

DUV11

OFLNE RCVR TSTS  
CNDURAO

AH-T466A-MC  
FICHE 1 OF 1

MAY 1983  
COPYRIGHT © 82-83  
MADE IN USA



Microfiche grid containing multiple frames of data, including tables and diagrams, arranged in a grid pattern on the left side of the page.



.REM \*

I D E N T I F I C A T I O N

PRODUCT NAME: CNDURAO DUV11 OFLNE RCVR TSTS

PRODUCT CODE:AC-T465A-MC

PRODUCT DATE:DEC, 1982,1983

MAINTAINER :DIAGNOSTICS SERVICES/ISS

AUTHOR: K.LIND

\*  
.REM \*

COPYRIGHT (C) 1982,1983  
DIGITAL EQUIPMENT CORPORATION, MAYNARD MASS.

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.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OF RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

\*

GENERAL DESCRIPTION

THIS DIAGNOSTIC CAN CHAIN 16 DUV11'S. THIS MEANS THAT 16 DEVICES CAN BE SEQUENTIALLY EXERCISED. THE DIAGNOSTIC MAKES ONE PASS BEFORE PROCEEDING TO THE NEXT DEVICE, AND CONTINUES EXERCISING ALL DEVICES IN THIS FASHION UNTIL HALTED.

.REM \*

1. THE DUV11 OFFLINE RECEIVER TESTS VERIFY THAT THE RECEIVER CHIP/LOGIC WORKS PROPERLY

\* .REM \*

2. REQUIREMENTS

PDP-11/21 COMPUTER (LSI)

DUV11 SYNCHRONOUS/ISOCRONOUS OPTION

ONE CONSOLE TELETYPE OR EQUIVALENT

\* .REM \*

2.2 STORAGE

THE PROGRAM LOADS INTO 4K OF MEMORY WITH BOOTSTRAP

3. LOADING PROCEDURE

THE STANDARD PROCEDURE FOR LOADING ABSOLUTE BINARY TAPES IS TO BE USED.

	STARTING ADDRESS FOR ABSOLUTE LOADER
4K	017500
8K	037500
12K	057500
16K	077500
20K	117500
24K	137500
28K	157500

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

NOTE: ALL SWITCHES RESIDE INTERNAL TO THE CPU AT ADDRESS 176. THESE MAY BE SET VIA THE CONSOLE TTY BY DIRECTLY MODIFYING LOC. 176.

NOTE: RUNNING UNDER APT-11, THERE IS A USER SWITCH REGISTER CALLED '\$USWR'. IN ORDER TO BE FLEXIBLE ON THE AVAILABILITY OF THE H315 CONNECTOR, ONE BIT PASSES STATUS TO APT-11. BIT 0 IN \$USWR REFLECTS THIS STATUS, A 0 = CONNECTOR PRESENT, A 1 = CONNECTOR NOT AVAILABLE. THE USER CHANGES THE CONTENTS OF THIS LOCATION

WHEN BUILDING THE E TABLE, BY ANSWERING THE  
PROMPT "SWITCH 2".

- 4.1.1 AFTER PROGRAM LOAD (INITIAL PROGRAM START)  
ALL CONSOLE SWITCHES DOWN
- 4.1.2 TO MODIFY DEVICE VECTOR AND CONTROL REGISTER ADDRESSES  
AFTER PROGRAM RESTART OR TO RUN MULTIPLE DEVICES  
SW00=1
- 4.1.3 TO START PROGRAM AT SELECTED TEST AFTER A PROGRAM RESTART  
(ONLY IN SINGLE DEVICE TESTS)  
SW01=1

- 4.1.4 TO LOCK ON SELECTED TEST AFTER A PROGRAM RESTART  
(ONLY IN SINGLE DEVICE TESTS)  
SW14=1  
NOTE1: IN GENERAL SW01 WILL BE USED WHEN SW14=1 IS USED  
NOTE2: WITHOUT SW01=1 "LOCK ON TEST" WILL DEFAULT TO TEST 1
- 4.2 STARTING ADDRESS

THE STARTING ADDRESS FOR ALL TESTS IS 000200  
THE RETARTING ADDRESS FOR ALL TESTS IS 000200  
THE STARTING ADDRESS TO ENTER A SELECTED TEST IS 000200  
THE STARTING ADDRESS TO LOCK ON TEST IS 000200

4.3 PROGRAM AND/OR OPERATOR ACTION

4.3.1 INITIAL PROGRAM START

- 4.3.1.1 LOAD PROGRAM INTO MEMORY WITH ABSOLUTE LOADER
- 4.3.1.2 SET SWITCH REGISTER (LOC. 176) TO ZERO.
- 4.3.1.3 TYPE 200G.
- 4.3.1.4 PROGRAM WILL START.

4.3.1.5 THE PROGRAM WILL TYPE 'DUV11 CNDUR-A TAPE B' (ONCE ONLY)

4.3.1.6 THE PROGRAM WILL TYPE 'R' TO INDICATE THAT IT IS ABOUT  
TO START TESTING ,AND THEN TESTING WILL BEGIN

4.3.2 PROGRAM RESTART WITH ALL SWITCHES DOWN

4.3.2.1 THE PROGRAM WILL TYPE 'R' AND WILL COMMENCE TESTING

4.3.3 PROGRAM RESTART WITH SW00=1

\* .REM \*  
\* .REM \*

- 4.3.3.1 SET SWITCH REGISTER (LOC. 176) TO A 000001.
- 4.3.3.2 TYPE 200G.
- 4.3.3.3 PROGRAM WILL START.
- 4.3.3.4 THE PROGRAM WILL TYPE " 1ST DEVICE: RECEIVER CONTROL REGISTER ADDRESS" AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD
- 4.3.3.5 TYPE IN THE ADDRESS OF THE FIRST RECEIVER CONTROL REGISTER ADDRESS OF THE DUV11 TO BE TESTED FOLLOWED BY A <CARRIAGE RETURN>

IF AN INCORRECT ADDRESS IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL THEN REPEAT THE MESSAGE OF 4.3.3.4

- 4.3.3.6 THE PROGRAM WILL TYPE "VECTOR ADDRESS-" AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD
- 4.3.3.7 TYPE IN THE BASE RECEIVER INTERRUPT VECTOR ADDRESS FOR THE DUV11 TO BE TESTED FOLLOWED BY A <CARRIAGE RETURN>

IF AN INCORRECT ADDRESS IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL THEN REPEAT THE MESSAGE OF 4.3.3.6

- 4.3.3.8 THE PROGRAM WILL TYPE "ARE YOU RUNNING MULTIPLE DEVICES ?" (Y OR N)-" AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD
- 4.3.3.9 TYPE IN THE APPROPRIATE ANSWER YES OR NO FOLLOWED BY A <CARRIAGE RETURN>

IF AN INCORRECT ANSWER IS GIVEN, THE PROGRAM WILL TYPE "?" AND WILL THEN REPEAT THE MESSAGE OF 4.3.3.8

IF A "NO" ANSWER IS GIVEN: JUMP TO SECTION 4.3.3.12  
IF A "YES" ANSWER IS GIVEN:THE NEXT QUESTION IS ASKED

- 4.3.3.10 THE PROGRAM WILL TYPE "LAST DEVICE:RECEIVER CONTROL REGISTER ADDRESS-" AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD
- 4.3.3.11 TYPE IN THE ADDRESS OF THE LAST RECEIVER CONTROL REGISTER ADDRESS OF THE DUV11 TO BE TESTED FOLLOWED BY A <CARRIAGE RETURN>

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL THEN REPEAT THE MESSAGE OF 4.3.3.10  
NOTE:ALL ADDRESSES SHALL BE CONTIGUOUS

- 4.3.3.11.1 IF AN "OUT OF RANGE" ADDRESS IS TYPED IE. MORE THAN 16 (10) DEVICES AWAY (UPWARDS).....THE PROGRAM WILL TYPE "OUT OF RANGE:RETYPE LAST DEVICE RXCSR ADDRESS-" AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD

4.3.3.11.2 TYPE IN THE ADDRESS OF THE LAST RECEIVER CONTROL REGISTER ADDRESS OF THE DUV11 TO BE TESTED FOLLOWED BY A <CARRIAGE RETURN>

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL REPEAT THE MESSAGE OF 4.3.3.11.1

IF A DEVICE ADDRESS LOWER THAN 1ST DEVICE ADDRESS IS TYPED.....  
.....SCHOOLS OUT.....THERE IS NO PROTECTION FOR THIS.  
THE PROGRAM WILL DEFAULT TO TWO DEVICES ACTIVE (UPWARDS FROM 1ST DEVICE ADDRESS).THE SAME APPLIES TO IDENTICAL ADDRESSES TYPED FOR FIRST AND LAST DEVICE.  
OBSERVE LOCATION @ ACTREG: SEE SECTION 7.2

4.3.3.12 THE PROGRAM WILL TYPE '# OF SYNC CHARS SELECTED (1 OR 2)-' AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD. REFER TO MANUAL FOR PROPER SWITCH SETTINGS OF SWITCH E55-4.

4.3.3.13 TYPE IN THE APPROPRIATE ANSWER '1' OR '2' FOLLOWED BY A <CARRIAGE RETURN>. (NOTE: ALL MULTIPLE DEVICES MUST BE THE SAME)

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL REPEAT THE MESSAGE OF 4.3.3.12

4.3.3.14 THE PROGRAM WILL TYPE 'IS SEC XMIT SWITCH E55-2 ON? (Y OR N)-' AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD

4.3.3.15 TYPE IN THE APPROPRIATE ANSWER YES OR NO FOLLOWED BY A <CARRIAGE RETURN>. (NOTE THAT ALL MULTIPLE DEVICES MUST BE THE SAME)

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL REPEAT THE MESSAGE OF 4.3.3.14

4.3.3.16 THE PROGRAM WILL TYPE 'IS SEC REC SWITCH E55-3 ON? (Y OR N)-' AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD

4.3.3.17 TYPE IN THE APPROPRIATE ANSWER YES OR NO FOLLOWED BY A <CARRIAGE RETURN>. (NOTE: ALL MULTIPLE DEVICES MUST BE THE SAME)

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?" AND WILL REPEAT THE MESSAGE OF 4.3.3.16

4.3.3.18 THE PROGRAM WILL TYPE 'IS OPT CLR ENABLE SWITCH E55-1 ON? (Y OR N)-' AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD

4.3.3.19 TYPE IN THE APPROPRIATE ANSWER YES OR NO FOLLOWED BY A <CARRIAGE RETURN>. (NOTE: ALL MULTIPLE DEVICES MUST BE THE SAME)

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?"  
AND WILL REPEAT THE MESSAGE OF 4.3.3.18

4.3.3.20 THE PROGRAM WILL TYPE "ARE YOU RUNNING IN MAINT.  
MODE EXTERNAL ? AND .....DO YOU HAVE THE EXTERNAL MODEM  
BYPASS JUMPER CONNECTOR ON ? (Y OR N)-" AND WAIT FOR AN  
INPUT FROM THE TELETYPE KEYBOARD

4.3.3.21 TYPE IN THE APPROPRIATE ANSWER YES OR NO FOLLOWED BY  
A <CARRIAGE RETURN>. (NOTE: ALL MULTIPLE DEVICES MUST BE THE SAME)

IF AN INCORRECT ANSWER IS TYPED ,THE PROGRAM WILL TYPE "?"  
AND WILL REPEAT THE MESSAGE OF 4.3.3.20

4.3.3.22 THE PROGRAM WILL TYPE 'R' TO INDICATE THAT IT  
HAS STARTED AND WILL COMMENCE TESTING AT TEST 1

4.3.4 PROGRAM RESTART WITH SW01=1  
NOTE: THIS WILL ONLY WORK WHEN A SINGLE DEVICE IS SELECTED  
...IT WILL NOT WORK IF MULTIPLE DEVICES ARE SELECTED

IF MULTIPLE DEVICES WERE PREVIOUSLY SELECTED,LOAD 000200,  
AND SELECT SW00=1 AND ANSWER 'NO' TO THE MULTIPLE DEVICE QUESTION  
SEE 4.3.3

4.3.4.1 SET SW01=1 IN SWITCH REG (LOC. 176)

4.3.4.2 TYPE 200G.

4.3.4.3 PROGRAM WILL START.

4.3.4.4 THE PROGRAM WILL TYPE "TEST PC-" AND WAIT FOR AN INPUT FROM  
THE TELETYPE KEYBOARD

4.3.4.5 TYPE IN THE ADDRESS OF THE TEST AT WHICH THE PROGRAM IS TO  
BE STARTED FOLLOWED BY A <CARRIAGE RETURN>

4.3.4.6 THE PROGRAM WILL TYPE 'R' TO INDICATE THAT IT HAS STARTED  
TESTING AT THE SELECTED TEST

NOTE: CARE MUST BE TAKEN WHEN THIS FEATURE IS USED  
SINCE THERE IS NO PROTECTION AGAINST SELECTING AN ADDRESS  
THAT IS IN THE MIDDLE OF A TEST

4.3.5 PROGRAM RESTART WITH SW14 =1  
NOTE: THIS WILL ONLY WORK WHEN A SINGLE DEVICE IS SELECTED  
SEE NOTE IN 4.3.4 FOR MORE DETAILS

4.3.5.1 SET SW14=1 IN SWITCH REG. (LOC. 176)

4.3.5.2 TYPE 200G.

4.3.5.3 PROGRAM WILL START.

4.3.5.4 THE PROGRAM WILL TYPE 'LOCK ON SELETED TEST ? (Y OR N)-'  
AND WAIT FOR AN INPUT FROM THE TELETYPE KEYBOARD

4.3.5.5 TYPE IN THE APPROPRIATE ANSWER YES OR NO FOLLOWED BY A  
<CARRIAGE RETURN>

IF A NO ANSWER IS GIVEN: THIS LOCK ON TEST WILL BE IGNORED  
AND THE PROGRAM WILL TYPE 'R' TO INDICATE THAT IT HAS STARTED  
TESTING AT TEST 1

4.3.5.6 IF A YES ANSWER WAS GIVEN:THE PROGRAM WILL ACT AS FOLLOWS...  
THE PROGRAM WILL TYPE 'R' TO INDICATE THAT IT HAS STARTED  
TESTING AT TEST 1 AND WILL REMAIN IN TEST 1 UNTIL HALTED  
OR IF ANY KEY IS STRUCK ON THE TELETYPE THE PROGRAM  
WILL FREEZE ON THE NEXT TEST UNTIL A KEY IS STRUCK ON  
THE TELETYPE AND SO FORTH THRU THE PROGRAM. IF SW01 =1 IT  
WILL PERFORM AS IN SECTION 4.3.4 ALLOWING ONE TO FREEZE  
ON A SELECTED TEST RATHER THAN DEFAULTING TO TEST 1

## 5. OPERATING PROCEDURE

### 5.1 OPERATIONAL SWITCH SETTINGS (INTERNAL TO THE CPU, ACCESSED VIA LOC. 176).

SW15 =1 HALT ON ERROR  
SW14 =1 LOOP ON CURRENT TEST  
SW13 =1 INHIBIT ERROR TYPEOUT  
SW11 =1 INHIBIT ITERATIONS  
SW10 =1 ESCAPE TO NEXT TEST ON ERROR  
SW09 =1 LOOP ON ERROR  
SW01 =1 RESTART PROGRAM AT SELECTED TEST  
SW00 =1 RESELECT VECTOR AND CONTROL REGISTER ADDRESSES  
&PARAMETERS AFTER A PROGRAM RESTART  
TO INHIBIT 'END OF PASS' TYPEOUT - TURN TELETYPE OFF

## 6. ERRORS

### 6.1 ERROR HALTS (UNDER LSI ALL HALT ERRORS RETURN CONTROL TO O.D.T.) THERE ARE FOUR DISTINCT ERROR TYPEOUTS

#### 6.1.1 PC+2 = ERROR PC WHERE PC +2 IS THE ADDRESS OF THE CALL TO THE ERROR HANDLER +2

REFER TO THE ABOVE 'HLT' IN DIAGNOSTIC FOR ERROR DESCRIPTION

CHECK ADDRESS @ RXCSR: TO LOCATE THE DEVICE PRESENTLY UNDER  
TEST WHEN RUNNING MULTIPLE DEVICES

#### 6.1.2 PC +2 = REGISTER ERROR PC

REGISTER	EXPECTED	ACTUAL
16XXXX	YYYYYY	ZZZZZZ

WHERE 16XXXX IS THE ADDRESS OF THE FAILING DEVICE REGISTER

WHERE YYYYYY IS THE EXPECTED CONTENTS OF THAT REGISTER



WHERE ZZZZZZ IS THE ACTUAL CONTENTS OF THAT REGISTER

6.1.3 PC +2 = RECEIVER ERROR PC  
REGISTER EXPECTED ACTUAL  
16XXXX YYYYYY ZZZZZZ

WHERE 16XXXX IS THE ADDRESS OF THE FAILING RECEIVER (RXDBUF) REGISTER

WHERE YYYYYY IS THE EXPECTED DATA CONTENTS OF THAT REGISTER

WHERE ZZZZZZ IS THE ACTUAL DATA CONTENTS OF THAT REGISTER

6.1.4 PC +2 = TRANSMITTER ERROR PC  
REGISTER EXPECTED ACTUAL  
16XXXX YYYYYY ZZZZZZ

WHERE 16XXXX IS THE ADDRESS OF THE FAILING TRANSMITTER (TXCSR) REGISTER

WHERE YYYYYY IS THE EXPECTED CONTENTS OF THAT REGISTER

WHERE ZZZZZZ IS THE ACTUAL CONTENTS OF THAT REGISTER

6.1.5 ERROR DESCRIPTIONS  
SEE LISTINGS FOR DETAILS OF ERRORS

6.2 ERROR RECOVERY

6.2.1 SW15 =0  
IF THE PROGRAM IS RUN WITH SW15 =0 ,NO OPERATOR ACTION IS  
REQUIRED TO CONTINUE TESTING

6.2.2 SW15 =1  
IF THE PROGRAM IS RUN WITH SW15 =1 ,TO CONTINUE TESTING  
AFTER THE PROGRAM HAS HALTED ,PRESS THE PROCESSOR  
CONSOLE "CONTINUE SWITCH"

NOTE: THE PC + 2 OF THE "HLT" WILL BE DISPLAYED IN THE DATA LIGHTS

6.2.3 ILLEGAL INTERRUPTS  
IF AN INTERRUPT OCCURS TO A VECTOR ADDRESS NOT SELECTED  
DURING PROGRAM INITIALIZATION, THE PROGRAM WILL HALT IN  
THE TRAPCATCHER. THE ADDRESS AT WHICH THE PROGRAM  
HALTS IS 2 GREATER THAN THE ADDRESS TO WHICH THE INTERRUPT  
OCCURED. THE PROGRAM MUST BE RESTARTED AT 000200 TO  
RECOVER FROM THIS ERROR.

6.2.4 ADDITIONAL TROUBLESHOOTING AIDS ERRCNT: & PASCNT:  
CHECK THESE TWO TAG LOCATIONS FOR TOTAL # OF ERRORS AND PASSES RESPECTIVELY.  
LOADING 000200 AND RESTARTING WILL CLEAR THESE LOCATIONS.

6.3 END OF PASS ROUTINE  
THIS TYPEOUT IS MENTIONED HERE FOR CONVENIENCE  
IT IS IN THE FORM:

END OF PASS TAPE Y

16XXXX = DEVICE

WHERE Y IS THE TAPE LOADED

WHERE 16XXXX IS THE DEVICE'S BASE REGISTER ADDRESS

TO INHIBIT THIS TYPEOUT - TURN TELETYPE OFF

## 7. RESTRICTIONS

### 7.1 MULTIPLE DEVICES

UP TO 16(10) DEVICES MAY BE TESTED. HOWEVER, THEY MUST HAVE CONTIGUOUS ADDRESSES AND VECTORS

NOTE: IF ALL DEVICES UNDER TEST HAVE THE SAME INTERRUPT VECTOR YOU CAN CHANGE "ZERO: ADD #10,BASEIV ;NEXT BLOCK (VECTORS)" TO "ZERO: ADD #0,BASEIV"; THEREBY THE VECTOR ADDRESSES WILL NOT BE UPDATED AFTER EACH PASS.

### 7.2 DISQUALIFYING DEVICES WHEN RUNNING MULTIPLE DEVICES

WHEN RUNNING MULTIPLE DEVICES AN ACTIVE BIT IS SET FOR EACH DEVICE RUNNING UNDER TEST IE. BIT 0 FOR DEVICE 0 ,BIT 15 FOR DEVICE 15 TO DISQUALIFY DEVICES:

7.2.1 IF DEVICE 0 IS TO BE DISQUALIFIED ,SIMPLY RESTART PROGRAM WITH SW00 =1 AND OMIT THE FIRST DEVICE.

7.2.2 IF HOWEVER, DEVICES 1 THRU 15 OR ANY COMBINATION THEREOF ARE TO BE DISQUALIFIED....LOAD THE LOCATION OF ACTREG: OBSERVE THE ACTIVE BITS (ACTIVE =1, NONACTIVE = 0) AND DEPOSIT 0 WHERE THOSE DEVICES ARE TO BE DISQUALIFIED

7.2.2.1 TO RESTART...TYPE 200G...  
THE PROGRAM WILL CONTINUE WITH THE DEVICE IT WAS IN BEFORE HALTING.

7.2.2.2 .....OR .....SET SW00=1 IN SWITCH REG (LOC. 176) AND TYPE 200G....  
ANSWER THE QUESTION :1ST DEVICE : ETC.....  
.....THE PROGRAM WILL CONTINUE WITH DEVICE 0

7.2.2.3 IF ALL DEVICES ARE DISQUALIFIED BY MISTAKE THE PROGRAM WILL TYPEOUT AN ERROR MESSAGE.....TYPE 200G.

### 7.3 CABLE DELAYS

NOTE: EXTERNAL LOOP BACK TESTS ONLY (MODEM CABLE WITH H315 CONNECTOR ON)

7.3.1 TO PROVIDE SUFFICIENT DELAY FOR CLOCK SIGNAL OVER THE CABLE, LOCATION 'HOLD:' MUST BE MODIFIED TO ACCOMODATE FOR FASTER MACHINES. PRESENTLY 'HOLD:' =20 IS SUFFICIENT TIME ON AN 11/21 MACHINE.

BASICALLY DON'T TRY TO EXCEED 10K TO 12K RATE USING THE EIA DRIVERS

7.4 TO USE THE "XOR" TESTER ,THE BRANCH AROUND THE "XOR" CODE MUST BE PATCHED TO A 'NOP'. (SEE LISTINGS FOR DETAILS)

8. DEFAULT PARAMETERS:  
1ST DEVICE: RECEIVER CONTROL REGISTER ADDRESS- RXCSR: 174300  
VECTOR ADDRESS- DURIV: 330  
ARE YOU RUNNING MULTIPLE DEVICES ?- NO MULTD: 0  
LAST DEVICE: RECEIVER CONTROL REGISTER ADDRESS- LASTADD: 0  
# OF SYNC CHARS SELECTED - 2 SYNCNO: 377  
IS SEC XMIT SWITCH E55-2 ON?- YES SEXMIT: 377  
IS SEC REC SWITCH E55-3 ON?- YES SEREC: 377  
IS OPT CLR ENABLE SWITCH E55-1 ON?- YES OPTCLR: 377  
DO YOU HAVE THE EXTERNAL MODEM BYPASS JUMPER  
CONNECTOR ON (H315)- YES JMRBY: 377

9. PROGRAM DESCRIPTION

9.1 THIS PROGRAM PERFORMS THE OFFLINE RECEIVER SECTION TESTING  
OF THE DEVICE  
SEE LISTING FOR DETAILS

10. FLOW CHARTS: RECEIVER FLOW,TRANSMITTER FLOW,TRANSMITTER & RECEIVER FLOW

11. REVISION HISTORY  
DZDURB1 WAS MODIFIED TO WORK IN SBC 11/21 PROCESSOR  
PRIORITY 340 WAS CHANGED TO 300 WHEREVER ENCOUNTERED  
DEFAULT CSR AND VECTOR ADDRESSES WERE CHANGED  
.INIT CALL INCLUDED FOR INITIALIZATION  
DIAGNOSTIC WAS RENAMED TO CNDURA0.

