

DUV11

OFLNE COMBINED  
CNDUVAO

AH-T460A-MC  
FICHE 1 OF 1

MAY 1983  
COPYRIGHT © 82-83  
MADE IN USA



Table with multiple columns and rows of data, including headers and numerical values. The text is faint and difficult to read.





.REM \*

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

PRODUCT NAME: CNDUVA0 DUV11 OFLNE COMBINED

PRODUCT CODE: AC-T459A-MC

PRODUCT DATE:DEC, 1982

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 COMBINED TESTS VERIFY THAT THE TRANSMITTER AND RECEIVER CAN TALK THRU THE EXTERNAL MODEM CABLE PROVIDING THAT THE H315 CONNECTOR IS ON

\* .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  
STARTING ADDRESS

4.2

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 CNDUV-A TAPE F" (ONCE ONLY)

\* .REM \*

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

\* .REM \*

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



- 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 SELECTED 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 TIMEOUT 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 COMBINED (TRANSMITTER & RECEIVER)  
CABLE TESTING OF THE DEVICE  
SEE LISTING FOR DETAILS

\* .REM \*

\* .REM \*

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

11. REVISION HISTORY  
DZDUVB1 DIAGNOSTIC WAS MODIFIED TO WORK IN 11/21 PROCESSOR  
PRIORITY340 WAS CHANGED TO 300 WHEREVER ENCOUNTERED  
DEFAULT CSR AND VECTOR HAVE BEEN CHANGED  
.INIT CALL ADDED AT THE END.  
DIAGNOSTIC WAS RENAMED TO CNDUVA0.  
12. LISTINGS

\*



6489  
 6588  
 6589  
 6590  
 6601  
 6615  
 6616  
 6617  
 6678  
 6679  
 6884  
 7012  
 7013

.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 $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 000112 006200 ASR R0 ;;GET MESSAGE LNGTH IN WORDS
(1) 000114 010067 001424 MOV R0,$MSGGLT ;;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 012716 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)                                ;;LOG FLAG
(1) 000244      000          $FFLG: .BYTE 0    ;;FATAL FLAG
(1)                                .EVEN
(1)                                APTSIZE=200
(1)                                APTENV=001
(1)                                APTSPool=100
(1)                                APTCSUP=040
(1)                                $TN=1
7252
7378
7382
7417
7434
7447
7459
7472

```



7495  
7561  
7567  
7703  
7731  
7764  
7805  
7836  
7887  
7935  
7988  
8003  
8015  
8117





```

(2)          000300      PR6=      300          ;;PRIORITY LEVEL 6
(2)          000340      PR7=      340          ;;PRIORITY LEVEL 7
(2)
(2)          :*'SWITCH REGISTER' SWITCH DEFINITIONS
(2)          100000      SW15=     100000
(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)          ;*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)
(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)
(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)                                     ;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)
(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)
(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)                                     .EVEN
(2) 001176      001176   .EVEN
(2) 001400      001400   .=1400
(2)

```



```
(2)  
(2)  
(2)  
(2)  
(2)          ;INSTRUCTION DEFINITIONS  
(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   RECACT=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   SYNCH=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 001400
(2) 001400 000000
(2) 001402 000
(2) 001403 000
(2) 001404 000000
(2) 001406 000000
(2) 001410 000000
(2) 001412 000000
(2) 001414 000
(2) 001415 001
(2) 001416 000000
(2) 001420 000000
(2) 001422 000000
(2) 001424 000000
(2) 001426 000000
(2) 001430 000000
(2) 001432 000000
(2) 001434 000
(2) 001435 000
(2) 001436 000000
(2) 001440 177570
(2) 001442 177570
(2) 001444 177560
(2) 001446 177562
(2) 001450 177564
(2) 001452 177566
(2) 001454 000
(2) 001455 002
(2)
(2) 001456 012
(2) 001457 000
(2) 001460 000000
(2)
(4) 001462 000000
(4) 001464 000000
(4) 001466 000000
(4) 001470 000000
(4) 001472 000000
(4) 001474 000000
(4) 001476 000000
(4) 001500 000000
(4) 001502 000000
(4) 001504 000000
(4) 001506 000000
(4) 001510 000000
(2) 001512 000000
(2) 001514 000000
(2) 001516 177607 000377
(2) 001522 077
```

```

$CMTAG:      =.
$STNM: .WORD 0
$ERFLG: .BYTE 0
$ICNT: .WORD 0
$LPADR: .WORD 0
$LPERR: .WORD 0
$ERTTL: .WORD 0
$ITEMB: .BYTE 0
$ERMAX: .BYTE 1
$ERRPC: .WORD 0
$GDADR: .WORD 0
$BDADR: .WORD 0
$GDDAT: .WORD 0
$BDDAT: .WORD 0
$AUTOB: .BYTE 0
$INTAG: .BYTE 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
$TKS: 177560
$TKB: 177562
$TPS: 177564
$TPB: 177566
$NULL: .BYTE 0
$FILLS: .BYTE 2
$FILLC: .BYTE 12
$TPFLG: .BYTE 0
$REGAD: .WORD 0
$REG0: .WORD 0
$REG1: .WORD 0
$REG2: .WORD 0
$REG3: .WORD 0
$REG4: .WORD 0
$REG5: .WORD 0
$TMP0: .WORD 0
$TMP1: .WORD 0
$TMP2: .WORD 0
$TMP3: .WORD 0
$TMP4: .WORD 0
$TMP5: .WORD 0
$TIMES: 0
$ESCAPE: 0
$BELL: .ASCIZ <207><377><377>
$QUES: .ASCIZ /?/
```

