RCSR WITH RESET
1824	005532	010377	175106		5\$:	MOV	R3,@RVECT			:RESTORE RECEIVE VECTOR
1825	005536	012737	000012	001002	TCLOCK:	MOV	#12,\$TSTNM			:ADJUST TEST NUMBER TO (NEXT TEST - 1)

1826

```
.SBTTL TEST # 13 - TEST ABILITY TO ACCESS LKS
:*****
:*TEST 13 - TEST ABILITY TO ACCESS LKS
:*****
```

```
TST13: SCOPE
1827 005544 000004          TST      CTSTFL      ;IS CONSOLE UNDER TEST?
      005546 005737 002624  BEQ      TST14      ;IF NOT, SKIP THIS TEST
      005552 001420          BIT      #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
      005554 032777 000100 173256 BNE     TST14      ;IF YES, SKIP THIS TEST
      005562 001014          MOV     @#4,R3     ;SAVE TIMEOUT VECTOR
1828 005564 013703 000004  MOV     #1$,@#4   ;SET UP TIMEOUT VECTOR
1829 005570 012737 005604 000004 TST     @LKS      ;ACCESS LKS
1830 005576 005777 175072          BR      2$        ;NO TIMEOUT - BR TO END OF TEST
1831 005602 000402
1832
1833 005604 022626          1$:    CMP     (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
1834 005606 104010          ERROR  +10       ;CAN NOT ACCESS LKS
1835
1836 005610 010337 000004          2$:    MOV     R3,@#4   ;RESTORE TIMEOUT VECTOR
1837
```

1838

.SBTTL TEST # 14 - TEST THAT BIT6 OF LKS CAN BE SET & RESET

\*\*\*\*\*  
 :\*TEST 14 - TEST THAT BIT6 OF LKS CAN BE SET & RESET  
 \*\*\*\*\*

```

1839 005614 000004
      005616 005737 002624
      005622 001460
      005624 032777 000100 173206
      005632 001054
1840 005634 000005
1841 005636 017703 175034
1842 005642 012777 005672 175026
1843 005650 004737 014400
1844 005654 000340
1845 005656 032777 000100 175010
1846 005664 001404
1847 005666 104033
1848
1849 005670 000402
1850
1851 005672 022626
1852 005674 104034
1853
1854
1855 005676 052777 000100 174770
1856 005704 032777 000100 174762
1857 005712 001001
1858
1859 005714 104035
1860
1861
1862 005716 042777 000100 174750
1863 005724 032777 000100 174742
1864 005732 001401
1865 005734 104036
1866
1867 005736 052777 000100 174730
1868 005744 000005
1869 005746 032777 000100 174720
1870 005754 001401
1871
1872 005756 104037
1873
1874 005760 010377 174712
    
```

TST14: SCOPE

```

TST CTSTFL ;IS CONSOLE UNDER TEST?
BEQ TST15 ;IF NOT, SKIP THIS TEST
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
BNE TST15 ;IF YES, SKIP THIS TEST
RESET
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV #1@,@RTCVT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
JSR PC,WRPSW ;SET PSW TO PRIORITY 7
        .WORD 340
BIT #BIT6,@LKS ;TEST BIT6 OF LKS
BEQ 2$
ERROR +33
        ;BIT6 OF LKS NOT CLEAR AFTER RESET
BR 2$
1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
    ERROR +34
        ;LKS INTERRUPT WITH PRIORITY=7
2$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
    BIT #BIT6,@LKS ;TEST BIT6 OF LKS
    BNE 3$ ;BR IF SET
    ERROR +35
        ;CANNOT SET BIT6 OF LKS
3$: BIC #BIT6,@LKS ;CLEAR BIT6 OF LKS
    BIT #BIT6,@LKS ;TEST BIT6 OF LK
    BEQ 4$
    ERROR +36
        ;CANNOT CLEAR BIT6 OF LKS
4$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
    RESET ;CLEAR BIT6 OF LKS WITH RESET
    BIT #BIT6,@LKS ;TEST BIT6 OF LKS
    BEQ 5$ ;BR IF CLEAR
    ERROR +37
        ;CANNOT CLEAR BIT6 OF LKS WITH RESET
5$: MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
    
```

1875

.SBTTL TEST # 15 - TEST FOR DUAL ADDRESSING OF REGISTERS  
 :\*\*\*\*\*  
 :\*TEST 15 - TEST FOR DUAL ADDRESSING OF REGISTERS  
 :\*\*\*\*\*

						TST15: SCOPE	
1876	005764	000004				MOV @#4,R3	;SAVE TIMEOUT VECTOR
1877	005772	013703	000004			MOV @#6,R4	;SAVE TIMEOUT PSW
1878	005776	012737	006130	000004		MOV #4,@#4	;SET UP TIMEOUT VECTOR
1879	006004	012737	000340	000006		MOV #340,@#6	;KEEP PRIO=7
1880	006012	000005				RESET	;CLEAR EVERYTHING
1881	006014	012700	000002			MOV #BIT1,R0	;SET UP BIT MASK
1882	006020	032777	000020	173012		BIT #BIT4,@SWR	;CLOCK TEST ONLY?
1883	006026	001404				BEQ 1\$	;BR IF NOT
1884	006030	013737	002674	001020		MOV LKS,\$GDADR	;ELSE, MOVE GOOD LKS ADDRESS INTO \$GDADR
1885	006036	000403				BR 2\$	
1886	006040	013737	002634	001020	1\$:	MOV RCSR,\$GDADR	;MOVE GOOD RCSR ADDRESS INTO \$GDADR
1887	006046	013737	001020	001022	2\$:	MOV \$GDADR,\$BDADR	;MOVE GOOD ADDRESS INTO TEST ADDRESS LOCATION
1888	006054	040037	001022			BIC R0,\$BDADR	;CREATE BAD ADDRESS BY COMPLEMENTING ONE BIT
1889	006060	023737	001020	001022		CMP \$GDADR,\$BDADR	;ARE ADDRESSES IDENTICAL?
1890	006066	001002				BNE 3\$	;IF NOT, TEST THIS ADDRESS
1891	006070	050037	001022			BIS R0,\$BDADR	;ELSE, BIT SET THIS BIT POSITION TO GENERATE BAD ADDRESS
1892	006074	017737	172722	001024	3\$:	MOV @\$BDADR,\$GDDAT	;SAVE CONTENTS OF BAD ADDRESS IF IT EXISTS
1893	006102	052777	000100	172712		BIS #BIT6,\$BDADR	;SET BIT6 USING BAD ADDRESS
1894	006110	032777	000100	172702		BIT #BIT6,\$GDADR	;CHECK TO SEE IF GOOD ADDRESS CONTAINS BIT6
1895	006116	001011				BNE 6\$	;BR IF SET ---> ERROR
1896	006120	013777	001024	172674		MOV \$GDDAT,\$BDADR	;RESTORE ANY MEMORY LOCATION THAT WAS ALTERED
1897	006126	000401				BR 5\$	;BR TO CONTINUE TEST
1898	006130	022626			4\$:	CMP (SP)+,(SP)+	;RESTORE SP AFTER TIMEOUT
1899	006132	006300			5\$:	ASL R0	;SHIFT BIT MASK TO NEXT POSITION
1900	006134	105700				TSTB R0	;COMPLEMENTED ALL BITS FROM 1 - 7?
1901	006136	100343				BPL 2\$	;BR, IF NOT.
1902	006140	000401				BR 7\$	;BR TO NEXT TEST
1903							
1904	006142	104040			6\$:	ERROR +40	;DUAL ADDRESSING ERROR
1905							; \$BDADR = DUAL ADDRESS
1906							; \$GDADR = GOOD ADDRESS
1907							
1908	006144	010337	000004		7\$:	MOV R3,@#4	;RESTORE TIMEOUT VECTOR
1909	006150	010437	000006			MOV R4,@#6	;RESTORE TIMEOUT PSW

1910

.SBTTL TEST # 16 - TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED  
 :\*\*\*\*\*  
 :\*TEST 16 - TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED  
 :\*\*\*\*\*

1911	006154	000004				TST16: SCOPE		
	006156	005737	002624			TST	CTSTFL	; IS CONSOLE UNDER TEST?
	006162	001437				BEQ	TST17	; IF NOT, SKIP THIS TEST
	006164	032777	000100	172646		BIT	#BIT6,@SWR	; ARE LINE CLOCK TESTS INHIBITED?
	006172	001033				BNE	TST17	; IF YES, SKIP THIS TEST
1912	006174	000005				RESET		; CLEAR EVERYTHING & SET BIT7 OF LKS
1913	006176	105777	174472		1\$:	TSTB	@LKS	; TEST FOR BIT7 OF LKS
1914	006202	100401				BMI	2\$	; BR IF SET
1915								
1916	006204	104041				ERROR	+41	; BIT7 OF LKS DID NOT SET WITH RESET
1917								
1918	006206	042777	000200	174460	2\$:	BIC	#BIT7,@LKS	; CLEAR BIT7 OF LKS
1919	006214	032777	000200	174452		BIT	#BIT7,@LKS	; TEST BIT7 OF LKS
1920	006222	001410				BEQ	3\$	
1921	006224	042777	000200	174442		BIC	#BIT7,@LKS	; TRY ONE MORE TIME BECAUSE THE CLOCK
1922	006232	032777	000200	174434		BIT	#BIT7,@LKS	; MAY HAVE SET IMMEDIATELY AFTER THE FIRST CLEAR
1923	006240	001401				BEQ	3\$	
1924								
1925	006242	104042				ERROR	+42	; CAN NOT CLEAR BIT7 OF LKS
1926								
1927	006244	005000			3\$:	CLR	RO	; CLEAR TIMER
1928	006246	105777	174422		CONT:	TSTB	@LKS	; TEST FOR BIT7 OF LKS
1929	006252	100403				BMI	TST17	; BR, IF SET
1930	006254	005200				INC	RO	; INCREMENT TIMER
1931	006256	001373				BNE	CONT	; CONTINUE UNTIL TIME EXPIRES
1932								
1933	006260	104043				ERROR	+43	; BIT7 OF LKS DOES NOT SET



1934

.SBTTL TEST # 17 - TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY

\*\*\*\*\*  
 \*TEST 17 - TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY  
 \*\*\*\*\*

```

TST17: SCOPE
1935 006262 000004          TST      CTSTFL          ;IS CONSOLE UNDER TEST?
      006264 005737 002624 BEQ      TST20           ;IF NOT, SKIP THIS TEST
      006270 001503          BIT      #BIT6,@SWR     ;ARE LINE CLOCK TESTS INHIBITED?
      006272 032777 000100 172540 BNE     TST20           ;IF YES, SKIP THIS TEST
      006300 001077          JSR     PC,WRPSW       ;SET PSW TO PRIORITY 7
1936 006302 004737 014400          .WORD   340
1937 006306 000340          MOV     @RTCVT,R3      ;SAVE LINE CLOCK VECTOR
1938 006310 017703 174362          MOV     @RTCPW,R4     ;SAVE LINE CLOCK PSW VECTOR
1939 006314 017704 174360          MOV     #2,@RTCVT     ;SET RTC INTERRUPT VECTOR TO ERROR REPORT
1940 006320 012777 006356 174350          MOV     #340,@RTCPW   ;KEEP PRIORITY AT 7
1941 006326 012777 000340 174344          MOV     #BIT6,@LKS    ;CLEAR DONE FLAG AND SET CLOCK INTERRUPT ENABLE
1942 006334 012777 000100 174332          MOV     @LKS          ;WAIT FOR RTC DONE(INTERRUPT REQUEST)
1943 006342 105777 174326 1$: TSTB   @LKS
      006346 100375          BPL     1$
1945 006350 000240          NOP
1946 006352 000240          NOP          ;GIVE TIME FOR ANY INTERRUPTS
1947 006354 000402          BR      3$          ;GIVE TIME FOR ANY INTERRUPTS
                        ;BR, IF NO INTERRUPT OCCURS
1949 006356 022626          2$: CMP   (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1950 006360 104044          ERROR  +44         ;RTC INTERRUPTS AT PRIORITY 7
1952 006362 005077 174306          3$: CLR   @LKS      ;DISABLE RTC INTERRUPTS & CLEAR DONE
1953 006366 012777 006416 174302          MOV   #4,@RTCVT    ;SET RTC INTERRUPT VECTOR FOR ERROR
1954 006374 004737 014400          JSR   PC,WRPSW     ;CHANGE PSW TO PRIORITY 5
1955 006400 000240          .WORD 240
1956 006402 105777 174266          20$: TSTB  @LKS     ;WAIT FOR DONE (INTERRUPT REQUEST)
1957 006406 100375          BPL   20$
1958 006410 000240          NOP
1959 006412 000240          NOP          ;GIVE TIME FOR ANY INTERRUPT
1960 006414 000402          BR    5$          ;GIVE TIME FOR ANY INTERRUPT
                        ;IF NO INTERRUPT - BR TO CONTINUE TEST
1962 006416 022626          4$: CMP   (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1963 006420 104045          ERROR  +45         ;RTC INTERRUPTS WITH INTERRUPTS DISABLED
1965 006422 012777 006460 174246          5$: MOV   #7,@RTCVT  ;POINT RTC VECTOR TO END OF TEST
1966 006430 042777 000200 174236          BIC   #BIT7,@LKS   ;CLEAR CLOCK DONE FLAG
1967 006436 052777 000100 174230          BIS   #BIT6,@LKS   ;ALLOW INTERRUPTS
1968 006444 105777 174224          6$: TSTB  @LKS     ;WAIT FOR RTC DONE
1969 006450 100375          BPL   6$
1970 006452 000240          NOP          ;GIVE TIME FOR INTERRUPT
1972 006454 104046          ERROR  +46         ;RTC INTERRUPT DID NOT OCCUR
1973 006456 000401          BR    8$          ;BRANCH OVER STACK CORRECTION
1974 006460 022626          7$: CMP   (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1975 006462 042777 000100 174204          8$: BIC   #BIT6,@LKS  ;DISABLE INTERRUPTS
1976 006470 010377 174202          MOV   R3,@RTCVT    ;RESTORE LINE CLOCK VECTOR
1977 006474 010477 174200          MOV   R4,@RTCPW    ;RESTORE LINE CLOCK PSW VECTOR
  
```

1978

.SBTTL TEST # 20 - TEST RTC FOR DOUBLE INTERRUPTS

\*\*\*\*\*  
 :\*TEST 20 - TEST RTC FOR DOUBLE INTERRUPTS  
 \*\*\*\*\*

```

1979 006500 000004
1979 006502 005737 002624
      006506 001463
      006510 032777 000100 172322
      006516 001057
1980 006520 000005
1981 006522 017703 174150
1982 006526 017704 174146
1983 006532 012777 006604 174136
1984 006540 012777 000340 174132
1985 006546 004737 014400
1986 006552 000240
1987 006554 012777 000100 174112
1988 006562 105777 174106 1$:
1989 006566 100375
1990 006570 000240
1991 006572 000240
1992 006574 000240
1993 006576 000240
1994
1995 006600 104047
1996 006602 000401
1997 006604 022626 2$:
1998 006606 012777 006634 174062 25$:
1999 006614 004737 014400
2000 006620 000240
2001 006622 000240
2002 006624 000240
2003 006626 000240
2004 006630 000240
2005 006632 000402
2006
2007 006634 022626 3$:
2008 006636 104050
2009
2010
2011 006640 042777 000100 174026 4$:
2012 006646 010377 174024
2013 006652 010477 174022
    
```

```

TST20: SCOPE
      TST CTSTFL ;IS CONSOLE UNDER TEST?
      BEQ TST21 ;IF NOT, SKIP THIS TEST
      BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
      BNE TST21 ;IF YES, SKIP THIS TEST
      RESET ;CLEAR EVERYTHING
      MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
      MOV @RTCPW,R4 ;SAVE LINE CLOCK PSW VECTOR
      MOV #2,@RTCVT ;SET UP RTC INTERRUPT VECTOR
      MCV #340,@RTCPW ;DISALLOW INTERRUPTS AFTER THE INTERRUPT
      JSR PC,WRPSW ;SET PRIORITY TO 5
      .WORD 240
      MOV #BIT6,@LKS ;CLEAR DONE FLAG AND SET CLOCK INTERRUPT ENABLE
      TSTB @LKS ;WAIT FOR DONE
      BPL 1$
      NOP ;GIVE TIME FOR ANY INTERRUPT
      NOP ;GIVE TIME FOR ANY INTERRUPT
      NOP ;GIVE TIME FOR ANY INTERRUPT
      NOP ;GIVE TIME FOR ANY INTERRUPT
      ERROR +47 ;RTC INTERRUPT DID NOT OCCUR
      BR 25$ ;BRANCH OVER STACK CORRECTION
      CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
      MOV #3,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT
      JSR PC,WRPSW ;SET PSW TO PRIORITY 5
      .WORD 240
      NOP ;GIVE SOME TIME FOR AN INTERRUPT
      NOP ;GIVE SOME TIME FOR AN INTERRUPT
      NOP ;GIVE SOME TIME FOR AN INTERRUPT
      NOP ;GIVE SOME TIME FOR AN INTERRUPT
      BR 4$ ;NO INTERRUPT - BR TO END OF TEST
      CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
      ERROR +50 ;INTERRUPT SEQUENCE DID NOT CLEAR
      ;INTERRUPT REQUEST
      BIC #BIT6,@LKS ;DISABLE CLOCK INTERRUPTS
      MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
      MOV R4,@RTCPW ;RESTORE LINE CLOCK PSW VECTOR
    
```

2014

.SBTTL TEST # 21 - TEST THAT RTC INTERRUPT CLEARS WITH RESET  
 :\*\*\*\*\*  
 :\*TEST 21 - TEST THAT RTC INTERRUPT CLEARS WITH RESET  
 :\*\*\*\*\*

2015	006656	000004			TST21: SCOPE		
	006660	005737	002624		TST	CTSTFL	; IS CONSOLE UNDER TEST?
	006664	001442			BEQ	TST22	; IF NOT, SKIP THIS TEST
	006666	032777	000100	172144	BIT	#BIT6,@SWR	; ARE LINE CLOCK TESTS INHIBITED?
	006674	001036			BNE	TST22	; IF YES, SKIP THIS TEST
2016	006676	004737	014400		JSR	PC,WRPSW	; SET PRIORITY TO 7
2017	006702	000340				.WORD 340	
2018	006704	017703	173766		MOV	@RTCVT,R3	; SAVE LINE CLOCK VECTOR
2019	006710	012777	006762	173760	MOV	#2\$,@RTCVT	; POINT RTC VECTOR TO ERROR REPORT
2020	006716	012777	000100	173750	MOV	#BIT6,@LKS	; CLEAR DONE FLAG AND SET CLOCK INTERRUPT ENABLE
2021	006724	105777	173744		1\$: TSTB	@LKS	; WAIT FOR DONE (INTERRUPT REQUEST)
2022	006730	100375			BPL	1\$	
2023	006732	000005			RESET		; CLEAR PENDING INTERRUPT WITH RESET
2024	006734	004737	014400		JSR	PC,WRPSW	; SET PRIORITY TO 5
2025	006740	000240				.WORD 240	
2026	006742	000240			NOP		; GIVE TIME FOR ANY INTERRUPT
2027	006744	000240			NOP		; GIVE TIME FOR ANY INTERRUPT
2028	006746	000240			NOP		; GIVE TIME FOR ANY INTERRUPT
2029	006750	000240			NOP		; GIVE TIME FOR ANY INTERRUPT
2030	006752	042777	000100	173714	BIC	#BIT6,@LKS	; DISALLOW INTERRUPTS
2031	006760	000402			BR	3\$	; BR TO END OF TEST
2032							
2033	006762	022626			2\$: CMP	(SP)+,(SP)+	; RESTORE SP AFTER INTERRUPT
2034	006764	104051			ERROR	+51	; RESET DID NOT CLEAR INTERRUPT
2035							
2036	006766	010377	173704		3\$: MOV	R3,@RTCVT	; RESTORE LINE CLOCK VECTOR