12. LISTINGS

\* .REM \*  
\* .REM \*

6488  
 6585  
 6586  
 6587  
 6598  
 6612  
 6613  
 6614  
 6675  
 6676  
 6881  
 7009  
 7010

.SBTTL APT COMMUNICATIONS ROUTINE

```

(1)
(2)
(1) 000000 112767 000001 000236 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
(1) 000006 112767 000001 000226 $ATY3:  MOVB  #1,$MFLG      ;;TO TYPE A MESSAGE
(1) 000014 000403                BR      $ATYC
(1) 000016 112767 000001 000220 $ATY4:  MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
(1) 000024                $ATYC:
(3) 000024 010046                MOV    R0,-(SP)      ;;PUSH R0 ON STACK
(3) 000026 010146                MOV    R1,-(SP)      ;;PUSH R1 ON STACK
(1) 000030 105767 000206                TSTB  $MFLG          ;;SHOULD TYPE A MESSAGE?
(1) 000034 001450                BEQ   5$              ;;IF NOT: BR
(1) 000036 122767 000001 001502                CMPB  #APTENV,$ENV   ;;OPERATING UNDER APT?
(1) 000044 001031                BNE   3$              ;;IF NOT: BR
(1) 000046 132767 000100 001473                BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 000054 001425                BEQ   3$              ;;IF NOT: BR
(1) 000056 017600 000004                MOV    @4(SP),R0     ;;GET MESSAGE ADDR.
(1) 000062 062766 000002 000004                ADD    #2,4(SP)      ;;BUMP RETURN ADDR.
(1) 000070 005767 001432                1$:  TST    $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
(1) 000074 001375                BNE   1$              ;;IF NOT: WAIT
(1) 000076 010067 001440                MOV    R0,$MSGAD
(1)                ;;PUT ADDR IN MAILBOX
(1) 000102 105720                2$:  TSTB  (R0)+         ;;FIND END OF MESSAGE
(1) 000104 001376                BNE   2$
(1) 000106 166700 001430                SUB    $MSGAD,R0     ;;SUB START OF MESSAGE
(1) 000112 006200                ASR   R0              ;;GET MESSAGE LNGTH IN WORDS
(1) 000114 010067 001424                MOV    R0,$MSGLGT    ;;PUT LENGTH IN MAILBOX
(1) 000120 012767 000004 001400                MOV    #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
(1) 000126 000413                BR    5$
(1) 000130 017667 000004 000016 3$:  MOV    @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
(1) 000136 062766 000002 000004                ADD    #2,4(SP)      ;;BUMP RETURN ADDRESS
(3) 000144 016746 177626                MOV    177776,-(SP)  ;;PUSH 177776 ON STACK
(1) 000150 004767 012566                JSR   PC,$TYPE       ;;CALL TYPE MACRO
(1) 000154 000000                4$:  .WORD  0
(1) 000156                5$:
(1) 000156 105767 000062                10$: TSTB  $FFLG          ;;SHOULD REPORT FATAL ERROR?
(1) 000162 001416                BEQ   12$            ;;IF NOT: BR
(1) 000164 005767 001356                TST   $ENV           ;;RUNNING UNDER APT?
(1) 000170 001413                BEQ   12$            ;;IF NOT: BR
(1) 000172 005767 001330                11$: TST   $MSGTYPE       ;;FINISHED LAST MESSAGE?
(1) 000176 001375                BNE   11$           ;;IF NOT: WAIT
(1) 000200 017667 000004 001322                MOV    @4(SP),$FATAL ;;GET ERROR #
(1) 000206 062766 000002 000004                ADD    #2,4(SP)      ;;BUMP RETURN ADDR.
(1) 000214 005267 001306                INC   $MSGTYPE       ;;TELL APT TO TAKE ERROR
  
```

```
(1) 000220 105067 000020      12$: CLRB $FFLG      ;;CLEAR FATAL FLAG
(1) 000224 105067 000013      CLRB $LFLG      ;;CLEAR LOG FLAG
(1) 000230 105067 000006      CLRB $MFLG      ;;CLEAR MESSAGE FLAG
(3) 000234 012601      MOV (SP)+,R1    ;;POP STACK INTO R1
(3) 000236 012600      MOV (SP)+,R0    ;;POP STACK INTO R0
(1) 000240 000207      RTS PC          ;;RETURN
(1) 000242 000      $MFLG: .BYTE 0  ;;MESSG. FLAG
(1) 000243 000      $LFLG: .BYTE 0
(1)      $FFLG: .BYTE 0 ;;LOG FLAG
(1) 000244 000      ;;FATAL FLAG
(1)      .EVEN
(1) 000200      APTSIZE=200
(1) 000001      APTENV=001
(1) 000100      APTSPool=100
(1) 000040      APTCSUP=040
7249 000001      $TN=1
7375
7379
7414
7431
7444
7456
7469
```

- 7492
- 7558
- 7564
- 7700
- 7728
- 7761
- 7802
- 7833
- 7884
- 7932
- 7985
- 8000
- 8012
- 8114



```

(2)      040000      SW14= 40000
(2)      020000      SW13= 20000
(2)      010000      SW12= 10000
(2)      004000      SW11= 4000
(2)      002000      SW10= 2000
(2)      001000      SW09= 1000
(2)      000400      SW08= 400
(2)      000200      SW07= 200
(2)      000100      SW06= 100
(2)      000040      SW05= 40
(2)      000020      SW04= 20
(2)      000010      SW03= 10
(2)      000004      SW02= 4
(2)      000002      SW01= 2
(2)      000001      SW00= 1
(2)      .EQUIV      SW09,SW9
(2)      .EQUIV      SW08,SW8
(2)      .EQUIV      SW07,SW7
(2)      .EQUIV      SW06,SW6
(2)      .EQUIV      SW05,SW5
(2)      .EQUIV      SW04,SW4
(2)      .EQUIV      SW03,SW3
(2)      .EQUIV      SW02,SW2
(2)      .EQUIV      SW01,SW1
(2)      .EQUIV      SW00,SW0
(2)
(2)      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(2)      100000      BIT15= 100000
(2)      040000      BIT14= 40000
(2)      020000      BIT13= 20000
(2)      010000      BIT12= 10000
(2)      004000      BIT11= 4000
(2)      002000      BIT10= 2000
(2)      001000      BIT09= 1000
(2)      000400      BIT08= 400
(2)      000200      BIT07= 200
(2)      000100      BIT06= 100
(2)      000040      BIT05= 40
(2)      000020      BIT04= 20
(2)      000010      BIT03= 10
(2)      000004      BIT02= 4
(2)      000002      BIT01= 2
(2)      000001      BIT00= 1
(2)      .EQUIV      BIT09,BIT9
(2)      .EQUIV      BIT08,BIT8
(2)      .EQUIV      BIT07,BIT7
(2)      .EQUIV      BIT06,BIT6
(2)      .EQUIV      BIT05,BIT5
(2)      .EQUIV      BIT04,BIT4
(2)      .EQUIV      BIT03,BIT3
(2)      .EQUIV      BIT02,BIT2
(2)      .EQUIV      BIT01,BIT1
(2)      .EQUIV      BIT00,BIT0
(2)
(2)      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(2)      000004      ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
  
```



(2)	000010	RESVEC= 10	::RESERVED AND ILLEGAL INSTRUCTIONS
(2)	000014	TBITVEC=14	::"T" BIT
(2)	000014	TRTVEC= 14	::TRACE TRAP
(2)	000014	BPTVEC= 14	::BREAKPOINT TRAP (BPT)
(2)	000020	IOTVEC= 20	::INPUT/OUTPUT TRAP (IOT) **SCOPE**
(2)	000024	PWRVEC= 24	::POWER FAIL
(2)	000030	EMTVEC= 30	::EMULATOR TRAP (EMT) **ERROR**
(2)	000034	TRAPVEC=34	::"TRAP" TRAP
(2)	000060	TKVEC= 60	::TTY KEYBOARD VECTOR
(2)	000064	TPVEC= 64	::TTY PRINTER VECTOR
(2)		::***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED	
(2)	000100	LKVEC= 100	::LINE CLOCK VECTOR
(2)	000140	BRKVEC= 140	::BREAK VECTOR
(2)	000240	PIRQVEC=240	::PROGRAM INTERRUPT REQUEST VECTOR

```
(2) ;STANDARD INTERRUPT VECTORS
(2)
(2)
(2) 000174 000174 . =174
(2) 000174 000000 DISPREG:0
(2) 000176 000000 SWREG:0
(2) 000200 000200 . =200
(2) 000200 000167 001746 JMP .START ;GO TO START OF PROGRAM
(2)
(2)
(2) 001100 001100 . =1100
(2) 001100 000000 .WORD 0
(2) 001102 177570 LIGHTS:177570
(2)
(2)
(2) ;PROGRAM CONTROL PARAMETERS
(2) 001104 000000 RETURN: 0
(2) 001106 000000 NEXT: 0 ;ADDRESS OF NEXT TEST TO BE EXECUTED
(2) 001110 000000 LOCK: 0 ;ADDRESS FOR LOCK ON CURRENT DATA
(2) 001112 000000 PASCNT: 0 ;ADDRESS CONTAINING PASS COUNT
(2) 001114 000000 ERRCNT: 0 ;ERROR COUNT
(2) 001116 000000 SAVSP: 0 ;STACK POINTER STORAGE
(2)
(2) ;PROGRAM VARIABLES
(2) 001120 000020 HOLD: 20 ;TEMPORARY STORAGE=DELAY TIME FOR CABLES
(2) 001122 000000 SHIFT: 0 ;TEMPORARY STORAGE= # OF SHIFTS PER CHAR
(2) 001124 000000 COUNT: 0 ;TEMPORARY STORAGE= # OF TIMES A CHAR WILL BE SENT
(2) 001126 000000 SAVPC: 0 ;PROGRAM COUNTER STORAGE
(2) 001130 000000 HLD0: 0
(2) 001132 000000 HLD1: 0
(2) 001134 000000 HLD2: 0
(2) 001136 000000 HLD3: 0
(2) 001140 000000 HLD4: 0
(2) 001142 000000 HLD5: 0
(2) 001144 000000 HLD6: 0
(2)
```

```
(2)                                     ;PROGRAM CONVERSATIONAL PARAMETERS
(2) 001146      377      SYNCNO: .BYTE 377      ;# OF SYNC CHARS REQ'D FOR SYNC'ZATION
(2) 001147      377      SEXMIT: .BYTE 377      ;SEC XMIT JUMPER 'IN'
(2) 001150      377      SEREC:  .BYTE 377      ;SEC REC JUMPER 'IN'
(2) 001151      377      OPTCLR: .BYTE 377      ;OPTIONAL JUMPER CLR 'IN'
(2) 001152      000      MULTD:  .BYTE 0        ;NO MULTIPLE DEVICE FLAG
(2) 001153      377      JMRBY:  .BYTE 377      ;EXTERNAL MODEM BYPASS JUMPER 'IN'
(2)                                     .EVEN
(2)                                     ;PROGRAM MULTIPLE DEVICE PARAMETERS
(2) 001154      000000   BASEADD: 0          ;PROG CONTROLLED 1ST DEVICE ADDR
(2) 001156      000000   KEEPADD: 0          ;SAVED 1ST DEVICE ADDR
(2) 001160      000000   LASTADD: 0          ;LAST DEVICE RXCSR ADDR
(2) 001162      000000   BASEIV:  0          ;PROG CONTROLLED IV
(2) 001164      000000   KEEPIV:  0          ;SAVED INTR VECTOR
(2) 001166      000000   ACTREG:  0          ;ACTIVE REGISTER ...MODIFY THIS
(2)                                     ;LOCATION TO DISQUALIFY OR QUALIFY
(2)                                     ;DEVICES (1= RUN,,,0= DON'T RUN)
(2) 001170      000000   ROTADD:  0          ;ROTATING POINTER FOR ACTREG..POINTS
(2)                                     ;TO DEVICE PRESENTLY UNDER TEST WHEN RUNNING MULTIPLE DEVICES
(2)                                     ;PROGRAM CONTROL FLAGS
(2) 001172      000      INIFLG: .BYTE 0        ;PROGRAM INITIALIZATION FLAG
(2) 001173      000      STFLG:  .BYTE 0        ;TEST START FLAG
(2) 001174      000      LOKFLG: .BYTE 0        ;LOCK ON CURRENT TEST FLAG
(2)                                     .EVEN
(1)                                     .=1400
(2)
(2)
```

```
(2)
(2)
(2)
(2) ; INSTRUCTION DEFINITIONS
(2)
(2) 005746 PUSH1SP=5746 ; DECREMENT PROCESSOR STACK 1 WORD =TST -(SP)
(2) 005726 POP1SP=5726 ; INCREMENT PROCESSOR STACK 1 WORD =TST (SP)+
(2) 010046 PUSHRO=10046 ; SAVE R0 ON STACK =MOV R0,-(SP)
(2) 012600 POPRO=12600 ; RESTORE R0 FROM STACK =MOV (SP)+,R0
(2) 024646 PUSH2SP=24646 ; DECREMENT STACK TWICE =CMP -(SP),-(SP)
(2) 022626 POP2SP=22626 ; INCREMENT STACK TWICE =CMP (SP)+,(SP)+
(2) ; REGISTER DEFINITIONS
(2) ; RXCSR BIT DEFINITIONS
(2) 100000 DSC=BIT15 ; DATA SET CHANGE
(2) 040000 RING=BIT14 ; RING
(2) 020000 CTS=BIT13 ; CLR TO SEND
(2) 010000 CARDET=BIT12 ; CARRIER DETECT
(2) 004000 REACT=BIT11 ; REC ACTIVE
(2) 002000 SRD=BIT10 ; SEC REC DATA
(2) 001000 DSR=BIT9 ; DATA SET RDY
(2) 000400 STPSYN=BIT8 ; STRIP SYNC
(2) 000200 RXDONE=BIT7 ; REC DONE
(2) 000100 RINTEN=BIT6 ; REC INTR ENABLE
(2) 000040 DSINTE=BIT5 ; DSC INTR ENABLE
(2) 000020 SYNSCH=BIT4 ; SYNC SEARCH
(2) 000010 STD=BIT3 ; SEC XMIT DATA
(2) 000004 RTS=BIT2 ; REQ TO SEND
(2) 000002 DTR=BIT1 ; DATA TERM RDY
(2) 000001 VOID=BIT0
(2) ; RXDBUF BIT DEFINITIONS
(2) 100000 RXERR=BIT15 ; REC ERROR
(2) 040000 OVRRUN=BIT14 ; OVERRUN
(2) 020000 FRMERR=BIT13 ; FRAME ERROR
(2) 010000 PARER=BIT12 ; PARITY ERROR
(2) ; PARCSR BIT DEFINITIONS
(2) 001000 PAREN=BIT9 ; PARITY ENABLE
(2) 000400 EVPAR=BIT8 ; EVEN PARITY SENSE
(2) ; PARCSR WRD DEFINITIONS
(2) 030000 SYNINT=30000 ; SYNC EXTERNAL MODE
(2) 020000 SYNEXT=20000 ; SYNC INTERNAL MODE
(2) 000000 ISYMOD=0 ; ISOC MODE
(2) 000000 FIVE=0 ; WORD LENGTH 5 BITS
(2) 002000 SIX=2000 ; WORD LENGTH 6 BITS
(2) 004000 SEVEN=4000 ; WORD LENGTH 7 BITS
(2) 006000 EIGHT=6000 ; WORD LENGTH 8 BITS
(2) 000000 NOPAR=0 ; NO PARITY
(2) 001000 ODDPAR=1000 ; ODD PARITY
(2) 001400 EVEPAR=1400 ; EVEN PARITY
(2) ; TXCSR BIT DEFINITIONS
(2) 100000 DNA=BIT15 ; DATA NOT AVAILABLE
(2) 040000 MTDATA=BIT14 ; MAINT DATA
(2) 020000 CLK=BIT13 ; CLK
(2) 002000 BITW=BIT10 ; BIT WINDOW
(2) 000400 MRESET=BIT8 ; MASTER RESET
(2) 000200 TXDONE=BIT7 ; XMIT DONE
(2) 000100 TXINTE=BIT6 ; XMIT INTR ENABLE
```

(2)	000040	DNAINTE=BIT5	;DNA INTR ENAB
(2)	000020	SEND=BIT4	;SEND
(2)	000010	HDXEN=BIT3	;HDX/FDX
(2)	000001	BREAK=BIT0	;BREAK
(2)		;TXCSR WRD DEFINITIONS	
(2)	000000	USER=0	;USER MODE
(2)	004000	MINT=4000	;MAINT INT MODE
(2)	010000	MEXT=10000	;MAINT EXT MODE
(2)	014000	SYSTST=14000	;SYSTEM TEST MODE

```

(2)          .SBTTL COMMON TAGS
(2)
(3)          ::*****
(2)          :*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(2)          :*USED IN THE PROGRAM.
(2)
(2)          001400          .=.          ;;START OF COMMON TAGS
(2) 001400 000000          $CMTAG: .WORD 0          ;;CONTAINS THE TEST NUMBER
(2) 001402 000          $TSTNM: .BYTE 0          ;;CONTAINS ERROR FLAG
(2) 001403 000          $ERFLG: .BYTE 0          ;;CONTAINS SUBTEST ITERATION COUNT
(2) 001404 000000          $ICNT: .WORD 0          ;;CONTAINS SCOPE LOOP ADDRESS
(2) 001406 000000          $LPADR: .WORD 0          ;;CONTAINS SCOPE RETURN FOR ERRORS
(2) 001410 000000          $LPERR: .WORD 0          ;;CONTAINS TOTAL ERRORS DETECTED
(2) 001412 000000          $ERTTL: .WORD 0          ;;CONTAINS ITEM CONTROL BYTE
(2) 001414 000          $ITEMB: .BYTE 0          ;;CONTAINS MAX. ERRORS PER TEST
(2) 001415 001          $ERMAX: .BYTE 1          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
(2) 001416 000000          $ERRPC: .WORD 0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
(2) 001420 000000          $GDADR: .WORD 0          ;;CONTAINS ADDRESS OF 'BAD' DATA
(2) 001422 000000          $BDADR: .WORD 0          ;;CONTAINS 'GOOD' DATA
(2) 001424 000000          $GDDAT: .WORD 0          ;;CONTAINS 'BAD' DATA
(2) 001426 000000          $BDDAT: .WORD 0          ;;RESERVED--NOT TO BE USED
(2) 001430 000000          .WORD 0
(2) 001432 000000          .WORD 0
(2) 001434 000          $AUTOB: .BYTE 0          ;;AUTOMATIC MODE INDICATOR
(2) 001435 000          $INTAG: .BYTE 0          ;;INTERRUPT MODE INDICATOR
(2) 001436 000000          .WORD 0
(2) 001440 177570          $SWR: .WORD DSWR          ;;ADDRESS OF SWITCH REGISTER
(2) 001442 177570          $DISPLAY: .WORD DDISP          ;;ADDRESS OF DISPLAY REGISTER
(2) 001444 177560          $TKS: 177560          ;;TTY KBD STATUS
(2) 001446 177562          $TKB: 177562          ;;TTY KBD BUFFER
(2) 001450 177564          $TPS: 177564          ;;TTY PRINTER STATUS REG. ADDRESS
(2) 001452 177566          $TPB: 177566          ;;TTY PRINTER BUFFER REG. ADDRESS
(2) 001454 000          $NULL: .BYTE 0          ;;CONTAINS NULL CHARACTER FOR FILLS
(2) 001455 002          $FILLS: .BYTE 2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
(2)          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
(2) 001456 012          $FILLC: .BYTE 12          ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
(2) 001457 000          $TPFLG: .BYTE 0          ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(2) 001460 000000          $REGAD: .WORD 0          ;;CONTAINS THE ADDRESS FROM
(2)          ;;WHICH ($REGO) WAS OBTAINED
(4) 001462 000000          $REG0: .WORD 0          ;;CONTAINS (($REGAD)+0)
(4) 001464 000000          $REG1: .WORD 0          ;;CONTAINS (($REGAD)+2)
(4) 001466 000000          $REG2: .WORD 0          ;;CONTAINS (($REGAD)+4)
(4) 001470 000000          $REG3: .WORD 0          ;;CONTAINS (($REGAD)+6)
(4) 001472 000000          $REG4: .WORD 0          ;;CONTAINS (($REGAD)+10)
(4) 001474 000000          $REG5: .WORD 0          ;;CONTAINS (($REGAD)+12)
(4) 001476 000000          $TMP0: .WORD 0          ;;USER DEFINED
(4) 001500 000000          $TMP1: .WORD 0          ;;USER DEFINED
(4) 001502 000000          $TMP2: .WORD 0          ;;USER DEFINED
(4) 001504 000000          $TMP3: .WORD 0          ;;USER DEFINED
(4) 001506 000000          $TMP4: .WORD 0          ;;USER DEFINED
(4) 001510 000000          $TMP5: .WORD 0          ;;USER DEFINED
(2) 001512 000000          $TIMES: 0          ;;MAX. NUMBER OF ITERATIONS
(2) 001514 000000          $ESCAPE: 0          ;;ESCAPE ON ERROR ADDRESS
(2) 001516 177607 000377          $BELL: .ASCII <207><377><377>          ;;CODE FOR BELL
(2) 001522 077          $QUES: .ASCII /?/          ;;QUESTION MARK
  
```

```

(2) 001523 015 $CRLF: .ASCII <15> ::CARRIAGE RETURN
(2) 001524 000012 $LF: .ASCIIZ <12> ::LINE FEED
(3) ::*****
(3) .SBTTL APT MAILBOX-ETABLE
(4) ::*****
(3) .EVEN
(3) 001526 $MAIL: ::APT MAILBOX
(3) 001526 000000 $MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
(3) 001530 000000 $FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
(3) 001532 000000 $TESTN: .WORD ATESTN ::TEST NUMBER
(3) 001534 000000 $PASS: .WORD APASS ::PASS COUNT
(3) 001536 000000 $DEVCT: .WORD ADEVCT ::DEVICE COUNT
(3) 001540 000000 $UNIT: .WORD AUNIT ::I/O UNIT NUMBER
(3) 001542 000000 $MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
(3) 001544 000000 $MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
(3) 001546 $ETABLE: ::APT ENVIRONMENT TABLE
(3) 001546 000 $ENV: .BYTE AENV ::ENVIRONMENT BYTE
(3) 001547 000 $ENVM: .BYTE AENVM
(3) ::ENVIRONMENT MODE BITS
(3) 001550 000000 $SWREG: .WORD ASWREG ::APT SWITCH REGISTER
(3) 001552 000000 $USWR: .WORD AUSWR ::USER SWITCHES
(3) 001554 000000 $CPUOP: .WORD ACPUOP ::CPU TYPE,OPTIONS
(3) * BIT 15-11=CPU TYPE
(3) * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(3) * 11/70=06,PDQ=07,Q=10
(3) * BIT 10=REAL TIME CLOCK
(3) * BIT 9=FLOATING POINT PROCESSOR
(3) * BIT 8=MEMORY MANAGEMENT
(3) 001556 000 $MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
(3) 001557 000 $MTYP1: .BYTE AMTYP1 ::MEM. TYPE,BLK#1
(3) * MEM.TYPE BYTE -- (HIGH BYTE)
(3) * 900 NSEC CORE=001
(3) * 300 NSEC BIPOLAR=002
(3) * 500 NSEC MOS=003
(3) 001560 000000 $MADR1: .WORD AMADR1 ::HIGH ADDRESS,BLK#1
(3) * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(3) 001562 000 $MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS,M.S. BYTE
(3) 001563 000 $MTYP2: .BYTE AMTYP2 ::MEM.TYPE,BLK#2
(3) 001564 000000 $MADR2: .WORD AMADR2 ::MEM.LAST ADDRESS,BLK#2
(3) 001566 000 $MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS,M.S.BYTE
(3) 001567 000 $MTYP3: .BYTE AMTYP3 ::MEM.TYPE,BLK#3
(3) 001570 000000 $MADR3: .WORD AMADR3 ::MEM.LAST ADDRESS,BLK#3
(3) 001572 000 $MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS,M.S.BYTE
(3) 001573 000 $MTYP4: .BYTE AMTYP4 ::MEM.TYPE,BLK#4
(3) 001574 000000 $MADR4: .WORD AMADR4 ::MEM.LAST ADDRESS,BLK#4
(3) 001576 000000 $VECT1: .WORD AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
(3) 001600 000000 $VECT2: .WORD AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
(3) 001602 000000 $BASE: .WORD ABASE
(3) ::BASE ADDRESS OF EQUIPMENT UNDER TEST
(3) 001604 000000 $DEVN: .WORD ADEVN ::DEVICE MAP
(3) 001606 000000 $CDW1: .WORD ACDW1 ::CONTROLLER DESCRIPTION WORD#1
(3) 001610 000000 $CDW2: .WORD ACDW2 ::CONTROLLER DESCRIPTION WORD#2
(3) 001612 000000 $DDW0: .WORD ADDW0 ::DEVICE DESCRIPTOR WORD#0
(3) 001614 000000 $DDW1: .WORD ADDW1 ::DEVICE DESCRIPTOR WORD#1
(3) 001616 000000 $DDW2: .WORD ADDW2 ::DEVICE DESCRIPTOR WORD#2

```







CNDUR-A MACY11 30(1046) 14-DEC-82 09:58 PAGE 61-11  
CNDURA.M11 29-OCT-82 16:31 APT MAILBOX-ETABLE

SEQ 0025

(4)	000040	DNAINTE=BIT5	:DNA INTR ENAB
(4)	000020	SEND=BIT4	:SEND
(4)	000010	HDXEN=BIT3	:HDX/FDX
(4)	000001	BREAK=BIT0	:BREAK
(4)		:TXCSH WRD DEFINITIONS	
(4)	000000	USER=0	:USER MODE
(4)	004000	MINT=4000	:MAINT INT MODE
(4)	010000	MEXT=10000	:MAINT EXT MODE
(4)	014000	SYSTST=14000	:SYSTEM TEST MODE

```

(2) .SBTTL ERROR POINTER TABLE
(2)
(2) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(2) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(2) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(2) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(2) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(2)
(2) ;* EM ;:POINTS TO THE ERROR MESSAGE
(2) ;* DH ;:POINTS TO THE DATA HEADER
(2) ;* DT ;:POINTS TO THE DATA
(2) ;* DF ;:POINTS TO THE DATA FORMAT
(2)
(2) $ERRTB:
(2) ;ERROR TABLE
(1) 001652
(1) 001652 001762 EM1 ;ERROR 1 REGISTER ERROR
(1) 001654 002067 DH1
(1) 001656 002116 DT1
(1) 001660 002132 DF1
(1) 001662 002022 EM2 ;ERROR 2 RECEIVER ERROR
(1) 001664 002067 DH1
(1) 001666 002116 DT1
(1) 001670 002132 DF1
(1) 001672 002043 EM3 ;ERROR 3 TRANSMITTER ERROR
(1) 001674 002067 DH1
(1) 001676 002116 DT1
(1) 001700 002132 DF1
(1) 001702 001746 EM4 ;ERROR 4 BIT ERROR (GENERAL)
(1) 001704 000000 0
(1) 001706 002126 DT4
(1) 001710 002132 DF1
(1)
(1) ;DEFAULT DU ADDRESSES
(1) 001712 174300 RXCSR: 174300
(1) 001714 174301 HRXCSR: 174301
(1) 001716 174302 RXDBUF: 174302
(1) 001720 174303 HRXDBUF: 174303
(1) 001722 174302 PARCSR: 174302
(1) 001724 174303 HPARCSR: 174303
(1) 001726 174304 TXCSR: 174304
(1) 001730 174305 HTXCSR: 174305
(1) 001732 174306 TXDBUF: 174306
(1) 001734 174307 HTXDBUF: 174307
(1)
(1) ;DEFAULT DU VECTORS
(1) 001736 000330 DURIV: 330 ;REC INTR VECTOR
(1) 001740 000332 DURIS: 332 ;REC INTR STATUS
(1) 001742 000334 DUTIV: 334 ;XMIT INTR VECTOR
(1) 001744 000336 DUTIS: 336 ;XMIT INTR STATUS
(1)
(1) ;ERROR MESSAGES
(1) 001746 020040 051105 047522 EM4: .ASCIZ / ERROR PC /
(1) 001754 020122 041520 000040
(1) 001762 020040 047503 050115 EM1: .ASCIZ / COMPARISON ERROR ON REGISTERS/
(1) 001770 051101 051511 047117
(1) 001776 042440 051122 051117
(1) 002004 047440 020116 042522
  