```

::START OF COMMON TAGS
::CONTAINS THE TEST NUMBER
::CONTAINS ERROR FLAG
::CONTAINS SUBTEST ITERATION COUNT
::CONTAINS SCOPE LOOP ADDRESS
::CONTAINS SCOPE RETURN FOR ERRORS
::CONTAINS TOTAL ERRORS DETECTED
::CONTAINS ITEM CONTROL BYTE
::CONTAINS MAX. ERRORS PER TEST
::CONTAINS PC OF LAST ERROR INSTRUCTION
::CONTAINS ADDRESS OF 'GOOD' DATA
::CONTAINS ADDRESS OF 'BAD' DATA
::CONTAINS 'GOOD' DATA
::CONTAINS 'BAD' DATA
::RESERVED--NOT TO BE USED
::AUTOMATIC MODE INDICATOR
::INTERRUPT MODE INDICATOR
::ADDRESS OF SWITCH REGISTER
::ADDRESS OF DISPLAY REGISTER
::TTY KBD STATUS
::TTY KBD BUFFER
::TTY PRINTER STATUS REG. ADDRESS
::TTY PRINTER BUFFER REG. ADDRESS
::CONTAINS NULL CHARACTER FOR FILLS
::CONTAINS # OF FILLER CHARACTERS REQUIRED
::INSERT FILL CHARS. AFTER A 'LINE FEED'
::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
::CONTAINS THE ADDRESS FROM
*WHICH ($REG0) WAS OBTAINED
::CONTAINS (($REGAD)+0)
::CONTAINS (($REGAD)+2)
::CONTAINS (($REGAD)+4)
::CONTAINS (($REGAD)+6)
::CONTAINS (($REGAD)+10)
::CONTAINS (($REGAD)+12)
::USER DEFINED
::USER DEFINED
::USER DEFINED
::USER DEFINED
::USER DEFINED
::USER DEFINED
::USER DEFINED
::MAX. NUMBER OF ITERATIONS
::ESCAPE ON ERROR ADDRESS
::CODE FOR BELL
::QUESTION MARK
```

```

(2) 001523 015 $CRLF: .ASCII <15> ::CARRIAGE RETURN
(2) 001524 000012 $LF: .ASCIZ <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

```



(3)	001620	000000	\$DDW3:	.WORD	ADDW3	::DEVICE	DESCRIPTOR	WORD#3
(3)	001622	000000	\$DDW4:	.WORD	ADDW4	::DEVICE	DESCRIPTOR	WORD#4
(3)	001624	000000	\$DDW5:	.WORD	ADDW5	::DEVICE	DESCRIPTOR	WORD#5
(3)	001626	000000	\$DDW6:	.WORD	ADDW6	::DEVICE	DESCRIPTOR	WORD#6
(3)	001630	000000	\$DDW7:	.WORD	ADDW7	::DEVICE	DESCRIPTOR	WORD#7
(3)	001632	000000	\$DDW8:	.WORD	ADDW8	::DEVICE	DESCRIPTOR	WORD#8
(3)	001634	000000	\$DDW9:	.WORD	ADDW9	::DEVICE	DESCRIPTOR	WORD#9
(3)	001636	000000	\$DDW10:	.WORD	ADDW10	::DEVICE	DESCRIPTOR	WORD#10
(3)	001640	000000	\$DDW11:	.WORD	ADDW11	::DEVICE	DESCRIPTOR	WORD#11
(3)	001642	000000	\$DDW12:	.WORD	ADDW12	::DEVICE	DESCRIPTOR	WORD#12
(3)	001644	000000	\$DDW13:	.WORD	ADDW13	::DEVICE	DESCRIPTOR	WORD#13
(3)	001646	000000	\$DDW14:	.WORD	ADDW14	::DEVICE	DESCRIPTOR	WORD#14
(3)	001650	000000	\$DDW15:	.WORD	ADDW15	::DEVICE	DESCRIPTOR	WORD#15
(3)								
(3)								
(3)	001652		SETEND:					
(3)								
(4)								
(4)								





CNDUV-A MACY11 30(1046) 14-DEC-82 10:03 PAGE 61-11  
CNDUVA.M11 30-OCT-82 12:45 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)		;TXCSR 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) 001652 $ERRTB:
(1) ;ERROR TABLE
(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) ;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) ;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) ;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)
(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)
(1) .EVEN
(1)
(1) 002116 001416 001130 001132 DT1: ;DATA TABLES FOR ERROR MESSAGES
(1) 002124 000000 .WORD $ERRPC,HLD0,HLD1,0
(1)
(1) 002126 001416 000000 DT4: .WORD $ERRPC,0
(1)
(1) 002132 000 000 000 DF1: .BYTE 0,0,0,0
(1) 002135 000
(1)
(2) .EVEN
(2) .SBTTL ACT11 HOOKS
(3)
(2) ;*****
(2) ;HOOKS REQUIRED BY ACT11
(2) 002136 $SVPC=. ;SAVE PC
(2) 000046 012674 .=46 ;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(2) 000052 000052 $ENDAD ;
(2) 002136 .WORD 0 ;2)SET LOC.52 TO ZERO
(2) .SBTTL APT PARAMETER BLOCK ; RESTORE PC
(3)
(2) ;*****
(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
(2) 002136
(3)
(2) ;*****
(2) ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(2) ;INTERFACE SPEC.
(2)
(2) 002136 $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    #%CMTAG,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 016320 000020   MOV    #%SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(3) 002200 012737 000300 000022   MOV    #PR6,@#IOTVEC+2 ;;LEVEL 6
(3) 002206 012737 014210 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 016654 000034   MOV    #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(3) 002230 012737 000300 000036   MOV    #PR6,@#TRAPVEC+2;LEVEL 6
(3) 002236 012737 015012 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    #$$SWREG,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 015012 000024   MOV    #.PFAIL,@#24 ;;SET UP POWER FAIL VECTOR