2037

.SBTTL TEST # 22 - TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS  
 :\*\*\*\*\*  
 :\*TEST 22 - TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS  
 :\*\*\*\*\*

2038	006772	000004			TST22: SCOPE		
	006774	005737	002624		TST	CTSTFL	:IS CONSOLE UNDER TEST?
	007000	001447			BEQ	TST23	:IF NOT, SKIP THIS TEST
	007002	032777	000100	172030	BIT	#BIT6,@SWR	:ARE LINE CLOCK TESTS INHIBITED?
	007010	001043			BNE	TST23	:IF YES, SKIP THIS TEST
2039	007012	004737	014400		JSR	PC,WRPSW	:SET PRIORITY TO 7
2040	007016	000340				.WORD 340	
2041	007020	017703	173652		MOV	@RTCVT,R3	:SAVE LINE CLOCK VECTOR
2042	007024	012777	007102	173644	MOV	#2,@RTCVT	:POINT RTC VECTOR TO ERROR REPORT
2043	007032	012777	000100	173634	MOV	#BIT6,@LKS	:CLEAR DONE FLAG AND SET CLOCK INTERRUPT ENABLE
2044	007040	105777	173630		1\$: TSTB	@LKS	:WAIT FOR DONE (INTERRUPT REQUEST)
2045	007044	100375			BPL	1\$	
2046	007046	042777	000200	173620	BIC	#BIT7,@LKS	:CLEAR DONE & INTERRUPT
2047	007054	004737	014400		JSR	PC,WRPSW	:ALLOW INTERRUPTS
2048	007060	000240				.WORD 240	
2049	007062	000240			NOP		:GIVE TIME FOR ANY INTERRUPT
2050	007064	000240			NOP		:GIVE TIME FOR ANY INTERRUPT
2051	007066	000240			NOP		:GIVE TIME FOR ANY INTERRUPT
2052	007070	000240			NOP		:GIVE TIME FOR ANY INTERRUPT
2053	007072	042777	000100	173574	BIC	#BIT6,@LKS	:DISALLOW INTERRUPTS
2054	007100	000402			BR	3\$	:TO END OF TEST
2055							
2056							
2057	007102	022626			2\$: CMP	(SP)+,(SP)+	:RESTORE SP AFTER INTERRUPT
2058	007104	104052			ERROR	+52	:CLEARING BIT7 OF LKS DID NOT CLEAR INTERRUPT
2059							
2060	007106	010377	173564		3\$: MOV	R3,@RTCVT	:RESTORE LINE CLOCK VECTOR
2061	007112	004737	014400		JSR	PC,WRPSW	:RESTORE PRIORITY TO 7
2062	007116	000340				.WORD 340	

2063

.SBTTL TEST # 23 - TEST CLOCK REPEATABILITY  
 :\*\*\*\*\*  
 :\*TEST 23 - TEST CLOCK REPEATABILITY  
 :\*\*\*\*\*

2064	007120	000004				TST23: SCOPE		
	007122	065737	002624			TST	CTSTFL	:IS CONSOLE UNDER TEST?
	007126	001467				BEQ	TST24	:IF NOT, SKIP THIS TEST
	007130	032777	000100	171702		BIT	#BIT6,@SWR	:ARE LINE CLOCK TESTS INHIBITED?
	007136	001063				BNE	TST24	:IF YES, SKIP THIS TEST
2065	007140	042777	000100	173526		BIC	#BIT6,@LKS	:DISALLOW INTERRUPTS
2066								
2067	007146	005000				CLR	R0	:CLEAR A TIMER
2068	007150	012701	177777			MOV	#-1,R1	:SET A FLAG INDICATING FIRST PASS THRU THIS LOOP
2069	007154	005002			1\$:	CLR	R2	:CLEAR CLOCK COUNTER
2070	007156	005077	173512			CLR	@LKS	:CLEAR DONE
2071	007162	105777	173506		2\$:	TSTB	@LKS	:SYNC ON DONE
2072	007166	100375				BPL	2\$	
2073	007170	005077	173500			CLR	@LKS	:CLEAR DONE
2074	007174	105777	173474		3\$:	TSTB	@LKS	:IS CLOCK DONE?
2075	007200	100003				BPL	4\$	:BR IF NOT, TO INCREMENT TIMER
2076	007202	005202				INC	R2	:IF DONE, INCREMENT CLOCK COUNT
2077	007204	005077	173464			CLR	@LKS	:CLEAR DONE
2078	007210	005200			4\$:	INC	R0	:INCREMENT TIMER
2079	007212	001370				BNE	3\$	:BR IF TIME REMAINS
2080	007214	005201				INC	R1	:INCREMENT LOOP PASS FLAG
2081	007216	001003				BNF	CMPARE	:BR IF TWO PASSES HAVE BEEN MADE
2082	007220	010237	002336			MOV	R2,FIRST	:IF NOT, STORE FIRST CLOCK COUNT
2083	007224	000753				BR	1\$	:DO LOOP AGAIN
2084	007226	013701	002336		CMPARE:	MOV	FIRST,R1	:RECALL FIRST CLOCK COUNT
2085	007232	160201				SUB	R2,R1	:CALCULATE DIFFERENCE OF TWO COUNTS
2086	007234	100001				BPL	TOLER	:IF POSITIVE,SKIP NEGATION OF DIFFERENCE
2087	007236	005401				NEG	R1	:MAKE DIFFERENCE A POSITIVE NUMBER
2088	007240	032737	000001	003002	TOLER:	BIT	#BIT0,FLAG44	** IS THIS A 11/44
2089	007246	001403				BEQ	6\$	** NO
2090	007250	020127	000003			CMP	R1,#3	** YES; COMPARE DIFFERENCE WITH DESIRED
2091								** TOLERANCE OF 3
2092	007254	000402				BR	7\$	**
2093	007256	020127	000001		6\$:	CMP	R1,#1	:COMPARE DIFFERENCE WITH DESIRED TOLERANCE
2094	007262	003403			7\$:	BLE	5\$	:BR, IF LOWER/EQUAL TO TOLERANCE
2095								
2096	007264	010237	002616			MOV	R2,SECND	:STORE SECOND COUNT
2097	007270	104053				ERROR	+5\$	:CLOCK REPEATABILITY ERROR
2098								
2099	007272	032777	000020	171540	5\$:	BIT	#BIT4,@SWR	:CLOCK TESTS ONLY?
2100	007300	001402				BEQ	TST24	:BR IF NOT
2101	007302	000137	014224			JMP	\$EOP	:ELSE, JUMP TO END OF PASS ROUTINE

2102

```
.SBTTL TEST # 24 - TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
:*****
:*TEST 24 - TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
:*****
```

```
TST24: SCOPE
2103 007306 000004          BIC    #BIT6,@TCSR    ;CLEAR TRANSMIT INTERRUPT ENABLE
2104 007310 042777 000100 173322    MOV    @TVECT,R3     ;SAVE XMIT VECTOR
2105 007322 012777 007346 173320    MOV    #2$,@TVECT    ;POINT XMIT VECTOR TO ERROR REPORT
2106 007330 105777 173304    1$:   TSTB   @TCSR      ;WAIT FOR DONE
2107 007334 100375          BPL    1$
2108 007336 004737 014400    JSR    PC,WRPSW      ;SET PSW TO PRIORITY 3
2109 007342 000140          .WORD  140
2110 007344 000402          BR     3$
2111
2112 007346 022626          2$:   CMP    (SP)+,(SP)+  ;RESTORE SP AFTER INTERRUPT
2113 007350 104054          ERROR +54
2114
2115 007352 012777 007402 173270    3$:   MOV    #4$,@TVECT    ;XMIT INTERRUPTS WITH INTERRUPT ENABLE CLEAR
2116 007360 052777 000100 173252    BIS    #BIT6,@TCSR    ;SET XMIT VECTOR TO END OF TEST
2117 007366 000240          NOP                                ;ENABLE INTERRUPTS
2118 007370 000240          NOP                                ;**
2119 007372 000240          NOP                                ;**
2120 007374 000240          NOP                                ;**
2121
2122 007376 104055          ERROR +55                    ;XMIT DID NOT INTERRUPT
2123 007400 000401          BR     5$                    ;BRANCH OVER STACK CORRECTION
2124 007402 022626          4$:   CMP    (SP)+,(SP)+  ;RESTORE SP AFTER INTERRUPT
2125 007404 042777 000100 173226    5$:   BIC    #BIT6,@TCSR    ;DISABLE INTERRUPTS
2126 007412 010377 173232    MOV    R3,@TVECT      ;RESTORE XMIT VECTOR
```

2127

.SBTTL TEST # 25 - TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED  
 :\*\*\*\*\*  
 :\*TEST 25 - TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED  
 :\*\*\*\*\*

```

TST25: SCOPE
2128 007416 000004
2128 007420 042777 000100 173212 BIC #BIT6,@TCSR ;DISABLE INTERRUPT
2129 007426 004737 014400 JSR PC,WRPSW ;SET PSW TO PRIORITY 7
2130 007432 000340 .WORD 340
2131 007434 017703 173210 MOV @TVECT,R3 ;SAVE XMIT VECTOR
2132 007440 012777 007474 173202 MOV #2$,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT
2133 007446 105777 173166 1$: TSTB @TCSR ;WAIT FOR DONE
2134 007452 100375 BPL 1$
2135 007454 052777 000100 173156 BIS #BIT6,@TCSR ;ENABLE INTERRUPT
2136 007462 000240 NOP ;**
2137 007464 000240 NOP ;**
2138 007466 000240 NOP ;**
2139 007470 000240 NOP ;**
2140 007472 000402 BR 3$ ;CONTINUE TEST
2141
2142 007474 022626 2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
2143 007476 104056 ERROR +56
2144 ;XMIT INTERRUPTS AT PRIORITY=7
2145 007500 042777 000100 173132 3$: BIC #BIT6,@TCSR ;CLEAR INTERRUPT ENABLE
2146 007506 012777 007534 173134 MOV #4$,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT
2147 007514 004737 014400 JSR PC,WRPSW ;SET PSW TO PRIORITY 3
2148 007520 000140 .WORD 140
2149 007522 000240 NOP ;**
2150 007524 000240 NOP ;**
2151 007526 000240 NOP ;**
2152 007530 000240 NOP ;**
2153 007532 000402 BR 5$ ;BR TO END OF TEST-NO INTERRUPT
2154
2155 007534 022626 4$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
2156 007536 104057 ERROR +57
2157 ;XMIT INTERRUPT OCCURES WITH BIT6 CLEAR
2158 007540 010377 173104 5$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
    
```

2159

.SBTTL TEST # 26 - TEST TRANSMITTER FOR DOUBLE INTERRUPTS  
 :\*\*\*\*\*  
 :\*TEST 26 - TEST TRANSMITTER FOR DOUBLE INTERRUPTS  
 :\*\*\*\*\*

2160	007544	000004				TST26: SCOPE		
2160	007546	042777	000100	173064		BIC	#BIT6,@TCSR	;CLEAR INTERRUPT ENABLE
2161	007554	017703	173070			MOV	@TVECT,R3	;SAVE XMIT VECTOR
2162	007560	017704	173066			MOV	@TPSW,R4	;SAVE XMIT PSW VECTOR
2163	007564	012777	007636	173056		MOV	#2\$,@TVECT	;SET UP XMIT VECTOR
2164	007572	012777	000340	173052		MOV	#340,@TPSW	;SET PIO 7 AFTER INTERRUPT
2165	007600	004737	014400			JSR	PC,WRPSW	;SET PSW TO PRIORITY 3
2166	007604	000140					.WORD 140	
2167	007606	105777	173026		1\$:	TSTB	@TCSR	;WAIT FOR DONE
2168	007612	100375				BPL	1\$	
2169	007614	052777	000100	173016		BIS	#BIT6,@TCSR	;ENABLE INTERRUPTS
2170	007622	000240				NOP		;**
2171	007624	000240				NOP		;**
2172	007626	000240				NOP		;**
2173	007630	000240				NOP		;**
2174								
2175	007632	104060				ERROR	+60	;XMIT INTERRUPT DID NOT OCCUR
2176	007634	000401				BR	25\$	;BRANCH OVER STACK CORRECTION INTERRUPT
2177	007636	022626			2\$:	CMP	(SP)+,(SP)+	;RESTORE SP AFTER INTERRUPT
2178	007640	012777	007674	173002	25\$:	MOV	#4\$,@TVECT	;POINT XMIT VECTOR TO ERROR
2179	007646	004737	014400			JSR	PC,WRPSW	;SET PSW TO PRIORITY 3
2180	007652	000140					.WORD 140	
2181	007654	000240				NOP		;GIVE TIME FOR ANY INTERRUPTS
2182	007656	000240				NOP		;GIVE TIME FOR ANY INTERRUPTS
2183	007660	000240				NOP		;GIVE TIME FOR ANY INTERRUPTS
2184	007662	000240				NOP		;GIVE TIME FOR ANY INTERRUPTS
2185	007664	042777	000100	172746		BIC	#BIT6,@TCSR	;DISABLE INTERRUPTS
2186	007672	000402				BR	5\$	;BR TO END OF TEST
2187								
2188	007674	022626			4\$:	CMP	(SP)+,(SP)+	;RESTORE SP AFTER INTERRUPT
2189	007676	104061				ERROR	+61	
2190								;XMIT RE-INTERRUPTED
2191	007700	010377	172744		5\$:	MOV	R3,@TVECT	;RESTORE XMIT VECTOR
2192	007704	010477	172742			MOV	R4,@TPSW	;RESTORE XMIT PSW VECTOR
2193								



2194

.SBTTL TEST # 27 - TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF  
 :\*\*\*\*\*  
 :\*TEST 27 - TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF  
 :\*\*\*\*\*

```

TST27: SCOPE
2195 007710 000004          BIC    #BIT6,@TCSR    ;DISABLE INTERRUPTS
2196 007712 042777 000100 172720 JSR    PC,WRPSW      ;SET PSW TO PRIORITY 7
2197 007720 004737 014400      JSR    PC,WRPSW      ;SET PSW TO PRIORITY 7
2198 007724 000340          .WORD  340
2199 007726 017703 172716      MOV    @TVECT,R3     ;SAVE XMIT VECTOR
2200 007732 012777 010020 172710 MOV    #2,@TVECT     ;POINT XMIT VECTOR TO ERROR
2201 007740 052777 000004 172672 BIS    #BIT2,@TCSR   ;** ENABLE MAINT. WRAP
2202 007746 052777 000100 172664 BIS    #BIT6,@TCSR   ;ENABLE INTERRUPTS
2203 007754 005077 172662      CLR    @TBUF        ;LOAD TBUF
2204 007760 105777 172654      CLR    @TBUF        ;LOAD TBUF
2205 007766 005077 172650      TSTB  @TCSR        ;WAIT FOR DONE (INTERRUPT)
2206 007772 004737 014400      BPL    1$           ;WAIT FOR DONE (INTERRUPT)
2207 007776 000140          CLR    @TBUF        ;FILL SECOND BUFFER TO RESET INT.
2208 010000 000240          JSR    PC,WRPSW      ;ALLOW INTERRUPTS
2209 010002 000240          .WORD  140
2210 010004 000240          NOP                    ;GIVE TIME FOR ANY INTERRUPTS
2211 010006 000240          NOP                    ;GIVE TIME FOR ANY INTERRUPTS
2212 010010 042777 000100 172622 NOP                    ;GIVE TIME FOR ANY INTERRUPTS
2213 010016 000405          BIC    #BIT6,@TCSR   ;GIVE TIME FOR ANY INTERRUPTS
2214 010018 000405          BR     3$           ;DISABLE INTERRUPTS
2215 010020 022626          BR     3$           ;BR TO END OF TEST
2216 010022 042777 000004 172630 CMP    (SP)+,(SP)+   ;RESTORE SP AFTER INTERRUPT
2217 010030 104062          BIC    #BIT2,@TCSR   ;** DISABLE MAINTENANCE MODE FOR
2218 010032 010377 172612      BR     40$          ;** CONSOLE TO ALLOW FOR COMMUNICATION
2219 010036 005000          ERROR  +62         ;** WITH TERMINAL.
2220 010040 012701 000002      MOV    R3,@TVECT    ;LOADING TBUF DID NOT CLEAR INTERRUPT.
2221 010044 005300          MOV    R0            ;RESTORE XMIT VECTOR
2222 010046 001376          CLR    R0            ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
2223 010050 005301          MOV    #2,R1        ;** THAT MIGHT BE IN THE PROCESS OF BEING
2224 010052 001374          DEC    R0            ;** TRANSMITTED TO FINISH BEFORE MAINTENANCE
2225 010054 001374          BNE   40$           ;** WRAP FOR THE UART UNDER TEST IS DISABLED.
2226 010056 001374          DEC    R1            ;** THIS WILL INHIBIT ANY COMMUNICATION
2227 010058 001374          DEC    R1            ;** TO HARDWARE MEDIA SUCH AS TUSB THAT MIGHT
2228 010060 001374          BNE   40$           ;** BE ATTACHED TO UART UNDER TEST.
  