```

```

(1) 002012 044507 052123 051105
(1) 002020 000123
(1) 002022 020040 042522 042503 EM2: .ASCIZ / RECEIVER ERROR/
(1) 002030 053111 051105 042440
(1) 002036 051122 051117 000
(1) 002043 040 052040 040522 EM3: .ASCIZ / TRANSMITTER ERROR/
(1) 002050 051516 044515 052124
(1) 002056 051105 042440 051122
(1) 002064 051117 000
(1) 002067 105 051122 041520 DH1: ;DATA HEADERS FOR ERROR MESSAGES
(1) 002074 020040 040527 052116 ;ASCIZ /ERRPC WANTED ACTUAL/
(1) 002102 042105 020040 041501
(1) 002110 052524 046101 000
(1) 002116 001416 001130 001132 DT1: ;DATA TABLES FOR ERROR MESSAGES
(1) 002124 000000 ;WORD $ERRPC,HLDO,HLDI,0
(1) 002126 001416 000000 DT4: .WORD $ERRPC,0
(1) 002132 000 000 000 DF1: .BYTE 0,0,0,0
(1) 002135 000
(1) .EVEN
(2) .SBTTL ACT11 HOOKS
(2)
(3) ;*****
(2) ;HOOKS REQUIRED BY ACT11
(2) 002136 $SVPC=. ;SAVE PC
(2) 000046 =46
(2) 000046 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(2) 000052 =52
(2) 000052 .WORD 0 ;:2)SET LOC.52 TO ZERO
(2) 002136 =$SVPC ;: RESTORE PC
(2) .SBTTL APT PARAMETER BLOCK
(3) ;*****
(2) ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(3) ;*****
(2) 002136 $.X=. ;:SAVE CURRENT LOCATION
(2) 000024 =24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
(2) 000024 200 ;:FOR APT START UP
(2) 000044 =44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
(2) 000044 $APTHDR ;:POINT TO APT HEADER BLOCK
(2) 002136 =$.X ;:RESET LOCATION COUNTER
(3) ;*****
(2) ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(2) ;INTERFACE SPEC.
(2)
(2) $APTHD:
(2) 002136 000000 $HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(2) 002140 001526 $MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
(2) 002142 000010 $TSTM: .WORD 10 ;:RUN TIM OF LONGEST TEST
(2) 002144 000010 $PASTM: .WORD 10 ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(2) 002146 000000 $UNITM: .WORD ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(2) 002150 000052 .WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)

```

```

(1)
(2)
(2)          ;PROGRAM INITIALIZATION
(2)          ;LOCK OUT INTERRUPTS
(2)          ;SET UP PROCESSOR STACK
(2)          ;SET UP POWER FAIL VECTOR
(2)          ;CLEAR PROGRAM CONTROL FLAGS AND COUNTS
(2)          ;TYPE TITLE MESSAGE
(2)
(2) 002152   .START:
(3)          .SBTTL INITIALIZE THE COMMON TAGS
(3)          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(3) 002152   012706 001400   MOV    #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(3) 002156   005026          CLR    (R6)+          ;;CLEAR MEMORY LOCATION
(3) 002160   022706 001440   CMP    #SWR,R6      ;;DONE?
(3) 002164   001374          BNE    -6            ;;LOOP BACK IF NO
(3) 002166   012706 001100   MOV    ##STACK,SP   ;;SETUP THE STACK POINTER
(3)          ;;INITIALIZE A FEW VECTORS
(3) 002172   012737 016170 000020   MOV    #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(3) 002200   012737 000300 000022   MOV    #PR6,@IOTVEC+2 ;;LEVEL 6
(3) 002206   012737 014060 000030   MOV    #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(3) 002214   012737 000300 000032   MOV    #PR6,@EMTVEC+2 ;;LEVEL 6
(3)          ;;BIT02
(3) 002222   012737 016524 000034   MOV    #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(3) 002230   012737 000300 000036   MOV    #PR6,@TRAPVEC+2;LEVEL 6
(3) 002236   012737 014662 000024   MOV    #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
(3) 002244   012737 000300 000026   MOV    #PR6,@PWRVEC+2 ;;LEVEL 6
(3) 002252   005067 177234          CLR    TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
(3) 002256   005067 177232          CLR    $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(3) 002262   112767 000001 177125   MOV    #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
(3) 002270   012767 002270 177110   MOV    #,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(3) 002276   012767 002276 177104   MOV    #,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
(4)          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(4)          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(4) 002304   013746 000004          MOV    @ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(4) 002310   012737 002344 000004   MOV    #64$,@ERRVEC  ;;SET UP ERROR VECTOR
(4) 002316   012767 177570 177114   MOV    #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
(4) 002324   012767 177570 177110   MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(4) 002332   022777 177777 177100   CMP    #-1,@SWR     ;;TRY TO REFERENCE HARDWARE SWR
(4) 002340   001012          BNE    66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(4)          ;;AND THE HARDWARE SWR IS NOT = -1
(4) 002342   000403          BR     65$          ;;BRANCH IF NO TIMEOUT
(4) 002344   012716 002352   64$: MOV    #65$,(SP)    ;;SET UP FOR TRAP RETURN
(4) 002350   000002          RTI
(4) 002352   012767 000176 177060   65$: MOV    #SWREG,SWR   ;;POINT TO SOFTWARE SWR
(4) 002360   012767 000174 177054   MOV    #DISPREG,DISPLAY
(4) 002366   012637 000004   66$: MOV    (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
(3)
(4) 002372   005067 177136          CLR    $PASS        ;;CLEAR PASS COUNT
(4) 002376   132767 000200 177143   BITB  #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(4) 002404   001403          BEQ   67$          ;;YES,USE NON-APT SWITCH
(4) 002406   012767 001550 177024   MOV    #SSWREG,SWR  ;;NO,USE APT SWITCH REGISTER
(4) 002414          67$:
(2) 002414   012706 001100          MOV    #STACK,SP    ;;SET STACK
(2) 002420   106427 000300          MTPS  #300          ;;LOCK INTERRUPTS
(2) 002424   012737 014662 000024   MOV    #.PFAIL,@#24 ;;SET UP POWER FAIL VECTOR
  
```

```

(2) 002432 105067 176535      CLR      STFLG      ;CLEAR START FLAG
(2) 002436 005067 176450      CLR      PASCNT     ;CLEAR PASS COUNT
(2) 002442 105067 176735      CLR      $ERFLG     ;CLEAR ERROR FLAG
(2) 002446 005067 176740      CLR      $ERTTL     ;CLEAR ERROR COUNT
(2) 002452 005067 176740      CLR      $ERRPC     ;CLEAR LAST ERROR POINTER
(2) 002456 012767 000001 176716  MOV      #1,$STSTNM ;SET UP FOR TEST 1
(2) 002464 012767 002152 176412  MOV      #.START,RETURN ;SET UP FOR POWER FAIL BEFORE
(2)                                     ;TESTING STARTS
(2) 002472 013746 000006      MOV      @#6,-(SP)
(2) 002476 013746 000004      MOV      @#4,-(SP)
(2) 002502 012737 002516 000004  MOV      #1$,@#4
(2) 002510 005777 176724      TST      @SWR
(2) 002514 000407                BR      2$
(2) 002516 012767 000176 176714 1$:  MOV      #SWREG,SWR
(2) 002524 012767 000174 176710  MOV      #DISPREG,DISPLAY
(2) 002532 022626                CMP      (SP)+,(SP)+
(2) 002534 012637 000004      2$:  MOV      (SP)+,@#4
(2) 002540 012637 000006      MOV      (SP)+,@#6
(2) 002544 022767 000176 176666  CMP      #SWREG,SWR
(2) 002552 001007                BNE      3$
(2) 002554 005737 000042      TST      @#42      ;CHECK FOR CHAIN
(2) 002560 001402                BEQ      33$
(2) 002562 000167 000522      JMP      .BEGIN
(2) 002566 004767 010054      33$:  JSR      PC,CNTLU
(2) 002572 105767 176374      3$:  TSTB     INIFLG     ;HAS INITIALIZATION BEEN PERFORMED
(2) 002576 001004                BNE      ONCE
(2) 002600 104401 015022      TYPE     ,MTITLE     ;TYPE TITLE MESSAGE
(2) 002604 105167 176362      COMB     INIFLG     ;IF NOT SET FLAG AND DO
(2) 002610 105767 176732      ONCE:  TSTB     $ENV     ;APT CONTROL?
(2) 002614 001410                BEQ      11$      ;BR IF NO
(2) 002616 032767 000001 176726  BIT      #1,$USWR   ;EXTENAL JUMPER ON?
(2) 002624 001002                BNE      12$      ;NO
(2) 002626 105067 176321      CLRB     JMRBY      ;CLEAR FLAG
(2) 002632 000167 000452      12$:  JMP      .BEGIN     ;GO DO IT
(2) 002636 032777 000001 176574 11$:  BIT      #SW00,@SWR ;RESELECT VECTOR & CONTROL REG?
(2) 002644 001002                BNE      1$
(2) 002646 000167 000436      JMP      .BEGIN
(2) 002652 012700 000300      1$:  MOV      #300,R0 ;RESTORE VECTOR AREA TO TRAPCATCHER
(2) 002656 012701 000302      MOV      #302,R1 ;START AT LOCATION 300
(2) 002662 012702 000004      MOV      #4,R2
(2) 002666 010110      2$:  MOV      R1,(R0)
(2) 002670 005011                CLR      (R1)
(2) 002672 060200                ADD      R2,R0
(2) 002674 060201                ADD      R2,R1
(2) 002676 022701 001000      CMP      #1000,R1 ;END AT LOCATION 776
(2) 002702 002771                BLT      2$
(2) 002704 104406      INSTR   ;OUTPUT MESSAGE & GET INPUT STRING
(2) 002706 015070      MREGAD  ;MESSAGE
(2) 002710 104410      PARAM   ;CONVERT STRING
(2) 002712 174000      174000 ;LOW LIMIT
(2) 002714 177776      177776 ;HIGH LIMIT
(2) 002716 017020      DUBASE  ;STORE AT THIS LOCATION
(2) 002720      001      .BYTE 1 ;MASK
(2) 002721      001      .BYTE 1 ;HOW MANY TIMES + 2
(1) 002722 016767 014072 176226  MOV      DUBASE,KEEPADD ;SAVE
(1) 002730 004767 013732      JSR      PC,DUADDR
  
```

(1)	002734	016767	176216	176212		MOV	KEEPADD,BASEADD ;RESTORE FOR ROTATION
(2)	002742	104406				INSTR	;OUTPUT MESSAGE & GET INPUT STRING
(2)	002744	015055				MVECTO	;MESSAGE
(2)	002746	104410				PARAM	;CONVERT STRING
(2)	002750	000300				300	;LOW LIMIT
(2)	002752	000776				776	;HIGH LIMIT
(2)	002754	001736				DURIV	;STORE AT THIS LOCATION
(2)	002756	001			.BYTE	1	;MASK
(2)	002757	004			.BYTE	4	;HOW MANY TIMES + 2
(1)	002760	016767	176752	176176		MOV	DURIV,KEEPIV ;SAVE
(1)	002766	016767	176744	176166		MOV	DURIV,BASEIV ;SET UP FOR ROTATION
(2)	002774	104406				INSTR	;OUTPUT MESSAGE & GET INPUT STRING
(2)	002776	015120				MMULT	;MESSAGE
(2)	003000	104414				SETFLG	;SET FLAG BASED UPON INPUT STRING
(2)	003002	001152				MULTD	;THIS FLAG
(1)	003004	105767	176142			TSTB	MULTD ;ARE THERE MULTIPLE DEVICES ;ON THE SYSTEM ?
(1)	003010	100406				BMI	BBB ;YES,ASK NEXT QUESTION
(1)	003012	005067	176150			CLR	ACTREG
(1)	003016	005067	176146			CLR	ROTADD
(1)	003022	000167	000140			JMP	OUTMUL ;JUMP AROUND NEXT QUESTION
(1)	003026				BBB:		
(2)	003026	104406				INSTR	;OUTPUT MESSAGE & GET INPUT STRING
(2)	003030	015147				MLASTD	;MESSAGE
(2)	003032	104410				PARAM	;CONVERT STRING
(2)	003034	174000				174000	;LOW LIMIT
(2)	003036	177776				177776	;HIGH LIMIT
(2)	003040	001160				LASTADD	;STORE AT THIS LOCATION
(2)	003042	001			.BYTE	1	;MASK
(2)	003043	001			.BYTE	1	;HOW MANY TIMES + 2
(1)	003044	012767	000001	176116	1\$:	MOV	#1,ROTADD ;SET UP POINTER
(1)	003052	005067	176110			CLR	ACTREG ;CLR ACTIVE REGISTER
(1)	003056	056767	176106	176102	2\$:	BIS	ROTADD,ACTREG ;MAKE THIS DEVICE ACTIVE
(1)	003064	000241				CLC	
(1)	003066	006167	176076			ROL	ROTADD ;SET UP POINTER
(1)	003072	103421				BCS	3\$ ;ARE YOU OUT OF RANGE ?
(1)	003074	062767	000010	176052		ADD	#10,BASEADD ;SET UP BASE ADDRESS
(1)	003102	026767	176052	176044		CMP	LASTADD,BASEADD ;IS THIS THE LAST DEVICE ?
(1)	003110	101362				BHI	2\$ ;NO DO IT AGAIN
(1)	003112	056767	176052	176046		BIS	ROTADD,ACTREG ;THIS ASSUMES THAT THERE ARE AT ;LEAST TWO DEVICES WHEN YOU ANSWER YES TO ;MULTIPLE DEVICE QUESTION
(1)	003120	012767	000001	176042	4\$:	MOV	#1,ROTADD ;SET UP FOR LATER USE IN END OF PASS ROUTINE
(1)	003126	016767	176024	176020		MOV	KEEPADD,BASEADD ;DITTO
(1)	003134	000414				BR	OUTMUL ;CONTINUE QUESTIONS
(1)	003136	016767	176014	176010	3\$:	MOV	KEEPADD,BASEADD ;RESTORE
(2)	003144	104406				INSTR	;OUTPUT MESSAGE & GET INPUT STRING
(2)	003146	015243				MRANGE	;MESSAGE
(2)	003150	104410				PARAM	;CONVERT STRING
(2)	003152	174000				174000	;LOW LIMIT
(2)	003154	177776				177776	;HIGH LIMIT
(2)	003156	001160				LASTADD	;STORE AT THIS LOCATION
(2)	003160	001			.BYTE	1	;MASK
(2)	003161	001			.BYTE	1	;HOW MANY TIMES + 2
(1)	003162	000167	177656			JMP	1\$ ;DO IT AGAIN

```

(1) 003166 012767 000300 013466 OUTMUL: MOV #300,DUPRT
(1) 003174 004767 013412 JSR PC,DULEV
(2) ;COMPARE THE FIRST CHARACTER IN THE TELETYPE INPUT
(2) ;BUFFER TO THE CHARACTERS '1' AND '2'.
(2) ;IF THE CHARACTER IS '1' CLEAR THE FLAG
(2) ;IF THE CHARACTER IS '2' SET THE FLAG
(2) 003200 AAA:
(2) 003200 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003202 015461 MSYNC ;MESSAGE
(2) 003204 122767 000061 012610 3$: CMPB #'1,INBUF ;IS IT '1' ?
(2) 003212 001003 BNE 1$
(2) 003214 105067 175726 CLRB SYNCNO ;000
(2) 003220 000412 BR 4$
(2) 003222 122767 000062 012572 1$: CMPB #'2,INBUF ;IS IT '2' ?
(2) 003230 001004 BNE 2$
(2) 003232 112767 177777 175706 MOVB #-1,SYNCNO ;377
(2) 003240 000402 BR 4$
(2) 003242 104407 2$: INSTER ;RETRY
(2) 003244 000757 BR 3$
(2) 003246 000240 4$: NOP
(2) 003250 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003252 015527 MWIRE6 ;MESSAGE
(2) 003254 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003256 001147 SEXMIT ;THIS FLAG
(2) 003260 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003262 015600 MWIRE5 ;MESSAGE
(2) 003264 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003266 001150 SEREC ;THIS FLAG
(2) 003270 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003272 015650 MWIRE4 ;MESSAGE
(2) 003274 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003276 001151 OPTCLR ;THIS FLAG
(2) 003300 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003302 015727 MEXTJ ;MESSAGE
(2) 003304 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003306 001153 JMRBY ;THIS FLAG
(2)
(2) ;TEST START AND RESTART
(2)
(2) 003310 012706 001100 .BEGIN: MOV #STACK,SP ;SET UP STACK
(2) 003314 106427 000300 MTPS #300 ;LOCK OUT INTERRUPTS
(2) 003320 032777 000002 176112 BIT #SW01,@SWR ;IF SW01=1, GET STARTING PC
(2) 003326 001413 BEQ 3$
(3) 003330 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(3) 003332 015413 MTSTPC ;MESSAGE
(3) 003334 104410 PARAM ;CONVERT STRING
(3) 003336 003374 TST1 ;LOW LIMIT
(3) 003340 017500 17500 ;HIGH LIMIT
(3) 003342 001402 $TSTNM ;STORE AT THIS LOCATION
(3) 003344 001 .BYTE 1 ;MASK
(3) 003345 001 .BYTE 1 ;HOW MANY TIMES + 2
(2) 003346 016767 176030 175530 MOV $TSTNM,RETURN
(2) 003354 000403 BR 4$
(2) 003356 012767 003374 175520 3$: MOV #TST1,RETURN ;START AT TEST 1
(2) 003364 104401 015407 4$: TYPE ,MR ;TYPE R
(2) 003370 000177 175510 JMP @RETURN ;START TESTING
  
```



8151

8152

::THIS TEST VERIFYS WORD LENGTH SELECT OF THE  
::RECEIVER SECTION,IT USES THE ERROR FLAGS  
::TO DETERMINE THAT IT WAS SELECTED CORRECTLY  
::(OVRUN,RXERR)  
::MODE:ISYMOD  
::LENGTH:FIVE  
::CHAR:37  
::

\*\*\*\*\*

(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(5)  
(4) 003374 000004  
(4)  
(3) 003376 052777 000400 176322  
(2) 003404 012777 000000 176310  
(3) 003412 052777 000400 176306  
(2)  
(2)  
(2) 003420 012777 064001 176300  
(2)  
(2)  
(2) 003426 012777 000000 176266  
(2) 003434 052777 000020 176250  
(2)  
(2) 003442 042777 020000 176256  
(2) 003450 052777 020000 176250  
(2)  
(2) 003456 042777 020000 176242  
(2) 003464 052777 020000 176234  
(1) 003472 016703 176220  
(1) 003476 012700 000037  
(1) 003502 012767 000007 175412  
(1) 003510 012767 000176 175762  
(1) 003516 004767 013300  
(1) 003522 105777 176164  
(1) 003526 100401  
(1) 003530 104004  
(1) 003532 017701 176160  
(1) 003536 020001  
(1) 003540 001401  
(1) 003542 104002  
(1)  
(1)  
(1)  
(1) 003544 012767 000007 175350  
(1) 003552 012767 000176 175720  
(1) 003560 004767 013236  
(1)  
(1) 003564 012767 000007 175330  
(1) 003572 012767 000176 175700  
(1) 003600 004767 013216  
(1) 003604 012700 140037  
(1)  
(1) 003610 017701 176102  
(1) 003614 020001  
(1) 003616 001401  
(1) 003620 104002  
(1)

TST1: SCOPE  
 BIS #MRESET,@TXCSR ;MASTER RESET  
 MOV #ISYMOD,@PARCSR ;SET THE MODE  
 BIS #MRESET,@TXCSR ;MASTER RESET  
 ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE  
 MOV #MTDATA!CLK!MINT!BREAK,@TXCSR  
 ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG  
 MOV #ISYMOD!FIVE!NOPAR!0,@PARCSR  
 BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH  
 ;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....  
 BIC #CLK,@TXCSR ;POKE CLK DOWN  
 BIS #CLK,@TXCSR ;POKE CLK UP  
 ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION  
 BIC #CLK,@TXCSR ;POKE CLK DOWN  
 BIS #CLK,@TXCSR ;POKE CLK UP  
 MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE  
 MOV #37,R0 ;EXPECTED  
 MOV #7,SHIFT ;# OF SHIFTS  
 MOV #176,\$TMP1 ;DATA CHAR  
 JSR PC,RPOKE ;SHIFT IN THIS CHAR  
 TSTB @RXCSR ;RXDONE ?  
 BMI ;+4  
 ERROR 4 ;RXDONE SHOULD BE SET  
 MOV @RXDBUF,R1 ;ACTUAL  
 CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL  
 BEQ ;+4  
 ERROR 2 ;RECEIVED DATA DID NOT MATCH  
 ;EXPECTED DATA - CHECK MAINT DATA  
 ;OR RECEIVER LOGIC  
 MOV #7,SHIFT ;# OF SHIFTS  
 MOV #176,\$TMP1 ;DATA CHAR  
 JSR PC,RPOKE ;SHIFT IN THIS CHAR  
 ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF  
 MOV #7,SHIFT ;# OF SHIFTS  
 MOV #176,\$TMP1 ;DATA CHAR  
 JSR PC,RPOKE ;SHIFT IN THIS CHAR  
 MOV #140000!37,R0 ;EXPECTED DATA PLUS  
 ;RXERR & OVRRUN  
 MOV @RXDBUF,R1 ;ACTUAL  
 CMP R0,R1 ;COMPARE EXP VS. ACT  
 BEQ ;+4  
 ERROR 2 ;SPECIFICALLY LOOK AT RXERR &  
 ;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

(1)
8153
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 003622 000004
(4)
(3) 003624 052777 000400 176074
(2) 003632 012777 000000 176062
(3) 003640 052777 000400 176060
(2)
(2)
(2) 003646 012777 064001 176052
(2)
(2)
(2) 003654 012777 000000 176040
(2) 003662 052777 000020 176022
(2)
(2) 003670 042777 020000 176030
(2) 003676 052777 020000 176022
(2)
(2) 003704 042777 020000 176014
(2) 003712 052777 020000 176006
(1) 003720 016703 175772
(1) 003724 012700 000000
(1) 003730 012767 000007 175164
(1) 003736 012767 000100 175534
(1) 003744 004767 013052
(1) 003750 105777 175736
(1) 003754 100401
(1) 003756 104004
(1) 003760 017701 175732
(1) 003764 020001
(1) 003766 001401
(1) 003770 104002
(1)
(1)
(1) 003772 012767 000007 175122
(1) 004000 012767 000100 175472
(1) 004006 004767 013010
(1)
(1) 004012 012767 000007 175102
(1) 004020 012767 000100 175452
(1) 004026 004767 012770
(1) 004032 012700 140000
(1)
(1) 004036 017701 175654
(1) 004042 020001
(1) 004044 001401
(1) 004046 104002
(1)

```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: FIVE
:: CHAR: 0
::
:: *****
TST2:  SCOPE
      BIS    #MRESET,@TXCSR  :MASTER RESET
      MOV    #ISYMOD,@PARCSR  :SET THE MODE
      BIS    #MRESET,@TXCSR  :MASTER RESET

      :SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
      MOV    #MTDATA!CLK!MINT!BREAK,@TXCSR

      :SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
      MOV    #ISYMOD!FIVE!NOPAR!0,@PARCSR
      BIS    #SYNSCH,@RXCSR   :SET SYNC SEARCH
      :POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
      BIC    #CLK,@TXCSR     :POKE CLK DOWN
      BIS    #CLK,@TXCSR     :POKE CLK UP
      :POKE CLK TO GET LOGIC INTO SYNCRONIZATION
      BIC    #CLK,@TXCSR     :POKE CLK DOWN
      BIS    #CLK,@TXCSR     :POKE CLK UP
      MOV    RXDBUF,R3       :SET UP FOR ERROR MESSAGE
      MOV    #0,R0           :EXPECTED
      MOV    #7,SHIFT        :# OF SHIFTS
      MOV    #100,$TMP1      :DATA CHAR
      JSR    PC,RPOKE        :SHIFT IN THIS CHAR
      TSTB   @RXCSR ;RXDONE ?
      BMI    .+4
      ERROR  4               :RXDONE SHOULD BE SET
      MOV    @RXDBUF,R1      :ACTUAL
      CMP    R0,R1           :COMPARE EXPECTED VS. ACTUAL
      BEQ    .+4
      ERROR  2               :RECEIVED DATA DID NOT MATCH
                          :EXPECTED DATA - CHECK MAINT DATA
                          :OR RECEIVER LOGIC
      MOV    #7,SHIFT        :# OF SHIFTS
      MOV    #100,$TMP1      :DATA CHAR
      JSR    PC,RPOKE        :SHIFT IN THIS CHAR
      :NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
      MOV    #7,SHIFT        :# OF SHIFTS
      MOV    #100,$TMP1      :DATA CHAR
      JSR    PC,RPOKE        :SHIFT IN THIS CHAR
      MOV    #140000!0,R0    :EXPECTED DATA PLUS
                          :RXERR & OVRUN
      MOV    @RXDBUF,R1      :ACTUAL
      CMP    R0,R1           :COMPARE EXP VS. ACT
      BEQ    .+4
      ERROR  2               :SPECIFICALLY LOOK AT RXERR &
                          :OVRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8154
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 004050 000004
(4)
(3) 004052 052777 000400 175646
(2) 004060 012777 000000 175634
(3) 004066 052777 000400 175632
(2)
(2)
(2) 004074 012777 064001 175624
(2)
(2)
(2) 004102 012777 002000 175612
(2) 004110 052777 000020 175574
(2)
(2) 004116 042777 020000 175602
(2) 004124 052777 020000 175574
(2)
(2) 004132 042777 020000 175566
(2) 004140 052777 020000 175560
(1) 004146 016703 175544
(1) 004152 012700 000025
(1) 004156 012767 000010 174736
(1) 004164 012767 000252 175306
(1) 004172 004767 012624
(1) 004176 105777 175510
(1) 004202 100401
(1) 004204 104004
(1) 004206 017701 175504
(1) 004212 020001
(1) 004214 001401
(1) 004216 104002
(1)
(1)
(1) 004220 012767 000010 174674
(1) 004226 012767 000252 175244
(1) 004234 004767 012562
(1)
(1) 004240 012767 000010 174654
(1) 004246 012767 000252 175224
(1) 004254 004767 012542
(1) 004260 012700 140025
(1)
(1) 004264 017701 175426
(1) 004270 020001
(1) 004272 001401
(1) 004274 104002
(1)