```

```

(2) 002432 105067 176535 CLRB STFLG ;CLEAR START F: AG
(2) 002436 005067 176450 CLR PASCNT ;CLEAR PASS COUNT
(2) 002442 105067 176735 CLRB $ERFLG ;CLEAR ERROR FLAG
(2) 002446 005067 176740 CLR $ERTTL ;CLEAR ERROR COUNT
(2) 002452 005067 176740 CLR $ERRPC ;CLEAR LAST ERROR POINTER
(2) 002456 012767 000001 176716 MOV #1,$STSTM ;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 010204 33$: JSR PC,CNTLU
(2) 002572 105767 176374 3$: TSTB INIFLG ;HAS INITIALIZATION BEEN PERFORMED
(2) 002576 001004 BNE ONCE
(2) 002600 104401 015152 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 015220 MREGAD ;MESSAGE
(2) 002710 104410 PARAM ;CONVERT STRING
(2) 002712 174000 174000 ;LOW LIMIT
(2) 002714 177776 177776 ;HIGH LIMIT
(2) 002716 017150 DUBASE ;STORE AT THIS LOCATION
(2) 002720 001 .BYTE 1 ;MASK
(2) 002721 001 .BYTE 1 ;HOW MANY TIMES + 2
(1) 002722 016767 014222 176226 MOV DUBASE,KEEPADD ;SAVE
(1) 002730 004767 014062 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 015205                      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 015250                      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
(1)                                     ;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 015277                      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
(1)                                     ;LEAST TWO DEVICES WHEN YOU ANSWER YES TO
(1)                                     ;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 015373                      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 013616 OUTMUL: MOV #300,DUPRT
(1) 003174 004767 013542 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 015611 MSYNC :MESSAGE
(2) 003204 122767 000061 012740 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 012722 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$: NOP
(2) 003250 104406 INSTR :OUTPUT MESSAGE & GET INPUT STRING
(2) 003252 015657 MWIRE6 :MESSAGE
(2) 003254 104414 SETFLG :SET FLAG BASED UPON INPUT STRING
(2) 003256 001147 SEXMIT :THIS FLAG
(2) 003260 104406 INSTR :OUTPUT MESSAGE & GET INPUT STRING
(2) 003262 015730 MWIRE5 :MESSAGE
(2) 003264 104414 SETFLG :SET FLAG BASED UPON INPUT STRING
(2) 003266 001150 SEREC :THIS FLAG
(2) 003270 104406 INSTR :OUTPUT MESSAGE & GET INPUT STRING
(2) 003272 016000 MWIRE4 :MESSAGE
(2) 003274 104414 SETFLG :SET FLAG BASED UPON INPUT STRING
(2) 003276 001151 OPTCLR :THIS FLAG
(2) 003300 104406 INSTR :OUTPUT MESSAGE & GET INPUT STRING
(2) 003302 016057 MEXTJ :MESSAGE
(2) 003304 104414 SETFLG :SET FLAG BASED UPON INPUT STRING
(2) 003306 001153 JMRBY :THIS FLAG
(2)
(2) ;TEST START AND RESTART
(2)
(2) 003310 012706 001100 .BEGIN: MOV #STACK,SP ;SET UP STACK
(2) 003314 106427 000300 MTPS #300 ;LOCK OUT INTERRUPTS
(2) 003320 032777 000002 176112 BIT #SW01,@SWR ;IF SW01=1, GET STARTING PC
(2) 003326 001413 BEQ 3$
(3) 003330 104406 INSTR :OUTPUT MESSAGE & GET INPUT STRING
(3) 003332 015543 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$: MOV #TST1,RETURN ;START AT TEST 1
(2) 003364 104401 015537 4$: TYPE ,MR ;TYPE R
(2) 003370 000177 175510 JMP @RETURN ;START TESTING

```

```

8155
8156
8157
8158
8159
8160
8161
8161 (3) 003374 000004
8162 (3)
8163 003376 052777 000400 176322 BIS #MRESET,@TXCSR ;MASTER RESET
      (1) 003404 012777 020000 176310 MOV #SYNEXT,@PARCSR ;SET THE MODE
      (2) 003412 052777 000400 176306 BIS #MRESET,@TXCSR ;MASTER RESET
      (1)
      (1) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
      (1) 003420 012777 064001 176300 MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
      (1)
      (1) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
8164 003426 012777 026026 176266 MOV #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8164 003434 052777 000020 176250 BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
8165 ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
      (1) 003442 042777 020000 176256 BIC #CLK,@TXCSR ;POKE CLK DOWN
      (1) 003450 052777 020000 176250 BIS #CLK,@TXCSR ;POKE CLK UP
8166 003456 012777 003500 176252 MOV #1$,@DURIV ;SET UP TRAPCATCHER
8167 003464 016777 013322 176246 MOV DUPRT,@DURIS
8168 003472 106427 000000 MTPS #0 ;ALLOW INTERRUPTS
8169 003476 000424 BR 2$ ;JUMP AROUND INTERRUPT SVC ROUTINE
8170 ;THE FOLLOWING IS THE INTERRUPT SVC ROUTINE
8171 003500 106427 000300 1$: MTPS #300 ;DON'T ALLOW ANYMORE INTERRUPTS
8172 003504 042777 000100 176200 BIC #RINTEN,@RXCSR ;CLEAR INTERRUPT ENABLE
8173 003512 105777 176174 TSTB @RXCSR ;RXDONE=1?
8174 003516 100401 BMI .+4
8175 003520 104004 ERROR 4 ;FALSE INTERRUPT
8176 003522 012716 003714 MOV #3$(,SP) ;SET UP RETURN LOCATION
8177 003526 016777 176206 176202 MOV DURIS,@DURIV ;RESTORE TRAPCATCHER
8178 003534 012777 000000 176176 MOV #0,@DURIS
8179 003542 017701 176150 MOV @RXDBUF,R1 ;CLEAR INTERRUPT
8180 003546 000002 RTI
8181
8182 003550 052777 000100 176134 2$: BIS #RINTEN,@RXCSR ;SET INTERRUPT ENABLE
8183 003556 012767 000010 175336 MOV #8,SHIFT ;# OF SHIFTS
8184 003564 012767 000025 175706 MOV #25,$TMP1 ;TO BE SHIFTED CHARACTER
8185 ;THE FOLLOWING POKES THE MAINT DATA BASED UPON THE
8186 ;INFORMATION CONTAINED IN $TMP1 AND IT IS
8187 ;SHIFTED IN BY THE CONTENTS OF SHIFT
8188 003572 042777 040000 176126 5$: BIC #MTDATA,@TXCSR
8189 003600 000241 CLC
8190 003602 006067 175672 ROR $TMP1 ;FORCE CARRY
8191 003606 103003 BCC 4$
8192 003610 052777 040000 176110 BIS #MTDATA,@TXCSR
8193 003616 042777 020000 176102 4$: BIC #CLK,@TXCSR
8194 003624 052777 020000 176074 BIS #CLK,@TXCSR
8195 003632 005367 175264 DEC SHIFT
8196 003636 001355 BNE 5$
8197 ;INTERRUPT SHOULD NOW OCCUR
8198 003640 005000 CLR R0