```

2222

.SBTTL TEST # 30 - TEST THAT RCVR ACTIVE & DONE SET & CLEAR  
 :\*\*\*\*\*  
 :\*TEST 30 - TEST THAT RCVR ACTIVE & DONE SET & CLEAR  
 :\*\*\*\*\*

2223	010054	000004	000001	003002	TST30:	SCOPE			
2224	010056	032737				BIT	#BIT0,FLAG44	;	**
2225	010064	001067				BNE	RCVDON	;	**
2226	010066	000005				RESET		;	CLEAR EVERYTHING
2227	010070	052777	000004	172542		BIS	#BIT2,@TCSR	;	SET MAINTENANCE WRAP
	010076	005000				CLR	R0	;	** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	010100	012701	000002			MOV	#2,R1	;	** THAT MIGHT BE IN THE PROCESS OF BEING
								;	** RECEIVED TO FINISH AFTER MAINTENANCE
								;	** WRAP FOR THE UART UNDER TEST IS ENABLED.
								;	** READ TO CLEAR DONE
	010104	105777	172526		42\$:	TSTB	@RBUF	;	**
	010110	005300				DEC	R0	;	**
	010112	001374				BNE	42\$	;	**
	010114	005301				DEC	R1	;	**
	010116	001372				BNE	42\$	;	**
2228	010120	005000				CLR	R0	;	CLEAR A TIMER
2229	010122	005077	172514			CLR	@TBUF	;	LOAD TRANSMIT BUFFER
2230	010126	032777	004000	172500	WACTV:	BIT	#BIT11,@RCSR	;	TEST RCVR ACTIVE BIT
2231	010134	001006				BNE	2\$	;	BR IF SET
2232	010136	005200				INC	R0	;	INCREMENT TIMER IF NOT SET
2233	010140	001372				BNE	WACTV	;	CONTINUE WAIT IF TIME REMAINS
2234	010142	042777	000004	172510		BIC	#BIT2,@CTCSR	;	** DISABLE MAINTENANCE MODE FOR
								;	** CONSOLE TO ALLOW FOR COMMUNICATION
								;	** WITH TERMINAL.
2235									
2236	010150	104063				ERROR	+63	;	RCVR ACTIVE DID NOT SET WHILE RECEIVING
2237									
2238	010152				2\$:				
	010152	005000				CLR	R0	;	** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	010154	012701	000002			MOV	#2,R1	;	** THAT MIGHT BE IN THE PROCESS OF BEING
	010160	005300			40\$:	DEC	R0	;	** TRANSMITTED TO FINISH BEFORE MAINTENANCE
	010162	001376				BNE	40\$	;	** WRAP FOR THE UART UNDER TEST IS DISABLED.
	010164	005301				DEC	R1	;	** THIS WILL INHIBIT ANY COMMUNICATION
	010166	001374				BNE	40\$	;	** TO HARDWARE MEDIA SUCH AS TU58 THAT MIGHT
								;	** BE ATTACHED TO UART UNDER TEST.
2239	010170	000005				RESET			
2240	010172	032777	004000	172434		BIT	#BIT11,@RCSR	;	VERIFY "INIT" CLEARS RCV ACTIVE
2241	010200	001401				BEQ	3\$		
2242									
2243	010202	104115				ERROR	+115	;	INIT DID NOT CLEAR RCV ACTIVE
2244									
2245	010204	005000			3\$:	CLR	R0	;	CLEAR A TIMER
2246	010206	052777	000004	172424		BIS	#BIT2,@TCSR	;	SET MAINTENANCE WRAP
2247	010214	062700	000000		WT:	ADD	#0,R0	;	WAIT AT LEAST ONE BIT TIME
2248	010220	005200				INC	R0		
2249	010222	001374				BNE	WT		
2250	010224	033777	004000	172402		BIT	BIT11,@RCSR	;	VERIFY RCV ACTIVE STILL CLEAR
2251	010232	001404				BEQ	RCVDON	;	BR IF CLEAR
2252									
2253	010234	042777	000004	172416		BIC	#BIT2,@CTCSR	;	** DISABLE MAINTENANCE MODE FOR
								;	** CONSOLE TO ALLOW FOR COMMUNICATION
								;	** WITH TERMINAL.
2254	010242	104116				ERROR	+116	;	RCV ACTIVE WITHOUT "START" BIT

```

2255
2256 010244 052777 000004 172366 RCVDON: BIS #BIT2,@TCSR ;SET MAINTENCE WRAP
2257 010252 035000 CLR RO ;CLEAR TIMER
2258 010254 035077 172362 CLR @TBUF ;LOAD TRANSMIT BUFFER
2259 010260 105777 172350 WDONE: TSTB @RCSR ;CHECK FOR RECEIVER DONE
2260 010264 100406 BMI 5$ ;BR, IF DONE
2261 010266 005200 INC RO ;INCREMENT TIMER, IF NOT DONE
2262 010270 001373 BNE WDONE ;CONTINUE WAIT IF TIME REMAINS
2263 010272 042777 000004 172360 BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.

2264 010300 104064 ERROR +64
2265 ;RECEIVER DONE NEVER SET
2266
2267 010302 032737 000001 003002 5$: BIT #BIT0,FLAG44 ;** 11/44??
2268 010310 001010 BNE 6$ ;** DO NOT EXECUTE THIS SECTION
2269 010312 032777 004000 172314 BIT #BIT11,@RCSR ;CHECK FOR RCVR ACTIVE CLEAR
2270 010320 001404 BEQ 6$ ;BR, IF CLEAR
2271 010322 042777 000004 172330 BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.

2272 010330 104065 ERROR +65
2273 ;RCVR ACTIVE DID NOT CLEAR WITH RCVR DONE
2274
2275 010332 000005 6$: RESET ;CLEAR DONE WITH RESET
2276 010334 105777 172274 TSTB @RCSR ;CHECK FOR DONE CLEAR
2277 010340 001404 BEQ 7$
2278
2279 010342 042777 000004 172310 BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.

2280 010350 104066 ERROR +66
2281 ;RESET DID NOT CLEAR RCVR DONE
2282
2283 010352 7$:
2284 010352 042777 000004 172300 BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.
    
```

2285

```
.SBTTL TEST # 31 - TEST THAT READING RBUF CLEARS RECEIVER DONE
:*****
:*TEST 31 - TEST THAT READING RBUF CLEARS RECEIVER DONE
:*****
```

```
TST31: SCOPE
        RESET
2286 010360 000004
2287 010362 000005
2288 010364 052777 000004 172246
        BIS #BIT2,@TCSR ;CLEAR EVERYTHING
        CLR R0 ;SET MAINTENANCE WRAP
        MOV #2,R1 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
; ** THAT MIGHT BE IN THE PROCESS OF BEING
; ** RECEIVED TO FINISH AFTER MAINTENANCE
; ** WRAP FOR THE UART UNDER TEST IS ENABLED.
; ** READ TO CLEAR DONE

        010400 105777 172232 42$: TSTB @RBUF
        DEC R0
        BNE 42$
        DEC R1
        BNE 42$
2289 010414 005077 172222 CLR @TBUF ;LOAD TRANSMITTER
2290 010420 105777 172210 1$: TSTB @RCSR ;WAIT FOR RECEIVER DONE
2291 010424 100375 BPL 1$
2292 010426 017700 172204 MOV @RBUF,R0 ;READ RECEIVE BUFFER
2293 010432 042777 000004 172220 BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.
;CHECK FOR RECEIVE DONE CLEAR
; ** BR, IF CLEAR TO NEXT TEST

2294 010440 105777 172170 TSTB @RCSR
2295 010444 001401 BEQ 10$
2296 010446 104070 ERROR +70
2297 ;READING RBUF DID NOT CLEAR RCVR DONE
2298 010450 000240 10$: NOP
2299
```

2300

.SBTTL TEST # 32 - TEST THAT RDR ENABLE CLEARS RECEIVER DONE FLAG  
 :\*\*\*\*\*  
 :\*TEST 32 - TEST THAT RDR ENABLE CLEARS RECEIVER DONE FLAG  
 :\*\*\*\*\*

2301	010452	000004			TST32: SCOPE		
2301	010454	032737	000001	003002	BIT	#BIT0,FLAG44	;** 11/44 ??
2302	010462	001044			BNE	10\$	;** YES DO NOT EXECUTE THIS TEST
2303	010464	000005			RESET		;CLEAR EVERYTHING
2304	010466	052777	000001	172140	BIS	#BIT0,@RCSR	;SET RDR ENABLE
2305	010474	052777	000004	172136	BIS	#BIT2,@TCSR	;SET MAINTENANCE WRAP
2306	010502	005000			CLR	R0	;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	010504	012701	000002		MOV	#2,R1	;** THAT MIGHT BE IN THE PROCESS OF BEING
							;** RECEIVED TO FINISH AFTER MAINTENANCE
							;** WRAP FOR THE UART UNDER TEST IS ENABLED.
							;** READ TO CLEAR DONE
	010510	105777	172122		42\$: TSTB	@RBUF	
	010514	005300			DEC	R0	;**
	010516	001374			BNE	42\$	;**
	010520	005301			DEC	R1	;**
	010522	001372			BNE	42\$	;**
2307	010524	005077	172112		CLR	@TBUF	;LOAD TRANSMITTER
2308	010530	105777	172100		1\$: TSTB	@RCSR	;WAIT FOR RECEIVER DONE
2309	010534	100375			BPL	1\$	
2310	010536	032777	000001	172070	BIT	#BIT0,@RCSR	;VERIFY RCV ACTIVE CLEARED RDR ENABLE
2311	010544	001401			BEQ	2\$	;BR IF CLEAR
2312							
2313	010546	104117			ERROR	+117	;RDR ENABLE NOT CLEARED WITH RCV ACTIVE
2314							
2315	010550	052777	000001	172056	2\$: BIS	#BIT0,@RCSR	;CLEAR DONE BY SETTING RDR ENABLE
2316	010556	105777	172052		TSTB	@RCSR	;CHECK FOR DONE CLEAR
2317	010562	001404			BEQ	10\$	;** BR, IF CLEAR TO NEXT TEST
2318	010564	042777	000004	172066	BIC	#BIT2,@TCSR	;** DISABLE MAINTENANCE MODE FOR
							;** CONSOLE TO ALLOW FOR COMMUNICATION
							;** WITH TERMINAL.
2319	010572	104067			ERROR	+67	
2320							;SETTING RDR ENABLE DID NOT CLEAR RCVR DONE
2321	010574	000240			10\$: NOP		;**

```

2322 .SBTTL TEST # 33 - TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
*****
*TEST 33 - TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
*****
TST33: SCOPE
2323 010576 000004          BIC    #BIT6,@TCSR    ;DISABLE TRANSMIT INTERRUPTS
2324 010600 042777 000100 172032 BIC    #BIT6,@RCR    ;DISABLE RECEIVER INTERRUPTS
2325 010614 052777 000004 172020 BIS    #BIT2,@TCSR    ;SET MAINTENANCE WRAP
2326 010622 005000          CLR    R0              ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
    010624 012701 000002          MOV    #2,R1          ;** THAT MIGHT BE IN THE PROCESS OF BEING
    ;** RECEIVED TO FINISH AFTER MAINTENANCE
    ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
    ;** READ TO CLEAR DONE
    010630 105777 172002          42$:  TSTB   @RBUF      ;**
    010634 005300          DEC    R0              ;**
    010636 001374          BNE   42$              ;**
    010640 005301          DEC    R1              ;**
    010642 001372          BNE   42$              ;**
2327 010644 017703 171774          MOV    @RVECT,R3      ;SAVE RECEIVE VECTOR
2328 010650 012777 010706 171766 MOV    #2$,@RVECT     ;POINT RCV VECTOR TO ERROR REPORT
2329 010656 004737 014400          JSR   PC,WRPSW        ;SET PSW TO PRIORITY 3
2330 010662 000140          .WORD 140
2331 010664 005077 171752          CLR    @TBUF          ;SEND A CHARACTER
2332 010670 105777 171740          1$:  TSTB   @RCR          ;WAIT FOR RECEIVER DONE
2333 010674 100375          BPL   1$
2334 010676 042777 000004 171754 BIC    #BIT2,@CTCSR   ;** DISABLE MAINTENANCE MODE FOR
    ;** CONSOLE TO ALLOW FOR COMMUNICATION
    ;** WITH TERMINAL.
    2335 010704 000405          BR    3$              ;CONTINUE TEST
2336
2337 010706
2338 010706 042777 000004 171744 2$:  BIC    #BIT2,@CTCSR   ;** DISABLE MAINTENANCE MODE FOR
    ;** CONSOLE TO ALLOW FOR COMMUNICATION
    ;** WITH TERMINAL.
2339 010714 022626          CMP   (SP)+,(SP)+    ;RESTORE SP AFTER INTERRUPT
2340 010716 104071          ERROR +71           ;RECEIVER INTERRUPTS WITH INT. ENABLE CLEAR
2341
2342 010720 012777 010756 171716 3$:  MOV    #4$,@RVECT     ;POINT RCV VECTOR TO END OF TEST
2343 010726 052777 000100 171700 BIS    #BIT6,@RCR     ;ENABLE RCV INTERRUPTS
2344 010734 000240          NOP
2345 010736 000240          NOP
2346 010740 000240          NOP
2347 010742 000240          NOP
2348 010744 042777 000004 171706 BIC    #BIT2,@CTCSR   ;** DISABLE MAINTENANCE MODE FOR
    ;** CONSOLE TO ALLOW FOR COMMUNICATION
    ;** WITH TERMINAL.
    2349 010752 104072          ERROR +72           ;RCVR DID NOT INTERRUPT
2350 010754 000401          BR    5$              ;BRANCH OVER STACK CORRECTION
2351 010756 022626          4$:  CMP   (SP)+,(SP)+    ;RESTORE SP AFTER INTERRUPT
2352 010760 042777 000100 171646 5$:  BIC    #BIT6,@RCR     ;DISABLE INTERRUPT
2353 010766 042777 000004 171664 BIC    #BIT2,@CTCSR   ;** DISABLE MAINTENANCE MODE FOR
    ;** CONSOLE TO ALLOW FOR COMMUNICATION
    ;** WITH TERMINAL.
2354 010774 010377 171644          MOV    R3,@RVECT     ;RESTORE RECEIVE VECTOR
  
```

2355

.SBTTL TEST # 34 - TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED  
 :\*\*\*\*\*  
 :\*TEST 34 - TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED  
 :\*\*\*\*\*

```

2356 011000 000004
2356 011002 000005
2357 011004 004737 014400
2358 011010 000340
2359 011012 017703 171626
2360 011016 012777 011114 171620
2361 011024 052777 000004 171606
2362 011032 005000
    011034 012701 000002

    011040 105777 171572
    011044 005300
    011046 001374
    011050 005301
    011052 001372
2363 011054 105777 167770
2364 011060 100375
2365 011062 005077 171554
2366 011066 105777 171542
2367 011072 100375
2368 011074 052777 000100 171532
2369 011102 000240
2370 011104 000240
2371 011106 000240
2372 011110 000240
2373 011112 000405
2374 011114
2375 011114 042777 000004 171536

2376 011122 022626
2377 011124 104073
2378
2379 011126 042777 000100 171500
2380 011134 012777 011164 171502
2381 011142 004737 014400
2382 011146 000140
2383 011150 000240
2384 011152 000240
2385 011154 042777 000004 171476

2386 011162 000405
2387
2388 011164
2389 011164 042777 000004 171466

2390 011172 022626
2391 011174 104074
2392
  
```

```

TST34: SCOPE
        RESET
        JSR      PC,WRPSW          ;CLEAR EVERYTHING
        .WORD   340              ;SET PSW TO PRIORITY 7
        MOV     @RVECT,R3        ;SAVE RECEIVE VECTOR
        MOV     #2,@RVECT       ;POINT RCVR VECTOR TO ERROR REPORT
        BIS     #BIT2,@TCSR     ;SET MAINTENANCE WRAP
        CLR     R0              ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
        MOV     #2,R1           ;** THAT MIGHT BE IN THE PROCESS OF BEING
        ;** RECEIVED TO FINISH AFTER MAINTENANCE
        ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
        ;** READ TO CLEAR DONE
42$:   TSTB    @RBUF
        DEC    R0
        BNE    42$
        DEC    R1
        BNE    42$
6$:   TSTB    @STPS
        BPL    6$
        CLR    @TBUF
        TSTB   @RCSR
        BPL    1$
1$:   BIS     #BIT6,@RCSR       ;ENABLE INTERRUPTS
        NOP
        NOP
        NOP
        NOP
        BR     3$
2$:   BIC     #BIT2,@TCSR       ;** DISABLE MAINTENANCE MODE FOR
        ;** CONSOLE TO ALLOW FOR COMMUNICATION
        ;** WITH TERMINAL.
        CMP    (SP)+,(SP)+     ;RESTORE SP AFTER INTERRUPT
        ERROR  +73            ;RCVR INTERRUPTS AT PRIORITY 7
3$:   BIC     #BIT6,@RCSR       ;CLEAR INTERRUPT ENABLE
        MOV     #4,@RVECT      ;POINT RCVR VECTOR TO ERROR REPORT
        JSR     PC,WRPSW       ;SET PSW TO PRIORITY 3
        .WORD   140
        NOP
        NOP
        BIC     #BIT2,@TCSR     ;** DISABLE MAINTENANCE MODE FOR
        ;** CONSOLE TO ALLOW FOR COMMUNICATION
        ;** WITH TERMINAL.
        BR     5$
5$:   BR     5$                ;BR TO END OF TEST, IF NO INTERRUPT
4$:   BIC     #BIT2,@TCSR       ;** DISABLE MAINTENANCE MODE FOR
        ;** CONSOLE TO ALLOW FOR COMMUNICATION
        ;** WITH TERMINAL.
        CMP    (SP)+,(SP)+     ;RESTORE SP AFTER INTERRUPT
        ERROR  +74
        ;RCVR INTERRUPT REQUEST PASSED WITH BIT6 CLEAR
  
```

```

2393 011176 010377 171442 5$: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
2394 .SBTTL TEST # 35 - TEST RECEIVER FOR DOUBLE INTERRUPTS
;*****
;TEST 35 - TEST RECEIVER FOR DOUBLE INTERRUPTS
;*****
TST35: SCOPE
RESET ;CLEAR EVERYTHING
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV @RPSW,R4 ;SAVE RECEIVE PSW VECTOR
MOV #2,@RVECT ;POINT RCV VECTOR TO CONTINUE TEST
MOV #340,@RPSW ;SET PRIORITY TO 7 AFTER INTERRUPT
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
;WORD 140
2395 011202 000004
2395 011204 000005
2396 011206 017703 171432
2397 011212 017704 171430
2398 011216 012777 011332 171420
2399 011224 012777 000340 171414
2400 011232 004737 014400
2401 011236 000140
2402 011240 052777 000004 171372 BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
2403 011246 005000 CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
011250 012701 000002 MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS ENABLED.
;** READ TO CLEAR DONE
011254 105777 171356 42$: TSTB @RBUF
011260 005300 DEC R0
011262 001374 BNE 42$
011264 005301 DEC R1
011266 001372 BNE 42$
2404 011270 005077 171346 CLR @TBUF ;SEND A CHARACTER
2405 011274 105777 171334 1$: TSTB @RCSR ;WAIT FOR RCVR DONE
2406 011300 100375
2407 011302 042777 000004 171350 BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
2408 011310 052777 000100 171316 BIS #BIT6,@RCSR ;ENABLE RCV INTERRUPTS
2409 011316 000240 NOP ;** GIVE SOME TIME
2410 011320 000240 NOP ;** GIVE SOME TIME
2411 011322 000240 NOP ;** GIVE SOME TIME
2412 011324 000240 NOP ;** GIVE SOME TIME
2413
2414 011326 104075 ERROR +75 ;RCVR INTERRUPT DID NOT OCCUR
2415 011330 000401 BR 25$ ;BRANCH OVER STACK CORRECTION
2416 011332 022626 2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
2417 011334 012777 011400 171302 25$: MOV #3,@RVECT ;POINT RCV VECTOR TO ERROR REPORT
2418 011342 004737 014400 JSR PC,WRPSW ;RESET PSW TO PRIORITY 3
2419 011346 000140 .WORD 140
2420 011350 000240 NOP ;** GIVE SOME TIME
2421 011352 000240 NOP ;** GIVE SOME TIME
2422 011354 000240 NOP ;** GIVE SOME TIME
2423 011356 000240 NOP ;** GIVE SOME TIME
2424 011360 042777 000100 171246 BIC #BIT6,@RCSR ;CLEAR INTERRUPT ENABLE
2425 011366 010377 171252 MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
2426 011372 010477 171250 MOV R4,@RPSW ;RESTORE RECEIVE PSW VECTOR
2427 011376 000402 BR 4$ ;BR TO END OF TEST
2428
2429 011400 022626 3$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
2430 011402 104076 ERROR +76
2431
2432 011404 010377 171234 4$: MOV R3,@RVECT ;RECEIVER RE-INTERRUPTED
;RESTORE RECEIVE VECTOR
    
```



2433

.SBTTL TEST # 36 - TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF

\*\*\*\*\*  
 :\*TEST 36 - TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF  
 \*\*\*\*\*