```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: SIX
:: CHAR: 25
::
*****
TST3: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SIX!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #25,R0 ;EXPECTED
MOV #8,SHIFT ;# OF SHIFTS
MOV #252,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI .+4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ .+4
ERROR 2 ;RECEIVED DATA D;D NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #8,SHIFT ;# OF SHIFTS
MOV #252,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #8,SHIFT ;# OF SHIFTS
MOV #252,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!25,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ .+4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8155
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 004276 000004
(4)
(3) 004300 052777 000400 175420
(2) 004306 012777 000000 175406
(3) 004314 052777 000400 175404
(2)
(2)
(2) 004322 012777 064001 175376
(2)
(2)
(2) 004330 012777 002000 175364
(2) 004336 052777 000020 175346
(2)
(2) 004344 042777 020000 175354
(2) 004352 052777 020000 175346
(2)
(2) 004360 042777 020000 175340
(2) 004366 052777 020000 175332
(1) 004374 016703 175316
(1) 004400 012700 000052
(1) 004404 012767 000010 174510
(1) 004412 012767 000324 175060
(1) 004420 004767 012376
(1) 004424 105777 175262
(1) 004430 100401
(1) 004432 104004
(1) 004434 017701 175256
(1) 004440 020001
(1) 004442 001401
(1) 004444 104002
(1)
(1)
(1) 004446 012767 000010 174446
(1) 004454 012767 000324 175016
(1) 004462 004767 012334
(1)
(1) 004466 012767 000010 174426
(1) 004474 012767 000324 174776
(1) 004502 004767 012314
(1) 004506 012700 140052
(1)
(1) 004512 017701 175200
(1) 004516 020001
(1) 004520 001401
(1) 004522 104002
(1)

```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: SIX
:: CHAR: 52
::
:: *****
TST4: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SIX!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #52,R0 ;EXPECTED
MOV #8,,SHIFT ;# OF SHIFTS
MOV #324,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI +4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ +4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #8,,SHIFT ;# OF SHIFTS
MOV #324,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #8,,SHIFT ;# OF SHIFTS
MOV #324,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!52,R0 ;EXPECTED DATA PLUS
;RXERR & OVRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ +4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8156
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 004524 000004
(4)
(3) 004526 052777 000400 175172
(2) 004534 012777 000000 175160
(3) 004542 052777 000400 175156
(2)
(2) 004550 012777 064001 175150
(2)
(2) 004556 012777 002000 175136
(2) 004564 052777 000020 175120
(2) 004572 042777 020000 175126
(2) 004600 052777 020000 175120
(2) 004606 042777 020000 175112
(2) 004614 052777 020000 175104
(1) 004622 016703 175070
(1) 004626 012700 000077
(1) 004632 012767 000010 174262
(1) 004640 012767 000376 174632
(1) 004646 004767 012150
(1) 004652 105777 175034
(1) 004656 100401
(1) 004660 104004
(1) 004662 017701 175030
(1) 004666 020001
(1) 004670 001401
(1) 004672 104002
(1)
(1)
(1) 004674 012767 000010 174220
(1) 004702 012767 000376 174570
(1) 004710 004767 012106
(1)
(1) 004714 012767 000010 174200
(1) 004722 012767 000376 174550
(1) 004730 004767 012066
(1) 004734 012700 140077
(1)
(1) 004740 017701 174752
(1) 004744 020001
(1) 004746 001401
(1) 004750 104002
(1)

```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: SIX
:: CHAR: 77
::
:: *****
TSTS: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET
;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SIX!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #77,R0 ;EXPECTED
MOV #8,,SHIFT ;# OF SHIFTS
MOV #376,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI +4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ +4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #8,,SHIFT ;# OF SHIFTS
MOV #376,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #8,,SHIFT ;# OF SHIFTS
MOV #376,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!77,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ +4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8157
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 004752 000004
(4)
(3) 004754 052777 000400 174744
(2) 004762 012777 000000 174732
(3) 004770 052777 000400 174730
(2)
(2)
(2) 004776 012777 064001 174722
(2)
(2)
(2) 005004 012777 002000 174710
(2) 005012 052777 000020 174672
(2)
(2) 005020 042777 020000 174700
(2) 005026 052777 020000 174672
(2)
(2) 005034 042777 020000 174664
(2) 005042 052777 020000 174656
(1) 005050 016703 174642
(1) 005054 012700 000000
(1) 005060 012767 000010 174034
(1) 005066 012767 000200 174404
(1) 005074 004767 011722
(1) 005100 105777 174606
(1) 005104 100401
(1) 005106 104004
(1) 005110 017701 174602
(1) 005114 020001
(1) 005116 001401
(1) 005120 104002
(1)
(1)
(1) 005122 012767 000010 173772
(1) 005130 012767 000200 174342
(1) 005136 004767 011660
(1)
(1) 005142 012767 000010 173752
(1) 005150 012767 000200 174322
(1) 005156 004767 011640
(1) 005162 012700 140000
(1)
(1) 005166 017701 174524
(1) 005172 020001
(1) 005174 001401
(1) 005176 104002
(1)

```

```

::THIS TEST VERIFYS WORD LENGTH SELECT OF THE
::RECEIVER SECTION,IT USES THE ERROR FLAGS
::TO DETERMINE THAT IT WAS SELECTED CORRECTLY
::(OVRUN,RXERR)
::MODE:ISYMOD
::LENGTH:SIX
::CHAR:0
::
*****
TST6: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SIX!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #0,R0 ;EXPECTED
MOV #8,SHIFT ;# OF SHIFTS
MOV #200,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI .+4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ .+4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #8,SHIFT ;# OF SHIFTS
MOV #200,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #8,SHIFT ;# OF SHIFTS
MOV #200,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!0,R0 ;EXPECTED DATA PLUS
;RXERR & OVRUN
MOV @RX.BUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ .+4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8158
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 005200 000004
(4)
(3) 005202 052777 000400 174516
(2) 005210 012777 000000 174504
(3) 005216 052777 000400 174502
(2)
(2)
(2) 005224 012777 064001 174474
(2)
(2)
(2) 005232 012777 004000 174462
(2) 005240 052777 000020 174444
(2)
(2) 005246 042777 020000 174452
(2) 005254 052777 020000 174444
(2)
(2) 005262 042777 020000 174436
(2) 005270 052777 020000 174430
(1) 005276 016703 174414
(1) 005302 012700 000125
(1) 005306 012767 000011 173606
(1) 005314 012767 000652 174156
(1) 005322 004767 011474
(1) 005326 105777 174360
(1) 005332 100401
(1) 005334 104004
(1) 005336 017701 174354
(1) 005342 020001
(1) 005344 001401
(1) 005346 104002
(1)
(1)
(1)
(1) 005350 012767 000011 173544
(1) 005356 012767 000652 174114
(1) 005364 004767 011432
(1)
(1) 005370 012767 000011 173524
(1) 005376 012767 000652 174074
(1) 005404 004767 011412
(1) 005410 012700 140125
(1)
(1) 005414 017701 174276
(1) 005420 020001
(1) 005422 001401
(1) 005424 104002
(1)
  
```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: SEVEN
:: CHAR: 125
::
:: *****
TST7: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SEVEN!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #125,R0 ;EXPECTED
MOV #9,SHIFT ;# OF SHIFTS
MOV #652,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI +4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ +4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #9,SHIFT ;# OF SHIFTS
MOV #652,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #9,SHIFT ;# OF SHIFTS
MOV #652,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!125,R0 ;EXPECTED DATA PLUS
;RXERR & OVRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ +4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRUN BITS...THEY BOTH SHOULD BE SET
  
```

```

(1)
8159
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 005426 000004
(4)
(3) 005430 052777 000400 174270
(2) 005436 012777 000000 174256
(3) 005444 052777 000400 174254
(2)
(2)
(2) 005452 012777 064001 174246
(2)
(2)
(2) 005460 012777 004000 174234
(2) 005466 052777 000020 174216
(2)
(2) 005474 042777 020000 174224
(2) 005502 052777 020000 174216
(2)
(2) 005510 042777 020000 174210
(2) 005516 052777 020000 174202
(1) 005524 016703 174166
(1) 005530 012700 000052
(1) 005534 012767 000011 173360
(1) 005542 012767 000524 173730
(1) 005550 004767 011246
(1) 005554 105777 174132
(1) 005560 100401
(1) 005562 104004
(1) 005564 017701 174126
(1) 005570 020001
(1) 005572 001401
(1) 005574 104002
(1)
(1)
(1)
(1) 005576 012767 000011 173316
(1) 005604 012767 000524 173666
(1) 005612 004767 011204
(1)
(1) 005616 012767 000011 173276
(1) 005624 012767 000524 173646
(1) 005632 004767 011164
(1) 005636 012700 140052
(1)
(1) 005642 017701 174050
(1) 005646 020001
(1) 005650 001401
(1) 005652 104002
(1)

```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: SEVEN
:: CHAR: 52
::
::*****
TST10: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SEVEN!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #52,R0 ;EXPECTED
MOV #9,SHIFT ;# OF SHIFTS
MOV #524,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI +4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ +4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #9,SHIFT ;# OF SHIFTS
MOV #524,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #9,SHIFT ;# OF SHIFTS
MOV #524,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!52,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ +4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```



```

(1) 8160
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 005654 000004
(4)
(3) 005656 052777 000400 174042
(2) 005664 012777 000000 174030
(3) 005672 052777 000400 174026
(2)
(2)
(2) 005700 012777 064001 174020
(2)
(2)
(2) 005706 012777 004000 174006
(2) 005714 052777 000020 173770
(2)
(2) 005722 042777 020000 173776
(2) 005730 052777 020000 173770
(2)
(2) 005736 042777 020000 173762
(2) 005744 052777 020000 173754
(1) 005752 016703 173740
(1) 005756 012700 000177
(1) 005762 012767 000011 173132
(1) 005770 012767 000776 173502
(1) 005776 004767 011020
(1) 006002 105777 173704
(1) 006006 100401
(1) 006010 104004
(1) 006012 017701 173700
(1) 006016 020001
(1) 006020 01401
(1) 006022 104002
(1)
(1)
(1) 006024 012767 000011 173070
(1) 006032 012767 000776 173440
(1) 006040 004767 010756
(1)
(1) 006044 012767 000011 173050
(1) 006052 012767 000776 173420
(1) 006060 004767 010736
(1) 006064 012700 140177
(1)
(1) 006070 017701 173622
(1) 006074 020001
(1) 006076 001401
(1) 006100 104002
(1)

```

```

:::THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:::RECEIVER SECTION,IT USES THE ERROR FLAGS
:::TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:::(OVRUN,RXERR)
:::MODE:ISYMOD
:::LENGTH:SEVEN
:::CHAR:177
:::
*****
TST11: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

:SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

:SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!SEVEN!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
:POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
:POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #177,R0 ;EXPECTED
MOV #9,SHIFT ;# OF SHIFTS
MOV #776,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI +4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ +4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #9,SHIFT ;# OF SHIFTS
MOV #776,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
:NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #9,SHIFT ;# OF SHIFTS
MOV #776,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!177,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ +4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1) 8161                                     ::THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1)                                     ::RECEIVER SECTION,IT USES THE ERROR FLAGS
(1)                                     ::TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1)                                     ::(OVRUN,RXERR)
(1)                                     ::MODE:ISYMOD
(1)                                     ::LENGTH:SEVEN
(1)                                     ::CHAR:0
(1)                                     ::
(5)                                     ::*****
(4) 006102 000004                         TST12: SCOPE
(4)
(3) 006104 052777 000400 173614          BIS    #MRESET,@TXCSR ;MASTER RESET
(2) 006112 012777 000000 173602          MOV    #ISYMOD,@PARCSR ;SET THE MODE
(3) 006120 052777 000400 173600          BIS    #MRESET,@TXCSR ;MASTER RESET
(2)
(2) 006126 012777 064001 173572          ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2)                                     MOV    #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2) 006134 012777 004000 173560          ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 006142 052777 000020 173542          MOV    #ISYMOD!SEVEN!NOPAR!0,@PARCSR
(2)                                     BIS    #SYNSCH,@RXCSR ;SET SYNC SEARCH
(2)                                     ;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
(2) 006150 042777 020000 173550          BIC    #CLK,@TXCSR ;POKE CLK DOWN
(2) 006156 052777 020000 173542          BIS    #CLK,@TXCSR ;POKE CLK UP
(2)                                     ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 006164 042777 020000 173534          BIC    #CLK,@TXCSR ;POKE CLK DOWN
(2) 006172 052777 020000 173526          BIS    #CLK,@TXCSR ;POKE CLK UP
(1) 006200 016703 173512                  MOV    RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 006204 012700 000000                  MOV    #0,R0 ;EXPECTED
(1) 006210 012767 000011 172704          MOV    #9,SHIFT ;# OF SHIFTS
(1) 006216 012767 000400 173254          MOV    #400,$TMP1 ;DATA CHAR
(1) 006224 004767 010572                  JSR    PC,RPOKE ;SHIFT IN THIS CHAR
(1) 006230 105777 173456                  TSTB  @RXCSR ;RXDONE ?
(1) 006234 100401                          BMI    .+4
(1) 006236 104004                          ERROR  4 ;RXDONE SHOULD BE SET
(1) 006240 017701 173452                  MOV    @RXDBUF,R1 ;ACTUAL
(1) 006244 C20001                          CMP    R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 006246 001401                          BEQ    .+4
(1) 006250 104002                          ERROR  2 ;RECEIVED DATA DID NOT MATCH
(1)                                     ;EXPECTED DATA - CHECK MAINT DATA
(1)                                     ;OR RECEIVER LOGIC
(1) 006252 012767 000011 172642          MOV    #9,SHIFT ;# OF SHIFTS
(1) 006260 012767 000400 173212          MOV    #400,$TMP1 ;DATA CHAR
(1) 006266 004767 010530                  JSR    PC,RPOKE ;SHIFT IN THIS CHAR
(1)                                     ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 006272 012767 000011 172622          MOV    #9,SHIFT ;# OF SHIFTS
(1) 006300 012767 000400 173172          MOV    #400,$TMP1 ;DATA CHAR
(1) 006306 004767 010510                  JSR    PC,RPOKE ;SHIFT IN THIS CHAR
(1) 006312 012700 140000                  MOV    #140000!0,R0 ;EXPECTED DATA PLUS
(1)                                     ;RXERR & OVRRUN
(1) 006316 017701 173374                  MOV    @RXDBUF,R1 ;ACTUAL
(1) 006322 020001                          CMP    R0,R1 ;COMPARE EXP VS. ACT
(1) 006324 001401                          BEQ    .+4
(1) 006326 104002                          ERROR  2 ;SPECIFICALLY LOOK AT RXERR &
(1)                                     ;OVRRUN BITS...THEY BOTH SHOULD BE SET
    
```

```

(1) 8162
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 006330 000004
(4)
(3) 006332 052777 000400 173366
(2) 006340 012777 000000 173354
(3) 006346 052777 000400 173352
(2)
(2)
(2) 006354 012777 064001 173344
(2)
(2)
(2) 006362 012777 006000 173332
(2) 006370 052777 000020 173314
(2)
(2) 006376 042777 020000 173322
(2) 006404 052777 020000 173314
(2)
(2) 006412 042777 020000 173306
(2) 006420 052777 020000 173300
(1) 006426 016703 173264
(1) 006432 012700 000125
(1) 006436 012767 000012 172456
(1) 006444 012767 001252 173026
(1) 006452 004767 010344
(1) 006456 105777 173230
(1) 006462 100401
(1) 006464 104004
(1) 006466 017701 173224
(1) 006472 020001
(1) 006474 001401
(1) 006476 104002
(1)
(1)
(1) 006500 012767 000012 172414
(1) 006506 012767 001252 172764
(1) 006514 004767 010302
(1)
(1) 006520 012767 000012 172374
(1) 006526 012767 001252 172744
(1) 006534 004767 010262
(1) 006540 012700 140125
(1)
(1) 006544 017701 173146
(1) 006550 020001
(1) 006552 001401
(1) 006554 104002
(1)
  
```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: EIGHT
:: CHAR: 125
:: *****
TST13: SCOPE
      BIS #MRESET,@TXCSR ;MASTER RESET
      MOV #ISYMOD,@PARCSR ;SET THE MODE
      BIS #MRESET,@TXCSR ;MASTER RESET
;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
      MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
      MOV #ISYMOD!EIGHT!NOPAR!0,@PARCSR
      BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
      BIC #CLK,@TXCSR ;POKE CLK DOWN
      BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
      BIC #CLK,@TXCSR ;POKE CLK DOWN
      BIS #CLK,@TXCSR ;POKE CLK UP
      MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
      MOV #125,R0 ;EXPECTED
      MOV #10,SHIFT ;# OF SHIFTS
      MOV #1252,$TMP1 ;DATA CHAR
      JSR PC,RPOKE ;SHIFT IN THIS CHAR
      TSTB @RXCSR ;RXDONE ?
      BMI .+4
      ERROR 4 ;RXDONE SHOULD BE SET
      MOV @RXDBUF,R1 ;ACTUAL
      CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
      BEQ .+4
      ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
      MOV #10,SHIFT ;# OF SHIFTS
      MOV #1252,$TMP1 ;DATA CHAR
      JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
      MOV #10,SHIFT ;# OF SHIFTS
      MOV #1252,$TMP1 ;DATA CHAR
      JSR PC,RPOKE ;SHIFT IN THIS CHAR
      MOV #14000!125,R0 ;EXPECTED DATA PLUS
;RXERR & OVRUN
      MOV @RXDBUF,R1 ;ACTUAL
      CMP R0,R1 ;COMPARE EXP VS. ACT
      BEQ .+4
      ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRUN BITS...THEY BOTH SHOULD BE SET
  
```

```

(1)
8163
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 006556 000004
(4)
(3) 006560 052777 000400 173140 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 006566 012777 000000 173126 MOV #ISYMOD,@PARCSR ;SET THE MODE
(3) 006574 052777 000400 173124 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) 006602 012777 064001 173116 ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)
(2) 006610 012777 006000 173104 ;SET MODE,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!EIGHT!NOPAR!0,@PARCSR
(2) 006616 052777 000020 173066 BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCRONIZATION....
(2) 006624 042777 020000 173074 BIC #CLK,@TXCSR ;POKE CLK DOWN
(2) 006632 052777 020000 173066 BIS #CLK,@TXCSR ;POKE CLK UP
(2)
(2) 006640 042777 020000 173060 ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
(2) 006646 052777 020000 173052 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 006654 016703 173036 MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 006660 012700 000252 MOV #252,R0 ;EXPECTED
(1) 006664 012767 000012 172230 MOV #10,,SHIFT ;# OF SHIFTS
(1) 006672 012767 001524 172600 MOV #1524,$TMP1 ;DATA CHAR
(1) 006700 004767 010116 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) 006704 105777 173002 TSTB @RXCSR ;RXDONE ?
(1) 006710 100401 BMI .+4
(1) 006712 104004 ERROR 4 ;RXDONE SHOULD BE SET
(1) 006714 017701 172776 MOV @RXDBUF,R1 ;ACTUAL
(1) 006720 020001 CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 006722 001401 BEQ .+4
(1) 006724 104002 ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
(1)
(1) 006726 012767 000012 172166 MOV #10,,SHIFT ;# OF SHIFTS
(1) 006734 012767 001524 172536 MOV #1524,$TMP1 ;DATA CHAR
(1) 006742 004767 010054 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1)
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 006746 012767 000012 172146 MOV #10,,SHIFT ;# OF SHIFTS
(1) 006754 012767 001524 172516 MOV #1524,$TMP1 ;DATA CHAR
(1) 006762 004767 010034 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) 006766 012700 140252 MOV #140000!252,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
(1)
(1) 006772 017701 172720 MOV @RXDBUF,R1 ;ACTUAL
(1) 006776 020001 CMP R0,R1 ;COMPARE EXP VS. ACT
(1) 007000 001401 BEQ .+4
(1) 007002 104002 ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1) 8164
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 007004 000004
(4)
(3) 007006 052777 000400 172712
(2) 007014 012777 000000 172700
(3) 007022 052777 000400 172676
(2)
(2)
(2) 007030 012777 064001 172670
(2)
(2)
(2) 007036 012777 006000 172656
(2) 007044 052777 000020 172640
(2)
(2) 007052 042777 020000 172646
(2) 007060 052777 020000 172640
(2)
(2) 007066 042777 020000 172632
(2) 007074 052777 020000 172624
(1) 007102 016703 172610
(1) 007106 012700 000377
(1) 007112 012767 000012 172002
(1) 007120 012767 001776 172352
(1) 007126 004767 007670
(1) 007132 105777 172554
(1) 007136 100401
(1) 007140 104004
(1) 007142 017701 172550
(1) 007146 020001
(1) 007150 001401
(1) 007152 104002
(1)
(1)
(1) 007154 012767 000012 171740
(1) 007162 012767 001776 172310
(1) 007170 004767 007626
(1)
(1) 007174 012767 000012 171720
(1) 007202 012767 001776 172270
(1) 007210 004767 007606
(1) 007214 012700 140377
(1)
(1) 007220 017701 172472
(1) 007224 020001
(1) 007226 001401
(1) 007230 104002
(1)

```

```

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: EIGHT
:: CHAR: 377
::
:: *****
TST15: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET
;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!EIGHT!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #377,R0 ;EXPECTED
MOV #10,SHIFT ;# OF SHIFTS
MOV #1776,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI .+4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ .+4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #10,SHIFT ;# OF SHIFTS
MOV #1776,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #10,SHIFT ;# OF SHIFTS
MOV #1776,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!377,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ .+4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8165
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 007232 000004
(4)
(3) 007234 052777 000400 172464
(2) 007242 012777 000000 172452
(3) 007250 052777 000400 172450
(2)
(2)
(2) 007256 012777 064001 172442
(2)
(2)
(2) 007264 012777 006000 172430
(2) 007272 052777 000020 172412
(2)
(2) 007300 042777 020000 172420
(2) 007306 052777 020000 172412
(2)
(2) 007314 042777 020000 172404
(2) 007322 052777 020000 172376
(1) 007330 016703 172362
(1) 007334 012700 000000
(1) 007340 012767 000012 171554
(1) 007346 012767 001000 172124
(1) 007354 004767 007442
(1) 007360 105777 172326
(1) 007364 100401
(1) 007366 104004
(1) 007370 017701 172322
(1) 007374 020001
(1) 007376 001401
(1) 007400 104002
(1)
(1)
(1) 007402 012767 000012 171512
(1) 007410 012767 001000 172062
(1) 007416 004767 007400
(1)
(1) 007422 012767 000012 171472
(1) 007430 012767 001000 172042
(1) 007436 004767 007360
(1) 007442 012700 140000
(1)
(1) 007446 017701 172244
(1) 007452 020001
(1) 007454 001401
(1) 007456 104002
(1)

:: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
:: RECEIVER SECTION, IT USES THE ERROR FLAGS
:: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
:: (OVRUN, RXERR)
:: MODE: ISYMOD
:: LENGTH: EIGHT
:: CHAR: 0
::
:: *****
TST16: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #ISYMOD,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET

;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR

;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #ISYMOD!EIGHT!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #0,R0 ;EXPECTED
MOV #10,,SHIFT ;# OF SHIFTS
MOV #1000,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI .-4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ .+4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #10,,SHIFT ;# OF SHIFTS
MOV #1000,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #10,,SHIFT ;# OF SHIFTS
MOV #1000,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!0,R0 ;EXPECTED DATA PLUS
;RXERR & OVRRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ .+4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRRUN BITS...THEY BOTH SHOULD BE SET

```

```

(1)
8166
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 007460 000004
(4)
(3) 007462 052777 000400 172236
(2) 007470 012777 020000 172224
(3) 007476 052777 000400 172222
(2)
(2)
(2) 007504 012777 064001 172214
(2)
(2)
(2) 007512 012777 020000 172202
(1) 007520 052777 000020 172164
(2)
(2) 007526 042777 020000 172172
(2) 007534 052777 020000 172164
(1) 007542 016703 172150
(1) 007546 012700 000025
(1) 007552 012767 000005 171342
(1) 007560 012767 000025 171712
(1) 007566 004767 007230
(1) 007572 105777 172114
(1) 007576 100401
(1) 007600 104004
(1) 007602 017701 172110
(1) 007606 020001
(1) 007610 001401
(1) 007612 104002
(1)
(1)
(1)
(1) 007614 012767 000005 171300
(1) 007622 012767 000025 171650
(1) 007630 004767 007166
(1)
(1) 007634 012767 000005 171260
(1) 007642 012767 000025 171630
(1) 007650 004767 007146
(1) 007654 012700 140025
(1)
(1) 007660 017701 172032
(1) 007664 020001
(1) 007666 001401
(1) 007670 104002
(1)
(1)
8167
(1)

```