```



```

8199 003642 005200          INC      R0          ;WAIT FOR INTERRUPT
8200 003644 001376          BNE     .-2
8201 003646 016777 176066 176062  MOV     DURIS,@DURIV ;RESTORE TRAPCATCHER
8202 003654 012777 000000 176056  MOV     #0,@DURIS   ;
8203 003662 016703 176030          MOV     RXDBUF,R3   ;FOR ERROR MESSAGE
8204 003666 012700 000025          MOV     #25,R0     ;EXPECTED
8205 003672 017701 176020          MOV     @RXDBUF,R1
8206 003676 042777 000100 176006  BIC     #RINTEN,@RXCSR ;CLR INTR ENABLE
8207 003704 020001          CMP     R0,R1
8208 003706 001401          BEQ     .+4
8209 003710 104002          ERROR  2          ;CHARACTERS SHOULD COMPARE
8210 003712 104004          ERROR  4          ;INTERRUPT FAILED TO OCCUR
8211
8212 003714 106427 000300 3$:  MTPS  #300  ;
8213
8214
8215
8216
8217
8218
8219
8220
8221
8221 (3) 003720 000004          ;*****
8221 (3)          TST2:  SCOPE
8222 003722 105767 175225          TSTB   JMRBY          ;IN MAINT. EXTERNAL?
8223 003726 100402          BMI     .+6          ;IF ANSWER WAS YES DO THIS TEST
8224 003730 000167 000402          JMP     1$          ;IF ANSWER WAS NO JUMP AROUND TEST
8225 003734 052777 000400 175764  BIS     #MRESET,@TXCSR ;MASTER RESET
8225 (1) 003742 012777 020000 175752  MOV     #SYNEXT,@PARCSR ;SET THE MODE
8225 (2) 003750 052777 000400 175750  BIS     #MRESET,@TXCSR ;MASTER RESET
8225 (1)
8225 (1)          ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
8225 (1) 003756 012777 064001 175742  MOV     #MTDATA!CLK!MINT!BREAK,@TXCSR
8225 (1)
8225 (1)          ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
8226 003772 052777 000020 175712  MOV     #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8227 004000 042777 020000 175720  BIS     #SYNSCH,@RXCSR ;SET SEARCH SYNC
8227 (1)          ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
8227 (1) 004006 052777 020000 175712  BIC     #CLK,@TXCSR   ;POKE CLK DOWN
8228 004014 012777 004036 175714  BIS     #CLK,@TXCSR   ;POKE CLK UP
8229 004022 016777 012764 175710  MOV     #2$,@DURIV   ;SET UP TRAPCATCHER
8230 004030 106427 000000          MOV     DUPRT,@DURIS ;
8231 004034 000454          MTPS   #0          ;ALLOW INTERRUPT
8231          BR      3$          ;JUMP AROUND SVC ROUTINE
8232
8233          ;THE FOLLOWING IS THE 1ST INTERRUPT SVC ROUTINE
8234 004036 106427 000300 2$:  MTPS   #300          ;DON'T ALLOW ANY MORE INTERRUPTS
8235 004042 105777 175644          TSTB   @RXCSR          ;RXDONE = 1 ?
8236 004046 100401          BMI     .+4
8237 004050 104004          ERROR  4          ;FALSE INTERRUPT
8238 004052 012716 004332          MOV     #5$, (SP)    ;SET UP RETURN LOCATION
8239 004056 012777 004136 175652  MOV     #4$,@DURIV   ;SET UP TRAPCATCHER FOR SECOND
8240          ;INTERRUPT
8241 004064 052777 000002 175620  BIS     #DTR,@RXCSR  ;TRY TO CAUSE SECOND INTERRUPT
8242 004072 017701 175620          MOV     @RXDBUF,R1  ;JUST READ RXDBUF TO CLR RXDONE
  
```



```

8243
8244 004076 106427 000000          MTPS #0          ;TO ALLOW SECOND INTERRUPT
8245 004102 005000          CLR RO          ;ALLOW INTERRUPT
8246 004104 005200          INC RO          ;WAIT FOR INTERRUPT
8247 004106 001376          BNE -.2
8248 004110 042777 000140 175574    BIC #RINTEN!DSINTE,@RXCSR ;CLR INTR ENABLES
8249 004116 104004          ERROR 4        ;2ND INTERRUPT FAILED TO OCCUR
8250
8251 004120 016777 175614 175610 6$:  MOV DURIS,@DURIV ;RESTORE TRAPCATCHER
8252 004126 012777 000000 175604    MOV #0,@DURIS ;
8253 004134 000002          RTI
8254
8255          ;THE FOLLOWING IS THE 2ND INTERRUPT SVC ROUTINE
8256 004136 106427 000300 4$:  MTPS #300      ;DON'T ALLOW ANYMORE INTERRUPTS
8257 004142 005777 175544          TST @RXCSR      ;DSC = 1 ?
8258 004146 100401          BMI .+4
8259 004150 104004          ERROR 4
8260 004152 042777 000140 175532    BIC #RINTEN!DSINTE,@RXCSR ;FALSE INTERRUPT
8261 004160 012716 004120          MOV #6$(,SP)    ;CLR BOTH INTR ENABLES
8262 004164 000002          RTI             ;SET UP RETURN LOCATION
8263
8264 004166 052777 000140 175516 3$:  BIS #RINTEN!DSINTE,@RXCSR ;SET INTERRUPT ENABLES
8265 004174 012767 000010 174720    MOV #8$,SHIFT  ;# OF SHIFTS
8266 004202 012767 000025 175270    MOV #25,$TMP1
8267          ;THE FOLLOWING POKES THE MAINT DATA BASED UPON THE
8268          ;INFORMATION CONTAINED IN $TMP1 AND IT IS
8269          ;SHIFTED IN BY THE CONTENTS OF SHIFT
8270 004210 042777 040000 175510 8$:  BIC #MTDATA,@TXCSR
8271 004216 000241          CLR
8272 004220 006067 175254          ROR $TMP1      ;FORCE CARRY
8273 004224 103003          BCC 7$
8274 004226 052777 040000 175472    BIS #MTDATA,@TXCSR
8275 004234 042777 020000 175464 7$:  BIC #CLK,@TXCSR
8276 004242 052777 020000 175456    BIS #CLK,@TXCSR
8277 004250 005367 174646          DEC SHIFT
8278 004254 001355          BNE 8$
8279          ;1ST INTERRUPT SHOULD NOW OCCUR
8280 004256 005000          CLR RO
8281 004260 005200          INC RO          ;WAIT FOR INTERRUPT
8282 004262 001376          BNE -.2
8283 004264 016777 175450 175444    MOV DURIS,@DURIV ;RESTORE TRAPCATCHER
8284 004272 012777 000000 175440    MOV #0,@DURIS ;
8285 004300 016703 175412          MOV RXDBUF,R3  ;FOR ERROR MESSAGE
8286 004304 012700 000025          MOV #25,RO     ;EXPECTED
8287 004310 017701 175402          MOV @RXDBUF,R1
8288 004314 042777 000140 175370    BIC #RINTEN!DSINTE,@RXCSR ;CLR INTERRUPT ENABLES
8289 004322 020001          CMP RO,R1
8290 004324 001401          BEQ .+4
8291 004326 104002          ERROR 2
8292 004330 104004          ERROR 4        ;CHARACTERS SHOULD COMPARE
8293          ;INTERRUPT FAILED TO OCCUR
8294 004332 106427 000300 5$:  MTPS #300      ;DON'T ALLOW ANY MORE INTERRUPTS
8295 004336
8296
8297          ;;THIS TEST VERIFYS THAT DNA CAUSES AN INTERRUPT
8298          ;;MODE: SYNC EXTERNAL