2434	011410	000004			TST36:	SCOPE			
	011412	000005				RESET			:CLEAR EVERYTHING
2435	011414	004737	014400			JSR	PC,WRPSW		:SET PSW PRIORITY TO 7
2436	011420	000340					.WORD 340		
2437	011422	017703	171216			MOV	@RVECT,R3		:SAVE RECEIVE VECTOR
2438	011426	012777	011544	171210		MOV	#2,@RVECT		:POINT RCV VECTOR TO ERROR REPORT
2439	011434	052777	000100	171172		BIS	#BIT6,@RCSR		:SET RCVR INTERRUPT ENABLE
2440	011442	052777	000004	171170		BIS	#BIT2,@TCSR		:SET MAINTENANCE WRAP
2441	011450	005000				CLR	R0		:** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	011452	012701	000002			MOV	#2,R1		:** THAT MIGHT BE IN THE PROCESS OF BEING
									:** RECEIVED TO FINISH AFTER MAINTENANCE
									:** WRAP FOR THE UART UNDER TEST IS ENABLED.
									:** READ TO CLEAR DONE
	011456	105777	171154		42\$:	TSTB	@RBUF		
	011462	005300				DEC	R0		:**
	011464	001374				BNE	42\$		:**
	011466	005301				DEC	R1		:**
	011470	001372				BNE	42\$		:**
2442	011472	005077	171144			CLR	@TBUF		:SEND A CHARACTER
2443	011476	105777	171132		1\$:	TSTB	@RCSR		:WAIT FOR DONE (INTERRUPT)
2444	011502	100375				BPL	1\$		
2445	011504	042777	000004	171146		BIC	#BIT2,@TCSR		:** DISABLE MAINTENANCE MODE FOR
									:** CONSOLE TO ALLOW FOR COMMUNICATION
									:** WITH TERMINAL.
2446	011512	005077	171120			CLR	@RBUF		:READ RBUF TO CLEAR PENDING INTERRUPT
2447	011516	004737	014400			JSR	PC,WRPSW		:SET PSW TO PRIORITY 3
2448	011522	000140					.WORD 140		
2449	011524	000240				NOP			:** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
2450	011526	000240				NOP			:** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
2451	011530	000240				NOP			:** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
2452	011532	000240				NOP			:** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
2453	011534	042777	000100	171072		BIC	#BIT6,@RCSR		:NO INTERRUPT-CLEAR INT. ENABLE
2454	011542	000402				BR	3\$		
2455									
2456	011544	022626			2\$:	CMP	(SP)+,(SP)+		:RESTORE SP AFTER INTERRUPT
2457	011546	104077				ERROR	+77		:READING RBUF DID NOT CLEAR INTERRUPT
2458									
2459	011550	010377	171070		3\$:	MOV	R3,@RVECT		:RESTORE RECEIVE VECTOR

```

2460          .SBTTL TEST # 37 - TEST THAT RESET CLEARS RECEIVE INTERRUPT
              :*****
              :*TEST 37 - TEST THAT RESET CLEARS RECEIVE INTERRUPT
              :*****
2461 011554 000004          TST37: SCOPE
2462 011556 000005          RESET          ;CLEAR EVERYTHING
2463 011564 004737 014400    JSR          PC,WRPSW          ;SET PSW TO PRIORITY 7
2464 011566 017703 171052    .WORD 340
2465 011572 012777 011702 171044    MOV @RVECT,R3          ;SAVE RECEIVE VECTOR
2466 011600 052777 000100 171026    MOV #2$,@RVECT        ;POINT RCV VECTOR TO ERROR REPORT
2467 011606 052777 000004 171024    BIS #BIT6,@RCSR       ;SET RCV INTERRUPT ENABLE
2468 011614 005000          CLR R0          ;SET MAINTENANCE WRAP
011616 012701 000002          MOV #2,R1          ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
              ;** THAT MIGHT BE IN THE PROCESS OF BEING
              ;** RECEIVED TO FINISH AFTER MAINTENANCE
              ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
              ;** READ TO CLEAR DONE
011622 105777 171010    42$: TSTB @RBUF
011626 005300          DEC R0
011630 001374          BNE 42$
011632 005301          DEC R1
011634 001372          BNE 42$
2469 011636 012777 000377 170776    MOV #377,@TBUF        ;SEND AN ALL 1'S CHARACTER
2470 011644 105777 170764    1$: TSTB @RCSR          ;WAIT FOR RCV DONE
2471 011650 100375          BPL 1$
2472 011652 000005          RESET          ;CLEAR RCV INTERRUPT & RBUF
2473 011654 004737 014400    JSR          PC,WRPSW          ;SET PSW TO PRIORITY 3
2474 011660 000140          .WORD 140
2475 011662 000240          NOP
2476 011664 000240          NOP
2477 011666 000240          NOP
2478 011670 000240          NOP
2479 011672 042777 000100 170734    BIC #BIT6,@RCSR       ;** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
2480 011700 000402          BR 3$          ;** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
2481
2482
2483 011702 022626    2$: CMP (SP)+,(SP)+        ;** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
2484 011704 104100          ERROR +100          ;** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
2485
2486 011706 010377 170732    3$: MOV R3,@RVECT        ;NO INTERRUPT-CLEAR INT. ENABLE
              ;CONTINUE TEST
              ;RESTORE SP AFTER INTERRUPT
              ;RESET DID NOT CLEAR RCVR INTERRUPT
              ;RESTORE RECEIVE VECTOR
  
```





2557

.SBTTL TEST # 42 - TEST THAT 'FR' ERROR CAN BE SET DURING BREAK

\*\*\*\*\*  
 \*TEST 42 - TEST THAT 'FR' ERROR CAN BE SET DURING BREAK  
 \*\*\*\*\*

2558	012312	000004	002000	166516	TST42: SCOPE		
2559	012314	032777			BIT #BIT110,@SWR		:IS THE 'TEST ERROR FLAGS' BIT SET
2560	012322	001464			BEQ TST43		:BR TO NEXT TEST, IF NOT SET
2561	012324	032777	000400	166506	BIT #BIT8,@SWR		:IS BREAK FUNCTION ENABLED
2562	012332	001460			BEQ TST43		:BR TO NEXT TEST, IF NOT SET
2563	012334	032737	000001	003002	BIT #BIT0,FLAG44		:** IS THIS A 11/44
2564	012342	001407			BEQ 9\$		:** NO
2565	012344	005737	002624		TST CTSTFL		:** YES THIS IS 11/44. IS THIS THE CONSOLE
2566	012350	001404			BEQ 9\$		:** SLU
2567	012352	032777	000010	166460	BIT #BIT03,@SWR		:** NO
2568	012360	001445			BEQ TST43		:** THIS IS THE CONSOLE SLU.SHOULD THE FRAME
2569	012362	000005			9\$: RESET		:** ERROR TEST BE PERFORMED
2570	012364	052777	000004	170246	BIS #BIT2,@TCSR		:** NO
2571	012372	005000			CLR R0		:CLEAR EVERYTHING
2572	012374	012701	000002		MOV #2,R1		:SET MAINTENANCE WRAP
	012400	105777	170232		42\$: TSTB @RBUF		:** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	012404	005300			DEC R0		:** THAT MIGHT BE IN THE PROCESS OF BEING
	012406	001374			BNE 42\$		:** RECEIVED TO FINISH AFTER MAINTENANCE
	012410	005301			DEC R1		:** WRAP FOR THE UART UNDER TEST IS ENABLED.
	012412	001372			BNE 42\$		:** READ TO CLEAR DONE
2573	012414	052777	000001	170216	BIS #BIT0,@TCSR		:SEND BREAK
2574	012422	005077	170214		CLR @TBUF		:TRANSMIT A CHARACTER TO TIME BREAK
2575	012426	105777	170202		1\$: TSTB @RCSR		:WAIT FOR RCVR DONE
2576	012432	100375			BPL 1\$		
2577	012434	042777	000001	170176	BIC #BIT0,@TCSR		:CLEAR BREAK BIT
2578	012442	042777	000004	170210	BIC #BIT2,@TCSR		:** DISABLE MAINTENANCE MODE FOR
	012450	032777	020000	170160	BIT #BIT13,@RBUF		:** CONSOLE TO ALLOW FOR COMMUNICATION
2580	012456	001001			BNE 2\$		:** WITH TERMINAL.
2581	012460	104104			ERROR +104		:CHECK FOR FRAMING ERROR FLAG
2582	012462	032777	100000	170146	2\$: BIT #BIT15,@RBUF		:BR, IF SET
2583	012470	001001			BNE 3\$		:BREAK DID NOT SET FRAMING ERROR
2584	012472	104114			ERROR +114		:TEST 'ERROR' FLAG
2585	012474				3\$:		:BR, IF SET
2586							: 'ERROR' FLAG DID NOT SET WITH 'OR' FLAG
2587							
2588							
2589							

2590

```
.SBTTL TEST # 43 - TEST DATA PATH FROM XMIT TO REC USING MAINT WRAP
*****
*TEST 43 - TEST DATA PATH FROM XMIT TO REC USING MAINT WRAP
*****
TST43: SCOPE
```

2591 012474 000004

2592

2593 012476 000005 000004 170132

2594

2595

2596

2597 012506 005000 000002  
 012510 012701

```
RESET
BIS #BIT2,@TCSR
```

```
;CLEAR EVERYTHING
;SET MAINTENANCE WRAP
;TRANSMIT A BINARY COUNT PATTERN - UP
;TO THE BIT POSITION INDICATED BY THE
;CONTENTS OF LOCATION '$USWR'
; ** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
; ** THAT MIGHT BE IN THE PROCESS OF BEING
; ** RECEIVED TO FINISH AFTER MAINTENANCE
; ** WRAP FOR THE UART UNDER TEST IS ENABLED.
; ** READ TO CLEAR DONE
```

012514 105777 170116

012520 005300

012522 001374

012524 005301

012526 001372

2598 012530 005001

2599 012532 105201

2600 012534 032737 000001 003002

2601 012542 001406

2602 012544 122701 000020

2603 012550 001770

2604 012552 122701 000220

2605 012556 001765

2606 012560 010177 170056

2607 012564 105777 170044

2608 012570 100375

2609 012572 017702 170040

2610 012576 043701 001112

2611 012602 020102

2612 012604 001003

2613 012606 105701

2614 012610 001411

2615 012612 000747

2616 012614 010137 001024

2617 012620 010237 001026

2618 012624 042777 000004 170026

```
42$: TSTB @RBUF
DEC R0
BNE 42$
DEC P1
BNE 42$
CLR R1
1$: INCB R1
BIT #BIT0,FLAG44
BEQ 5$
CMPB #20,R1
BEQ 1$
CMPB #220,R1
BEQ 1$
5$: MOV R1,@TBUF
2$: TSTB @RCSR
BPL 2$
MOV @RBUF,R2
BIC @#$USWR,R1
CMP R1,R2
BNE 3$
TSTB R1
BEQ 4$
BR 1$
3$: MOV R1,$GDDAT
MOV R2,$BDDAT
BIC #BIT2,@CTCSR
```

```
;CLEAR REGISTER FOR TEST DATA
;INCREMENT THE TEST DATA
; ** 11/44 CPU
; ** TEST ALL DATA
; ** CHECK FOR CONT-P IN TEST DATA
; ** DO NOT XMIT ^P
; ** DO NOT XMIT 220
;XMIT A CHARACTER
;WAIT FOR RECEIVER DONE
;GET RECEIVED CHARACTER
;CLEAR LOWEST UNUSED DATA BIT POSITITON IN TEST DATA
;COMPARE DATA
;BR, IF NON-COMPARE
;TEST XMIT DATA FOR ZERO
;BR, IF FINISHED
;CONTINUE IF NOT
;STORE THE EXPECTED DATA
;STORE RECEIVED DATA
; ** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.
;DATA COMPARE DATA
```

2619 012632 104105

2620

2621

2622 012634 042777 000004 170016

```
ERROR +105
4$: BIC #BIT2,@CTCSR
```

```
; ** DISABLE MAINTENANCE MODE FOR
; ** CONSOLE TO ALLOW FOR COMMUNICATION
; ** WITH TERMINAL.
```

2623

2624

2625

2626

.SBTTL TEST # 44 - TEST DATA PATHS USING LOOP-BACK CONNECTOR

\*\*\*\*\*  
 :\*TEST 44 - TEST DATA PATHS USING LOOP-BACK CONNECTOR  
 \*\*\*\*\*

2627	012642	000004			TST44:	SCOPE			
2628	012644	032777	000200	166166		BIT	#BIT7,@SWR		:IS THIS TEST ENABLED
2629	012652	001442				BEQ	TST45		:BR, IF NOT
2630	012654	000005				RESET			:CLEAR EVERYTHING
2631	012656	005001				CLR	R1		:CLEAR REGISTER FOR TEST DATA
2632									:TRANSMIT A BINARY COUNT PATTERN - UP
2633									:TO THE BIT POSITION INDICATED BY THE
2634	012660	105201			1\$:	INCB	R1		:CONTENTS OF LOCATION '\$USWR'
2635	012662	032737	000001	003002		BIT	#BIT0,FLAG44		:INCREMENT THE TEST DATA
2636	012670	001404				BEQ	5\$		:11/44 CPU
2637	012672	023727	002634	177560		CMP	RCSR,#177560		:TEST ALL DATA
2638	012700	001427				BEQ	TST45		:IS THIS 11/44 CONSOLE TERMINAL SLU?
2639	012702	010177	167734		5\$:	MOV	R1,@TBUF		:YES, SKIP THIS TEST
2640	012706	005000				CLR	R0		:XMIT A CHARACTER
2641	012710	105777	167720		2\$:	TSTB	@RCSR		:CLEAR A TIMER
2642	012714	100403				BMI	3\$		:WAIT FOR RECEIVER DONE
2643	012716	005200				INC	R0		:BR IF DONE
2644	012720	001373				BNE	2\$		:INCREMENT TIMER IF NOT
2645									:BR IF TIME REMAINS
2646	012722	104064				ERROR	+64		:RECEIVER DONE NOT SET
2647									
2648	012724	017702	167706		3\$:	MOV	@RBUF,R2		:GET RECEIVED CHARACTER
2649	012730	043701	001112			BIC	@\$USWR,R1		:CLEAR LOWEST UNUSED DATA BIT POSITITON IN TEST DATA
2650	012734	020102				CMP	R1,R2		:COMPARE DATA
2651	012736	001003				BNE	4\$		:BR, IF NON-COMPARE
2652	012740	105701				TSTB	R1		:TEST XMIT DATA FOR ZERO
2653	012742	001406				BEQ	TST45		:BR, IF FINISHED
2654	012744	000745				BR	1\$		:CONTINUE IF NOT
2655	012746	010137	001024		4\$:	MOV	R1,\$GDDAT		:STORE EXPECTED DATA
2656	012752	010237	001026			MOV	R2,\$BDDAT		:STORE RECEIVED DATA
2657									
2658	012756	104106				ERROR	+106		:DATA COMPARE ERROR USING LOOP-BACK CONNECTOR

TEST # 45 - TEST DL11-W LOGIC BY EXERCISING THE XMIT, REC, & CL  
2659

SBTTL TEST # 45 - TEST DL11-W LOGIC BY EXERCISING THE XMIT, REC, & CLOCK  
\*\*\*\*\*  
\*TEST 45 - TEST DL11-W LOGIC BY EXERCISING THE XMIT, REC, & CLOCK  
\*\*\*\*\*

2660	012760	000004			TST45:	SCOPE			
	012762	000005				RESET			:CLEAR EVERYTHING
2661	012764	005037	002444			CLR	CLKCNT		:INITIALIZE CLOCK COUNT TO ZERO
2662	012770	004737	014400			JSR	PC,WRPSW		:SET PRIORITY TO 7
2663	012774	000340				.WORD	340		
2664	012776	017703	167646			MOV	@TVECT,R3		:SAVE XMIT VECTOR
2665	013002	017704	167636			MOV	@RVECT,R4		:SAVE RECEIVE VECTOR
2666	013006	017705	167664			MOV	@RTCVT,R5		:SAVE CLOCK VECTOR
2667	013012	017737	167634	002446		MOV	@TPSW,STPSW		:SAVE XMIT PSW VECTOR
2668	013020	017737	167622	002450		MOV	@RPSW,SRPSW		:SAVE RECEIVE PSW VECTOR
2669	013026	017737	167646	002452		MOV	@RTCPSW,SCPSW		:SAVE CLOCK PSW VECTOR
2670	013034	012777	013440	167606		MOV	#XMIT,@TVECT		:POINT TRANSMIT VECTOR TO TRANSMIT ROUTINE
2671	013042	012777	000200	167602		MOV	#200,@TPSW		:NO MULTIPLE INTERRUPTS ALLOWED
2672	013050	012777	013512	167566		MOV	#RCV,@RVECT		:POINT RECEIVE VECTOR TO RECEIVE ROUTINE
2673	013056	012777	000200	167562		MOV	#200,@RPSW		:NO MULT INTERRUPTS
2674	013064	005737	002624			TST	CTSTFL		:IS CONSOLE UNDER TEST?
2675	013070	001415				BEQ	1\$		:IF NOT SKIP CLOCK SET UP
2676	013072	032777	000100	165740		BIT	#BIT6,@SWR		:IF YES, ARE CLOCK TEST DISABLED?
2677	013100	001011				BNE	1\$		:IF YES, SKIP CLOCK SET UP
2678	013102	012777	013526	167566		MOV	#CLK,@RTCVT		:POINT VECTOR TO CLOCK INTERRUPT ROUTINE
2679	013110	012777	000300	167562		MOV	#300,@RTCPSW		:NO MULT INTERRUPTS
2680	013116	052777	000100	167550		BIS	#BIT6,@LKS		:ENABLE CLOCK INTERRUPTS
2681	013124	052777	000004	167506	1\$:	BIS	#BIT2,@TCSR		:SET MAINTENANCE WRAP
2682	013132	005000				CLR	R0		** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
	013134	012701	000002			MOV	#2,R1		** THAT MIGHT BE IN THE PROCESS OF BEING
									** RECEIVED TO FINISH AFTER MAINTENANCE
									** WRAP FOR THE UART UNDER TEST IS ENABLED.
									** READ TO CLEAR DONE
	013140	105777	167472		42\$:	TSTB	@RBUF		
	013144	005300				DEC	R0		
	013146	001374				BNE	42\$		
	013150	005301				DEC	R1		
	013152	001372				BNE	42\$		
2683	013154	052777	000100	167456		BIS	#BIT6,@TCSR		:ENABLE TRANSMIT INTERRUPTS
2684	013162	052777	000100	167444		BIS	#BIT6,@RCSR		:ENABLE RECEIVE INTERRUPTS
2685	013170	005037	002442			CLR	XMTCNT		:CLEAR XMIT INTERRUPT COUNTER
2686	013174	005037	002440			CLR	RCVCNT		:CLEAR RCV INTERRUPT COUNTER
2687	013200	005001				CLR	R1		:CLEAR A REGISTER FOR TEST DATA USE
2688	013202	005000				CLR	R0		:CLEAR TIMER
2689	013204	012702	002454			MOV	#BUF,R2		:POINT R2 TO RECEIVE DATA STORAGE
2690	013210	005077	167426			CLR	@TBUF		:SEND FIRST CHARACTER
2691	013214	004737	014400			JSR	PC,WRPSW		:SET PSW TO PRIORITY 3
2692	013220	000140				.WORD	140		
2693									:WAIT FOR INTERRUPTS
2694	013222	000240			2\$:	NOP			:STALL
2695	013224	000240				NOP			:STALL
2696	013226	062700	000000			ADD	#0,R0		:ADD INSTRUCTIONS ARE USED TO LENGTHEN LOOP TIME
2697	013232	062700	000001			ADD	#1,R0		: TO COVER THE SLOWEST BAUD RATE ON THE FASTEST CPU
2698	013236	001371				BNE	2\$		
2699	013240	032777	000100	167372		BIT	#BIT6,@TCSR		:FINISHED ENTIRE TRANSMISSION
2700	013246	001402				BEQ	3\$		:BR, IF INTERRUPTS ARE DISABLED (FINISHED)
2701	013250	000005				RESET			:CLEAR EVERYTHING
2702									
2703	013252	104107				ERROR	+107		:TRANSMIT INTERRUPT TIMEOUT IN MAIN. DATA TEST



```

2704
2705 013254 023737 002442 002440 3$:   CMP      XMTCNT,RCVCNT   ;COMPARE THE NUMBER OF INTERRUPTS
2706 013262 001402                BEQ      4$             ;BR, IF EQUAL
2707 013264 000005                RESET                      ;CLEAR EVERYTHING
2708
2709 013266 104110                ERROR   +110           ;RECEIVER DID NOT GET FULL TRANSMISSION
2710                                ;   IF RCVCNT=0, NO DATA RECEIVED
2711                                ;   IF RCVCNT=?, THEN (XMTCNT-RCVCNT)
2712                                ;   EQUALS THE NO, OF INTERRUPTS LOST.
2713 013270 005737 002624          4$:   TST      CTSTFL       ;IS CONSOLE UNDER TEST?
2714 013274 001411                BEQ      5$             ;IF NOT, SKIP CLOCK COUNT CHECK
2715 013276 032777 000100 165534    BIT      #BIT6,@SWR     ;IF YES, ARE CLOCK TESTS DISABLED?
2716 013304 001005                BNE     5$             ;IF YES, SKIP CLOCK COUNT CHECK
2717 013306 005737 002444          TST      CLKCNT        ;CHECK FOR AT LESST ONE CLOCK INTERRUPT
2718 013312 001002                BNE     5$             ;BR IF INTERRUPTS OCCURRED
2719 013314 000005                RESET                      ;CLEAR MAINTENANCE WRAPAROUND
2720 013316 104113                ERROR   +113           ;NO CLOCK INTERRUPTS IN EXERCISER
2721
2722 013320 000005          5$:   RESET                      ;CLEAR EVERYTHING
2723 013322 012700 002454          MOV      #BUF,R0       ;LOAD RECEIVED DATA POINTER TO R0
2724 013326 005001                CLR      R1             ;SET UP REGISTER FOR COMPARISON
2725 013330 022001          COMP:  CMP      (R0)+,R1   ;COMPARE XMIT & RCV DATA
2726 013332 001014                BNE     6$             ;BR, IF NOT EQUAL
2727 013334 105201          9$:   INCB     R1           ;INCREMENT COMPARE DATA
2728 013336 032737 000001 003002    BIT      #BIT0,FLAG44  ;11/44 CPU?
2729 013344 001403                BEQ      8$             ;YES, SKIP
2730 013346 122701 000020          CMPB    #20,R1        ;SKIP 20 DATA IF 11/44
2731 013352 001770                BEQ      9$             ;SKIP 20
2732 013354 032701 000040          8$:   BIT      #BIT5,R1   ;FINISHED CHECKING RECEIVED DATA?
2733 013360 001763                BEQ     COMP           ;BR, IF NOT FINISHED
2734 013362 000405                BR      7$             ;BR TO END OF TEST
2735
2736 013364 014037 001026          6$:   MOV      -(R0),%BDAT  ;STORE BAD DATA FOR ERROR REPORT
2737 013370 010137 001024          MOV      R1,%GDAT     ;STORE GOOD DATA FOR ERROR REPORT
2738 013374 104111                ERROR   +111           ;DATA COMPARE ERROR IN EXERCISER
2739
2740 013376 010377 167246          7$:   MOV      R3,@TVECT   ;RESTORE XMIT VECTOR
2741 013402 010477 167236          MOV      R4,@RVECT   ;RESTORE RECEIVE VECTOR
2742 013406 010577 167264          MOV      R5,@RTCVT   ;RESTORE CLOCK VECTOR
2743 013412 013777 002446 167232    MOV      STPSW,@TPSW  ;RESTORE XMIT PSW VECTOR
2744 013420 013777 002450 167220    MOV      SRPSW,@RPSW ;RESTORE RECEIVE PSW VECTOR
2745 013426 013777 002452 167244    MOV      SCPSW,@RTCPSW ;RESTORE CLOCK PSW VECTOR
2746 013434 000137 014170          JMP      ENDEV        ;GOTO NEXT TEST
2747
2748 013440 005237 002442          XMIT:  INC      XMTCNT   ;INCREMENT XMIT INTERRUPT COUNTER
2749 013444 105201          1$:   INCB     R1           ;INCREMENT TEST DATA
2750 013446 032737 000001 003002    BIT      #BIT0,FLAG44  ;11/44 CPU
2751 013454 001403                BEQ      2$             ;TEST ALL DATA
2752 013456 122701 000020          CMPB    #20,R1        ;CHECK DATA FOR ^P
2753 013462 001770                BEQ      1$             ;DO NOT XMIT ^P
2754 013464 032701 000040          2$:   BIT      #BIT5,R1   ;SEND DATA PATTERN FROM 00 --> 37
2755 013470 001404                BEQ     XCONT          ;BR, IF MORE DATA TO BE SENT
2756 013472 042777 000100 167140    BIC     #BIT6,@TCSR   ;CLEAR XMIT INTERRUPT ENABLE
2757 013500 000402                BR      XRET          ;RETURN, WITHOUT SENDING ANY MORE DATA
2758 013502 110177 167134          XCONT: MOVB    R1,@TBUF  ;SEND NEW CHARACTER
2759 013506 005000          XRET:  CLR      R0           ;CLEAR TIMER