```

::THIS TEST VERIFYS WORD LENGTH SELECT OF THE
::RECEIVER SECTION,IT USES THE ERROR FLAGS
::TO DETERMINE THAT IT WAS SELECTED CORRECTLY
::(OVRUN,RXERR)
::MODE:SYNEXT
::LENGTH:FIVE
::CHAR:25
:*****
TST17: SCOPE
BIS #MRESET,@TXCSR ;MASTER RESET
MOV #SYNEXT,@PARCSR ;SET THE MODE
BIS #MRESET,@TXCSR ;MASTER RESET
;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #SYNEXT!FIVE!NOPAR!0,@PARCSR
BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
BIC #CLK,@TXCSR ;POKE CLK DOWN
BIS #CLK,@TXCSR ;POKE CLK UP
MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
MOV #25,R0 ;EXPECTED
MOV #5,SHIFT ;# OF SHIFTS
MOV #25,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
TSTB @RXCSR ;RXDONE ?
BMI +4
ERROR 4 ;RXDONE SHOULD BE SET
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
BEQ +4
ERROR 2 ;RECEIVED DATA DID NOT MATCH
;EXPECTED DATA - CHECK MAINT DATA
;OR RECEIVER LOGIC
MOV #5,SHIFT ;# OF SHIFTS
MOV #25,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
MOV #5,SHIFT ;# OF SHIFTS
MOV #25,$TMP1 ;DATA CHAR
JSR PC,RPOKE ;SHIFT IN THIS CHAR
MOV #140000!25,R0 ;EXPECTED DATA PLUS
;RXERR & OVRUN
MOV @RXDBUF,R1 ;ACTUAL
CMP R0,R1 ;COMPARE EXP VS. ACT
BEQ +4
ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
;OVRUN BITS...THEY BOTH SHOULD BE SET

```

```

::THIS TEST VERIFYS WORD LENGTH SELECT OF THE
::RECEIVER SECTION,IT USES THE ERROR FLAGS

```

```

(1)                                     ::TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1)                                     ::(OVRRUN,RXERR)
(1)                                     ::MODE:SYNEXT
(1)                                     ::LENGTH:FIVE
(1)                                     ::CHAR:12
(1)                                     ::
(5)                                     ::*****
(4) 007672 000004                       TST20: SCOPE
(4)
(3) 007674 052777 000400 172024         BIS    #MRESET,@TXCSR  ;MASTER RESET
(2) 007702 012777 020000 172012         MOV    #SYNEXT,@PARCSR ;SET THE MODE
(3) 007710 052777 000400 172010         BIS    #MRESET,@TXCSR  ;MASTER RESET
(2)
(2)                                     ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 007716 012777 064001 172002         MOV    #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)                                     ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 007724 012777 020000 171770         MOV    #SYNEXT!FIVE!NOPAR!0,@PARCSR
(1) 007732 052777 000020 171752         BIS    #SYNSCH,@RXCSR  ;SET SEARCH SYNC
(2)                                     ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 007740 042777 020000 171760         BIC    #CLK,@TXCSR     ;POKE CLK DOWN
(2) 007746 052777 020000 171752         BIS    #CLK,@TXCSR     ;POKE CLK UP
(1) 007754 016703 171736                 MOV    RXDBUF,R3       ;SET UP FOR ERROR MESSAGE
(1) 007760 012700 000012                 MOV    #12,R0          ;EXPECTED
(1) 007764 012767 000005 171130         MOV    #5,SHIFT        ;# OF SHIFTS
(1) 007772 012767 000012 171500         MOV    #12,$TMP1       ;DATA CHAR
(1) 010000 004767 007016                 JSR    PC,RPOKE         ;SHIFT IN THIS CHAR
(1) 010004 105777 171702                 TSTB   @RXCSR          ;RXDONE ?
(1) 010010 100401                         BMI    +4
(1) 010012 104004                         ERROR  4               ;RXDONE SHOULD BE SET
(1) 010014 017701 171676                 MOV    @RXDBUF,R1      ;ACTUAL
(1) 010020 020001                         CMP    R0,R1           ;COMPARE EXPECTED VS. ACTUAL
(1) 010022 001401                         BEQ    +4
(1) 010024 104002                         ERROR  2               ;RECEIVED DATA DID NOT MATCH
(1)                                     ;EXPECTED DATA - CHECK MAINT DATA
(1)                                     ;OR RECEIVER LOGIC
(1) 010026 012767 000005 171066         MOV    #5,SHIFT        ;# OF SHIFTS
(1) 010034 012767 000012 171436         MOV    #12,$TMP1       ;DATA CHAR
(1) 010042 004767 006754                 JSR    PC,RPOKE         ;SHIFT IN THIS CHAR
(1)                                     ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 010046 012767 000005 171046         MOV    #5,SHIFT        ;# OF SHIFTS
(1) 010054 012767 000012 171416         MOV    #12,$TMP1       ;DATA CHAR
(1) 010062 004767 006734                 JSR    PC,RPOKE         ;SHIFT IN THIS CHAR
(1) 010066 012700 140012                 MOV    #140000!12,R0   ;EXPECTED DATA PLUS
(1)                                     ;RXERR & OVRRUN
(1) 010072 017701 171620                 MOV    @RXDBUF,R1      ;ACTUAL
(1) 010076 020001                         CMP    R0,R1           ;COMPARE EXP VS. ACT
(1) 010100 001401                         BEQ    +4
(1) 010102 104002                         ERROR  2               ;SPECIFICALLY LOOK AT RXERR &
(1)                                     ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
(1)                                     ::THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1)                                     ::RECEIVER SECTION,IT USES THE ERROR FLAGS
(1)                                     ::TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1)                                     ::(OVRRUN,RXERR)
(1)                                     ::MODE:SYNEXT
  
```

8168



```

(1)          ::LENGTH:FIVE
(1)          ::CHAR:37
(1)          ::
(5)          ::*****
(4) 010104 000004  TST21: SCOPE
(4)
(3) 010106 052777 000400 171612  BIS    #MRESET,@TXCSR ;MASTER RESET
(2) 010114 012777 020000 171600  MOV    #SYNEXT,@PARCSR ;SET THE MODE
(3) 010122 052777 000400 171576  BIS    #MRESET,@TXCSR ;MASTER RESET
(2)
(2)          ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 010130 012777 064001 171570  MOV    #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)          ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 010136 012777 020000 171556  MOV    #SYNEXT!FIVE!NOPAR!0,@PARCSR
(1) 010144 052777 000020 171540  BIS    #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2)          ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 010152 042777 020000 171546  BIC    #CLK,@TXCSR ;POKE CLK DOWN
(2) 010160 052777 020000 171540  BIS    #CLK,@TXCSR ;POKE CLK UP
(1) 010166 06703 171524  MOV    RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 010172 012700 000037  MOV    #37,R0 ;EXPECTED
(1) 010176 012767 000005 170716  MOV    #5,SHIFT ;# OF SHIFTS
(1) 010204 012767 000037 171266  MOV    #37,$TMP1 ;DATA CHAR
(1) 010212 004767 006604  JSR    PC,RPOKE ;SHIFT IN THIS CHAR
(1) 010216 105777 171470  TSTB   @RXCSR ;RXDONE ?
(1) 010222 100401  BMI    .+4
(1) 010224 104004  ERROR  4 ;RXDONE SHOULD BE SET
(1) 010226 017701 171464  MOV    @RXDBUF,R1 ;ACTUAL
(1) 010232 020001  CMP    R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 010234 001401  BEQ    .+4
(1) 010236 104002  ERROR  2 ;RECEIVED DATA DID NOT MATCH
(1)          ;EXPECTED DATA - CHECK MAINT DATA
(1)          ;OR RECEIVER LOGIC
(1) 010240 012767 000005 170654  MOV    #5,SHIFT ;# OF SHIFTS
(1) 010246 012767 000037 171224  MOV    #37,$TMP1 ;DATA CHAR
(1) 010254 004767 006542  JSR    PC,RPOKE ;SHIFT IN THIS CHAR
(1)          ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 010260 012767 000005 170634  MOV    #5,SHIFT ;# OF SHIFTS
(1) 010266 012767 000037 171204  MOV    #37,$TMP1 ;DATA CHAR
(1) 010274 004767 006522  JSR    PC,RPOKE ;SHIFT IN THIS CHAR
(1) 010300 012700 140037  MOV    #140000!37,R0 ;EXPECTED DATA PLUS
(1)          ;RXERR & OVRRUN
(1) 010304 017701 171406  MOV    @RXDBUF,R1 ;ACTUAL
(1) 010310 020001  CMP    R0,R1 ;COMPARE EXP VS. ACT
(1) 010312 001401  BEQ    .+4
(1) 010314 104002  ERROR  2 ;SPECIFICALLY LOOK AT RXERR &
(1)          ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
(1)          ;THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1)          ;RECEIVER SECTION,IT USES THE ERROR FLAGS
(1)          ;TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1)          ;(OVRRUN,RXERR)
(1)          ;MODE:SYNEXT
(1)          ;LENGTH:FIVE
(1)          ;CHAR:0
(1)          ;
(1)          ;
8169

```

```
(5) ::*****  
(4) 010316 000004 TST22: SCOPE  
(4)  
(3) 010320 052777 000400 171400 BIS #MRESET,@TXCSR :MASTER RESET  
(2) 010326 012777 020000 171366 MOV #SYNEXT,@PARCSR :SET THE MODE  
(3) 010334 052777 000400 171364 BIS #MRESET,@TXCSR :MASTER RESET  
(2)  
(2) :SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE  
(2) 010342 012777 064001 171356 MOV #MTDATA!CLK!MINT!BREAK,@TXCSR  
(2)  
(2) :SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG  
(2) 010350 012777 020000 171344 MOV #SYNEXT!FIVE!NOPAR!0,@PARCSR  
(1) 010356 052777 000020 171326 BIS #SYNSCH,@RXCSR :SET SEARCH SYNC  
(2) :POKE CLK TO GET LOGIC INTO SYNCHRONIZATION  
(2) 010364 042777 020000 171334 BIC #CLK,@TXCSR :POKE CLK DOWN  
(2) 010372 052777 020000 171326 BIS #CLK,@TXCSR :POKE CLK UP  
(1) 010400 016703 171312 MOV RXDBUF,R3 :SET UP FOR ERROR MESSAGE  
(1) 010404 012700 000000 MOV #0,R0 :EXPECTED  
(1) 010410 012767 000005 170504 MOV #5,SHIFT :# OF SHIFTS  
(1) 010416 012767 000000 171054 MOV #0,$TMP1 :DATA CHAR  
(1) 010424 004767 006372 JSR PC,RPOKE :SHIFT IN THIS CHAR  
(1) 010430 105777 171256 TSTB @RXCSR ;RXDONE ?  
(1) 010434 100401 BMI .+4  
(1) 010436 104004 ERROR 4 ;RXDONE SHOULD BE SET  
(1) 010440 017701 171252 MOV @RXDBUF,R1 :ACTUAL  
(1) 010444 020001 CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL  
(1) 010446 001401 BEQ .+4  
(1) 010450 104002 ERROR 2 ;RECEIVED DATA DID NOT MATCH  
(1) ;EXPECTED DATA - CHECK MAINT LATA  
(1) ;OR RECEIVER LOGIC  
(1) 010452 012767 000005 170442 MOV #5,SHIFT :# OF SHIFTS  
(1) 010460 012767 000000 171012 MOV #0,$TMP1 :DATA CHAR  
(1) 010466 004767 006330 JSR PC,RPOKE :SHIFT IN THIS CHAR  
(1) :NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF  
(1) 010472 012767 000005 170422 MOV #5,SHIFT :# OF SHIFTS  
(1) 010500 012767 000000 170772 MOV #0,$TMP1 :DATA CHAR  
(1) 010506 004767 006310 JSR PC,RPOKE :SHIFT IN THIS CHAR  
(1) 010512 012700 140000 MOV #140000!0,R0 :EXPECTED DATA PLUS  
(1) ;RXERR & OVRRUN  
(1) 010516 017701 171174 MOV @RXDBUF,R1 :ACTUAL  
(1) 010522 020001 CMP R0,R1 ;COMPARE EXP VS. ACT  
(1) 010524 001401 BEQ .+4  
(1) 010526 104002 ERROR 2 ;SPECIFICALLY LOOK AT RXERR &  
(1) ;OVRRUN BITS...THEY BOTH SHOULD BE SET  
(1)  
8170 :THIS TEST VERIFYS WORD LENGTH SELECT OF THE  
(1) :RECEIVER SECTION,IT USES THE ERROR FLAGS  
(1) :TO DETERMINE THAT IT WAS SELECTED CORRECTLY  
(1) : (OVRRUN,RXERR)  
(1) :MODE:SYNEXT  
(1) :LENGTH:SIX  
(1) :CHAR:25  
(1) :  
(5) ::*****  
(4) 010530 000004 TST23: SCOPE  
(4)
```

```

(3) 010532 052777 000400 171166      BIS      #MRESET,@TXCSR ;MASTER RESET
(2) 010540 012777 020000 171154      MOV      #SYNEXT,@PARCSR ;SET THE MODE
(3) 010546 052777 000400 171152      BIS      #MRESET,@TXCSR ;MASTER RESET
(2)
(2) 010554 012777 064001 171144      :SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2)      MOV      #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2) 010562 012777 022000 171132      :SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2)      MOV      #SYNEXT!SIX!NOPAR!0,@PARCSR
(1) 010570 052777 000020 171114      BIS      #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2)      :POKE CLK TO GET LOGIC INTO SYNCHRONIZATION
(2) 010576 042777 020000 171122      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 010604 052777 020000 171114      BIS      #CLK,@TXCSR ;POKE CLK UP
(1) 010612 016703 171100      MOV      RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 010616 012700 000025      MOV      #25,R0 ;EXPECTED
(1) 010622 012767 000006 170272      MOV      #6,SHIFT ;# OF SHIFTS
(1) 010630 012767 000025 170642      MOV      #25,$TMP1 ;DATA CHAR
(1) 010636 004767 006160      JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 010642 105777 171044      TSTB    @RXCSR ;RXDONE ?
(1) 010646 100401      BMI     .+4
(1) 010650 104004      ERROR   4 ;RXDONE SHOULD BE SET
(1) 010652 017701 171040      MOV      @RXDBUF,R1 ;ACTUAL
(1) 010656 020001      CMP     R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 010660 001401      BEQ     .+4
(1) 010662 104002      ERROR   2 ;RECEIVED DATA DID NOT MATCH
(1)      ;EXPECTED DATA - CHECK MAINT DATA
(1)      ;OR RECEIVER LOGIC
(1) 010664 012767 000006 170230      MOV      #6,SHIFT ;# OF SHIFTS
(1) 010672 012767 000025 170600      MOV      #25,$TMP1 ;DATA CHAR
(1) 010700 004767 006116      JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1)      :NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 010704 012767 000006 170210      MOV      #6,SHIFT ;# OF SHIFTS
(1) 010712 012767 000025 170560      MOV      #25,$TMP1 ;DATA CHAR
(1) 010720 004767 006076      JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 010724 012700 140025      MOV      #140000!25,R0 ;EXPECTED DATA PLUS
(1)      ;RXERR & OVRRUN
(1) 010730 017701 170762      MOV      @RXDBUF,R1 ;ACTUAL
(1) 010734 020001      CMP     R0,R1 ;COMPARE EXP VS. ACT
(1) 010736 001401      BEQ     .+4
(1) 010740 104002      ERROR   2 ;SPECIFICALLY LOOK AT RXERR &
(1)      ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
8171      ;:THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1)      ;:RECEIVER SECTION,IT USES THE ERROR FLAGS
(1)      ;:TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1)      ;:(OVRRUN,RXERR)
(1)      ;:MODE:SYNEXT
(1)      ;:LENGTH:SIX
(1)      ;:CHAR:52
(1)      ;:
(5)      ;:*****
(4) 010742 000004      TST24: SCOPE
(4)
(3) 010744 052777 000400 170754      BIS      #MRESET,@TXCSR ;MASTER RESET
(2) 010752 012777 020000 170742      MOV      #SYNEXT,@PARCSR ;SET THE MODE
(3) 010760 052777 000400 170740      BIS      #MRESET,@TXCSR ;MASTER RESET
  
```

```

(2)
(2) 010766 012777 064001 170732 ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
MOV #MCDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2) 010774 012777 022000 170720 ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
MOV #SYNEXT!SIX!NOPAR!0,@PARCSR
(1) 011002 052777 000020 170702 BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 011010 042777 020000 170710 BIC #CLK,@TXCSR ;POKE CLK DOWN
(2) 011016 052777 020000 170702 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 011024 016703 170666 MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 011030 012700 000052 MOV #52,R0 ;EXPECTED
(1) 011034 012767 000006 170060 MOV #6,SHIFT ;# OF SHIFTS
(1) 011042 012767 000052 170430 MOV #52,$TMP1 ;DATA CHAR
(1) 011050 004767 005746 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011054 105777 170632 TSTB @RXCSR ;RXDONE ?
(1) 011060 100401 BMI .+4
(1) 011062 104004 ERROR 4 ;RXDONE SHOULD BE SET
(1) 011064 017701 170626 MOV @RXDBUF,R1 ;ACTUAL
(1) 011070 020001 CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 011072 001401 BEQ .+4
(1) 011074 104002 ERROR 2 ;RECEIVED DATA DID NOT MATCH
(1) ;EXPECTED DATA - CHECK MAINT DATA
(1) ;OR RECEIVER LOGIC
(1) 011076 012767 000006 170016 MOV #6,SHIFT ;# OF SHIFTS
(1) 011104 012767 000052 170366 MOV #52,$TMP1 ;DATA CHAR
(1) 011112 004767 005704 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 011116 012767 000006 167776 MOV #6,SHIFT ;# OF SHIFTS
(1) 011124 012767 000052 170346 MOV #52,$TMP1 ;DATA CHAR
(1) 011132 004767 005664 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011136 012700 140052 MOV #140000!52,R0 ;EXPECTED DATA PLUS
(1) ;RXERR & OVRRUN
(1) 011142 017701 170550 MOV @RXDBUF,R1 ;ACTUAL
(1) 011146 020001 CMP R0,R1 ;COMPARE EXP VS. ACT
(1) 011150 001401 BEQ .+4
(1) 011152 104002 ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
(1) ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
8172 ;: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1) ;: RECEIVER SECTION,IT USES THE ERROR FLAGS
(1) ;: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1) ;: (OVRRUN,RXERR)
(1) ;: MODE:SYNEXT
(1) ;: LENGTH:SIX
(1) ;: CHAR:77
(1) ;:
(5) ;:*****
(4) 011154 000004 TST25: SCOPE
(4)
(3) 011156 052777 000400 170542 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 011164 012777 020000 170530 MOV #SYNEXT,@PARCSR ;SET THE MODE
(3) 011172 052777 000400 170526 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 011200 012777 064001 170520 MOV #MCDATA!CLK!MINT!BREAK,@TXCSR

```

```

(2)
(2) 011206 012777 022000 170506 ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(1) 011214 052777 000020 170470   MOV   #SYNEXT!SIX!NOPAR!0,@PARCSR
(2) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION   BIS   #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2) 011222 042777 020000 170476   BIC   #CLK,@TXCSR! ;POKE CLK DOWN
(2) 011230 052777 020000 170470   BIS   #CLK,@TXCS! ;POKE CLK UP
(1) 011236 016703 170454   MOV   RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 011242 012700 000077   MOV   #77,R0 ;EXPECTED
(1) 011246 012767 000006 167646   MOV   #6,SHIFT ;# OF SHIFTS
(1) 011254 012767 000077 170216   MOV   #77,$TMP1 ;DATA CHAR
(1) 011262 004767 005534   JSR   PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011266 105777 170420   TSTB  @RXCSR ;RXDONE ?
(1) 011272 100401   BMI  .+4
(1) 011274 104004   ERROR 4 ;RXDONE SHOULD BE SET
(1) 011276 017701 170414   MOV   @RXDBUF,R1 ;ACTUAL
(1) 011302 020001   CMP   R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 011304 001401   BEQ  .+4
(1) 011306 104002   ERROR 2 ;RECEIVED DATA DID NOT MATCH
(1) ;EXPECTED DATA - CHECK MAINT DATA
(1) ;OR RECEIVER LOGIC
(1) 011310 012767 000006 167604   MOV   #6,SHIFT ;# OF SHIFTS
(1) 011316 012767 000077 170154   MOV   #77,$TMP1 ;DATA CHAR
(1) 011324 004767 005472   JSR   PC,RPOKE ;SHIFT IN THIS CHAR
(1) ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 011330 012767 000006 167564   MOV   #6,SHIFT ;# OF SHIFTS
(1) 011336 012767 000077 170134   MOV   #77,$TMP1 ;DATA CHAR
(1) 011344 004767 005452   JSR   PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011350 012700 140077   MOV   #14000!77,R0 ;EXPECTED DATA PLUS
(1) ;RXERR & OVRRUN
(1) 011354 017701 170336   MOV   @RXDBUF,R1 ;ACTUAL
(1) 011360 020001   CMP   R0,R1 ;COMPARE EXP VS. ACT
(1) 011362 001401   BEQ  .+4
(1) 011364 104002   ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
(1) ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
(1)
8173 ;:THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1) ;:RECEIVER SECTION,IT USES THE ERROR FLAGS
(1) ;:TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1) ;:(OVRRUN,RXERR)
(1) ;:MODE:SYNEXT
(1) ;:LENGTH:SIX
(1) ;:CHAR:0
(1) ;:
(5) ;:*****
(4) 011366 000004   TST26: SCOPE
(4)
(3) 011370 052777 000400 170330   BIS   #MRESET,@TXCSR ;MASTER RESET
(2) 011376 012777 020000 170316   MOV   #SYNEXT,@PARCSR ;SET THE MODE
(3) 011404 052777 000400 170314   BIS   #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 011412 012777 064001 170306   MOV   #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 011420 012777 022000 170274   MOV   #SYNEXT!SIX!NOPAR!0,@PARCSR

```

```

(1) 011426 052777 000020 170256 BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 011434 042777 020000 170264 BIC #CLK,@TXCSR ;POKE CLK DOWN
(2) 011442 052777 020000 170256 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 011450 016703 170242 MOV RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 011454 012700 000000 MOV #0,R0 ;EXPECTED
(1) 011460 012767 000006 167434 MOV #6,SHIFT ;# OF SHIFTS
(1) 011466 012767 000000 170004 MOV #0,$TMP1 ;DATA CHAR
(1) 011474 004767 005322 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011500 105777 170206 TSTB @RXCSR ;RXDONE ?
(1) 011504 100401 BMI ;+4
(1) 011506 104004 ERROR 4 ;RXDONE SHOULD BE SET
(1) 011510 017701 170202 MOV @RXDBUF,R1 ;ACTUAL
(1) 011514 020001 CMP R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 011516 001401 BEQ ;+4
(1) 011520 104002 ERROR 2 ;RECEIVED DATA DID NOT MATCH
(1) ;EXPECTED DATA - CHECK MAINT DATA
(1) ;OR RECEIVER LOGIC
(1) 011522 012767 000006 167372 MOV #6,SHIFT ;# OF SHIFTS
(1) 011530 012767 000000 167742 MOV #0,$TMP1 ;DATA CHAR
(1) 011536 004767 005260 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 011542 012767 000006 167352 MOV #6,SHIFT ;# OF SHIFTS
(1) 011550 012767 000000 167722 MOV #0,$TMP1 ;DATA CHAR
(1) 011556 004767 005240 JSR PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011562 012700 140000 MOV #140000!0,R0 ;EXPECTED DATA PLUS
(1) ;RXERR & OVRRUN
(1) 011566 017701 170124 MOV @RXDBUF,R1 ;ACTUAL
(1) 011572 020001 CMP R0,R1 ;COMPARE EXP VS. ACT
(1) 011574 001401 BEQ ;+4
(1) 011576 104002 ERROR 2 ;SPECIFICALLY LOOK AT RXERR &
(1) ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
(1)
8174 ;: THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1) ;: RECEIVER SECTION, IT USES THE ERROR FLAGS
(1) ;: TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1) ;: (OVRRUN, RXERR)
(1) ;: MODE: SYNEXT
(1) ;: LENGTH: SEVEN
(1) ;: CHAR: 125
(1) ;:
(5) ;:*****
(4) 011600 000004 TST27: SCOPE
(4)
(3) 011602 052777 000400 170116 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 011610 012777 020000 170104 MOV #SYNEXT,@PARCSR ;SET THE MODE
(3) 011616 052777 000400 170102 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 011624 012777 064001 170074 MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 011632 012777 024000 170062 MOV #SYNEXT!SEVEN!NOPAR!0,@PARCSR
(1) 011640 052777 000020 170044 BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 011646 042777 020000 170052 BIC #CLK,@TXCSR ;POKE CLK DOWN

```