```

```

8299      ;; INTERRUPT VECTOR: DUTIV
8300
8301      ;; *****
(3) 004336 000004      TST3: SCOPE
(3)
8302
8303 004340 052777 000400 175360      BIS      #MRESET,@TXCSR ;MASTER RESET
(1) 004346 012777 020000 175346      MOV      #SYNEXT,@PARCSR ;SET THE MODE
(2) 004354 052777 000400 175344      BIS      #MRESET,@TXCSR ;MASTER RESET
(1)
(1)      ;SET MAINTENANCE MODE & SEND
(1)      ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 004362 012777 004020 175336      MOV      #MINT!SEND,@TXCSR
(1)
(1)      ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 004370 012777 026026 175324      MOV      #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8304 004376 112777 000025 175326      MOV      #25,@TXDBUF ;LOAD CHARACTER
8305 004404 012767 000010 174510      MOV      #8.,SHIFT
8306      ;POKE CLK TO GET INTO SYNCHRONIZATION
8307 004412 052777 020000 175306      BIS      #CLK,@TXCSR ;POKE CLK UP
(1) 004420 042777 020000 175300      BIC      #CLK,@TXCSR ;POKE CLK DOWN
8308
8309 004426      1$:
(1) 004426 052777 020000 175272      BIS      #CLK,@TXCSR ;POKE CLK UP
(1) 004434 042777 020000 175264      BIC      #CLK,@TXCSR ;POKE CLK DOWN
8310 004442 005367 174454      DEC      SHIFT ;LAST SHIFT?
8311 004446 001367      BNE      1$
8312 004450 012777 004516 175264      MOV      #2$,@DUTIV ;SET UP TRAPCATCHER
8313 004456 016777 012330 175260      MOV      DUPRT,@DUTIS ;
8314 004464 106427 000000      MTPS     #0 ;ALLOW INTERRUPTS
8315 004470 052777 000040 175230      BIS      #DNAINTE,@TXCSR ;ENABLE INTERRUPT
8316      ;NOW POKE CLK TO GET DNA
8317 004476 052777 020000 175222      BIS      #CLK,@TXCSR ;POKE CLK
8318 004504 005000      CLR      R0
8319 004506 005200      INC      R0 ;WAIT FOR INTERRUPT
8320 004510 001376      BNE      -2
8321 004512 104004      ERROR    4 ;INTERRUPT FAILED TO OCCUR
8322 004514 000422      BR       3$ ;JUMP AROUND SVC ROUTINE
8323      ;THE FOLLOWING IS THE INTERRUPT SERVICE ROUTINE
8324 004516 106427 000300      2$: MTPS     #300 ;DON'T ALLOW ANYMORE INTERRUPTS
8325 004522 005777 175200      TST      @TXCSR ;DNA?
8326 004526 100401      BMI     +4
8327 004530 104004      ERROR    4 ;FALSE INTERRUPT
8328 004532 042777 000040 175166      BIC      #DNAINTE,@TXCSR ;CLR INTR ENABLE
8329 004540 012716 004604      MOV      #4$(SP) ;SET UP RETURN LOCATION
8330 004544 016777 175174 175170      MOV      DUTIS,@DUTIV ;RESTORE TRAPCATCHER
8331 004552 012777 000000 175164      MOV      #0,@DUTIS ;
8332 004560 000002      RTI
8333
8334 004562 016777 175156 175152      3$: MOV      DUTIS,@DUTIV ;RESTORE TRAPCATCHER
8335 004570 012777 000000 175146      MOV      #0,@DUTIS ;
8336
8337 004576 042777 000040 175122      4$: BIC      #DNAINTE,@TXCSR ;CLR INTERRUPT ENABLE
8338 004604 106427 000300      MTPS     #300 ;RESTORE NO INTERRUPT STATUS
8339
8340