```

```
2760 013510 000002          RTI          ;RETURN
2761
2762 013512 017722 167120    RCV:    MOV    @RBUF,(R2)+  ;STORE RECEIVED DATA
2763 013516 005237 002440    INC    RVCNT  ;INCREMENT RCV INTERRUPT COUNTER
2764 013522 005000          CLR    R0     ;CLEAR TIMER
2765 013524 000002          RTI          ;RETURN
2766
2767 013526 005237 002444    CLK:    INC    CLKCNT     ;INCREMENT CLOCK INTERRUPT COUNT
2768 013532 000002          RTI          ;RETURN
```

2769  
2770  
2771  
2772  
2773  
2774  
2775  
2776  
2777  
2778  
2779  
2780  
2781

.SBTTL WRAPAROUND TESTING- AUTO INITIATION OF T/A CONSOLE TEST  
:\*\*\*\*\*  
: WRAPAROUND TESTING- AUTO INITIATION OF T/A CONSOLE TEST

176500  
176502  
176504  
176506  
003014  
024610

RCSR58 = 176500  
RBUF58 = 176502  
TCSR58 = 176504  
TBUF58 = 176506  
BGNADD = DEVADR  
ENDADD = ENDADR

2782

013534 000004

2783  
2784  
2785  
2786  
2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799  
2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828  
2829  
2830  
2831  
2832  
2833  
2834

```

.SBTTL TEST # 46 - TEST CONSOLE WITH WRAP AROUND
:*****
:TEST 46 - TEST CONSOLE WITH WRAP AROUND
:*****
TST46: SCOPE
:*****
:
: MAIN LINE TEST
:
: THIS TEST PUTS COMMANDS OUT OVER THE SERIAL LINE WHICH IS WRAPPED
: AROUND TO THE CONSOLE AND RECEIVES THE RESPONSES
:
: CONTROL P IS SEND TO GET THE CONSOLE'S ATTENTION AND THEN
: T/A(A FOR APT) IS SENT. SINCE THE CPU IS HALTED DURING THE TESTING
: THE PROGRAM CAN NOT SEE THE CHARS RETURNING, THUS THE PROGRAM WILL
: SIT IN A LOOP WAITING FOR A 'B' TO BE PRINTED BY THE CONSOLE AS A
: SIGNAL TO APT THAT THE TESTING IS DONE. ALSO SINCE THE TESTING
: INVOLVES WRITTING TO THE CPU'S MEMORY A CHECKSUM IS CALCULATED AND THE
: MEMORY TO BE USED INSIDE THIS PROGRAMS SPACE IS SAVE BEFORE TESTING
: AND RESTORED AFTER TEST WITH ANOTHER CHECKSUM CALCULATION
:*****
:
2802 013536 032777 000004 165274 WRAP: BIT #BIT2,@SWR ;RUN THIS TEST ONLY IF
2803 013544 001451 BEQ 10$ ;SW BIT 2 IS ON
2805 013546 013737 177776 002434 MOV PSW,SAVEPS ;SAVE OLD PSW
2806 013554 012737 000340 177776 MOV #340,PSW ;PUT IN NOW PSW
2807 013562 012706 002432 MOV #JIM$TK,SP ;USE MY STACK (SO NO CLOBFR)
2809 013566 005037 002340 CLR LOC1
2810 013572 012737 000100 002342 MOV #100,LOC2 ;TIMING LOOP COUNTERS
2812 013600 004537 014100 JSR R5,SAVE TE ;SAVE LOCATIONS T/A WRITES
2814 013604 004537 013706 JSR R5,CHKSUM ;CALCULATE CHECKSUM
2815 013610 010437 002436 MOV R4,OLDSUM ;SAVE OLD CHECKSUM
2817 013614 012700 002564 MOV #CNTLP,R0
2818 013620 004537 013726 JSR R5,PUTLIN ;SEND OUT CONTROL P
2820 013624 012700 002566 MOV #PROMPT,R0
2821 013630 004537 013760 JSR R5,GETLIN ;GET CRLF CONSOLE>>>
2823 013634 012700 002610 MOV #TA,R0
2824 013640 004537 013726 JSR R5,PUTLIN ;SEND OUT T/A<CRLF>
2826 013644 004537 014024 JSR R5,GETB ;GET THE A FROM '-TESTB'
2828 013650 004537 014134 JSR R5,RESTTE ;RESTORE LOCATIONS T/A WRITES
2830 013654 004537 013706 JSR R5,CHKSUM ;RECALCULATE CHECKSUM
2831 013660 020437 002436 CMP R4,OLDSUM
2832 013664 001401 BEQ 10$
2833 013666 000000 HALT
2834 013670 012706 001000 MOV #1000,SP ;RETURN THEIR SP
    
```

```

2835 013674 013737 002434 177776      MOV      SAVEPS,PSW      ;RETURN PSW
2836 013702 000137 014224      JMP      $EOP           ;BR TO END OF PASS ROUTINE
    
```

```

2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
    
```

\*\*\*\*\*

```

: ROUTINE TO CALCULATE CHECKSUM ON PROGRAM
: INPUT CONDITIONS
:     BGNADD = ADDRESS TO START CHECK SUM (INCLUSIVE)
:     ENDADD = ADDRESS TO END CHECK SUM (EXCLUSIVE)
: OUTPUT CONDITIONS
:     REG4 = CHECK SUM
    
```

\*\*\*\*\*

```

2855 013706 012700 003014      CHKSUM:  MOV      #BGNADD,R0      ;GET STARTING ADDRESS
2856 013712 005004              CLR      R4                    ;RESET SUM
2857 013714 062004      1$:     ADD      (R0)+,R4          ;ADD WORD TO SUM
2858 013716 022700 024610      CMP      #ENDADD,R0          ;CHECK FOR END
2859 013722 001374              BNE     1$                    ;IF NOT DONE LOOP
2860 013724 000205              RTS      R5                    ;DONE RETURN
    
```

```

2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
    
```

\*\*\*\*\*

```

: THIS ROUTINE OUTPUTS TO THE SERIAL LINE CHARS STARTING AT
: THE ADDRESS IN REG0 UNTIL IT HITS A <377>
    
```

\*\*\*\*\*

```

2871 013726 112001              PUTLIN:  MOVB     (R0)+,R1          ;GET DATA
2872 013730 001412              BEQ     10$                   ;END OF DATA
2873 013732 004537 014060      1$:     JSR      R5,TIMER          ;PROVIDE FOR TIMEOUT
2874 013736 032737 000200 176504  BIT      #BIT7,TCSR58         ;TEST FOR XMIT READY
2875 013744 001772              BEQ     1$                    ;WAIT FOR XMIT READY
2876 013746 110137 176506      MOVB     R1,TBUF58           ;OUTPUT CHAR
2877 013752 000137 013726      JMP      PUTLIN              ;REPETE
2878 013756 000205              RTS      R5                    ;RETURN
    
```

```

2879
2880
2881
2882
2883
2884
2885
2886
    
```

\*\*\*\*\*

```

: THIS ROUTINE INPUTS A CHARS FROM THE SERIAL LINE AND COMPARES
: IT WITH THE EXPECTED VALUES POINTED TO BY REG0 UNTIL NULL<00>
    
```

\*\*\*\*\*

```

2887
2888 013760 112001              GETLIN:  MOVB     (R0)+,R1          ;GET EXPECTED CHAR
2889 013762 105201              INCB    R1                    ;
2890 013764 001416              BEQ     10$                   ;IF NULL EXIT
2891 013766 105301              DECB    R1
    
```

```

2892 013770 004537 014060 1$: JSR R5,TIMER ;PROVIDE FOR TIMOUT
2893 013774 032737 000200 176500 BIT #BIT7,RCSR58 ;TEST FOR REC READY
2894 014002 001772 BEQ 1$ ;WAIT FOR REC READY
2895 014004 113702 176502 MOVB RBUF58,R2
2896 014010 042702 000200 BIC #BIT7,R2 ;STRIP PARITY
2897 014014 120102 CMPB R1,R2 ;ARE THEY THE SAME
2898 014016 001760 BEQ GETLIN ;SAME GET MORE
2899 014020 000000 HALT ;NOT SAME HALT
2900 014022 000205 10$: RTS R5 ;RETURN
2901
2902
2903 :*****
2904 :
2905 : THIS ROUTINE INPUTS CHARS UNTIL IT GETS THE CHAR 'B'
2906 : AND THEN RETURNS
2907 :
2908 :*****
2909
2910
2911 014024 004537 014060 GETB: JSR R5,TIMER ;OUT TIMING LOOP OUT
2912 014030 032737 000200 176500 BIT #BIT7,RCSR58 ;TEST OFR REC READY
2913 014036 001772 BEQ GETB ;NOT DONE YET
2914 014040 113702 176502 MOVB RBUF58,R2
2915 014044 042702 000200 BIC #BIT7,R2 ;STRIP PARITY
2916 014050 122702 000102 CMPB #102,R2 ;CHECK FOR 'B'
2917 014054 001363 BNE GETB ;NOT 'B' REPETE
2918 014056 000205 RTS R5 ;WAS 'B' RETURN
2919
2920 :*****
2921 :
2922 : THIS ROUTINE IS USED AS A TIME OUT FEATURE
2923 : WHEN THE TIMING LOOPS BOIH REACH 0 THIS ROUTINE WILL
2924 : CAUSE A HALT
2925 :
2926 :*****
2927
2928
2929 014060 005337 002340 TIMER: DEC LOC1
2930 014064 001004 BNE 10$ ;DECREPNT TIMING LOOPS
2931 014066 005337 002342 DEC LOC2
2932 014072 001001 BNE 10$
2933 014074 000000 HALT ;'F ZERO THIS ROUTINE
2934 014076 000205 10$: RTS R5 ;EXECUTED R3 TIMES
2935
2936
2937 :*****
2938 :
2939 : THIS ROUTINE SAVES THE LOCATIONS WRITTEN BY THE T/A CONSOLE TFST
2940 : SO THEY MAY BE RESTORED LATER
2941 :
2942 :*****
2943
2944 014100 013737 000000 002344 SAVETE: MOV 0,SAVE0 ;SAVE LOCATION 0
2945 014106 012702 000002 MOV #2,R2 ;SET UP INDIRECT PNTER
2946 014112 012701 002346 MOV #SAVLOC,R1 ;SET UP STORAGE LOC. PNTER
2947 014116 011221 1$: MOV (R2),(R1)+ ;GET WORD AND SAVE
    
```

```

2948 014120 000241          CLC                ;DONT ROT IN ANY BITS
2949 014122 006102          ROL                R2                ;SET NEXT ADDRESS
2950 014124 020227 024610  CMP                R2,#ENDADD        ;SEE IF AT END
2951 014130 100772          BMI                1$                ;NO REPETE
2952 014132 000205          RTS                R5
2953
2954
2955          :*****
2956          :
2957          : THIS ROUTINE RESTORES THE SAVED LOCATIONS THAT T/A WRITES INTO
2958          :
2959          :*****
2960
2961 014134 013737 002344 000000 RESTTE:  MOV                SAVE0,0
2962 014142 012702 000002          MOV                #2,R2                ;SET UP INDIRECT PNTER
2963 014146 012701 002346          MOV                #SAVLOC,R1          ;SET UP STORAGE LOC. PNTER
2964 014152 012112          MOV                (R1)+,(R2)        ;GET WORD AND RESTORE
2965 014154 000241          CLC                ;DONT ROT IN ANY BITS
2966 014156 006102          ROL                R2                ;SET NEXT ADDRESS
2967 014160 020227 024610  CMP                R2,#ENDADD        ;SEE IF AT END
2968 014164 100772          BMI                1$                ;NO REPETE
2969 014166 000205          RTS                R5
  
```

```
2970                                     ;END OF DEVICE PASS ROUTINE
2971 014170 005037 001002            ENDEV: CLR    $TSTNM            ;CLEAR TEST NO. COUNT FOR SCOPE ROUTINE
2972 014174 005237 001076                   INC    $DEVCT            ;INCREMENT DEVICE COUNTER
2973 014200 023737 002630 001076        CMP    TMP2,$DEVCT        ;ALL DEVICES TESTED
2974 014206 001002                   BNE    NOEOP            ;BR, IF NO
2975 014210 000137 013536                   JMP    WRAP            ;EXECUTE WRAP AROUND AFTER ALL DEVICES
2976 014214 005037 002624            NOEOP: CLR    CTSTFL        ;CLEAR CONSOLE UNDER TEST FLAG
2977 014220 000137 004050                   JMP    TSTDEV        ;GO TEST NEXT DEVICE
```



2979

```
.SBTTL END OF PASS ROUTINE  
:*****  
:* INCREMENT THE PASS NUMBER ($PASS)  
:* IF THERES A MONITOR GO TO IT  
:* IF THERE ISN'T JUMP TO GOAGIN  
$EOP:  
SCOPE  
CLR $STNM ;;ZERO THE TEST NUMBER  
INC $PASS ;;INCREMENT THE PASS NUMBER  
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER  
DEC (PC)+ ;;LOOP?  
$EOPCT: .WORD 1  
BGT $DOAGN ;;YES  
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER  
$ENDCT: .WORD 1  
$EOPCT  
TYPE ,ENDMG ;;TYPE 'END PASS'  
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS  
BEQ $DOAGN ;;BRANCH IF NO MONITOR  
RESET ;;CLEAR THE WORLD  
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR  
NOP ;;SAVE ROOM  
NOP ;;FOR  
NOP ;;ACT11  
$DOAGN:  
JMP @(PC)+ ;;RETURN  
$RTNAD: .WORD GOAGIN  
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING  
          .EVEN  
2980 014314 015 012 105 ENDMG: .ASCIZ <CR><LF>/END PASS /
```

```

2982 014330 005046          GOAGIN: CLR      -(SP)          ;CLEAR ANOTHER LOCATION ON STACK
2983 014332 005216          1$:  INC      (SP)          ;INCREMENT STACK LOCATION
2984 014334 000240          NOP          ;TAKE UP SOME MORE TIME
2985 014336 001375          BNE      1$          ;BRANCH BACK UNTIL ZERO AGAIN
2986 014340 062706 000002          ADD      #2,SP        ;CLEAN THE LOCATION OFF THE STACK
2987 014344 005037 001076          CLR      $DEVCT       ;CLEAR DEVICE COUNT
2988 014350 022737 000001 002630          CMP      #1,TMP2      ;IS THERE ONLY ONE DEVICE UNDER TEST?
2989 014356 001004          BNE      RSTRT       ;BR, IF NOT
2990 014360 012706 001000          MOV      #1000,SP     ;RESET STACK POINTER
2991 014364 000137 004172          JMP      TST1        ;GO DO ANOTHER PASS
2992
2993 014370 005037 001100          RSTRT: CLR      $UNIT   ;CLEAR UNIT NUMBER
2994 014374 000137 003774          JMP      BEGIN
2995
2996 014400 011646          WRPSW: MOV(SP),-(SP)    ;COPY RETURN PC
2997 014402 013616          MCV      @ (SP)+,(SP) ;MOVE NEW PSW TO STACK
2998 014404 062746 000002          ADD      #2,-(SP)    ;ADJUST JSR RETURN
2999 014410 000002          RTI          ;POP RETURN PC & NEW PSW
3000
3001          ;SUBROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
3002
3003 014412 012600          CATCH: MOV      (SP)+,R0 ;GET ADDRESS OF TRAP VECTOR + 4
3004 014414 162700 000004          SUB      #4,R0        ;ADJUST TO POINT TO TRAP ADDRESS
3005 014420 010037 002622          MOV      R0,BDVCT     ;STORE TRAP OR INTERRUPT ADDRESS
3006 014424 016637 000002 002620          MOV      2(SP),OLDPC  ;GET PC WHERE TRAP OR INTERRUPT OCCURRED
3007 014432 104112          ERRGR   +112        ;REPORT ERROR
3008
3009 014434 000000          HALT          ;PROGRAM MUST BE RESTARTED AT THIS POINT
  