```

(2) 011654 052777 020000 170044      BIS      #CLK,@TXCSR      ;POKE CLK UP
(1) 011662 016703 170030              MOV      RXDBUF,R3      ;SET UP FOR ERROR MESSAGE
(1) 011666 012700 000125              MOV      #125,R0      ;EXPECTED
(1) 011672 012767 000007 167222      MOV      #7,SHIFT      ;# OF SHIFTS
(1) 011700 012767 000125 167572      MOV      #125,$TMP1     ;DATA CHAR
(1) 011706 004767 005110              JSR      PC,RPOKE      ;SHIFT IN THIS CHAR
(1) 011712 105777 167774              TSTB    @RXCSR ;RXDONE ?
(1) 011716 100401                      BMI      +4
(1) 011720 104004                      ERROR   4 ;RXDONE SHOULD BE SET
(1) 011722 017701 167770              MOV      @RXDBUF,R1     ;ACTUAL
(1) 011726 020001                      CMP      R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 011730 001401                      BEQ     +4
(1) 011732 104002                      ERROR   2 ;RECEIVED DATA DID NOT MATCH
(1)                                     ;EXPECTED DATA - CHECK MAINT DATA
(1)                                     ;OR RECEIVER LOGIC
(1) 011734 012767 000007 167160      MOV      #7,SHIFT      ;# OF SHIFTS
(1) 011742 012767 000125 167530      MOV      #125,$TMP1     ;DATA CHAR
(1) 011750 004767 005046              JSR      PC,RPOKE      ;SHIFT IN THIS CHAR
(1)                                     ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 011754 012767 000007 167140      MOV      #7,SHIFT      ;# OF SHIFTS
(1) 011762 012767 000125 167510      MOV      #125,$TMP1     ;DATA CHAR
(1) 011770 004767 005026              JSR      PC,RPOKE      ;SHIFT IN THIS CHAR
(1) 011774 012700 140125              MOV      #140000!125,R0 ;EXPECTED DATA PLUS
(1)                                     ;RXERR & OVRRUN
(1) 012000 017701 167712              MOV      @RXDBUF,R1     ;ACTUAL
(1) 012004 020001                      CMP      R0,R1 ;COMPARE EXP VS. ACT
(1) 012006 001401                      BEQ     +4
(1) 012010 104002                      ERROR   2 ;SPECIFICALLY LOOK AT RXERR &
(1)                                     ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
8175                                     ;; THIS TEST VERIFYS WORD LENGTH SELECT OF THE
(1)                                     ;; RECEIVER SECTION, IT USES THE ERROR FLAGS
(1)                                     ;; TO DETERMINE THAT IT WAS SELECTED CORRECTLY
(1)                                     ;; (OVRRUN,RXERR)
(1)                                     ;; MODE:SYNEXT
(1)                                     ;; LENGTH:SEVEN
(1)                                     ;; CHAR:52
(1)                                     ;;
(5)                                     ;;*****
(4) 012012 000004      ;TST30: SCOPE
(4)
(3) 012014 052777 000400 167704      BIS      #MRESET,@TXCSR ;MASTER RESET
(2) 012022 012777 020000 167672      MOV      #SYNEXT,@PARCSR ;SET THE MODE
(3) 012030 052777 000400 167670      BIS      #MRESET,@TXCSR ;MASTER RESET
(2)
(2)                                     ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 012036 012777 064001 167662      MOV      #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)                                     ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 012044 012777 024000 167650      MOV      #SYNEXT!SEVEN!NOPAR!0,@PARCSR
(1) 012052 052777 000020 167632      BIS      #SYNSCH,@RXCSR ;SET SEARCH SYNC
(2)                                     ;POKE CLK TO GET LOGIC INTO SYNCHRONIZATION
(2) 012060 042777 020000 167640      BIC      #CLK,@TXCSR    ;POKE CLK DOWN
(2) 012066 052777 020000 167632      BIS      #CLK,@TXCSR    ;POKE CLK UP
(1) 012074 016703 167616              MOV      RXDBUF,R3      ;SET UP FOR ERROR MESSAGE
(1) 012100 012700 000052              MOV      #52,R0 ;EXPECTED

```

```

(1) 012104 012767 000007 167010      MOV      #7,SHIFT      ;# OF SHIFTS
(1) 012112 012767 000052 167360      MOV      #52,$TMP1     ;DATA CHAR
(1) 012120 004767 004676              JSR      PC,RPOKE      ;SHIFT IN THIS CHAR
(1) 012124 105777 167562              TSTB    @RXCSR ;RXDONE ?
(1) 012130 100401                      BMI      +4
(1) 012132 104004                      ERROR   4 ;RXDONE SHOULD BE SET
(1) 012134 017701 167556              MOV      @RXDBUF,R1    ;ACTUAL
(1) 012140 020001                      CMP      R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 012142 001401                      BEQ      +4
(1) 012144 104002                      ERROR   2 ;RECEIVED DATA DID NOT MATCH
(1)                                     ;EXPECTED DATA - CHECK MAINT DATA
(1)                                     ;OR RECEIVER LOGIC
(1) 012146 012767 000007 166746      MOV      #7,SHIFT      ;# OF SHIFTS
(1) 012154 012767 000052 167316      MOV      #52,$TMP1     ;DATA CHAR
(1) 012162 004767 004634              JSR      PC,RPOKE      ;SHIFT IN THIS CHAR
(1)                                     ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 012166 012767 000007 166726      MOV      #7,SHIFT      ;# OF SHIFTS
(1) 012174 012767 000052 167276      MOV      #52,$TMP1     ;DATA CHAR
(1) 012202 004767 004614              JSR      PC,RPOKE      ;SHIFT IN THIS CHAR
(1) 012206 012700 140052              MOV      #14000!52,R0  ;EXPECTED DATA PLUS
(1)                                     ;RXERR & OVRRUN
(1) 012212 017701 167500              MOV      @RXDBUF,R1    ;ACTUAL
(1) 012216 020001                      CMP      R0,R1 ;COMPARE EXP VS. ACT
(1) 012220 001401                      BEQ      +4
(1) 012222 104002                      ERROR   2 ;SPECIFICALLY LOOK AT RXERR &
(1)                                     ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
(1) 8176
(2)                                     ;END OF PASS
(2)                                     ;TYPE NAME OF TEST
(2)                                     ;UPDATE PASS COUNT
(2)                                     ;CHECK FOR EXIT TO ACT-11
(2)                                     ;RESTART TEST
(2)
(2) 012224 000004                      .EOP:  SCOPE
(2) 012226 004767 000340              JSR      PC,CKSWR
(2) 012232 104401                      TYPE                                     ;TYPE NAME OF TEST
(2) 012234 015362                      MEPASS
(2) 012236 104413 012470              CONVRT  ,OUTCRY
(2) 012242 104401 015201              TYPE   ,DEVICE
(2) 012246 105767 166700              TSTB    MULTD ;ARE YOU RUNNING MULTIPLE DEVICES ?
(2) 012252 001511                      BEQ      CCC ;NO, JUMP AROUND
(2) 012254 005767 166706              TST     ACTREG ;ARE ANY DEVICES ACTIVE ?
(2) 012260 001007                      BNE     RUNIT ;YES
(2) 012262 104401 015213              TYPE   ,MCOV ;NO
(2) 012266 016700 166674              MOV      ACTREG,R0 ;DISPLAY ACTREG
(2) 012272 000000                      HALT    ;SELECT SOMETHING TO RUN @ ACTREG:
(2)                                     ;SELECT SWITCHES & HIT CONTINUE (PUT SW00 =1)
(2) 012274 000167 167652              JMP     .START ;START OVER AGAIN.....YOU DESELECTED EVERYTHING
(2) 012300 062767 000010 166646 RUNIT: ADD #10,BASEADD ;NEXT BLOCK (ADDRESSES)
(2) 012306 062767 000010 166646 ZERO:  ADD #10,BASEIV ;NEXT BLOCK (VECTORS)
(2) 012314 000241                      CLC
(2) 012316 006167 166646              ROL     ROTADD ;UP DATE ROTATING POINTER
(2) 012322 103410                      BCS    2$ ;IS IT THE LAST DEVICE
(2)                                     ;TO BE TESTED IN THIS PASS ?
(2) 012324 036767 166640 166634      BIT     ROTADD,ACTREG ;TEST THIS DEVICE FOR ACTIVE STATUS

```



CNDUR-A MACY11 30(1046) 14-DEC-82 09:58 PAGE 61-42  
 CNDURA.M11 29-OCT-82 16:31 INITIALIZE THE COMMON TAGS

SEQ 0056

```

(2) 012332 001762          BEQ    RUNIT      ;IF NOT ACTIVE, TRY NEXT ADDRESS
(2) 012334 004767 000034   JSR    PC,REPLAY  ;CALCULATE NEW PARAMETERS
(2) 012340 000167 000210   JMP    RESTRT    ;YES IT WAS ACTIVE, TEST THIS DEVICE
(2) 012344 012767 000001 166616 2$:  MOV    #1,ROTADD  ;OK!,NOW SET UP ROTATING
(2)                                ;POINTER FOR NEXT MULTIPLE PASS
(2) 012352 016767 166600 166574   MOV    KEEPADD,BASEADD ;RESTORE BASE ADDRESS
(2) 012360 016767 166600 166574   MOV    KEEPIV,BASEIV  ;RESTORE BASE INTERRUPT VECTORS
(2) 012366 004767 000002          JSR    PC,REPLAY  ;CALC NEW PARAMETERS
(2) 012372 000441          BR     CCC        ;JUMP AROUND REPLAY
(2) 012374 016767 166554 004416  REPLAY: MOV    BASEADD,DUBASE ;SET UP FOR NEW ADDRESSES
(2) 012402 004767 004260          JSR    PC,DUADDR   ;CREATE NEW ADDRESSES
(2) 012406 016767 166550 167322   MOV    BASEIV,DURIV  ;CREATE DURIV
(2) 012414 062767 000002 166540   ADD    #2,BASEIV
(2) 012422 016767 166534 167310   MOV    BASEIV,DURIS  ;CREATE DURIS
(2) 012430 062767 000002 166524   ADD    #2,BASEIV
(2) 012436 016767 166520 167276   MOV    BASEIV,DUTIV  ;CREATE DUTIV
(2) 012444 062767 000002 166510   ADD    #2,BASEIV
(2) 012452 016767 166504 1c7264   MOV    BASEIV,DUTIS  ;CREATE DUTIS
(2) 012460 016767 167252 166474   MOV    DURIV,BASEIV  ;RESTORE
(2) 012466 000207          RTS    PC
(2)
(2) 012470 000001          OUTCRY: 1
(2) 012472 006          .BYTE 6,2
(2) 012474 001712          RXCSR
(2)
(2) 01c476          CCC:
(2) 012476 005067 166700          CLR    $STSTM      ;CLEAR TEST NUMBER
(2) 012502 005067 166710          CLR    $ERRPC     ;CLEAR LAST ERROR PC
(2) 012506 005067 166671          CLR    $ERFLG    ;CLEAR ERROR FLAG
(2) 012512 005267 166374          INC    PASCNT     ;UPDATE PASS COUNT
(2) 012516 016767 166370 166356   MOV    PASCNT,LIGHTS ;DISPLAY PASS COUNT
(2) 012524 016767 166362 167002   MOV    PASCNT,$PASS ;PASS COUNT TO APT
(2) 012532 013701 000042          MOV    @#42,R1    ;CHECK FOR ACT-11 OR DDP
(2) 012536 001406          BEQ    RESTRT    ;IF NO CONTINUE TESTING
(2) 012540 000005          RESET
(2) 012542 000005          RESET
(2) 012544 004711          SENDAD: JSR    PC,(R1)
(2) 012546 000240          NOP
(2) 012550 000240          NOP
(2) 012552 000240          NOP
(2) 012554          RESTRT:
(2) 012554 012767 003376 166624   MOV    #TST1+2,$LPADR ;LOAD LAST ADDR
(2) 012562 004767 000004          JSR    PC,CKSWR
(2) 012566 000167 170516          JMP    .BEGIN
(2)
(2)                                ;CHECK SWITCH REGISTER ROUTINE.
(2)                                ;CHECKS TO ALLOW FOR <^G> TO ALLOW
(2)                                ;THE CHANGING OF LOCATION 176
(2)
(2) 012572 005737 000042          CKSWR: TST    @#42
(2) 012576 001040          BNE    OUT
(2) 012600 022767 000176 166632   CMP    #SWREG,SWR  ;SOFTWARE SWR PRESENT?
(2) 012606 001034          BNE    OUT        ;NO--LEAVE
(2) 012610 105777 166630          TSTB  @STKS       ;CHECK TTY READY
(2) 012614 100031          BPL    OUT        ;NO--LEAVE
(2) 012616 017767 166624 000422   MOV    @STKB,.MSG  ;GET CHARACTER

```

```
(2) 012624 042767 177600 000414      BIC      #177600,.MSG      ;STRIP JUNK
(2) 012632 122767 000007 000406      CMPB     #7,.MSG          ;IS IT <^G> ?
(2) 012640 001017                    BNE      OUT              ;NO
(2) 012642 104401 015767                    TYPE     ,MCNTG
(2) 012646 005137 012706      CNTLU:  COM      @#RDSW
(2) 012652 104401 015777                    TYPE     ,MMSWR
(2) 012656 104413                    CONVRT
(2) 012660 012710                    SWREGL
(2) 012662 104406 016010                    INSTR,MMNEW
(2) 012666 104410                    PARAM
(2) 012670 000000                    0
(2) 012672 177777                    177777
(2) 012674 000176                    SWREG
(2) 012676      000      001      .BYTE 0,1
(2) 012700 005037 012706      OUT:    CLR      @#RDSW
(2) 012704 000207                    RTS      PC
(2) 012706 000000                    RDSW:   .WORD   0
(2) 012710 000001                    SWREGL: 1
(2) 012712      006      002      .BYTE 6,2
(2) 012714 000176                    SWREG
(2) 012716 000005                    5
(2) ;CHECK FOR FREEZE ON CURRENT DATA
(2)
(2) 012720 004767 177646 .SCOPI: JSR      PC,CKSWR
(2) 012724 032777 001000 166506      BIT      #SW09,@SWR
(2) 012732 001402                    BEQ      1$
(2) 012734 016716 166150                    MOV      LOCK,(SP)
(2) 012740 000002                    1$:    RTI
(2) .SBTTL TYPE ROUTINE
(2)
(2) ;*****
(2) ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(2) ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(2) ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(2) ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(2) ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(2) ;*
(2) ;*CALL:
(2) ;*1) USING A TRAP INSTRUCTION
(2) ;*      TYPE     ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(2) ;*OR
(2) ;*      TYPE     MESADR
(2) ;*
(2)
(2) 012742 105767 166511 $TYPE: TSTB     $TPFLG      ;;IS THERE A TERMINAL?
(2) 012746 100002                    BPL      1$              ;;BR IF YES
(2) 012750 000000                    HALT
(2) 012752 000430                    BR      3$              ;;HALT HERE IF NO TERMINAL
(2) 012754 010046                    BR      3$              ;;LEAVE
(2) 012756 017600 000002 1$:    MOV      RO,-(SP)      ;;SAVE RO
(2) 012762 122767 000001 166556      MOV      @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
(2) 012770 001011                    CMPB     #APTENV,$ENV    ;;RUNNING IN APT MODE
(2) 012772 132767 000100 166547      BNE      62$            ;;NO,GO CHECK FOR APT CONSOLE
(2)                                BITB     #APTPOOL,$ENVM   ;;SPOOL MESSAGE TO APT
```

```

(2) 013000 001405      BEQ      62$      ;;NO,GO CHECK FOR CONSOLE
(2) 013002 010067 000004  MOV      RO,61$  ;;SETUP MESSAGE ADDRESS FOR APT
(2) 013006 004767 164774  JSR      PC,$ATY3 ;;SPOOL MESSAGE TO APT
(2) 013012 000000      .WORD    0        ;;MESSAGE ADDRESS
(2) 013014 132767 000040 166525 62$:  BITB    #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(2) 013022 001003      BNE      60$      ;;YES,SKIP TYPE OUT
(2) 013024 112046      2$:  MOVB   (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(2) 013026 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
(2) 013030 005726      TST     (SP)+    ;;IF TERMINATOR POP IT OFF THE STACK
(2) 013032 012600      60$:  MOV     (SP)+,0    ;;RESTORE RO
(2) 013034 062716 000002  3$:  ADD     #2,(SP)   ;;ADJUST RETURN PC
(2) 013040 000002      RTI                    ;;RETURN
(2) 013042 122716 000011  4$:  CMPB   #HT,(SP)   ;;BRANCH IF <HT>
(2) 013046 001430      BEQ      8$      ;;BRANCH IF NOT <CRLF>
(2) 013050 122716 000200  CMPB   #CRLF,(SP)
(2) 013054 001006      BNE      5$      ;;POP <CR><LF> EQUIV
(2) 013056 005726      TST     (SP)+    ;;TYPE A CR AND LF
(2) 013060 104401      TYPE
(2) 013062 001523      $CRLF
(2) 013064 105067 000130  CLRB   $CHARCNT  ;;CLEAR CHARACTER COUNT
(2) 013070 000755      BR      2$      ;;GET NEXT CHARACTER
(2) 013072 004767 000056  5$:  JSR     PC,$TYPEC  ;;GO TYPE THIS CHARACTER
(2) 013076 126726 166354  6$:  CMPB   $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(2) 013102 001350      BNE      2$      ;;IF NO GO GET NEXT CHAR.
(2) 013104 016746 166344  MOV     $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
(2)                                ;;AND THE NULL CHAR.
(2) 013110 105366 000001  7$:  DECB   1(SP)     ;;DOES A NULL NEED TO BE TYPED?
(2) 013114 002770      BLT     6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
(2) 013116 004767 000032  JSR     PC,$TYPEC  ;;GO TYPE A NULL
(2) 013122 105367 000072  DECB   $CHARCNT  ;;DO NOT COUNT AS A COUNT
(2) 013126 000770      BR      7$      ;;LOOP
(2)
(2)                                ;HORIZONTAL TAB PROCESSOR
(2)
(2) 013130 112716 000040  8$:  MOVB   #' ,(SP)  ;;REPLACE TAB WITH SPACE
(2) 013134 004767 000014  9$:  JSR     PC,$TYPEC  ;;TYPE A SPACE
(2) 013140 132767 000007 000052  BITB   #7,$CHARCNT ;;BRANCH IF NOT AT
(2) 013146 001372      BNE      9$      ;;TAB STOP
(2) 013150 005726      TST     (SP)+    ;;POP SPACE OFF STACK
(2) 013152 000724      BR      2$      ;;GET NEXT CHARACTER
(2) 013154 105777 166270  $TYPEC: TSTB   @STPS  ;;WAIT UNTIL PRINTER IS READY
(2) 013160 100375      BPL     $TYPEC
(2) 013162 116677 000002 166262  MOVB   2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(2)
(2) 013170 122766 000015 000002  CMPB   #CR,2(SP)   ;;IS CHARACTER A CARRIAGE RETURN?
(2) 013176 001003      BNE      1$      ;;BRANCH IF NO
(2) 013200 105067 000014      CLRB   $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
(2) 013204 000406      BR      $TYPEX   ;;EXIT
(2) 013206 122766 000012 000002  1$:  CMPB   #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
(2) 013214 001402      BEQ     $TYPEX   ;;BRANCH IF YES
(2) 013216 105227      INCB   (PC)+    ;;COUNT THE CHARACTER
(2) 013220 000000      $CHARCNT: .WORD  0 ;;CHARACTER COUNT STORAGE
(2) 013222 000207      $TYPEX: RTS     PC
(2)
(2)
(2)

```

```

(2)
(2)
(2)
(2) 013224 017667 000000 000014 .INSTR: MOV @ (SP), .MSG ; PICK UP MESSAGE
(2) 013232 062716 000002 ADD #2, (SP) ; JUMP AROUND MESSAGE FOR RTI
(2) 013236 105767 166304 TSTB $ENV ; APT CONTROL
(2) 013242 001036 BNE INSTR2 ; YES NO TYPE
(2) 013244 104401 .INST1: TYPE
(2) 013246 000000 .MSG: 0
(2) 013250 012704 016022 MOV #INBUF, R4 ; GET STARTING LOC OF INBUF
(2) 013254 012703 000007 MOV #7, R3 ; MAX # OF CHARS
(2) 013260 105777 166160 1$: TSTB @STKS ; TTY FLAG
(2) 013264 100375 BPL 1$
(2) 013266 117714 166154 MOVB @STKB, (R4) ; TAKE CHAR
(2) 013272 142714 000200 BICB #200, (R4) ; STRIP
(2) 013276 121427 000025 CMPB (R4), #25 ; IS IT <^G>
(2) 013302 001760 BEQ .INST1
(2) 013304 122427 000015 CMPB (R4)+, #15 ; CHECK FOR CR
(2) 013310 001413 BEQ INSTR2
(2) 013312 105777 166132 2$: TSTB @STPS ; TEST FLAG
(2) 013316 100375 BPL 2$
(2) 013320 117777 166122 166124 MOVB @STKB, @STPB ; ECHO CHARACTER
(2) 013326 005303 DEC R3 ; DID YOU TYPE TOO MANY CHARS ?
(2) 013330 001353 BNE 1$
(2) 013332 104401 .INSTE: TYPE
(2) 013334 015307 MQM ;?
(2) 013336 000742 BR .INST1 ; RETRY
(2) 013340 000002 INSTR2: RTI

(2)
(2)
(2)
(2)
(2) 013342 011605 .PARAM: MOV (SP), R5 ; PUT CONTENTS OF SP INTO R5
(2) 013344 012567 000162 MOV (R5)+, LOLIM ; PUT LOW LIMIT INTO LOLIM
(2) 013350 012567 000160 MOV (R5)+, HILIM ; PUT HIGH LIMIT INTO HILIM
(2) 013354 012567 000156 MOV (R5)+, DEVADR ; PUT STORE LOC INTO DEVADR
(2) 013360 112567 000154 MOVB (R5)+, LOBITS ; PUT MASK INTO LOBITS
(2) 013364 112567 000151 MOVB (R5)+, ADRCNT ; PUT COUNT INTO ADRCNT
(2) 013370 010516 MOV R5, (SP) ; RESTORE RETURN ADDR ON STACK FOR RTI
(2) 013372 005005 PARAM1: CLR R5
(2) 013374 012704 016022 MOV #INBUF, R4
(2) 013400 122714 000015 CMPB #15, (R4) ; CR ?
(2) 013404 001420 BEQ PARERR ; YOU TYPED CR TOO SOON !
(2) 013406 121427 000060 1$: CMPB (R4), #60 ; LOW LIMIT ASCII 0
(2) 013412 002415 BLT PARERR
(2) 013414 121427 000067 CMPB (R4), #67 ; HIGH LIMIT ASCII 7
(2) 013420 003012 BGT PARERR
(2) 013422 142714 000060 BICB #60, (R4) ; CONVERT TO OCTAL
(2) 013426 152405 BISB (R4)+, R5 ; STORE AWAY ITS AN OK CHAR
(2) 013430 122714 000015 CMPB #15, (R4) ; CR ?
(2) 013434 001414 BEQ LIMITS ; NOW CHECK FOR HIGH & LOW LIMIT CONDS
(2) 013436 006305 ASL R5 ; ALLOCATE ROOM FOR NEXT CHAR
(2) 013440 006305 ASL R5
(2) 013442 006305 ASL R5
(2) 013444 000760 BR 1$
(2) 013446 122714 000015 PARERR: CMPB #15, (R4) ; CR?
(2) 013452 001003 BNE 120$

```

```
(2) 013454 005737 012706          TST      @#RDSW          ;CK SWR USED
(2) 013460 001023                    BNE     PARTI
(2) 013462 104407          120$:  INSTER ;RETRY
(2) 013464 000742                    BR      PARAM1
(2)
(2)                                ;TEST TO SEE IF NUMBER IS WITHIN LIMITS
(2) 013466 020567 000042          LIMITS: CMP     R5,HILIM
(2) 013472 101365                    BHI     PARERR ;THE # IS TOO HIGH
(2) 013474 020567 000032          CMP     R5,LOLIM
(2) 013500 103762                    BLO     PARERR ;THE # IS TOO LOW
(2) 013502 136705 000032          BITB   LOBITS,R5 ;TEST BY MASKINGTHE #
(2) 013506 001357                    BNE     PARERR
(2)
(2)                                ;STORE NUMBER AT SPECIFIED ADDRESS
(2) 013510 016704 000022          1$:   MOV     DEVADR,R4 ;GET STARTING ADDR OF
(2) 013514 010524                    MOV     R5,(R4)+ ;STORE AT THIS ADDR
(2) 013516 062705 000002          ADD     #2,R5
(2) 013522 105367 000013          DECB   ADRCNT ;HOW MANY TIMES + 2 ?
(2) 013526 001372                    BNE     1$
(2) 013530 000002          PARTI: RTI
(2) 013532 000000          LOLIM: 0
(2) 013534 000000          HILIM: 0
(2) 013536 000000          DEVADR: 0
(2) 013540 000000          LOBITS: 0
(2)                                ADRCNT=LOBITS+1
(2)
(2)                                ;SAVE PC OF TEST THAT FAILED AND R0-R5
(2) 013542 016667 000004 165356 .SAV05: MOV     4(SP),SAVPC
(2)
(2)                                ;SAVE R0-R5
(2) 013550 010567 165720          SV05:  MOV     R5,$REG5
(2) 013554 010467 165712          MOV     R4,$REG4
(2) 013560 010367 165704          MOV     R3,$REG3
(2) 013564 010267 165676          MOV     R2,$REG2
(2) 013570 010167 165670          MOV     R1,$REG1
(2) 013574 010067 165662          MOV     R0,$REG0
(2) 013600 000002          RTI
(2)
(2)                                ;RESTORE R0-R5
(2) 013602 016700 165654          .RES05: MOV     $REG0,R0
(2) 013606 016701 165652          MOV     $REG1,R1
(2) 013612 016702 165650          MOV     $REG2,R2
(2) 013616 016703 165646          MOV     $REG3,R3
(2) 013622 016704 165644          MOV     $REG4,R4
(2) 013626 016705 165642          MOV     $REG5,R5
(2) 013632 000002          RTI
(2)
(2)                                ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
(2) 013634 104401          .CONVR: TYPE
(2) 013636 015313          MCRLF ;CR LF
```