```



INITIALIZE THE COMMON TAGS

```
8341      ::THIS TEST VERIFYS THAT TXDONE CAUSES AN INTERRUPT
8342      ::INTERRUPT VECTOR: DUTIV
8343      ::NOTE: TXDONE = 1 AFTER A MASTER RESET
8344      ::
8345      ::*****
(3) 004610 000004      TST4: SCOPE
(3)
8346
8347 004612 052777 000400 175106      BIS      #MRESET,@TXCSR ;MASTER RESET
8348 004620 012777 004666 175114      MOV      #1$,@DUTIV ;SET UP TRAPCATCHER
8349 004626 016777 012160 175110      MOV      DUPRT,@DUTIS ;
8350 004634 106427 000000 ;ALLOW INTERPUTS
8351 004640 052777 000100 175060      BIS      #TXINTE,@TXCSR ;ENABLE INTERRUPT
8352 004646 005000      CLR      RO ;WAIT FOR INTERRUPT
8353 004650 005200      INC      RO ;
8354 004652 001376      BNE      -2 ;
8355 004654 042777 000100 175044      BIC      #TXINTE,@TXCSR ;CLR INTERRUPT ENABLE
8356 004662 104004      ERROR   4 ;INTERRUPT FAILED TO OCCUR
8357 004664 000422      BR       2$ ;JUMP AROUND SVC ROUTINE
8358
8359      ;THE FOLLOWING IS THE INTERRUPT SERVICE ROUTINE
8360 004666 106427 000300 175026      1$: MTPS   #300 ;DON'T ALLOW ANYMORE INTERRUPTS
8361 004672 042777 000100 175026      BIC      #TXINTE,@TXCSR ;CLR INTR ENABLE
8362 004700 105777 175022      TSTB    @TXCSR ;TXDONE?
8363 004704 100401      BMI     +4 ;
8364 004706 104004      ERROR   4 ;FALSE INTERRUPT
8365 004710 012716 004746      MOV     #3$, (SP) ;SET UP RETURN LOCATION
8366 004714 016777 175024 175020      MOV     DUTIS,@DUTIV ;RESTORE TRAPCATCHER
8367 004722 012777 000000 175014      MOV     #0,@DUTIS ;
8368 004730 000002      RTI
8369
8370 004732 016777 175006 175002      2$: MOV     DUTIS,@DUTIV ;RESTORE TRAPCATCHER
8371 004740 012777 000000 174776      MOV     #0,@DUTIS ;
8372
8373 004746 106427 000300      3$: MTPS   #300 ;RESTORE NO INTERRUPT STATUS
8374
8375
8376
8377      ::THIS TEST VERIFYS THAT TXDONE DOES NOT CAUSE AN INTERRUPT
8378      ::WHEN PROCESSOR PRIORITY LEVEL IS TOO HIGH
8379      ::INTERRUPT VECTOR: DUTIV
8380      ::NOTE: TXDONE = 1 AFTER A MASTER RESET
8381      ::
8382      ::*****
(3) 004752 000004      TST5: SCOPE
(3)
8383 004754 052777 000400 174744      BIS      #MRESET,@TXCSR ;MASTER RESET
8384 004762 012777 005046 174752      MOV      #1$,@DUTIV ;SET UP TRAPCATCHER
8385 004770 016777 012016 174746      MOV      DUPRT,@DUTIS ;
8386 004776 106427 000300 ;SET PS LEVEL TOO HIGH
8387 005002 052777 000100 174716      BIS      #TXINTE,@TXCSR ;ENABLE INTERRUPT
8388 005010 005000      CLR      RO ;WAIT FOR INTERRUPT
8389 005012 005200      INC      RO ;
8390 005014 001376      BNE      -2 ;
8391 005016 042777 000100 174702      BIC      #TXINTE,@TXCSR ;CLR INTR ENABLE
8392 005024 106427 000300      MTPS    #300 ;DON'T ALLOW INTERRUPTS
```



```

8393 005030 016777 174710 174704      MOV      DUTIS,@DUTIV      ;RESTORE TRAPCATCHER
8394 005036 012777 000000 174700      MOV      #0,@DUTIS        ;
8395 005044 000421                BR       2$                ;TEST IS OK....GET OUT OF TEST
8396                                ;THE FOLLOWING IS THE INTERRUPT SVC ROUTINE
8397 005046 106427 000300      1$:     MTPS      #300      ;DONT ALLOW ANYMORE INTERRUPTS
8398 005052 042777 000100 174646      BIC      #TXINTE,@TXCSR   ;CLR INTR ENABLE
8399 005060 012716 005102      MOV      #3$,(SP)        ;SET UP RETURN LOCATION
8400                                ;TO REPORT ERROR
8401 005064 016777 174654 174650      MOV      DUTIS,@DUTIV   ;RESTORE TRAPCATCHER
8402 005072 012777 000000 174644      MOV      #0,@DUTIS        ;
8403 005100 000002                RTI
8404                                ;END OF INTERRUPT SVC ROUTINE
8405
8406
8407
8408                                ;YOU SHOULD NOT GET INTO THIS FOLLOWING CODE UNLESS THERE
8409                                ;WAS AN ERROR
8410 005102 106427 000300      3$:     MTPS      #300      ;DON'T ALLOW ANYMORE INTERRUPTS
8411 005106 104004                ERROR    4                ;INTERRUPT SHOULD NOT OF OCCURED,CHECK
8412                                ;THE INTERRUPT LEVEL SELECTED OR CHECK
8413                                ;INTERRUPT LOGIC OR BOTH
8414 005110                2$:
8415
8416                                ;;THIS TEST VERIFYS THAT TXDONE CAUSES ONLY ONE INTERRUPT
8417                                ;;PROVIDING THAT TXCSR IS NOT READ
8418                                ;;AND TXDBUF IS NOT LOADED (WRITTEN)
8419                                ;;THIS TEST CHECKS THE ONCE ONLY FLIP/FLOP (V2)
8420                                ;;OF THE INTERRUPT CONTROL LOGIC
8421                                ;;INTERRUPT VECTOR: DUTIV
8422                                ;;NOTE: TXDONE = 1 AFTER A MASTER RESET
8423                                ;
8424                                ;*****
(3) 005110 000004      TST6:   SCOPE
(3)
8425 005112 052777 000400 174606      BIS      #MRESET,@TXCSR  ;MASTER RESET
8426 005120 012777 005160 174614      MOV      #1$,@DUTIV      ;SET UP TRAPCATCHER
8427 005126 016777 011660 174610      MOV      DUPRT,@DUTIS    ;
8428 005134 106427 000000      MTPS     #0                ;ALLOW INTERRUPTS
8429 005140 052777 000100 174560      BIS      #TXINTE,@TXCSR  ;ENABLE INTR ENABLE
8430 005146 005000                CLR      R0
8431 005150 005200                INC      R0
8432 005152 001376                BNE     ,-2
8433 005154 104004                ERROR    4                ;INTERRUPT FAILED TO OCCUR
8434 005156 000425                BR       4$
8435                                ;THE FOLLOWING IS THE INTR SVC ROUTINE
8436 005160 106427 000300      1$:     MTPS      #300      ;DON'T ALLOW ANYMORE INTR
8437 005164 012716 005224      MOV      #3$,(SP)        ;SET UP RETURN LOCATION
8438 005170 012777 005200 174544      MOV      #2$,@DUTIV      ;SET UP TRAPCATCHER TO
8439                                ;PROVE THAT THE INTERRUPT DOES NOT OCCUR
8440                                ;TWICE (AFTER RTI 'ING FROM THIS
8441                                ;SVC ROUTINE
8442 005176 000002                RTI
8443                                ;THE FOLLOWING INTERRUPT SVC ROUTINE WILL CATCH THE SECOND INTR
8444 005200 106427 000300      2$:     MTPS      #300      ;DON'T ALLOW INTER
8445 005204 012716 005232      MOV      #4$,(SP)        ;SET UP RETURN LOCATION
8446 005210 105777 174512      TSTB    @TXCSR ;TXDONE = 1?