```

3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023  
3024 014436  
3025 014436 105237 001003  
3026 014442 001775  
3027 014444 013777 001002 164370  
3028 014452 005237 001012  
3029 014456 011637 001016  
3030 014462 162737 000002 001016  
3031 014470 117737 164322 001014  
3032 014476 032777 020000 164334  
3033 014504 001004  
3034 014506 004737 014620  
3035 014512 104401 001063  
3036 014516  
3037 014516 122737 000001 001106  
3038 014524 001007  
3039 014526 113737 001014 014540  
3040 014534 004737 015326  
3041 014540 000  
3042 014541 000  
3043 014542 000777  
3044 014544 005777 164270  
3045 014550 100001  
3046 014552 000000  
3047 014554 104406  
3048 014556 032777 001000 164254  
3049 014564 001402  
3050 014566 013716 001010  
3051 014572 005737 001060  
3052 014576 001402  
3053 014600 013716 001060  
3054 014604  
3055 014604 022737 014274 000042  
3056 014612 001001  
3057 014614 000000  
3058 014616  
3059 014616 000002  
3060

```
*****  
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,  
*SAVE THE ERROR ITEM NUMBER AND ADDRESS OF THE ERROR CALL  
*AND GO TO $ERRTYP ON ERROR  
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  
*SW15=1 HALT ON ERROR  
*SW13=1 INHIBIT ERROR TYPEOUTS  
*SW09=1 LOOP IN ERROR  
*CALL  
* ERROR +N ;;ERROR=EMT AND N=ERROR ITEM NUMBER  
*****
```

```
$ERROR:  
7$: INCB $ERFLG ;SET THE ERROR FLAG  
BEQ 7$ ;DON'T LET FLAG GO TO ZERO  
MOV $STNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG  
INC $ERTTL ;INCREMENT ERROR COUNT  
MOV (SP),$ERRPC ;GET ADDRESS OF ERROR INSTRUCTION  
SUB #2,$ERRPC  
MOVB @ $ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE  
BIT #BIT13,@SWR ;SKIP TYPEOUT IF SET  
BNE 20$ ;SKIP TYPEOUTS  
JSR PC,$ERRTYP ;GO TO USER ERROR ROUTINE  
TYPE , $CRLF  
  
20$: CMPB #APTENV,$ENV ;RUNNING IN APT MODE  
BNE 2$ ;NO, SKIP APT ERROR REPORT  
MOVB $ITEMB,21$ ;SET ITEM NUMBER AS ERROR NUMBER  
JSR PC,$ATY4 ;REPORT FATAL ERROR TO APT  
  
21$: .BYTE 0  
 .BYTE 0  
  
22$: BR 22$ ;APT ERROR LOOP  
2$: TST @SWR ;HALT ON ERROR  
BPL 3$ ;SKIP IF CONTINUE  
HALT ;HALT ON ERROR!  
3$: CKSWR ;TEST FOR CHANGE IN SOFT-SWR  
BIT #BIT09,@SWR ;LOOP ON ERROR SWITCH SET?  
BEQ 4$ ;BR IF NO  
MOV $LPERR,(SP) ;FUDGE RETURN FOR LOOPING  
4$: TST $ESCAPE ;CHECK FOR AN ESCAPE ADDRESS  
BEQ 5$ ;BR IF NONE  
MOV $ESCAPE,(SP) ;FUDGE RETURN ADDRESS FOR ESCAPE  
  
5$: CMP #SENDAD,@#42 ;ACT-11 AUTO-ACCEPT?  
BNE 6$ ;BR IF NO  
HALT ;YES  
  
6$: RTI ;RETURN
```

3062  
 3063

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE
*****
*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
$ERRTYP:
014620 014620 104401 001063      TYPE      , $CRLF      ;; 'CARRIAGE RETURN' & 'LINE FEED'
014624 010046                    MOV      RO, -(SP)    ;; SAVE RO
014626 005000                    CLR      RO          ;; PICKUP THE ITEM INDEX
014630 153700 001014            BISB     @#$ITEMB,RO
014634 001004                    BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                                ;; TYPE THE PC OF THE ERROR
014636 013746 001016            MOV      $ERRPC, -(SP) ;; SAVE $ERRPC FOR TYPEOUT
                                ;; ERROR ADDRESS
                                ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
014642 104402                    TYPOC
014644 000426                    BR      6$          ;; GET OUT
014646 005300 1$:              DEC      RO          ;; ADJUST THE INDEX SO THAT IT WILL
014650 006300                    ASL     RO          ;; WORK FOR THE ERROR TABLE
014652 006300                    ASL     RO
014654 006300                    ASL     RO
014656 062700 001146            ADD      #$ERRTB,RO  ;; FORM TABLE POINTER
014662 012037 014672            MOV      (RO)+, 2$  ;; PICKUP 'ERROR MESSAGE' POINTER
014666 001404                    BEQ     3$          ;; SKIP TYPEOUT IF NO POINTER
014670 104401                    TYPE
014672 000000 2$:              .WORD   0          ;; 'ERROR MESSAGE' POINTER GOES HERE
014674 104401 001063            TYPE      , $CRLF  ;; 'CARRIAGE RETURN' & 'LINE FEED'
014700 012037 014710 3$:      MOV      (RO)+, 4$  ;; PICKUP 'DATA HEADFR' POINTER
014704 001404                    BEQ     5$          ;; SKIP TYPEOUT IF 0
014706 104401                    TYPE
014710 000000 4$:              .WORD   0          ;; 'DATA HEADER' POINTER GOES HERE
014712 104401 001063            TYPE      , $CRLF  ;; 'CARRIAGE RETURN' & 'LINE FEED'
014716 011000 5$:              MOV      (RO), RO    ;; PICKUP 'DATA TABLE' POINTER
014720 001004                    BNE     7$          ;; GO TYPE THE DATA
014722 012600 6$:              MOV      (SP)+, RO   ;; RESTORE RO
014724 104401 001063            TYPE      , $CRLF  ;; 'CARRIAGE RETURN' & 'LINE FEED'
014730 000207                    RTS     PC          ;; RETURN
014732 7$:
014732 013046                    MOV      @ (RO)+, -(SP) ;; SAVE @ (RO)+ FOR TYPEOUT
014734 104402                    TYPOC
014736 005710                    TST     (RO)        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
014740 001770                    BEQ     6$          ;; IS THERE ANOTHER NUMBER?
014742 104401 014750            TYPE      , 8$    ;; BR IF NO
014746 000771                    BR      7$          ;; TYPE TWO(2) SPACES
014750 040 040 000 8$:        .ASCIZ  / /          ;; LOOP
                                .EVEN          ;; TWO(2) SPACES

```

3064

```

3066
3067      .SBTTL  POWER DOWN AND UP ROUTINES
3068      ;:*****
3069      ;*POWER DOWN ROUTINE
3070      ;:*****
3071 014754 012737 015120 000024 $PWRDN: MOV    #SILLUP,@#PWRVEC ;SET FOR FAST UP
3072 014762 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;PRIO:7
3073 014770 010046                   MOV    R0,-(SP)      ;PUSH R0 ON STACK
3074 014772 010146                   MOV    R1,-(SP)      ;PUSH R1 ON STACK
3075 014774 010246                   MOV    R2,-(SP)      ;PUSH R2 ON STACK
3076 014776 010346                   MOV    R3,-(SP)      ;PUSH R3 ON STACK
3077 015000 010446                   MOV    R4,-(SP)      ;PUSH R4 ON STACK
3078 015002 010546                   MOV    R5,-(SP)      ;PUSH R5 ON STACK
3079 015004 017746 164030          MOV    @SWR,-(SP)    ;PUSH @SWR ON STACK
3080 015010 010637 015124          MOV    SP,$SAVR6    ;SAVE SP
3081 015014 012737 015026 000024  MOV    #SPWRUP,@#PWRVEC ;SET UP VECTOR
3082 015022 000000          HALT
3083 015024 000776          BR      .-2          ;HANG UP
3084
3085
3086      ;:*****
3087      ;*POWER UP ROUTINE
3088      ;:*****
3089 015026 012737 015120 000024 $PWRUP: MOV    #SILLUP,@#PWRVEC ;SET FOR FAST DOWN
3090 015034 013706 015124          MOV    $SAVR6,SP    ;GET SP
3091 015040 012677 163774          MOV    (SP)+,@SWR   ;POP STACK INTO @SWR
3092 015044 012605          MOV    (SP)+,R5
3093 015046 012604          MOV    (SP)+,R4    ;POP STACK INTO R4
3094 015050 012603          MOV    (SP)+,R3    ;POP STACK INTO R3
3095 015052 012602          MOV    (SP)+,R2    ;POP STACK INTO R2
3096 015054 012601          MOV    (SP)+,R1    ;POP STACK INTO R1
3097 015056 012600          MOV    (SP)+,R0    ;POP STACK INTO R0
3098 015060 012737 014754 000024  MOV    #SPWRDN,@#PWRVEC ;SET UP THE POWER DOWN VECTOR
3099 015066 012737 000340 000026  MOV    #340,@#PWRVEC+2 ;PRIO:7
3100 015074 005037 015124          CLR    $SAVR6      ;WAIT LOOP FOR THE TTY
3101 0 00 005237 015124 1$: INC    $SAVR6      ;WAIT FOR THE INC
3102 015104 001375          BNE    1$          ;OF WORD
3103 015106 104401          TYPE          ;REPORT THE POWER FAILURE
3104 015110 015126          $PWRMG: .WORD    $POWER ;POWER FAIL MESSAGE POINTER
3105 015112 013716 001006          MOV    $LPADR,(SP) ;CHOCOLATE FUDGE RETURN TO BEGINNING OF TEST
3106 015116 000002          RTI          ;RETURN TO THERE
3107 015120 000000          $ILLUP: HALT      ;THE POWER UP SEQUENCE WAS STARTED
3108 015122 000776          BR      .-2      ; BEFORE THE POWER DOWN WAS COMPLETE
3109 015124 000000          $SAVR6: 0        ;PUT THE SP HERE
3110 015126 015 012 120 $POWER: .ASCIZ <15><12>'POWER'
3111

```

3113  
3114

```

.SBTTL SCOPE HANDLER ROUTINE
:*****
:THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
:AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
:AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
:THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
:SW14=1      LOOP ON TEST
:SW09=1      LOOP ON ERROR
:CALL
:SCOPE      SCOPE      ;;SCOPE=IOT
$SCOPE:
015136      104406      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
015136      104406      1$: BIT      #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
015140      032777      040000      163672      BNE      $OVER      ;;YES IF SW14=1
015146      001052      ;*****START OF CODE FOR THE XOR TESTER*****
015150      000416      $XTSTR: BR      6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
015152      013746      000004      MOV      @#ERRVEC,-(SP)      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
015156      012737      015176      000004      MOV      #5$,@#ERRVEC      ;;SET FOR TIMEOUT
015164      005737      177060      TST      @#177060      ;;TIME OUT ON XOR?
015170      012637      000004      MOV      (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
015174      000421      BR      $SVLAD      ;;GO TO THE NEXT TEST
015176      022626      5$: CMP      (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
015200      012637      000004      MOV      (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
015204      000407      BR      7$      ;;LOOP ON THE PRESENT TEST
015206      6$::*****END OF CODE FOR THE XOR TESTER*****
015206      105737      001003      2$: TSTB      $ERFLG      ;;HAS AN ERROR OCCURRED?
015212      001412      BEQ      $SVLAD      ;;BR IF NO
015214      032777      001000      163616      BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
015222      001404      BEQ      4$      ;;BR IF NO
015224      013737      001010      001006      7$: MOV      $LPERR,$LPADR      ;;SET LOOP ADDRESS TO LAST SCOPE
015232      000420      BR      $OVER
015234      105037      001003      4$: CLRB      $ERFLG      ;;ZERO THE ERROR FLAG
015240      105237      001002      $SVLAD: INCB      $STNM      ;;COUNT TEST NUMBERS
015244      113737      001002      001072      MOVB      $STNM,$TESTN      ;;SET TEST NUMBER IN APT MAILBOX
015252      011637      001006      MOV      (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
015256      011637      001010      MOV      (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
015262      005037      001060      CLR      $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
015266      112737      000001      001015      MOVB      #1,$ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
015274      013777      001002      163540      $OVER: MOV      $STNM,@DISPLAY      ;;DISPLAY TEST NUMBER
015302      013716      001006      MOV      $_ADR,(SP)      ;;FUDGE RETURN ADDRESS
015306      000002      RTI      ;;FIXES PS

```

3115

```

3117
3118
3119
3120
3121
3122 015310 112737 000001 015554 SATY1:  MOVB  #1,$FFLG      ;TO REPORT FATAL ERROR
3123 015316 112737 000001 015552 SATY3:  MOVB  #1,$MFLG      ;TO TYPE A MESSAGE
3124 015324 000403
3125 015326 112737 000001 015554 SATY4:  MOVB  #1,$FFLG      ;TO ONLY REPORT FATAL ERROR
3126 015334
3127 015334 010046
3128 015336 010146
3129 015340 105737 015552
3130 015344 001450
3131 015346 122737 000001 001106
3132 015354 001031
3133 015356 132737 000100 001107
3134 015364 001425
3135 015366 017600 000004
3136 015372 062766 000002 000004
3137 015400 005737 001066 1$:
3138 015404 001375
3139 015406 010037 001102
3140 015412 105720 2$:
3141 015414 001376
3142 015416 163700 001102
3143 015422 006200
3144 015424 010037 001104
3145 015430 012737 000004 001066
3146 015436 000413
3147 015440 017637 000004 015464 3$:
3148 015446 062766 000002 000004
3149 015454 013746 177776
3150 015460 004737 015556
3151 015464 000000 4$:
3152 015466 5$:
3153 015466 105737 015554 10$:
3154 015472 001413
3155 015474 005737 001106
3156 015500 001410
3157 015502 005737 001066 11$:
3158 015506 001375
3159 015510 017637 000004 001070
3160 015516 005237 001066
3161 015522 062766 000002 000004 12$:
3162 015530 105037 015554
3163 015534 105037 015553
3164 015540 105037 015552
3165 015544 012601
3166 015546 012600
3167 015550 000207
3168 015552 000
3169 015553 000
3170 015554 000
3171
3172

```

```

:*****
.SBTTL  APT COMMUNICATIONS ROUTINE
:*****

```

```

MOV  R0,-(SP)      ;PUSH R0 ON STACK
MOV  R1,-(SP)      ;PUSH R1 ON STACK
TSTB $MFLG        ;SHOULD TYPE A MESSAGE?
BEQ  5$            ;IF NOT: BR
CMPB #APTENV,$ENV ;OPERATING UNDER APT?
BNE  3$            ;IF NOT: BR
BITB #APTSPOOL,$ENVM ;SHOULD SPOOL MESSAGE?
BEQ  3$            ;IF NOT: BR
MOV  @4(SP),R0     ;GET MESSAGE ADDRESS
ADD  #2,4(SP)      ;BUMP RETURN ADDRESS
TST  $MSGTYPE      ;SEE IF DONE W/ LAST XMISSION?
BNE  1$            ;IF NOT: WAIT
MOV  R0,$MSGAD     ;PUT ADDRESS IN MAILBOX
TSTB (R0)+         ;FIND END OF MESSAGE
BNE  2$            ;SUB START OF MESSAGE
SUB  $MSGAD,R0     ;GET MESSAGE LENGTH IN WORDS
ASR  R0            ;PUT LENGTH IN MAILBOX
MOV  R0,$MSGGLGT  ;TELL APT TO TAKE MESSAGE
BR   5$
MOV  @4(SP),4$     ;PUT MSG ADDR IN JSR LINKAGE
ADD  #2,4(SP)      ;BUMP RETURN ADDRESS
MOV  177776,-(SP) ;PUSH 177776 ON STACK
JSR  PC,$TYPE     ;CALL TYPE MACRO
WORD 0
TSTB $FFLG        ;SHOULD REPORT FATAL ERROR?
BEQ  12$          ;IF NOT: BR
TST  $ENV         ;RUNNING UNDER APT?
BEQ  12$          ;IF NOT: BR
TST  $MSGTYPE     ;FINISHED LAST MESSAGE?
BNE  11$          ;IF NOT: WAIT
MOV  @4(SP),$FATAL ;GET ERROR #
INC  $MSGTYPE     ;TELL APT TO TAKE ERROR
ADD  #2,4(SP)     ;BUMP RETURN ADDRESS
CLRB $FFLG       ;CLEAR FATAL FLAG
CLRB $LFLG       ;CLEAR LOG FLAG
CLRB $MFLG       ;CLEAR MESSAGE FLAG
MOV  (SP)+,R1    ;POP STACK INTO R1
MOV  (SP)+,R0    ;POP STACK INTO R1
RTS  PC          ;RETURN
$MFLG: .BYTE 0
$LFLG: .BYTE 0
$FFLG: .BYTE 0

```

.EVEN

CZDLDF0 DL11-W/1144 MFM SLU  
APT COMMUNICATIONS ROUTINE

MACRO M1113 05-NOV-80 12:32 PAGE <sup>E 8</sup>75-1 SEQUENCE 95

3173 000200  
3174 000001  
3175 000100  
3176 000040  
3177

APTSIZE=200  
APTENV=001  
APTSPool=100  
APTCSUP=040



```

.SBTTL TYPE ROUTINE
*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*
015556 105737 001057 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
015562 100002 BPL 1$ ;; BR IF YES
015564 000000 HALT ;; HALT HERE IF NO TERMINAL
015566 000430 BR 3$ ;; LEAVE
015570 010046 1$: MOV RO,-(SP) ;; SAVE RO
015572 017600 000002 MOV @2(SP),RO ;; GET ADDRESS OF ASCIZ STRING
015576 122737 000001 001106 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
015604 001011 BNE 62$ ;; NO,GO CHECK FOR APT CONSOLE
015606 132737 000100 001107 BITB #APTSPOOL,$ENVM ;; SPOOL MESSAGE TO APT
015614 001405 BEQ 62$ ;; NO,GO CHECK FOR CONSOLE
015616 010037 015626 MOV RO,61$ ;; SETUP MESSAGE ADDRESS FOR APT
015622 004737 015316 JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
015626 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
015630 132737 000040 001107 62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
015636 001003 BNE 60$ ;; YES,SKIP TYPE OUT
015640 112046 2$: MOV (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
015642 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
015644 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
015646 012600 60$: MOV (SP)+,RO ;; RESTORE RO
015650 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
015654 000002 RTI ;; RETURN
015656 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
015662 001430 BEQ 8$
015664 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
015670 001006 BNE 5$
015672 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
015674 104401 TYPE ;; TYPE A CR AND LF
015676 001063 $CRLF
015700 105037 016104 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
015704 000755 BR 2$ ;; GET NEXT CHARACTER
015706 004737 015770 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
015712 123726 001056 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
015716 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
015720 013746 001054 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
;; AND THE NULL CHAR.
015724 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
015730 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
015732 004737 015770 JSR PC,$TYPEC ;; GO TYPE A NULL
015736 105337 016104 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
015742 000770 BR 7$ ;; LOOP
;HORIZONTAL TAB PROCESSOR

```