```

(2) 013640 017601 000000      MOV      @ (SP), R1      ; PICK UP DATA POINTER
(2) 013644 062716 000002      ADD      #2, (SP)      ; SET UP SP FOR RTI
(2) 013650 012167 000130      MOV      (R1)+, WRDCNT ; PICK UP # OF WORDS FROM TABLE
(2) 013654 112167 000126      1$:     MOV      (R1)+, CHRCNT ; PICK UP # OF CHARS FROM TABLE
(2) 013660 112167 000123      MOV      (R1)+, SPACNT ; PICK UP # OF SPACES FROM TABLE
(2) 013664 013167 000120      MOV      @ (R1)+, BINWRD ; PICK UP ADDRESS OF MSG
(2)                                ; FROM TABLE
(2) 013670 016704 000114      2$:     MOV      BINWRD, R4 ; SAVE
(2) 013674 116705 000106      MOV      CHRCNT, R5    ; SAVE
(2) 013700 012700 016064      MOV      #TEMP, R0    ; STARTING ADDRESS OF TEMP BLOCK
(2) 013704 010403 3$:     MOV      R4, R3      ; SAVE
(2) 013706 042703 177770      BIC      #177770, R3   ; CLR OUT UPPER BITS .. SAVE CHAR
(2) 013712 062703 000260      ADD      #260, R3     ; CONVERT TO ASCII
(2) 013716 110320      MOV      R3, (R0)+    ; STORE AWAY
(2) 013720 006204      ASR      R4          ; SHIFT FOR NEXT #
(2) 013722 006204      ASR      R4          ; DITTO
(2) 013724 006204      ASR      R4          ; DITTO
(2) 013726 005305      DEC      R5          ; DEC CHAR COUNT
(2) 013730 001365      BNE      3$         ; DO IT AGAIN ?
(2) 013732 012703 016126      MOV      #MDATA, R3   ; STARTING ADDRESS OF MDATA BLOCK
(2) 013736 114023 4$:     MOV      -(R0), (R3)+  ; REVERSE THE ORDER OF NUMBERS
(2) 013740 105367 000042      DECB    CHRCNT      ; DEC CHAR COUNT
(2) 013744 001374      BNE      4$         ; DO IT AGAIN ?
(2) 013746 105767 000035      TSTB    SPACNT     ; HOW MANY SPACES ?
(2) 013752 001405      BEQ      6$         ; TYPE # IF BR = 0
(2) 013754 112723 000240      5$:     MOV      #240, (R3)+ ; "SPACE" IN ASCII
(2) 013760 105367 000023      DECB    SPACNT     ; DEC # OF SPACE COUNT
(2) 013764 001373      BNE      5$         ; DO IT AGAIN ?
(2) 013766 105013 6$:     CLRB    (R3)       ; INSERT '0' FOR TTY OUTPUT ROUTINE
(2) 013770 104401      TYPE
(2) 013772 016126      MDATA      ; THIS MESSAGE
(2) 013774 005367 000004      DEC      WRDCNT     ; HOW MANY #'S ?
(2) 014000 001325      BNE      1$         ; DO THIS ROUTINE AGAIN IF NOT EQUAL TO 0
(2) 014002 000002      RTI      ; RETURN TO PROGRAM
(2) 014004 000000      WRDCNT: 0
(2) 014006 000000      CHRCNT: 0
(2)                                SPACNT=CHRCNT+1
(2) 014010 000000      BINWRD: 0
(2)                                ; COMPARE THE FIRST CHARACTER IN THE TELETYPE INPUT
(2)                                ; BUFFER TO THE CHARACTERS 'N' AND 'Y'.
(2)                                ; IF THE CHARACTER IS 'N' CLEAR THE FLAG
(2)                                ; IF THE CHARACTER IS 'Y' SET THE FLAG
(2) 014012 017605 000000      .SETFLG: MOV      @ (SP), R5
(2) 014016 122767 000116 001776      CMPB    #'N', INBUF   ; IS IT 'N' ?
(2) 014024 001002      BNE      1$
(2) 014026 105015      CLRB    (R5)        ; 000
(2) 014030 000406      BR      2$
(2) 014032 122767 000131 001762 1$:     CMPB    #'Y', INBUF   ; IS IT 'Y' ?
(2) 014040 001005      BNE      3$
(2) 014042 112715 177777      MOV      #-1, (R5)   ; 377
(2) 014046 062716 000002      2$:     ADD      #2, (SP)
(2) 014052 000002      RTI
(2) 014054 104407      3$:     INSTER ; RETRY
(2) 014056 000755      BR      .SETFLG
  
```

```

(2)          .SBTTL  ERROR HANDLER ROUTINE
(2)
(3)          ;*****
(2)          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(2)          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(2)          ;*AND GO TO SAVIT ON ERROR
(2)          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(2)          ;*SW15=1      HALT ON ERROR
(2)          ;*SW13=1      INHIBIT ERROR TYPEOUTS
(2)          ;*SW10=1      BELL ON ERROR
(2)          ;*SW09=1      LOOP ON ERROR
(2)          ;*CALL
(2)          ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(2)          $ERROR:
(2) 014060      105267  165317      7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
(2) 014064      001775      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
(2) 014066      016777  165310  165346      MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(2) 014074      032777  002000  165336      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
(2) 014102      001402      BEQ      1$      ;;NO - SKIP
(2) 014104      104401  001516      TYPE      $BELL      ;;RING BELL
(2) 014110      005267  165276      1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
(2) 014114      011667  165276      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
(2) 014120      162767  000002  165270      SUB      #2,$ERRPC
(2) 014126      117767  165264  165260      MOV      @ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
(2) 014134      032777  020000  165276      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
(2) 014142      001004      BNE      20$      ;;SKIP TYPEOUTS
(2) 014144      004767  000072      JSR      PC,SAVIT      ;;GO TO USER ERROR ROUTINE
(2) 014150      104401  001523      TYPE      $CRLF
(2) 014154      122767  000001  165364      20$:      CMPB      #APTENV,$ENV      ;;RUNNING IN APT MODE
(2) 014162      001007      BNE      2$      ;;NO,SKIP APT ERROR REPORT
(2) 014164      116767  165224  000004      MOV      $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(2) 014172      004767  163620      JSR      PC,$ATY4      ;;REPORT FATAL ERROR TO APT
(2) 014176      000      21$:      .BYTE      0
(2) 014177      000      .BYTE      0
(2) 014200      000777      22$:      BR      22$      ;;APT ERROR LOOP
(2) 014202      005777  165232      2$:      TST      @SWR      ;;HALT ON ERROR
(2) 014206      100001      BPL      3$      ;;SKIP IF CONTINUE
(2) 014210      000000      HALT      ;;HALT ON ERROR!
(2) 014212      032777  001000  165220      3$:      BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
(2) 014220      001402      BEQ      4$      ;;BR IF NO
(2) 014222      016716  165162      MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
(2) 014226      005767  165262      4$:      TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
(2) 014232      001402      BEQ      5$      ;;BR IF NONE
(2) 014234      016716  165254      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
(2) 014240      000002      5$:      RTI      ;;RETURN
(2) 014242      010067  164662      SAVIT:  MOV      R0,HLD0
(2) 014246      010167  164660      MOV      R1,HLD1
(2) 014252      010267  164656      MOV      R2,HLD2
(2) 014256      010367  164654      MOV      R3,HLD3
(2) 014262      010467  164652      MOV      R4,HLD4
(2) 014266      010567  164650      MOV      R5,HLD5
(2) 014272      016767  165104  164644      MOV      $TSTNM,HLD6
(2)

```

(3)  
(3)  
(4)  
(3)  
(3)  
(3)  
(3)  
(3)  
(3) 014300  
(3) 014300 104401 001523  
(3) 014304 010046  
(3) 014306 005000  
(3) 014310 153700 001414  
(3) 014314 001004  
(3)  
(4) 014316 016746 165074  
(4)  
(4) 014322 104402  
(3) 014324 000426  
(3) 014326 005300  
(3) 014330 006300  
(3) 014332 006300  
(3) 014334 006300  
(3) 014336 062700 001652  
(3) 014342 012067 000004  
(3) 014346 001404  
(3) 014350 104401  
(3) 014352 000000  
(3) 014354 104401 001523  
(3) 014360 012067 000004  
(3) 014364 001404  
(3) 014366 104401  
(3) 014370 000000  
(3) 014372 104401 001523  
(3) 014376 011000  
(3) 014400 001004  
(3) 014402 012600  
(3)  
(3) 014404 104401 001523  
(3) 014410 000207  
(3) 014412  
(4) 014412 013046  
(4) 014414 104402  
(3) 014416 005710  
(3) 014420 001770  
(3) 014422 104401 014430  
(3) 014426 000771  
(3) 014430 020040 000  
(3) 014434  
(2)  
(2)  
(3)  
(2)  
(2)  
(2)  
(2)  
(2)

```
.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:
      TYPE      , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
      MOV      RO, -(SP)    ;; SAVE RO
      CLR      RO          ;; PICKUP THE ITEM INDEX
      BISB     @#$ITEMB, RO
      BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                          ;; TYPE THE PC OF THE ERROR
                          ;; SAVE $ERRPC FOR TYPEOUT
                          ;; ERROR ADDRESS
                          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
      MOV      $ERRPC, -(SP)
                          ;; GET OUT
                          ;; ADJUST THE INDEX SO THAT IT WILL
                          ;; WORK FOR THE ERROR TABLE
      TYPC     6$
      BR      1$
1$:   DEC      RO
      ASL     RO
      ASL     RO
      ASL     RO
      ADD     # $ERRTB, RO  ;; FORM TABLE POINTER
      MOV     (RO)+, 2$    ;; PICKUP "ERROR MESSAGE" POINTER
      BEQ     3$          ;; SKIP TYPEOUT IF NO POINTER
      TYPE   0           ;; TYPE THE "ERROR MESSAGE"
                          ;; "ERROR MESSAGE" POINTER GOES HERE
                          ;; "CARRIAGE RETURN" & "LINE FEED"
      .WORD  0           ;; "CARRIAGE RETURN" & "LINE FEED"
      MOV     (RO)+, 4$    ;; PICKUP "DATA HEADER" POINTER
      BEQ     5$          ;; SKIP TYPEOUT IF 0
      TYPE   0           ;; TYPE THE "DATA HEADER"
                          ;; "DATA HEADER" POINTER GOES HERE
                          ;; "CARRIAGE RETURN" & "LINE FEED"
      .WORD  0           ;; "CARRIAGE RETURN" & "LINE FEED"
      MOV     (RO), RO    ;; PICKUP "DATA TABLE" POINTER
      BNE     7$          ;; GO TYPE THE DATA
      MOV     (SP)+, RO   ;; RESTORE RO
      TYPE   , $CRLF     ;; "CARRIAGE RETURN" & "LINE FEED"
      RTS     PC         ;; RETURN
7$:   MOV     @ (RO)+, -(SP) ;; SAVE @ (RO)+ FOR TYPEOUT
      TYPC   6$         ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
      TST    (RO)       ;; IS THERE ANOTHER NUMBER?
      BEQ    6$         ;; BR IF NO
      TYPE  8$         ;; TYPE TWO(2) SPACES
      BR    7$         ;; LOOP
8$:   .ASCIZ  / /       ;; TWO(2) SPACES
      .EVEN
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

:*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV     NUM, -(SP)      ;; NUMBER TO BE TYPED
```



```

(2)          : *      TYPOS          ::CALL FOR TYPEOUT
(2)          : *      .BYTE      N          ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(2)          : *      .BYTE      M          ::M=1 OR 0
(2)          : *                                     :::1=TYPE LEADING ZEROS
(2)          : *                                     :::0=SUPPRESS LEADING ZEROS
(2)          : *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(2)          : *$TYPOS OR $TYPOC
(2)          : *CALL:
(2)          : *      MOV        NUM,-(SP)      ::NUMBER TO BE TYPED
(2)          : *      TYPON          ::CALL FOR TYPEOUT
(2)          : *
(2)          : *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(2)          : *CALL:
(2)          : *      MOV        NUM,-(SP)      ::NUMBER TO BE TYPED
(2)          : *      TYPOC          ::CALL FOR TYPEOUT
(2) 014434 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ::PICKUP THE MODE
(2) 014440 116667 000001 000211  MOVB     1(SP), $OFILL      ::LOAD ZERO FILL SWITCH
(2) 014446 112667 000207          MOVB     (SP)+, $OMODE+1    ::NUMBER OF DIGITS TO TYPE
(2) 014452 062716 000002          ADD      #2, (SP)        ::ADJUST RETURN ADDRESS
(2) 014456 000406          BR      $TYPON
(2) 014460 112767 000001 000171  $TYPOC: MOVB     #1, $OFILL      ::SET THE ZERO FILL SWITCH
(2) 014466 112767 000006 000165  MOVB     #6, $OMODE+1    ::SET FOR SIX(6) DIGITS
(2) 014474 112767 000005 000154  $TYPON: MOVB     #5, $OCNT      ::SET THE ITERATION COUNT
(2) 014502 010346          MOV      R3, -(SP)      ::SAVE R3
(2) 014504 010446          MOV      R4, -(SP)      ::SAVE R4
(2) 014506 010546          MOV      R5, -(SP)      ::SAVE R5
(2) 014510 116704 000145          MOVB     $OMODE+1, R4    ::GET THE NUMBER OF DIGITS TO TYPE
(2) 014514 005404          NEG      R4
(2) 014516 062704 000006          ADD      #6, R4          ::SUBTRACT IT FOR MAX. ALLOWED
(2) 014522 110467 000132          MOVB     R4, $OMODE      ::SAVE IT FOR USE
(2) 014526 116704 000125          MOVB     $OFILL, R4      ::GET THE ZERO FILL SWITCH
(2) 014532 016605 000012          MOV      12(SP), R5     ::PICKUP THE INPUT NUMBER
(2) 014536 005003          CLR      R3              ::CLEAR THE OUTPUT WORD
(2) 014540 006105          1$: ROL     R5              ::ROTATE MSB INTO 'C'
(2) 014542 000404          BR      3$              ::GO DO MSB
(2) 014544 006105          2$: ROL     R5              ::FORM THIS DIGIT
(2) 014546 006105          ROL     R5
(2) 014550 006105          ROL     R5
(2) 014552 010503          MOV      R5, R3
(2) 014554 006103          3$: ROL     R3              ::GET LSB OF THIS DIGIT
(2) 014556 105367 000076          DECB     $OMODE          ::TYPE THIS DIGIT?
(2) 014562 100016          BPL     7$              ::BR IF NO
(2) 014564 042703 177770          BIC     #177770, R3     ::GET RID OF JUNK
(2) 014570 001002          BNE     4$              ::TEST FOR 0
(2) 014572 005704          TST     R4              ::SUPPRESS THIS 0?
(2) 014574 001403          BEQ     5$              ::BR IF YES
(2) 014576 005204          4$: INC     R4              ::DON'T SUPPRESS ANYMORE 0'S
(2) 014600 052703 000060          BIS     #'0, R3         ::MAKE THIS DIGIT ASCII
(2) 014604 052703 000040          5$: BIS     #' , R3         ::MAKE ASCII IF NOT ALREADY
(2) 014610 110367 000040          MOVB     R3, 8$         ::SAVE FOR TYPING
(2) 014614 104401 014654          TYPE     8$              ::GO TYPE THIS DIGIT
(2) 014620 105367 000032          7$: DECB     $OCNT          ::COUNT BY 1
(2) 014624 003347          BGT     2$              ::BR IF MORE TO DO
(2) 014626 002402          BLT     6$              ::BR IF DONE
  
```

```

(2) 014630 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
(2) 014632 000744          BR       2$          ;;GO DO THE LAST DIGIT
(2) 014634 012605          6$: MOV    (SP)+,R5      ;;RESTORE R5
(2) 014636 012604          MOV    (SP)+,R4      ;;RESTORE R4
(2) 014640 012603          MOV    (SP)+,R3      ;;RESTORE R3
(2) 014642 016666 000002 000004 MOV    2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
(2) 014650 012616          MOV    (SP)+,(SP)
(2) 014652 000002          RTI
(2) 014654 000          8$: .BYTE 0          ;;RETURN
(2) 014655 000          .BYTE 0          ;;STORAGE FOR ASCII DIGIT
(2) 014656 000          $OCNT: .BYTE 0     ;;TERMINATOR FOR TYPE ROUTINE
(2) 014657 000          $OFILL: .BYTE 0    ;;OCTAL DIGIT COUNTER
(2) 014660 000000          $OMODE: .WORD 0    ;;ZERO FILL SWITCH
(2)                                ;NUMBER OF DIGITS TO TYPE
(2)                                ;ENTER HERE ON POWER FAILURE

(2) 014662          $PWRDN:
(2) 014662 010046          .PFAIL: MOV    R0,-(SP)      ;SAVE R0-R5 ON PROCESSOR STACK
(2) 014664 010146          MOV    R1,-(SP)
(2) 014666 010246          MOV    R2,-(SP)
(2) 014670 010346          MOV    R3,-(SP)
(2) 014672 010446          MOV    R4,-(SP)
(2) 014674 010546          MOV    R5,-(SP)
(2) 014676 016746 163122          MOV    24,-(SP)
(2) 014702 010667 164210          MOV    SP,SAVSP      ;SAVE STACK POINTER
(2) 014706 012767 014720 163110 MOV    #RESTART,24   ;SET UP FOR POWER UP TRAP
(2) 014714 000000          HALT
(2) 014716 000777          BR
(2)                                ;PROCESSOR WILL TRAP HERE WHEN POWER IS RESTORED

(2) 014720 016706 164172          RESTAR: MOV    SAVSP,SP      ;RESTORE STACK POINTER
(2) 014724 012605          MOV    (SP)+,R5      ;RESTORE R0-R5
(2) 014726 012604          MOV    (SP)+,R4
(2) 014730 012603          MOV    (SP)+,R3
(2) 014732 012602          MOV    (SP)+,R2
(2) 014734 012601          MOV    (SP)+,R1
(2) 014736 012600          MOV    (SP)+,R0
(2) 014740 012767 014662 163056 MOV    #.PFAIL,24    ;SET UP FOR POWER FAILURE
(2) 014746 106427 000300          MTPS    #300
(2) 014752 012706 001100          MOV    #STACK,SP
(2) 014756 005067 001102          CLR    TEMP
(2) 014762 005267 001076          INC    TEMP
(2) 014766 001375          BNE    -4
(2) 014770 104413          CONVRT
(2) 014772 015014          PFTAB
(2) 014774 104401          TYPE
(2) 014776 015316          MPFAIL
(2) 015000 005067 164377          CLR    $ERFLG
(2) 015004 005067 164406          CLR    $ERRPC
(2) 015010 000177 164070          JMP    @RETURN
(2) 015014 000001          PFTAB: 1
(2) 015016 006 002          .BYTE 6,2
(2) 015020 000207          RETURN
(2) 015022 005015 042012 053125 MTITLE: .ASCIZ <15><12><12>/DUV11 CNDUR-A TAPE B /<15><12>
(2) 015030 030461 041440 042116
  
```

(2)	015036	051125	040455	052040	
(2)	015044	050101	020105	020102	
(2)	015052	005015	000		
(2)	015055	015	053012	041505	MVECTO: .ASCIZ <15><12>/VEC ADD-/
(2)	015062	040440	042104	000055	
(2)	015070	005015	051461	020124	MREGAD: .ASCIZ <15><12>/1ST DEV: REC CSR ADD-/
(2)	015076	042504	035126	051040	
(2)	015104	041505	041440	051123	
(2)	015112	040440	042104	000055	
(2)	015120	005015	052515	052114	MMULT: .ASCIZ <15><12>/MULT DEV ? (Y OR N)-/
(2)	015126	042040	053105	037440	
(2)	015134	024040	020131	051117	
(2)	015142	047040	026451	000	
(2)	015147	015	046012	051501	MLASTD: .ASCIZ <15><12>/LAST DEV: REC CSR ADDR-/
(2)	015154	020124	042504	035126	
(2)	015162	051040	041505	041440	
(2)	015170	051123	040440	042104	
(2)	015176	026522	000		
(2)	015201	075	042504	044526	DEVICE: .ASCIZ /=DEVICE /
(2)	015206	042503	020040	000	
(2)	015213	015	051412	046105	MCOV: .ASCIZ <15><12>/SELECT TO RUN @ACTREG/
(2)	015220	041505	020124	047524	
(2)	015226	051040	047125	040040	
(2)	015234	041501	051124	043505	
(2)	015242	000			
(2)	015243	015	047412	043126	MRANGE: .ASCIZ <15><12>/OVFLO:RETYPE LAST DEV RXCSR ADDS-/
(2)	015250	047514	051072	052105	
(2)	015256	050131	020105	040514	
(2)	015264	052123	042040	053105	
(2)	015272	051040	041530	051123	
(2)	015300	040440	042104	026523	
(2)	015306	000			
(2)	015307	040	037440	000	MQM: .ASCIZ / ?/
(2)	015313	015	000012		MCRIF: .ASCIZ <15><12>
(2)	015316	043120	044501	026114	MPFAIL: .ASCIZ /PFAIL, RESTART AT TEST IN PROGRESS/
(2)	015324	020040	042522	052123	
(2)	015332	051101	020124	052101	
(2)	015340	052040	051505	020124	
(2)	015346	047111	050040	047522	
(2)	015354	051107	051505	000123	
(2)	015362	005015	047105	020104	MEPASS: .ASCIZ <15><12>/END OF PASS TAPE B/
(2)	015370	043117	050040	051501	
(2)	015376	020123	040524	042520	
(2)	015404	041040	000		
(2)	015407	015	051012	000	MR: .ASCIZ <15><12>/R/
(2)	015413	015	052012	051505	MTSTPC: .ASCIZ <15><12>/TEST PC-/
(2)	015420	020124	041520	000055	
(2)	015426	005015	047514	045503	MLOCK: .ASCIZ <15><12>/LOCK ON TEST? (Y OR N)-/
(2)	015434	047440	020116	052040	
(2)	015442	051505	037524	024040	
(2)	015450	020131	051117	047040	
(2)	015456	026451	000		
(2)	015461	015	021412	047440	MSYNC: .ASCIZ <15><12>/# OF SYNC CHARS SELECTED ( 1 OR 2)-/
(2)	015466	020106	054523	041516	
(2)	015474	041440	040510	051522	
(2)	015502	051440	046105	041505	

```

(2) 015510 042524 020104 020050
(2) 015516 020061 051117 031040
(2) 015524 026451 000
(2) 015527 015 044412 020123 MWIRE6: .ASCIIZ <15><12>/IS SEC XMIT SWITCH E55-2 IN? (Y OR N)-/
(2) 015534 042523 020103 046530
(2) 015542 052111 051440 044527
(2) 015550 041524 020110 032505
(2) 015556 026465 020062 047111
(2) 015564 020077 054450 047440
(2) 015572 020122 024516 000055
(2) 015600 005015 051511 051440 MWIRE5: .ASCIIZ <15><12>/IS SEC REC SWITCH E55-3 IN? (Y OR N)-/
(2) 015606 041505 051040 041505
(2) 015614 051440 044527 041524
(2) 015622 020110 032505 026465
(2) 015630 020063 047111 020077
(2) 015636 054450 047440 020122
(2) 015644 024516 000055
(2) 015650 005015 051511 047440 MWIRE4: .ASCIIZ <15><12>/IS OPT CLR ENABLE SWITCH E55-1 IN? (Y OR N)-/
(2) 015656 052120 041440 051114
(2) 015664 042440 040516 046102
(2) 015672 020105 053523 052111
(2) 015700 044103 042440 032465
(2) 015706 030455 044440 037516
(2) 015714 024040 020131 051117
(2) 015722 047040 026451 000
(2) 015727 015 005012 031510 MEXTJ: .ASCIIZ <15><12><12>/H315 CONNECTOR ON ?(Y OR N)-/
(2) 015734 032461 041440 047117
(2) 015742 042516 052103 051117
(2) 015750 047440 020116 024077
(2) 015756 020131 051117 047040
(2) 015764 026451 000
(2) 015767 015 020012 043536 MCNTG: .ASCIIZ <15><12>/ ^G /
(2) 015774 020040 000
(2) 015777 040 053523 036522 MMSWR: .ASCIIZ / SWR= /
(2) 016004 020040 000040
(2) 016010 020040 047040 053505 MMNEW: .ASCIIZ / NEW= /
(2) 016016 020075 000040
(2)
(2)
(2)
(2) 016022 000000 INBUF: 0
(2) 016064 016064 .=. +40
(2) 016064 000000 TEMP: 0
(2) 016126 016126 .=. +40
(2) 016126 000000 MDATA: 0
(2) 016170 016170 .=. +40
(3)
(3) .SBTTL SCOPE HANDLER ROUTINE STARS
(3) ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(3) ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(3) ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(3) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(3) ;*SW14=1 LOOP ON TEST
(3) ;*SW11=1 INHIBIT ITERATIONS
(3) ;*SW09=1 LOOP ON ERROR
(3) ;*SW08=1 LOOP ON TEST IN SWR<7:0>

```