```

```

8447 005214 100401      BMI      .+4
8448 005216 104004      ERROR    4      ;TXDONE SHOULD BE SET
8449 005220 104004      ERROR    4      ;THE INTERRUPT WAS TAKEN TWICE.....
8450                               ;CHECK OUT THE V2 FLIP/FLOP LOGIC
8451                               ;IN THE INTERRUPT CONTROL LOGIC
8452 005222 000002      RTI
8453 005224 005000      3$: CLR    R0      ;ALLOW TIME TO CATCH SECOND
8454 005226 005200      INC    R0      ;IF IT WERE TO OCCUR
8455 005230 001376      BNE    -2
8456 005232 016777 174506 174502 4$: MOV    DUTIS,@DUTIV ;RESTORE TRAPCATCHER
8457 005240 012777 000000 174476      MOV    #0,@DUTIS
8458 005246 042777 000100 174452      BIC    #TXINTE,@TXCSR ;CLR INTERRUPT ENABLE
8459 005254 106427 000300      MTPS   #300    ;RESTORE NO INTERRUPT STATUS
8460
8461
8462                               ::THIS TEST VERIFYS THAT TWO INTERRUPTS THAT TRAP
8463                               ::TO THE SAME VECTOR ARE BOTH EXECUTED
8464                               ::INTERRUPT VECTOR: DUTIV
8465                               ::MODE: SYNC EXTERNAL
8466
8467                               ::*****
(3) 005260 000004      TST7:  SCOPE
(3)
8468
8469 005262 052777 000400 174436      BIS    #MRESET,@TXCSR ;MASTER RESET
(1) 005270 012777 020000 174424      MOV    #SYNEXT,@PARCSR ;SET THE MODE
(2) 005276 052777 000400 174422      BIS    #MRESET,@TXCSR ;MASTER RESET
(1)
(1)                               ;SET MAINTENANCE MODE & SEND
(1)                               ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 005304 012777 004020 174414      MOV    #MINT!SEND,@TXCSR
(1)
(1)                               ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 005312 012777 026026 174402      MOV    #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8470 005320 112777 000025 174404      MOV    #25,@TXDBUF ;LOAD CHARACTER
8471 005326 012767 000010 173566      MOV    #8,SHIFT
8472                               ;POKE CLK TO GET INTO SYNCHRONIZATION
8473 005334 052777 020000 174364      BIS    #CLK,@TXCSR ;POKE CLK UP
(1) 005342 042777 020000 174356      BIC    #CLK,@TXCSR ;POKE CLK DOWN
8474
8475 005350      1$:
(1) 005350 052777 020000 174350      BIS    #CLK,@TXCSR ;POKE CLK UP
(1) 005356 042777 020000 174342      BIC    #CLK,@TXCSR ;POKE CLK DOWN
8476 005364 005367 173532      DEC    SHIFT ;LAST SHIFT?
8477 005370 001367      BNE    1$
8478 005372 012777 005432 174342      MOV    #2$,@DUTIV ;SET UP TRAPCATCHER
8479 005400 016777 011406 174336      MOV    DUPRT,@DUTIS
8480 005406 106427 000000      MTPS   #0 ;ALLOW INTERRUPTS
8481 005412 052777 000140 174306      BIS    #TXINTE!DNAINTE,@TXCSR ;ENABLE INTERRUPTS
8482 005420 005000      CLR    R0
8483 005422 005200      INC    R0 ;WAIT FOR INTERRUPT
8484 005424 001376      BNE    -2
8485 005426 104004      ERROR  4 ;INTERRUPT FAILED TO OCCUR
8486 005430 000461      BR    3$ ;JUMP AROUND SVC ROUTINES
8487
8488                               ;THE FOLLOWING IS THE 1ST INTERRUPT SVC ROUTINE
  
```



```

8489 005432 106427 000300      2$:  MTPS    #300    ;DON'T ALLOW ANYMORE INTERRUPTS
8490 005436 005777 174264      TST     @TXCSR  ;DNA=0 ?
8491 005442 100001      BPL     .+4
8492 005444 104004      ERROR   4      ;DNA SHOULD NOT BE ASSERTED
8493 005446 105777 174254      TSTB    @TXCSR  ;TXDONE = 1?
8494 005452 100401      BMI     .+4
8495 005454 104004      ERROR   4      ;FALSE INTERRUPT
8496 005456 012716 005616      MOV     #4$, (SP) ;SET UP RETURN LOCATION
8497 005462 012777 005544 174252      MOV     #5$, @DUTIV ;SET UP TRAPCATCHER
8498                                ;NOW POKE CLK TO BRING UP DNA
8499 005470 052777 020000 174230      BIS     #CLK, @TXCSR ;POKE CLK
8500 005476 112777 000025 174226      MOVB    #25, @TXDBUF ;JUST LOAD ANY CHAR TO CLR
8501                                ;TXDONE TO ALLOW SECOND INTERRUPT
8502 005504 106427 000000      MTPS    #0      ;ALLOW INTERRUPTS
8503 005510 005000      CLR     R0
8504 005512 005200      INC     R0      ;WAIT FOR INTERRUPT
8505 005514 001376      BNE     .-2
8506 005516 042777 000140 174202      BIC     #DNAINTE!TXINTE, @TXCSR ;CLR INTR ENABLES
8507 005524 104004      ERROR   4      ;2ND INTERRUPT FAILED TO OCCUR
8508
8509 005526 016777 174212 174206 6$:  MOV     DUTIS, @DUTIV ;RESTORE TRAPCATCHER
8510 005534 012777 000000 174202      MOV     #0, @DUTIS ;
8511 005542 000002      RTI
8512
8513      ;THE FOLLOWING IS THE 2ND INTERRUPT SVC ROUTINE
8514 005544 106427 000300      5$:  MTPS    #300
8515 005550 005777 174152      TST     @TXCSR      ;DNA
8516 005554 100401      BMI     .+4
8517 005556 104004      ERROR   4      ;FALSE INTERRUPT
8518 005560 042777 000140 174140      BIC     #DNAINTE!TXINTE, @TXCSR ;CLR BOTH INTR ENABLES
8519 005566 012716 005526      MOV     #6$, (SP) ;SETUP RETURN LOCATION
8520 005572 000002      RTI
8521
8522 005574 016777 174144 174140 3$:  MOV     DUTIS, @DUTIV ;RESTORE TRAPCATCHER
8523 005602 012777 000000 174134      MOV     #0, @DUTIS
8524
8525 005610 042777 000140 174110      BIC     #DNAINTE!TXINTE, @TXCSR ;CLR BOTH INTERRUPT
8526                                ;ENABLES
8527 005616 106427 000300      4$:  MTPS    #300    ;RESTORE NO INTERRUPT STATUS
8528
8529
8530
8531      ;;THIS TEST VERIFYS CTP MODE (IE SYSTST MODE)
8532      ;;IT BASICALLY CHECKS THE EXISTANCE OF
8533      ;;THE FREE RUNNING OSCILLATOR
8534      ;;MODE:  SYNEXT
8535      ;;LENGTH:  EIGHT
8536      ;;THIS TEST USES BOTH THE RECEIVER & TRANSMITTER LOGIC
8537
8538      ;*****
8539      (3) 005622 0000C4      TST10: SCOPE
8540      ;*****
8541      (3)
8542 005624 000167 000262      ;*****
8543                                JMP     1$      ;NOP THIS TEST
8544      ;*****
8545 005630 052777 000400 174070      ;*****
8546                                BIS     #MRESET, @TXCSR ;MASTER RESET