```

015744 112716 000040      8$:   MOVB   #' , (SP)           ;; REPLACE TAB WITH SPACE
015750 004737 015770      9$:   JSR    PC, $TYPEC          ;; TYPE A SPACE
015754 132737 000007 016104   BITB   #7, $CHARCNT         ;; BRANCH IF NOT AT
015762 001372           BNE    9$                  ;; TAB STOP
015764 005726           TST    (SP)+              ;; POP SPACE OFF STACK
015766 000724           BR     2$                  ;; GET NEXT CHARACTER
015770 105777 163054   $TYPEC: TSTB  @ $TPS          ;; WAIT UNTIL PRINTER IS READY
015774 100375           BPL   $TYPEC
015776 116677 000002 163046   MOVB   2(SP), @ $TPB       ;; LOAD CHAR TO BE TYPED INTO DATA REG.
016004 105777 163034   TSTB  @ $TKS              ;; SEE IF KEYBOARD IS TALKING.
016010 100021           BPL   2$                  ;; BRANCH IF IT ISN'T.
016012 017746 163030   MOV    @ $TKB, -(SP)       ;; PUSH CHARACTER ONTO STACK.
016016 042716 177600   BIC    #177600, (SP)       ;; BIT CLEAR TOP BYTE AND PARITY BIT.
016022 022726 000023   CMP    #23, (SP)+         ;; SEE IF THIS IS A ^S.
016026 001012           BNE    2$                  ;; BRANCH TO CONTINUE IF IT ISN'T.
016030 105777 163010      3$:   TSTB  @ $TKS              ;; WAIT FOR ANOTHER INPUT.
016034 100375           BPL   3$                  ;; BRANCH BACK IF NOT READY.
016036 017746 163004   MOV    @ $TKB, -(SP)       ;; PUSH NEXT CHARACTER ON STACK.
016042 042716 177600   BIC    #177600, (SP)       ;; BIT CLEAR TOP BYTE AND PARITY BIT.
016046 022726 000021   CMP    #21, (SP)+         ;; SEE IF THIS IS A ^Q.
016052 001366           BNE    3$                  ;; BRANCH BACK FOR MORE WAIT IF NOT.
016054 122766 000015 000002  2$:   CMPB  #CR, 2(SP)          ;; IS CHARACTER A CARRIAGE RETURN?
016062 001003           BNE    1$                  ;; BRANCH IF NO
016064 105037 016104   CLRB  $CHARCNT           ;; YES--CLEAR CHARACTER COUNT
016070 000406           BR     $TYPEX             ;; EXIT
016072 122766 000012 000002  1$:   CMPB  #LF, 2(SP)          ;; IS CHARACTER A LINE FEED?
016100 001402           BEQ   $TYPEX             ;; BRANCH IF YES
016102 105227           INCB  (PC)+              ;; COUNT THE CHARACTER
016104 000000   $CHARCNT: .WORD 0         ;; CHARACTER COUNT STORAGE
016106 000207   $TYPEX: RTS   PC

3181
.SBTL  BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV    NUM, -(SP)       ;; NUMBER TO BE TYPED
*   TYPOS  ;; CALL FOR TYPEOUT
*   .BYTE  N                ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M                ;; M=1 OR 0
*                               ;; 1=TYPE LEADING ZEROS
*                               ;; 0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV    NUM, -(SP)       ;; NUMBER TO BE TYPED
*   TYPON  ;; CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV    NUM, -(SP)       ;; NUMBER TO BE TYPED
*   TYPOC  ;; CALL FOR TYPEOUT
016110 017646 000000 016333  $TYPOS: MOV    @ (SP), -(SP)    ;; PICKUP THE MODE
016114 116637 000001   MOVB   1(SP), $OFILL      ;; LOAD ZERO FILL SWITCH
016122 112637 016335   MOVB   (SP)+, $OMODE+1    ;; NUMBER OF DIGITS TO TYPE

```

```

016126 062716 000002          ADD    #2,(SP)          ;;ADJUST RETURN ADDRESS
016132 000406          BR     $TYPON
016134 112737 000001 016333 $TYPOC: MOVB  #1,$OFILL          ;;SET THE ZERO FILL SWITCH
016142 112737 000006 016335          MOVB  #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
016150 112737 000005 016332 $TYPON: MOVB  #5,$OCNT          ;;SET THE ITERATION COUNT
016156 010346          MOV    R3,-(SP)        ;;SAVE R3
016160 010446          MOV    R4,-(SP)        ;;SAVE R4
016162 010546          MOV    R5,-(SP)        ;;SAVE R5
016164 113704 016335          MOVB  $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
016170 005404          NEG    R4
016172 062704 000006          ADD    #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
016176 110437 016334          MOVB  R4,$OMODE        ;;SAVE IT FOR USE
016202 113704 016333          MOVB  $OFILL,R4        ;;GET THE ZERO FILL SWITCH
016206 016605 000012          MOV    12(SP),R5       ;;PICKUP THE INPUT NUMBER
016212 005003          CLR    R3              ;;CLEAR THE OUTPUT WORD
016214 006105          1$:  ROL    R5          ;;ROTATE MSB INTO 'C'
016216 000404          BR     3$              ;;GO DO MSB
016220 006105          2$:  ROL    R5          ;;FORM THIS DIGIT
016222 006105          ROL    R5
016224 006105          ROL    R5
016226 010503          MOV    R5,R3
016230 006103          3$:  ROL    R3           ;;GET LSB OF THIS DIGIT
016232 105337 016334          DECB  $OMODE           ;;TYPE THIS DIGIT?
016236 100016          BPL    7$              ;;BR IF NO
016240 042703 177770          BIC    #177770,R3      ;;GET RID OF JUNK
016244 001002          BNE    4$              ;;TEST FOR 0
016246 005704          TST    R4              ;;SUPPRESS THIS 0?
016250 001403          BEQ    5$              ;;BR IF YES
016252 005204          4$:  INC    R4           ;;DON'T SUPPRESS ANYMORE 0'S
016254 052703 000060          BIS    #'0,R3         ;;MAKE THIS DIGIT ASCII
016260 052703 000040          5$:  BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
016264 110337 016330          MOVB  R3,8$           ;;SAVE FOR TYPING
016270 104401 016330          TYPE  ,8$             ;;GO TYPE THIS DIGIT
016274 105337 016332          7$:  DECB  $OCNT        ;;COUNT BY 1
016300 003347          BGT    2$              ;;BR IF MORE TO DO
016302 002402          BLT    6$              ;;BR IF DONE
016304 005204          INC    R4              ;;INSURE LAST DIGIT ISN'T A BLANK
016306 000744          BR     2$              ;;GO DO THE LAST DIGIT
016310 012605          6$:  MOV    (SP)+,R5     ;;RESTORE R5
016312 012604          MOV    (SP)+,R4     ;;RESTORE R4
016314 012603          MOV    (SP)+,R3     ;;RESTORE R3
016316 016666 000002 000004          MOV    2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
016324 012616          MOV    (SP)+,(SP)
016326 000002          RTI
016330 000          8$:  .BYTE  0           ;;RETURN
016331 000          .BYTE  0           ;;STORAGE FOR ASCII DIGIT
016332 000          $OCNT: .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
016333 000          $OFILL: .BYTE 0    ;;OCTAL DIGIT COUNTER
016334 000000          $OMODE: .WORD 0     ;;ZERO FILL SWITCH
                                ;;NUMBER OF DIGITS TO TYPE
    
```

3184  
3185

```

.SBTTL TTY INPUT ROUTINE
*****
.ENABL LSB
*****
*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.

016336 022737 000176 001040 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
016344 001074 BNE 15$ ;; BRANCH IF NO
016346 105777 162472 TSTB @STKS ;; CHAR THERE?
016352 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
016354 117746 162466 MOVB @STKB,-(SP) ;; SAVE THE CHAR
016360 042716 177600 BIC #^C177,(SP) ;; STRIP-OFF THE ASCII
016364 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
016370 001062 BNE 15$ ;; NO, RETURN TO USER
016372 123727 001034 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
016400 001456 BEQ 15$ ;; BRANCH IF YES
016402 104401 017063 TYPE , $CNTLG ;; ECHO THE CONTROL-G (^G)
016406 104401 017070 $GTSWR: TYPE , $MSWR ;; TYPE CURRENT CONTENTS
016412 013746 000176 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
016416 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
016420 104401 017101 TYPE , $MNEW ;; PROMPT FOR NEW SWR
016424 005046 19$: CLR -(SP) ;; CLEAR COUNTER
016426 005046 CLR -(SP) ;; THE NEW SWR
016430 105777 162410 7$: TSTB @STKS ;; CHAR THERE?
016434 100375 BPL 7$ ;; IF NOT TRY AGAIN
016436 117746 162404 MOVB @STKB,-(SP) ;; PICK UP CHAR
016442 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
016446 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
016452 001005 BNE 10$ ;; BRANCH IF NOT
016454 104401 017056 TYPE , $CNTLU ;; YES, ECHO CONTROL-U (^U)
016460 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
016464 000757 BR 19$ ;; LET'S TRY IT AGAIN
016466 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
016472 001022 BNE 16$ ;; BRANCH IF NO
016474 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
016500 001403 BEQ 11$ ;; BRANCH IF YES
016502 016677 000002 162330 MOV 2(SP),@SWR ;; SAVE NEW SWR
016510 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
016514 104401 001063 14$: TYPE , $CRLF ;; ECHO <CR> AND <LF>
016520 123727 001035 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
016526 001003 BNE 15$ ;; BRANCH IF NOT
016530 012777 000100 162306 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
016536 000002 15$: RTI ;; RETURN
016540 004737 015770 16$: JSR PC,$TYPEC ;; ECHO CHAR
016544 021627 000060 CMP (SP),#60 ;; CHAR < 0?
016550 002420 BLT 18$ ;; BRANCH IF YES
016552 021627 000067 CMP (SP),#67 ;; CHAR > 7?
016556 003015 BGT 18$ ;; BRANCH IF YES
016560 042726 000060 BIC #60,(SP)+ ;; STRIP-OFF ASCII
016564 005766 000002 TST 2(SP) ;; IS THIS THE FIRST CHAR
016570 001403 BEQ 17$ ;; BRANCH IF YES
016572 006316 ASL (SP) ;; NO, SHIFT PRESENT
016574 006316 ASL (SP) ;; CHAR OVER TO MAKE
016576 006316 ASL (SP) ;; ROOM FOR NEW ONE.

```

016600 005266 000002  
016604 056616 177776  
016610 000707  
016612 104401 001062  
016616 000720

```
17$: INC 2(SP) ;;KEEP COUNT OF CHAR
      BIS -2(SP),(SP) ;;SET IN NEW CHAR
      BR 7$ ;;GET THE NEXT ONE
18$: TYPE ,SQUES ;;TYPE ?<CR><LF>
      BR 20$ ;;SIMULATE CONTROL-U
```

.DSABL LSB  
\*\*\*\*\*

\*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

\*CALL:

```
* RDCHR ;;INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE ;;CHARACTER IS ON THE STACK
* ;;WITH PARITY BIT STRIPPED OFF
```

016620 011646  
016622 016666 000004 000002  
016630 105777 162210  
016634 100375  
016636 117766 162204 000004  
016644 042766 177600 000004  
016652 026627 000004 000023  
016660 001013  
016662 105777 162156  
016666 100375  
016670 117746 162152  
016674 042716 177600  
016700 022627 000021  
016704 001366  
016706 000750  
016710 026627 000004 000140  
016716 002407  
016720 026627 000004 000175  
016726 003003  
016730 042766 000040 000004  
016736 000002

```
$RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
          MOV 4(SP),2(SP) ;;SAVE THE PS
1$: TSTB @STKS ;;WAIT FOR
     BPL 1$ ;;A CHARACTER
     MOVB @STKB,4(SP) ;;READ THE TTY
     BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
     CMP 4(SP),#23 ;;IS IT A CONTROL-S?
     BNE 3$ ;;BRANCH IF NO
2$: TSTB @STKS ;;WAIT FOR A CHARACTER
     BPL 2$ ;;LOOP UNTIL ITS THERE
     MOVB @STKB,-(SP) ;;GET CHARACTER
     BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
     CMP (SP)+,#21 ;;IS IT A CONTROL-U?
     BNE 2$ ;;IF NOT DISCARD IT
     BR 1$ ;;YES, RESUME
3$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
     BLT 4$ ;;BRANCH IF YES
     CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
     BGT 4$ ;;BRANCH IF YES
     BIC #40,4(SP) ;;MAKE IT UPPER CASE
4$: RTI ;;GO BACK TO USER
```

\*\*\*\*\*

\*THIS ROUTINE WILL INPUT A STRING FROM THE TTY

\*CALL:

```
* RDLIN ;;INPUT A STRING FROM THE TTY
* RETURN HERE ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
* ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
```

016740 010346  
016742 012703 017046  
016746 022703 017056  
016752 101405  
016754 104407  
016756 112613  
016760 122713 000177  
016764 001003  
016766 104401 001062  
016772 000763  
016774 111337 017044  
017000 104401 017044  
017004 122723 000015  
017010 001356  
017012 105063 177777  
017016 104401 001064

```
$RDLIN: MOV R3,-(SP) ;;SAVE R3
1$: MOV #$TTYIN,R3 ;;GET ADDRESS
2$: CMP #$TTYIN+8.,R3 ;;BUFFER FULL?
     BLOS 4$ ;;BR IF YES
     RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
     MOVB (SP)+,(R3) ;;GET CHARACTER
10$: CMPB #177,(R3) ;;IS IT A RUBOUT
     BNE 3$ ;;SKIP IF NOT
4$: TYPE ,SQUES ;;TYPE A '?'
     BR 1$ ;;CLEAR THE BUFFER AND LOOP
3$: MOVB (R3),9$ ;;ECHO THE CHARACTER
     TYPE ,9$
     CMPB #15,(R3)+ ;;CHECK FOR RETURN
     BNE 2$ ;;LOOP IF NOT RETURN
     CLRB -1(R3) ;;CLEAR RETURN (THE 15)
     TYPE ,RLF ;;TYPE A LINE FEED
```

```

017024 011646                    MOV    (SP),-(SP)            ;;ADJUST THE STACK AND PUT ADDRESS OF THE
017026 016666    000004    000002    MOV    4(SP),2(SP)            ;;            FIRST ASCII CHARACTER ON IT
017034 012766    017046    000004    MOV    #STTYIN,4(SP)
017042 000002                    RTI                         ;;RETURN
017044        000                9$:    .BYTE    0                 ;;STORAGE FOR ASCII CHAR. TO TYPE
017045        000                       .BYTE    0                 ;;TERMINATOR
017046                            $TTYIN: .BLKB    8             ;;RESERVE 8 BYTES FOR TTY INPUT
017056        136        125        015    $CNTLU: .ASCIZ /^U/<15><12>    ;;CONTROL 'U'
017063        136        107        015    $CNTLG: .ASCIZ /^G/<15><12>    ;;CONTROL 'G'
017070        015        012        123    $MSWR:  .ASCIZ <15><12>/SWR = /
017101        040        040        116    $MNEW:  .ASCIZ / NEW = /
3186 017022 012603                MOV    (SP)+,R3            ;;RESTORE R3

```

3188  
3189  
3190

017112 010046  
017114 016600 000002  
017120 005740  
017122 111000  
017124 006300  
017126 016000 017146  
017132 000200

017134 011646  
017136 016666 000004 000002  
017144 000002

017146 017134  
017150 015556  
017152 016134  
017154 016110  
017156 016150  
017160 016406  
017162 016336  
017164 016620  
017166 016740

3191

.SBTTL TRAP DECODER

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

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

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

```
$TRAP2: MOV   (SP),-(SP)        ;;MOVE THE PC DOWN
        MOV   4(SP),2(SP)      ;;MOVE THE PSW DOWN
        RTI                          ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
\*BY THE "TRAP" INSTRUCTION.  
ROUTINE

```
$TRPAD: .WORD  $TRAP2
        $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPOC ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST (ALL)
        $GTSWR ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
        $CKSWR ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
        $RDCHR ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
```

3193  
3194  
3195  
3196  
3197  
3198  
3199

.SBTTL ECHO TEST

```

*****
: *THIS ROUTINE WILL ECHO ANY CHARACTER TYPED
: *AT THE CONSOLE
: *THE TEST IS HALTED BY TYPING A CONTROL-C
: *TEST CAN BE RESTARTED AFTER HALTING BY JUST CONTINUING
*****

```

ECHO:  
3200 017170  
3201 017170 000005  
3202 017172 112777 000052 163462  
3203 017200 105777 163450  
3204 017204 100375  
3205 017206 117777 163444 163446  
3206 017214 017700 163436  
3207 017220 100023  
3208 017222 052701 010000  
3209 017226 030100  
3210 017230 001403  
3211 017232 004737 017314  
3212 017236 017342  
3213 017240 006301  
3214 017242 030100  
3215 017244 001403  
3216 017246 004737 017314  
3217 017252 017353  
3218 017254 006301  
3219 017256 030100  
3220 017260 001403  
3221 017262 004737 017314  
3222 017266 017365  
3223 017270 042700 000200  
3224 017274 022700 000003  
3225 017300 001337  
3226 017302 004737 017314  
3227 017306 017400  
3228 017310 000000  
3229 017312 000726

```

RESET                                     ;CLEAR EVERYTHING
MOVW #*,@CTBUF                             ;TRANSMIT PROMPT '*'
2$: TSTB @CRCSR                             ;WAIT FOR INPUT
    BPL 2$
    MOVW @CRBUF,@CTBUF                       ;ECHO INPUT
    MOV @CRBUF,R0                            ;STORE INPUT
    BPL 5$                                   ;BR IF 'ERROR' NOT SET
    BIS #BIT12,R1                            ;SET PARITY ERROR TEST MASK
    BIT R1,R0                                ;CHECK FOR PARITY ERROR FLAG
    BEQ 3$                                   ;BR IF NOT SET
    JSR PC,MSG                               ;REPORT PARITY ERROR
3$: MPAR
    ASL R1                                    ;SHIFT MASK TO TEST 'FR' FLAG
    BIT R1,R0                                ;TEST FOR FRAMING ERROR FLAG
    BEQ 4$                                   ;BR IF NOT SET
    JSR PC,MSG                               ;REPSORT FRAMING ERROR
4$: MFR
    ASL R1                                    ;SHIFT MASK TO TEST 'OR' FLAG
    BIT R1,R0                                ;TEST FOR OVERFLOW ERROR
    BEQ 5$                                   ;BR IF NOT SET
    JSR PC,MSG                               ;REPORT OVERFLOW ERROR
5$: BIC #BIT7,R0                             ;CLEAR ANY PARITY BIT
    CMP #3,R0                                ;WAS INPUT CONTROL-C
    BNE 2$                                   ;BR IF NOT
    JSR PC,MSG                               ;REPORT PROGRAM STOP
MSTOP
HALT                                       ;END OF TEST HALT
BR ECHO                                    ;AFTER END OF TEST HALT
; PRESS CONTINUE TO RESTART ECHO TEST

```

3230  
3231  
3232 017314 013600  
3233 017316 062746 000002  
3234 017322 105777 163332  
3235 017326 100375  
3236 017330 112077 163326  
3237 017334 105710  
3238 017336 001371  
3239 017340 000207  
3240  
3241 017342 015 012 120  
3242 017353 015 012 106  
3243 017365 015 012 117  
3244 ^17400 015 012 123

```

MSG: MOV @(SP)+,R0                          ;PICK UP MESSAGE POINTER
      ADD #2,-(SP)                          ;ADJUST RETURN PC
WAIT: TSTB @CTCSR                            ;WAIT FOR XMIT DONE
      BPL WAIT
      MOVW (R0)+,@CTBUF                      ;SEND CHARACTER
      TSTB (R0)                              ;IS THIS END OF MESSAGE?
      BNE WAIT                               ;BR IF NOT
      RTS PC                                 ;RETURN
MPAR: .ASCIZ <CR><LF>/PARITY/
MFR:  .ASCIZ <CR><LF>/FRAMING/
MOR:  .ASCIZ <CR><LF>/OVERFLOW/
MSTOP: .ASCIZ <CR><LF>/STOP/

```



3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278  
3279

017410 012701 000040  
017414 012700 000040  
017420 105777 163234  
017424 100375  
017426 010177 163230  
017432 105201  
017434 005300  
017436 001370  
017440 004737 017314  
  
017444 001063  
017446 105777 163202  
017452 100404  
017454 032701 000200  
017460 001353  
017462 000754  
  