```

(3)          :*CALL
(3)          :*      SCOPE          ;;SCOPE=IOT
(3)          $SCOPE:
(5)          ;SCOPE LOOP AND INTERATION HANDLER
(5)          .SCOPE:
(5) 016170   JSR      PC,CKSWR
(5) 016170   CLR      $ERRPC          ;CLEAR LAST ERROR PC
(5) 016174   005067   174376   163216   CMP      #TST1+2,(SP)          ;IS SCOPE AT BEGINING OF TEST 1?
(5) 016200   022716   003376   BEQ      $XTSTR          ;YES NO LOOP.
(5) 016204   001422
(5) 016206   032777   040000   163224   TTST:   BIT      #BIT14,@SWR          ;THIS CODE IS FOR TESTING FOR BIT 14
(5) 016214   001412   BEQ      1$          ;ON LSI WHICH SYSMAC CANNOT HANDLE
(5) 016216   016767   163160   163162   MOV      $TSTNM,$LPADR
(5) 016224   000406   BR      1$
(5) 016226   105777   163212   TSTB    @TKS          ;KEYBOARD DONE?
(5) 016232   100123   BPL     $OVER          ;BR IF NO
(5) 016234   017766   163206   177776   MOV      @TKB,-2(SP)          ;CLEAR DONE BIT
(3) 016242   032777   040000   163170   1$:    BIT      #BIT14,@SWR          ;LOOP ON PRESENT TEST?
(3) 016250   001114   BNE     $OVER          ;YES IF SW14=1
(3)          ;#####START OF CODE FOR THE XOR TESTER#####
(3) 016252   000416   $XTSTR: BR      6$          ;IF RUNNING ON THE 'XOR' TESTER CHANGE
(3)          MOV      @ERRVEC,-(SP)          ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
(3) 016254   013746   000004   000004   MOV      #5,@ERRVEC          ;SAVE THE CONTENTS OF THE ERROR VECTOR
(3) 016260   012737   016300   TST     @177060          ;SET FOR TIMEOUT
(3) 016266   005737   177060   TST     (SP)+,@ERRVEC          ;TIME OUT ON XOR?
(3) 016272   012637   000004   MOV      $SVLAD          ;RESTORE THE ERROR VECTOR
(3) 016276   000463   BR      5$          ;GO TO THE NEXT TEST
(3) 016300   022626   5$:    CMP      (SP)+,(SP)+          ;CLEAR THE STACK AFTER A TIME OUT
(3) 016302   012637   000004   MOV      (SP)+,@ERRVEC          ;RESTORE THE ERROR VECTOR
(3) 016306   000423   BR      7$          ;LOOP ON THE PRESENT TEST
(3) 016310   6$:    ;#####END OF CODE FOR THE XOR TESTER#####
(3) 016310   032777   000400   163122   BIT      #BIT08,@SWR          ;LOOP ON SPEC. TEST?
(3) 016316   001404   BEQ      2$          ;BR IF NO
(3) 016320   127767   163114   163054   CMPB    @SWR,$TSTNM          ;ON THE RIGHT TEST? SWR<7:0>
(3) 016326   001465   BEQ     $OVER          ;BR IF YES
(3) 016330   105767   163047   2$:    TSTB    $ERFLG          ;HAS AN ERROR OCCURRED?
(3) 016334   001421   BEQ      3$          ;BR IF NO
(3) 016336   126767   163053   163037   CMPB    $ERMAX,$ERFLG          ;MAX. ERRORS FOR THIS TEST OCCURRED?
(3) 016344   101015   BHI      3$          ;BR IF NO
(3) 016346   032777   001000   163064   BIT      #BIT09,@SWR          ;LOOP ON ERROR?
(3) 016354   001404   BEQ      4$          ;BR IF NO
(3) 016356   016767   163026   163022   7$:    MOV      $LPERR,$LPADR          ;SET LOOP ADDRESS TO LAST SCOPE
(3) 016364   000446   BR      $OVER
(3) 016366   105067   163011   4$:    CLRB    $ERFLG          ;ZERO THE ERROR FLAG
(3) 016372   005067   163114   CLR     $TIMES          ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(3) 016376   000415   BR      1$          ;ESCAPE TO THE NEXT TEST
(3) 016400   032777   004000   163032   3$:    BIT      #BIT11,@SWR          ;INHIBIT ITERATIONS?
(3) 016406   001011   BNE     1$          ;BR IF YES
(3) 016410   005767   163120   TST     $PASS          ;IF FIRST PASS OF PROGRAM
(3) 016414   001406   BEQ      1$          ;INHIBIT ITERATIONS
(3) 016416   005267   162762   INC     $ICNT          ;INCREMENT ITERATION COUNT
(3) 016422   026767   163064   162754   CMP     $TIMES,$ICNT          ;CHECK THE NUMBER OF ITERATIONS MADE
(3) 016430   002024   BGE     $OVER          ;BR IF MORE ITERATION REQUIRED
  
```

```

(3) 016432 012767 000001 162744 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(3) 016440 016767 000056 163044 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(3) 016446 105267 162730 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
(3) 016452 116767 162724 163052 MOV $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(3) 016460 011667 162722 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(3) 016464 011667 162720 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(3) 016470 005067 163020 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(3) 016474 112767 000001 162713 MOV #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(3) 016502 016777 162674 162732 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
(3) 016510 016716 162672 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(5) 016514 000002 4$: RTI
(5) 016516 001407 BRW: 1407
(5) 016520 000432 BRX: 432
(3) 016522 000005 $MXCNT: 5 ;;MAX. NUMBER OF ITERATIONS

```

(2) .SBTTL TRAP DECODER

```

(3) *****
(2) *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(2) *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(2) *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(2) *GO TO THAT ROUTINE.

```

```

(2) 01652. C10046 $TRAP: MOV RO,-(SP) ;;SAVE RO
(2) 016526 016600 000002 MOV 2(SP),RO ;;GET TRAP ADDRESS
(2) 016532 005740 TST -(RO) ;;BACKUP BY 2
(2) 016534 111000 MOV (RO),RO ;;GET RIGHT BYTE OF TRAP
(2) 016536 006300 ASL RO ;;POSITION FOR INDEXING
(2) 016540 016000 016560 MOV $TRPAD(RO),RO ;;INDEX TO TABLE
(2) 016544 000200 RTS RO ;;GO TO ROUTINE

```

(2) ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

```

(2) 016546 011646 $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
(2) 016550 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
(2) 016556 000002 RTI ;;RESTORE THE PSW

```

(4) .SBTTL TRAP TABLE

```

(4) *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(4) *BY THE "TRAP" INSTRUCTION.

```

```

(4) : ROUTINE
(4) : -----
(4) $TRPAD: .WORD $TRAP2
(4) 016560 016546 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
(4) 016562 012742 $TYPC ;;CALL=TYPC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(4) 016564 014460 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(4) 016566 014434 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(2)
(2)
(3) 016572 012720 .SCOPI ;;CALL=SCOPI TRAP+5(104405)
(3) 016574 013224 .INSTR ;;CALL=INSTR TRAP+6(104406)
(3) 016576 013332 .INSTER ;;CALL=INSTER TRAP+7(104407)
(3) 016600 013342 .PARAM ;;CALL=PARAM TRAP+10(104410)
(3) 016602 013542 .SAVOS ;;CALL=SAVOS TRAP+11(104411)

```

```

(3) 016604 013602          .RES05  ;;CALL=RES05   TRAP+12(104412)
(3) 016606 013634          .CONVRT ;;CALL=CONVRT  TRAP+13(104413)
(3) 016610 014012          .SETFLG ;;CALL=SETFLG  TRAP+14(104414)
8177  ;*****
8178  ;UTILITIES
8179  ;*****
8180
8181  ;THIS UTILITY CALCULATES PRIORITY LEVEL
8182  016612 006367 000044 DULEV: ASL  DUPRT  ;SHIFT LEFT
8183  016616 006367 000040      ASL  DUPRT  ;
8184  016622 006367 000034      ASL  DUPRT  ;
8185  016626 006367 000030      ASL  DUPRT  ;
8186  016632 006367 000024      ASL  DUPRT  ;
8187  016636 016767 000020 000020 MOV  DUPRT,LESS1 ;MOVE THIS TO LESS1
8188  016644 162767 000001 000012 SUB  #1,LESS1 ;CREATE LESS1
8189  016652 042767 000037 000004 BIC  #37,LESS1 ;CLEAR TNZVC
8190  016660 000207          RTS  PC
8191  016662 000240          DUPRT: PR5
8192  016664 000200          LESS1: PR4 ;LEVEL TO ALLOW INTERRUPTS
8193
8194  ;NEW DU ADDRESSES
8195  016666 016767 000126 163016 DUADDR: MOV  DUBASE,RXCSR ;XXX0
8196  016674 005267 000120          INC  DUBASE
8197  016700 016767 000114 163006 MOV  DUBASE,HRXCSR ;XXX1
8198  016706 005267 000106          INC  DUBASE
8199  016712 016767 000102 162776 MOV  DUBASE,RXDBUF ;XXX2
8200  016720 016767 000074 162774 MOV  DUBASE,PARCSR ;XXX2
8201  016726 005267 000066          INC  DUBASE
8202  016732 016767 000062 162760 MOV  DUBASE,HRXDBUF ;XXX3
8203  016740 016767 000054 162756 MOV  DUBASE,HPARCSR ;XXX3
8204  016746 005267 000046          INC  DUBASE
8205  016752 016767 000042 162746 MOV  DUBASE,TXCSR ;XXX4
8206  016760 005267 000034          INC  DUBASE
8207  016764 016767 000030 162736 MOV  DUBASE,HTXCSR ;XXX5
8208  016772 005267 000022          INC  DUBASE
8209  016776 016767 000016 162726 MOV  DUBASE,TXDBUF ;XXX6
8210  017004 005267 000010          INC  DUBASE
8211  017010 016767 000004 162716 MOV  DUBASE,HTXDBUF ;XXX7
8212  017016 000207          RTS  PC
8213  017020 000000          DUBASE: 0
8214
8215
8216  ;THIS UTILITY POKES THE MAINT DATA BASED UPON THE
8217  ;INFORMATION CONTAINED IN $TMP1 AND IT IS
8218  ;SHIFTED IN BY THE CONTENTS OF SHIFT
8218  017022 042777 040000 162676 RPOKE: BIC  #MTDATA,@TXCSR
8219  017030 005067 162446          CLR  $TMP2
8220  017034 006067 162440          ROR  $TMP1 ;FORCE CARRY
8221  017040 006067 162436          ROR  $TMP2 ;PICK UP CARRY IN BIT 15
8222  017044 006267 162432          ASR  $TMP2 ;SHIFT INTO BIT 14
8223  017050 042767 100000 162424 BIC  #BIT15,$TMP2 ;CLR BIT 15
8224  017056 056777 162420 162642 BIS  $TMP2,@TXCSR ;POKE MAINT DATA
8225  017064 042777 020000 162634 BIC  #CLK,@TXCSR ;POKE CLK
8226  017072 052777 020000 162626 BIS  #CLK,@TXCSR ;
8227  017100 005367 162016          DEC  SHIFT
8228  017104 001346          BNE  RPOKE
8229  017106 000207          RTS  PC

```

```

8230      :THIS ROUTINE CALCULATES ODD PARITY FOR AN 8 BIT CHAR
8231 017110 016767 162364 162364 ODD8:  MOV  $TMP1,$TMP2  ;SAVE TEMP1
8232 017116 005067 162362          CLR  $TMP3
8233 017122 012727 000010          MOV  #8.,(PC)+
8234 017126 000000          4$:  0
8235 017130 006067 162346          1$:  ROR  $TMP2
8236 017134 005567 162344          ADC  $TMP3
8237 017140 005367 177762          DEC  4$
8238 017144 001371          BNE  1$
8239 017146 006067 162332          ROR  $TMP3
8240 017152 103404          BCS  2$
8241 017154 052767 000400 162316  BIS  #BIT8,$TMP1  ;SET ODD PARITY
8242 017162 000403          BR   3$
8243 017164 042767 000400 162306  2$:  BIC  #BIT8,$TMP1  ;CLR EVEN PARITY
8244          ;$TMP1 NOW HAS ODD PARITY CHARACTER
8245 017172 000207          3$:  RTS  PC
8246
8247      :THIS ROUTINE CALCULATES EVEN PARITY FOR AN 8 BIT CHARACTER
8248 017174 016767 162300 162300  EVEN8: MOV  $TMP1,$TMP2  ;SAVE TEMP1
8249 017202 005067 162276          CLR  $TMP3
8250 017206 012727 000010          MOV  #8.,(PC)+
8251 017212 000000          4$:  0
8252 017214 006067 162262          1$:  ROR  $TMP2
8253 017220 005567 162260          ADC  $TMP3
8254 017224 005367 177762          DEC  4$
8255 017230 001371          BNE  1$
8256 017232 006067 162246          ROR  $TMP3
8257 017236 103004          BCC  2$
8258 017240 052767 000400 162232  BIS  #BIT8,$TMP1  ;SET EVEN PARITY
8259 017246 000403          BR   3$
8260 017250 042767 000400 162222  2$:  BIC  #BIT8,$TMP1  ;CLR ODD FARITY
8261          ;$TMP1 NOW HAS EVEN PARITY CHARACTER
8262 017256 000207          3$:  RTS  PC
8263
8264 017260 062716 000002  TRPREG: ADD  #2,(SP) ;ALLOW IT TO 'CRUNCH' INTO HLT BACK
8265          ;IN MAIN PART OF THE PROGRAM
8266 017264 000002          RTI
8267          POINT=.          ;SAVE POINTER
      (1) 000100 017266          .=100
      (1) 000102 000300          $CLKVEC          ;LKVEC HANDLER
      (1) 000140 000140          300          ;INTERRUPT HANDLER PRI
      (1) 000140 170000          .=140          ;BRKVEC
      (1) 000142 000300          170000          ;ODT START ADDRESS
      (1) 017266 017266          300          ;PRIORITY
      (1) 017272 000000          .=POINT          ;RESTORE POINTER
      (1) 017274 005015 045514 042526  $CLKVEC:          TYPE,CLKMES
      (1) 017302 020103 047111 042524          HALT
      (1) 017310 051122 050125 020124  CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
      (1) 017316 020055 044504 041523
      (1) 017324 047117 042516 052103
      (1) 017332 046040 041524 000040
8268          .END
  
```



AAA	003200	8150#	
ABASE =	000000	8150	
ACDW1 =	000000	8150	
ACDW2 =	000000	8150	
ACPUOP=	000000	8150	
ACTREG	001166	8150#*	8176
ADDW0 =	000000	8150	
ADDW1 =	000000	8150	
ADDW10=	000000	8150	
ADDW11=	000000	8150	
ADDW12=	000000	8150	
ADDW13=	000000	8150	
ADDW14=	000000	8150	
ADDW15=	000000	8150	
ADDW2 =	000000	8150	
ADDW3 =	000000	8150	
ADDW4 =	000000	8150	
ADDW5 =	000000	8150	
ADDW6 =	000000	8150	
ADDW7 =	000000	8150	
ADDW8 =	000000	8150	
ADDW9 =	000000	8150	
ADEVCT=	000000	8150	
ADEVN =	000000	8150	
ADRCNT=	013541	8176#*	
AENV =	000000	8150	
AENVN =	000000	8150	
AFATAL=	000000	8150	
AMADR1=	000000	8150	
AMADR2=	000000	8150	
AMADR3=	000000	8150	
AMADR4=	000000	8150	
AMAMS1=	000000	8150	
AMAMS2=	000000	8150	
AMAMS3=	000000	8150	
AMAMS4=	000000	8150	
AMSGAD=	000000	8150	
AMSGLG=	000000	8150	
AMSGTY=	000000	8150	
AMTYP1=	000000	8150	
AMTYP2=	000000	8150	
AMTYP3=	000000	8150	
AMTYP4=	000000	8150	
APASS =	000000	8150	
APRIOR=	000000	8150	
APTCSU=	000040	7010#	8176
APTENV=	000001	7010#	8176
APTSIZ=	000200	7010#	8150
APTSP0=	000100	7010#	8176
ASWREG=	000000	8150	
ATESTN=	000000	8150	
AUNIT =	000000	8150	
AUSWR =	000000	8150	
AVECT1=	000000	8150	
AVECT2=	000000	8150	
BASEAD	001154	8150#*	8176*



DISPRE	000174	8150#																		
DNA	= 100000	8150#																		
DNAINT	= 000040	8150#																		
DSC	= 100000	8150#																		
DSINTE	= 000040	8150#																		
DSR	= 001000	8150#																		
DSWR	= 177570	8150#																		
DTR	= 000002	8150#																		
DT1	002116	8150#																		
DT4	002126	8150#																		
DUADDR	016666	8150	8176	8195#																
DUBASE	017020	8150	8176*	8195	8196*	8197	8198*	8199	8200	8201*	8202	8203	8204*	8205						
		8206*	8207	8208*	8209	8210*	8211	8213#												
DULEV	016612	8150	8182#																	
DUPRT	016662	8150*	8182*	8183*	8184*	8185*	8186*	8187	8191#											
DURIS	001740	8150#	8176*																	
DURIV	001736	8150#	8176*																	
DUTIS	001744	8150#	8176*																	
DUTIV	001742	8150#	8176*																	
EIGHT	= 006000	8150#	8162	8163	8164	8165														
EMTVEC	= 000030	8150#*																		
EM1	001762	8150#																		
EM2	002022	8150#																		
EM3	002043	8150#																		
EM4	001746	8150#																		
ERRCNT	001114	8150#																		
ERRVEC	= 000004	8150#*	8176*																	
EVEN8	017174	8248#																		
EVEPAR	= 001400	8150#																		
EVPAR	= 000400	8150#																		
FIVE	= 000000	8150#	8152	8153	8166	8167	8168	8169												
FRMERR	= 020000	8150#																		
GNS	= ***** U	8176																		
HDXEN	= 000010	8150#																		
HILIM	013534	8176#*																		
HLD0	001130	8150#	8176*																	
HLD1	001132	8150#	8176*																	
HLD2	001134	8150#	8176*																	
HLD3	001136	8150#	8176*																	
HLD4	001140	8150#	8176*																	
HLD5	001142	8150#	8176*																	
HLD6	001144	8150#	8176*																	
HOLD	001120	8150#																		
HPARCS	001724	8150#	8203*																	
HRXCSR	001714	8150#	8197*																	
HRXDBU	001720	8150#	8202*																	
HT	= 000011	8150#	8176																	
HTXCSR	001730	8150#	8207*																	
HTXDBU	001734	8150#	8211*																	
INBUF	016022	8150	8176#																	
INIFLG	001172	8150#*																		
INSTER	= 104407	8150	8176#																	
INSTR	= 104406	8150	8176#																	
INSTR2	013340	8176#																		
IOTVEC	= 000020	8150#*																		
ISYMOD	= 000000	8150#	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163						



OVRRUN=	040000	8150#																		
PARAM =	104410	8150	8176#																	
PARAM1	013372	8176#																		
PARCSR	001722	8150#	8152*	8153*	8154*	8155*	8156*	8157*	8158*	8159*	8160*	8161*	8162*	8163*						
		8164*	8165*	8166*	8167*	8168*	8169*	8170*	8171*	8172*	8173*	8174*	8175*	8200*						
PAREN =	001000	8150#																		
PARER =	010000	8150#																		
PARERR	013446	8176#																		
PARTI	013530	8176#																		
PASCNT	001112	8150#*	8176*																	
PFTAB	015014	8176#																		
PIRQ =	177772	8150#																		
PIRQVE=	000240	8150#																		
POINT =	017266	8267#																		
POPPO =	012600	8150#																		
POP1SP=	005726	8150#																		
POP2SP=	022626	8150#																		
PRO =	000000	8150#																		
PR1 =	000040	8150#																		
PR2	000100	8150#																		
PR3 =	000140	8150#																		
PR4 =	000200	8150#	8192																	
PR5 =	000240	8150#	8191																	
PR6 =	000300	8150#																		
PR7 =	000340	8150#																		
PS =	177776	8150#																		
PSW =	177776	8150#																		
PUSHRO=	010046	8150#																		
PUSH1S=	005746	8150#																		
PUSH2S=	024646	8150#																		
PWRVEC=	000024	8150#*																		
RDSW	012706	8176#*																		
REACT=	004000	8150#																		
REPLAY	012374	8176#																		
RESTAR	014720	8176#																		
RESTRT	012554	8176#																		
RESVEC=	000010	8150#																		
RESOS =	104412	8176#																		
RETURN	001104	8150#*	8176																	
RING =	040000	8150#																		
RINTEN=	000100	8150#																		
ROTADD	001170	8150#*	8176*																	
RPOKE	017022	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163	8164						
		8165	8166	8167	8168	8169	8170	8171	8172	8173	8174	8175	8218#	8228						
RTS =	000004	8150#																		
RUNA =	***** U	5678	5862	6003	6417															
RUNB =	000000	5676#	5703	5871	6010	6426														
RUNC =	***** U	5729	5879	6017	6435															
RUND =	***** U	5754	5888	6024	6444															
RUNE =	***** U	5779	5898	6031	6453															
RUNF =	***** U	5804	5908	6038	6462															
RUNIT	012300	8176#																		
RXCSR	001712	8150#	8152*	8153*	8154*	8155*	8156*	8157*	8158*	8159*	8160*	8161*	8162*	8163*						
		8164*	8165*	8166*	8167*	8168*	8169*	8170*	8171*	8172*	8173*	8174*	8175*	8176						
		8195*																		
RXDBUF	001716	8150#	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163						





\$BASE	001602	8150#		
\$BDADR	001422	8150#		
\$BDDAT	001426	8150#		
\$BELL	001516	8150#	8176	
\$CDW1	001606	8150#		
\$CDW2	001610	8150#		
\$CHARC	013220	8176#*		
\$CKSWR=	***** U	8176		
\$CLKVE	017266	8267#		
\$CMTAG	001400	8150#		
\$CM1	= 000006	8150#		
\$CM2	= 000014	8150#		
\$CM3	= 000006	8150#		
\$CM4	= 000006	8150#		
\$CPUOP	001554	8150#		
\$CRLF	001523	8150#	8176	
\$DDW0	001612	8150#		
\$DDW1	001614	8150#		
\$DDW10	001636	8150#		
\$DDW11	001640	8150#		
\$DDW12	001642	8150#		
\$DDW13	001644	8150#		
\$DDW14	001646	8150#		
\$DDW15	001650	8150#		
\$DDW2	001616	8150#		
\$DDW3	001620	8150#		
\$DDW4	001622	8150#		
\$DDW5	001624	8150#		
\$DDW6	001626	8150#		
\$DDW7	001630	8150#		
\$DDW8	001632	8150#		
\$DDW9	001634	8150#		
\$DEVCT	001536	8150#		
\$DEVN	001604	3150#		
\$E	= 000002	6838#		
\$ENDAD	012544	8150	8176#	
\$ENV	001546	7010	8150#	8176
\$ENVN	001547	7010	8150#	8176
\$ERFLG	001403	8150#*	8176*	
\$ERMAX	001415	8150#*	8176*	
\$ERROR	014060	8150	8176#	
\$ERRPC	001416	8150#*	8176*	
\$ERRTB	001652	8150#	8176	
\$ERRTY	014300	8176#		
\$ERTTL	001412	8150#*	8176*	
\$ESCAP	001514	8150#*	8176*	
\$ETABL	001546	8150#		
\$ETEND	001652	8150#		
\$FATAL	001530	7010*	8150#	
\$FFLG	000244	7010#*		
\$FILLC	001456	8150#	8176	
\$FILLS	001455	8150#	8176	
\$GDADR	001420	8150#		
\$GDDAT	001424	8150#		
\$GTSWR=	***** U	8176		
\$HIBTS	002136	8150#		



SICNT	001404	8150#	8176*																		
SINTAG	001435	8150#																			
SITEMB	001414	8150#	8176*																		
SLF	001524	8150#	8176																		
SLFLG	000243	7010#*																			
SLPADR	001406	8150#*	8176*																		
SLPERR	001410	8150#*	8176*																		
SMADR1	001560	8150#																			
SMADR2	001564	8150#																			
SMADR3	001570	8150#																			
SMADR4	001574	8150#																			
SMAIL	001526	8150#	8176																		
SMAMS1	001556	8150#																			
SMAMS2	001562	8150#																			
SMAMS3	001566	8150#																			
SMAMS4	001572	8150#																			
SMBADR	002140	8150#																			
SMFLG	000242	7010#*																			
SMSGAD	001542	7010#	8150#																		
SMSGLG	001544	7010#	8150#																		
SMSGTY	001526	7010#	8150#																		
SMTYP1	001557	8150#																			
SMTYP2	001563	8150#																			
SMTYP3	001567	8150#																			
SMTYP4	001573	8150#																			
SMXCNT	016522	8176#																			
SN	= 000000	6837#	8176#																		
SNULL	001454	8150#	8176																		
SNWTST	= 000000	8152#	8153#	8154#	8155#	8156#	8157#	8158#	8159#	8160#	8161#	8162#	8163#	8164#							
		8165#	8166#	8167#	8168#	8169#	8170#	8171#	8172#	8173#	8174#	8175#									
SOCNT	014656	8176#*																			
SOMODE	014660	8176#*																			
SOVER	016502	8176#																			
SPASS	001534	8150#*	8170#																		
SPASTM	002144	8150#																			
SPWRDN	014662	8150	8176#																		
SQUES	001522	8150#	8176																		
SRDCHR	= *****	8176																			
SRDDEC	= *****	8176																			
SRDLIN	= *****	8176																			
SRDOCT	= *****	8176																			
SREGAD	001460	8150#																			
SREG0	001462	8150#	8176*																		
SREG1	001464	8150#	8176*																		
SREG2	001466	8150#	8176*																		
SREG3	001470	8150#	8176*																		
SREG4	001472	8150#	8176*																		
SREG5	001474	8150#	8176*																		
SR2A	= *****	8176																			
SSAVRE	= *****	8176																			
SSCOPE	016170	8150	8176#																		
SSETUP	= 000017	8150#	8176																		
SSTUP	= 177777	8150#																			
SSVLAD	016446	8176#																			
SSVPC	= 002136	8150#																			
SSWR	= 177400	6497#	8150	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162							



.SAV05	013542	8176#	
.SCOPE	016170	8176#	
.SCOP1	C12720	8176#	
.SETFL	014012	8176#	
.START	002152	8150#	8176
.SASTA=	***** U	7010	
.SX =	002136	8150#	





.\$DIV	4587#		
.\$EOP	2214#	6494#	
.\$ERRO	2700#	6495#	8176
.\$ERRT	2896#	6495#	8176
.\$MULT	4523#		
.\$POWE	4229#	6495#	
.\$RAND	4307#		
.\$RDDE	3891#		
.\$RDOC	3797#		
.\$READ	3395#		
.\$R2AZ	4958#		
.\$SAVE	3969#		
.\$SB2D	4771#		
.\$SB2C	4874#		
.\$SCOP	2454#	6495#	8176
.\$SIZE	4361#		
.\$SUPR	4913#		
.\$STRAP	4073#	6495#	8176
.\$TYPB	3287#		
.\$TYPD	3209#		
.\$TYPE	2985#	6494#	8176
.\$TYPO	3112#	6495#	8176
.\$4OCA	972#		

. ABS. 017340 000

ERRORS DETECTED: 0

CNDURA,CNDURA/CRF/NL:TOC=CNMAC2.SML,CNDUR1.M11,CNDUR2.M11,CNDURA.M11  
RUN-TIME: 17 16 1 SECONDS  
RUN-TIME RATIO: 81/35=2.3  
CORE USED: 43K (86 PAGES)