017464 005077 163166  
017470 000000  
017472 000746

```

.EVEN
.SBTTL  TERMINAL OUTPUT TEST
;*****
; *THIS ROUTINE WILL OUTPUT ALL WRITABLE CHARACTERS FOR THE
; *THE OCTAL CODE 040 --> 377
; *32 CHARACTERS ARE PRINTED ON EACH LINE
; *THE PATTERN IS REPEATED EVERY THREE LINES
; *
;*****
OUTTST: MOV     #40,R1           ;LOAD FIRST WRITABLE CHARACTER
1$:    MOV     #40,R0           ;LOAD CHAR COUNT PER LINE
2$:    TSTB   @CTCSR           ;WAIT FOR DONE
        BPL    2$
        MOV   R1,@CTBUF        ;TRANSMIT A CHARACTER
        INCB  R1               ;INCREMENT CHARACTER CODE
        DEC   R0               ;DECREMENT CHAR COUNT
        BNE   2$               ;BR IF LINE NOT COMPLETE
        JSR   PC,MSG           ;SSUE CR,LINE FEED
        $CRLF
        TSTB  @CRCSR           ;ANY CHARACTER RECEIVED?
        BMI   3$               ;BR IF YES
        BIT   #BIT7,R1         ;FINISHED ONE PASS OF WRITABLE CHARACTERS?
        BNE   OUTTST           ;BR IF YES
        BR    1$               ;IF NOT WRITE NEXT LINE
3$:    CLR    @CRBUF           ;CLEAR RECEIVER
        HALT
        BR    OUTTST           ;RESTART TEST IF CONTINUED

```

3281									
3282									
3283	017474	103	101	116	EM1:	.ASCIZ	/CAN NOT ACCESS TCSR/		
3284	017520	103	101	116	EM2:	.ASCIZ	/CAN NOT ACCESS TBUF/		
3285	017544	124	103	123	EM3:	.ASCIZ	/TCSR DONE NOT CLEARED WITH TBUF FULL/		
3286	017611	124	103	123	EM4:	.ASCIZ	/TCSR DONE NOT SET/		
3287	017633	124	103	123	EM5:	.ASCIZ	/TCSR DONE NOT SET WITH RESET/		
3288	017670	103	101	116	EM6:	.ASCIZ	/CAN NOT ACCESS RCSR/		
3289	017714	103	101	116	EM7:	.ASCIZ	/CAN NOT ACCESS RBUF/		
3290	017740	103	101	116	EM10:	.ASCIZ	/CAN NOT ACCESS LKS/		
3291	017763	102	111	124	EM11:	.ASCIZ	/BIT0 OF TCSR NOT CLEAR AFTER RESET/		
3292	020026	103	101	116	EM12:	.ASCIZ	/CAN NOT SET BIT0 OF TCSR/		
3293	020057	103	101	116	EM13:	.ASCIZ	/CAN NOT CLEAR BIT0 OF TCSR/		
3294	020112	122	105	123	EM14:	.ASCIZ	/RESET DID NOT CLEAR BIT0 OF TCSR/		
3295	020153	102	111	124	EM15:	.ASCIZ	/BIT2 OF TCSR NOT CLEAR AFTER RESET/		
3296	020216	103	101	116	EM16:	.ASCIZ	/CAN NOT SET BIT2 OF TCSR/		
3297	020247	103	101	116	EM17:	.ASCIZ	/CAN NOT CLEAR BIT2 OF TCSR/		
3298	020302	122	105	123	EM20:	.ASCIZ	/RESET DID NOT CLEAR BIT2 OF TCSR/		
3299	020343	102	111	124	EM21:	.ASCIZ	/BIT6 OF TCSR NOT CLEAR AFTER RESET/		
3300	020406	130	115	111	EM22:	.ASCIZ	/XMIT INTERRUPT AT PRIORITY 7/		
3301	020443	103	101	116	EM23:	.ASCIZ	/CAN NOT SET BIT6 OF TCSR/		
3302	020474	103	101	116	EM24:	.ASCIZ	/CAN NOT CLEAR BIT6 OF TCSR/		
3303	020527	122	105	123	EM25:	.ASCIZ	/RESET DID NOT CLEAR BIT6 OF TCSR/		
3304	020570	102	111	124	EM26:	.ASCIZ	/BIT6 OF RCSR NOT CLEAR AFTER RESET/		
3305	020633	122	103	126	EM27:	.ASCIZ	/RCVR INTERRUPT WITH PRIORITY 7/		
3306	020672	103	101	116	EM30:	.ASCIZ	/CAN NOT SET BIT6 OF RCSR/		
3307	020723	103	101	116	EM31:	.ASCIZ	/CAN NOT CLEAR BIT6 OF RCSR/		
3308	020756	103	101	116	EM32:	.ASCIZ	/CAN NOT CLEAR BIT6 OF RCSR WITH RESET/		
3309	021024	102	111	124	EM33:	.ASCIZ	/BIT6 OF LKS NOT CLEAR AFTER RESET/		
3310	021066	114	113	123	EM34:	.ASCIZ	/LKS INTERRUPT WITH PRIORITY 7/		
3311	021124	103	101	116	EM35:	.ASCIZ	/CAN NOT SET BIT6 OF LKS/		
3312	021154	103	101	116	EM36:	.ASCIZ	/CAN NOT CLEAR BIT6 OF LKS/		
3313	021206	122	105	123	EM37:	.ASCIZ	/RESET DID NOT CLEAR BIT6 OF LKS/		
3314	021246	104	125	101	EM40:	.ASCIZ	/DUAL ADDRESSING ERROR/		
3315	021274	102	111	124	EM41:	.ASCIZ	/BIT6 OF LKS NOT SET AFTER RESET/		
3316	021334	103	101	116	EM42:	.ASCIZ	/CAN NOT CLEAR BIT7 OF LKS/		
3317	021366	102	111	124	EM43:	.ASCIZ	/BIT7 OF LKS DOES NOT SET/		
3318	021417	122	124	103	EM44:	.ASCIZ	/RTC INTERRUPT AT PRIORITY 7/		
3319	021453	122	124	103	EM45:	.ASCIZ	/RTC INTERRUPTS WHEN DISABLED/		
3320	021510				EM47:				
3321	021510	122	124	103	EM46:	.ASCIZ	/RTC INTERRUPT DID NOT OCCUR/		
3322	021544	122	124	103	EM50:	.ASCIZ	/RTC DOUBLE INTERRUPT/		
3323	021571	122	105	123	EM51:	.ASCIZ	/RESET DID NOT INTERRUPT/		
3324	021621	122	124	103	EM52:	.ASCIZ	/RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS/		
3325	021676	103	114	117	EM53:	.ASCIZ	/CLOCK REPEATABILITY/		
3326	021722	130	115	111	EM54:	.ASCIZ	/XMIT INTERRUPTS WHEN DISABLED/		
3327	021760	130	115	111	EM56:	.ASCIZ	/XMIT INTERRUPTS AT PRIORITY 7/		
3328	022016	130	115	111	EM57:	.ASCIZ	/XMIT INTERRUPTS WITH ENABLE CLEAR/		
3329	022060				EM55:				
3330	022060	130	115	111	EM60:	.ASCIZ	/XMIT DID NOT INTERRUPT/		
3331	022107	130	115	111	EM61:	.ASCIZ	/XMIT RE-INTERRUPTED/		
3332	022133	114	117	101	EM62:	.ASCIZ	/LOADING TBUF DID NOT CLEAR INTERRUPT/		
3333	022200	122	103	126	EM63:	.ASCIZ	/RCVR ACTIVE NOT SET/		
3334	022224	122	103	126	EM64:	.ASCIZ	/RCVR DONE NEVER SET/		
3335	022250	122	103	126	EM65:	.ASCIZ	/RCVR ACTIVE NOT CLEARED WITH DONE SET/		
3336	022316	122	105	123	EM66:	.ASCIZ	/RESET DID NOT CLEAR RCVR DONE/		
3337	022354	122	104	122	EM67:	.ASCIZ	/RDR ENABLE DID NOT CLEAR RCVR DONE/		



CZDLDFG DL11-W/1144 MFM SLU  
TERMINAL OUTPUT TEST

MACRO M1113 05-NOV-80 12:32 PAGE 81-2<sup>D 9</sup> SEQUENCE 107

3394  
3395

000001

.END

SYMBOL TABLE

ABASE = 176500	BGNADD= 003014	DH110 024227	EM27 020633	GETLIN 013760
ACDW1 = 000000	BIT0 = 000001	DH112 024273	EM3 017544	GOAGIN 014330
ACDW2 = 000000	BIT00 = 000001	DH2 023702	EM30 020672	GTSWR = 104405
ACPUOP= 000000	BIT01 = 000002	DH40 024025	EM31 020723	HT = 000011
ADDW0 = 000000	BIT02 = 000004	DH53 024061	EM32 020756	ID 004620
ADDW1 = 000000	BIT03 = 000010	DH6 023727	EM33 021024	INIT 003462
ADDW10= 000000	BIT04 = 000020	DH7 023754	EM34 021066	IOTVEC= 000020
ADDW11= 000000	BIT05 = 000040	DISPLA 001042	EM35 021124	JIMSTK 002432
ADDW12= 000000	BIT06 = 000100	DISPRE 000174	EM36 021154	LF = 000012
ADDW13= 000000	BIT07 = 000200	DSWR = 177570	EM37 021206	LKS 002674
ADDW14= 000000	BIT08 = 000400	DT1 024434	EM4 017611	LOC1 002340
ADDW15= 000000	BIT09 = 001000	DT10 024474	EM40 021246	LOC2 002342
ADDW2 = 000000	BIT1 = 000002	DT103 024532	EM41 021274	MANL 003502
ADDW3 = 000000	BIT10 = 002000	DT105 024544	EM42 021334	MFPT = 000007
ADDW4 = 000000	BIT11 = 004000	DT110 024560	EM43 021366	MFR 017353
ADDW5 = 000000	BIT12 = 010000	DT112 024574	EM44 021417	MOR 017365
ADDW6 = 000000	BIT13 = 020000	DT2 024444	EM45 021453	MPAR 017342
ADDW7 = 000000	BIT14 = 040000	DT40 024504	EM46 021510	MSG 017314
ADDW8 = 000000	BIT15 = 100000	DT53 024516	EM47 021510	MSTOP 017400
ADDW9 = 000000	BIT2 = 000004	DT6 024454	EM5 017633	M1 024344
ADEVCT= 000000	BIT3 = 000010	DT7 024464	EM50 021544	M2 024402
ADEVM = 000000	BIT4 = 000020	DVADT 003664	EM51 021571	M2A 024404
ADR 004132	BIT5 = 000040	ECHO 017170	EM52 021621	NOEOP 014214
ADRTBL 002702	BIT6 = 000100	EMTVEC= 000030	EM53 021676	OLDPC 002620
AENV = 000000	BIT7 = 000200	EM1 017474	EM54 021722	OLDSUM 002436
AENVM = 000000	BIT8 = 000400	EM10 017740	EM55 022060	OUTTST 017410
AFATAL= 000000	BIT9 = 001000	EM100 022753	EM56 021760	PIRQ = 177772
AMADR1= 000000	BPT = 000003	EM101 023016	EM57 022016	PIRQVE= 000240
AMADR2= 000000	BPTVEC= 000014	EM102 023044	EM6 017670	PROMPT 002566
AMADR3= 000000	BUF 002454	EM103 023103	EM60 022060	PRO = 000000
AMADR4= 000000	CATCH 014412	EM104 023141	EM61 022107	PR1 = 000040
AMAMS1= 000000	CHKSUM 013706	EM105 023176	EM62 022133	PR2 = 000100
AMAMS2= 000000	CKSWR = 104406	EM106 023221	EM63 022200	PR3 = 000140
AMAMS3= 000000	CLK 013526	EM107 023256	EM64 022224	PR4 = 000200
AMAMS4= 000000	CLKCNT 002444	EM11 017763	EM65 022250	PR5 = 000240
AMSGAD= 000000	COMPARE 007226	EM110 023310	EM66 022316	PR6 = 000300
AMSGLG= 000000	CNTLP 002564	EM111 023340	EM67 022354	PR7 = 000340
AMSGTY= 000000	COMP 013330	EM112 023400	EM7 017714	PS = 177776
AMTYP1= 000000	CONT 006246	EM113 023415	EM70 022417	PSW = 177776
AMTYP2= 000000	CONT41 012252	EM114 023453	EM71 022464	PUTLIN 013726
AMTYP3= 000000	CR = 000015	EM115 023512	EM72 022633	PWRVEC= 000024
AMTYP4= 000000	CRBUF 002656	EM116 023551	EM73 022526	RBUF 002636
APASS = 000000	CRCSR 002654	EM117 023610	EM74 022564	RBUF58= 176502
APRIOR= 000000	CRLF = 000200	EM12 020026	EM75 022633	RCSR 002634
APTCUSU= 000040	CRPSW 002666	EM13 020057	EM76 022662	RCSR58= 176500
APTENV= 000001	CRVECT 002664	EM14 020112	EM77 022706	RCV 013512
APTSIZ= 000200	CTBUF 002662	EM15 020153	ENDADD= 024610	RCVCNT 002440
APTSPO= 000100	CTCSR 002660	EM16 020216	ENDADR 024610	RCVDON 010244
APTSZD 003636	CTPSW 002672	EM17 020247	ENDB7 004676	RDCHR = 104407
ASWREG= 000000	CTSTFL 002624	EM2 017520	ENDEV 014170	RDLIN = 104410
ATESTN= 000000	CTVECT 002670	EM20 020302	ENDMG 014314	RESTIE 014134
AUNIT = 000000	DDISP = 177570	EM21 020343	ENDSTK 002412	RESVEC= 000010
AUSWR = 000400	DEVADR 003014	EM22 020406	ERROR = 104000	RPSW 002646
AVECT1= 000300	DH1 023655	EM23 020443	ERRVEC= 000004	RSTRT 014370
AVECT2= 000000	DH10 024001	EM24 020474	FIRST 002336	RTCPSW 002700
BDVECT 002622	DH103 024126	EM25 020527	FLAG44 003002	RTCVT 002676
BEGIN 003774	DH105 024163	EM26 020570	GETB 014024	RVECT 002644

R6	=%000006	TCSR	002640	TST7	005012	\$ERMAX	001015	\$PASS	001074
R7	=%000007	TCSR58=	176504	TURBUF	003006	\$ERROR	014436	\$PASTM	000506
SAVEPS	002434	TIMER	014060	TURCSR	003004	\$ERRPC	001016	\$POWER	015126
SAVETE	014100	TKVEC =	000060	TUTBUF	003012	\$ERRTB	001146	\$PWRDN	014754
SAVEO	002344	TMP1	002626	TUTCSR	003010	\$ERRTY	014620	\$PWRMG	015110
SAVLOC	002346	TMP2	002630	TVECT	002650	\$ERTTL	001012	\$PWRUP	015026
SCOPE =	000004	TMP3	002632	TYPE =	104401	\$ESCAP	001060	\$QUES	001062
SCPSW	002452	TOLER	007240	TYPOC =	104402	\$ETABL	001106	\$RDCHR	016620
SECND	002616	TPSW	002652	TYPON =	104404	\$ETEND	001146	\$RDLIN	016740
SETADR	004100	TPVEC =	000064	TYPOS =	104403	\$FATAL	001070	\$RDSZ =	000010
SHIFT	003740	TRAPVE=	000034	VCTADR	003670	\$FFLG	015554	\$RTNAD	014306
SIZE	003512	TRTVEC=	000014	VCTTBL	002742	\$FILLC	001056	\$SAVR6	015124
SRPSW	002450	TSTDEV	004050	VECT	004152	\$FILLS	001055	\$SCOPE	015136
STACK =	001100	TSTDVM	003646	WACTV	010126	\$GDADR	001020	\$SETUP=	000137
START	003046	TST1	004172	WAIT	017322	\$GDDAT	001024	\$STUP =	177777
STKMT=	177774	TST10	005044	WDONE	010260	\$GET42	014264	\$SVLAD	015240
STPSW	002446	TST11	005252	WRAP	013536	\$GTSWR	016406	\$SVPC =	000500
SWR	001040	TST12	005404	WRPSW	014400	\$HIBTS	000500	\$SWR =	161000
SWREG	000176	TST13	005544	WT	010214	\$ICNT	001004	\$SWREG	001110
SW0	= 000001	TST14	005614	XCONT	013502	\$ILLUP	015120	\$SWRMK=	000000
SW00	= 000001	TST15	005764	XMIT	013440	\$INTAG	001035	\$TESTN	001072
SW01	= 000002	TST16	006154	XMTCNT	002442	\$ITEMB	001014	\$TKB	001046
SW02	= 000004	TST17	006262	XRET	013506	\$LF	001064	\$TKS	001044
SW03	= 000010	TST2	004244	\$APTHD	000500	\$LFLG	015553	\$TN =	000047
SW04	= 000020	TST20	006500	\$ATYC	015334	\$LPADR	001006	\$TPB	001052
SW05	= 000040	TST21	006656	\$ATY1	015310	\$LPERR	001010	\$TPFLG	001057
SW06	= 000100	TST22	006772	\$ATY3	015316	\$MADR1	001120	\$TPS	001050
SW07	= 000200	TST23	007120	\$ATY4	015326	\$MADR2	001124	\$TRAP	017112
SW08	= 000400	TST24	007306	\$AUTOB	001034	\$MADR3	001130	\$TRAP2	017134
SW09	= 001000	TST25	007416	\$BASE	001142	\$MADR4	001134	\$TRP =	000011
SW1	= 000002	TST26	007544	\$BDADR	001022	\$MAIL	001066	\$TRPAD	017146
SW10	= 002000	TST27	007710	\$BDDAT	001026	\$MAMS1	001116	\$TSTM	000504
SW11	= 004000	TST3	004316	\$CHARC	016104	\$MAMS2	001122	\$TSTM	001002
SW12	= 010000	TST30	010054	\$CKSWR	016336	\$MAMS3	001126	\$TTYIN	017046
SW13	= 020000	TST31	010360	\$CMTAG	001000	\$MAMS4	001132	\$TYPE	015556
SW14	= 040000	TST32	010452	\$CM3 =	000000	\$MBADR	000502	\$TYPEC	015770
SW15	= 100000	TST33	010576	\$CNTLG	017063	\$MFLG	015552	\$TYPEX	016106
SW2	= 000004	TST34	011000	\$CNTLU	017056	\$MNEW	017101	\$TYPOC	016134
SW3	= 000010	TST35	011202	\$CPUOP	001114	\$MSGAD	001102	\$TYPON	016150
SW4	= 000020	TST36	011410	\$CRLF	001063	\$MSGLG	001104	\$TYPOS	016110
SW5	= 000040	TST37	011554	\$DEVCT	001076	\$MSGTY	001066	\$UNIT	001100
SW6	= 000100	TST4	004466	\$DEVM	001144	\$MSWR	017070	\$UNITM	000510
SW7	= 000200	TST40	011712	\$DOAGN	014304	\$MTYP1	001117	\$USWR	001112
SW8	= 000400	TST41	012076	\$ENDAD	014274	\$MTYP2	001123	\$VECT1	001136
SW9	= 001000	TST42	012312	\$ENDCT	014254	\$MTYP3	001127	\$VECT2	001140
TA	002610	TST43	012474	\$ENULL	014310	\$MTYP4	001133	\$XTSTR	015150
TBITVE=	000014	TST44	012642	\$ENV	001106	\$NULL	001054	\$SGET4=	000000
TBUF	002642	TST45	012760	\$ENVM	001107	\$NWTST=	000001	\$OFILL	016333
TBUF58=	176506	TST46	013534	\$EOP	014224	\$OCNT	016332	\$.SERRT	001146
TCLOCK	005536	TST5	004714	\$EOPCT	014246	\$OMODE	016334	\$.SX =	000500
TCONS	004020	TST6	004760	\$ERFLG	001003	\$OVER	015274		
. ABS.	024612								
	000000	000							
		001							

CZDLDF0 DL11-W/1144 MFM SLU      MACRO M11'3    05-NOV-80 12:32    PAGE 81-5<sup>G 9</sup>    SEQUENCE 110  
SYMBOL TABLE

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 36008 WORDS ( 141 PAGES)  
DYNAMIC MEMORY: 20346 WORDS ( 78 PAGES)  
ELAPSED TIME: 00:06:14  
CZDLDF.BIN,CZDLDF.SEQ/-SP/NL:TOC=CZDLDF.MLB/ML,CZDLDF.P11