```



```

8543 005636 012777 020000 174056      MOV      #SYNEXT,@PARCSR ;LOAD THE MODE
8544 005644 052777 000400 174054      BIS      #MRESET,@TXCSR ;MASTER RESET
8545 005652 012777 026026 174042      MOV      #SYNEXT!EIGHT!NOPAR!26,@PARCSR ;LOAD THE MODE,
8546                                     ;# OF BITS PER CHAR,PARITY SENSE(NO PARITY),
8547                                     ;&SYNC CHARACTER (26)
8548 005660 112777 000025 174044      MOV      #25,@TXDBUF ;LOAD THE CHAR
8549 005666 012777 005764 174042      MOV      #2$,@DURIV ;SET UP TRAPCATCHER
8550 005674 016777 011112 174036      MOV      DUPRT,@DURIS ;
8551 005702 106427 000000 ;ALLOW INTERRUPTS
8552 005706 016703 174004      MOV      RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
8553 005712 012700 000025      MOV      #25,R0 ;EXPECTED
8554 005716 012777 014020 174002      MOV      #SYSTST!SEND,@TXCSR ;OK NOW LOAD SEND &
8555                                     ;MAINT. MODE
8556 005724 052777 000120 173760      BIS      #SYNSCH!RINTEN,@RXCSR ;SET SEARCH SYNC &
8557                                     ;RECEIVER INTERRUPT
8558                                     ;ENABLE & WAIT FOR INTERRUPT
8559 005732 005067 173552      CLR      $TMP5
8560 005736 005002      3$: CLR      R2
8561 005740 005202      INC      R2 ;WAIT FOR INTERRUPT
8562 005742 001376      BNE      #-2
8563 005744 005267 173540      INC      $TMP5
8564 005750 022767 000003 173532      CMP      #3,$TMP5
8565 005756 002367      BGE      3$
8566 005760 104004      ERROR   4 ;INTERRUPT DID NOT OCCUR
8567 005762 000422      BR      4$
8568
8569 ;THE FOLLOWING IS THE INTERRUPT SVC ROUTINE
8570 005764 106427 000300      2$: MTPS   #300 ;PREVENT INTERRUPTS
8571 005770 017704 173716      MOV      @RXCSR,R4 ;SAVE
8572 005774 017701 173716      MOV      @RXDBUF,R1 ;ACTUAL
8573 006000 016777 173734 173730      MOV      DURIS,@DURIV ;RESTORE TRAPCATCHER
8574 006006 012777 000000 173724      MOV      #0,@DURIS ;
8575 006014 012716 006060      MOV      #5$,(SP) ;SET UP RETURN
8576 006020 042777 000100 173664      BIC      #RINTEN,@RXCSR ;CLR INTERRUPT ENABLE
8577 006026 000002      RTI
8578
8579 006030 042777 000100 173654      4$: BIC      #RINTEN,@RXCSR ;CLR INTERRUPT ENABLE
8580 006036 106427 000300      MTPS   #300 ;PREVENT INTERRUPTS
8581 006042 016777 173672 173666      MOV      DURIS,@DURIV ;RESTORE TRAPCATCHER
8582 006050 012777 000000 173662      MOV      #0,@DURIS ;
8583 006056 000415      BR      1$
8584
8585 006060 020001      5$: CMP      R0,R1
8586 006062 001401      BEQ     +4
8587 006064 104002      ERROR   2 ;CHARACTERS DID NOT MATCH
8588 006066 016703 173620      MOV      RXCSR,R3 ;SETUP FOR ERROR MESSAGE
8589 006072 012700 000200      MOV      #200,R0 ;EXPECTED
8590 006076 010401      MOV      R4,R1 ;ACTUAL
8591 006100 042701 177577      BIC      #177577,R1 ;SAVE ONLY RXDONE
8592 006104 020001      CMP      R0,R1
8593 006106 001401      BEQ     +4
8594 006110 104001      ERROR   1 ;FALSE INTERRUPT
8595
8596 006112      1$:
8597 ;:THIS TEST VERIFYS CTP MODE (IE SYSTST MODE)
8598 ;:IT BASICALLY CHECKS THE EXISTANCE OF

```

```

8599      ;;THE FREE RUNNING OSCILLATOR
8600      ;;MODE: SYNINT
8601      ;;LENGTH: EIGHT
8602      ;;THIS TEST USES BOTH THE RECEIVER & TRANSMITTER LOGIC
8603      ;;
8604      ;;*****
(3) 006112 000004      TST11: SCOPE
(3)
8605 006114 052777 000400 173604      BIS      #MRESET,@TXCSR ;MASTER RESET
(1) 006122 012777 030000 173572      MOV      #SYNINT,@PARCSR ;SET THE MODE
(2) 006130 052777 000400 173570      BIS      #MRESET,@TXCSR ;MASTER RESET
(1)
(1)      ;SET MAINTENANCE MODE & SEND
(1)      ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 006136 012777 014020 173562      MOV      #SYSTST!SEND,@TXCSR
(1)
(1)      ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 006144 012777 036026 173550      MOV      #SYNINT!EIGHT!NOPAR!26,@PARCSR
8606 006152 052777 000420 173532      BIS      #SYNSCH!STPSYN,@RXCSR ;SET SEARCH SYNC &
8607      ;STRIP SYNC SO THAT RXDONE ASSERTS
8608      ;WHEN CHAR '25' ARRIVES AND NOT BEFORE...
8609      ;...THEREFORE,SET STRIP SYNC
8610      ;...WAIT FOR SYNSCH TO BE
8611      ;CLOCKED IN BY SYSTST CLK
8612 006160 005067 173324      CLR      $TMP5
8613 006164 005002      CLR      R2
8614 006166 005202      INC      R2 ;WAIT
8615 006170 001376      BNE      #-2
8616 006172 005267 173312      INC      $TMP5
8617 006176 022767 000003 173304      CMP      #3,$TMP5
8618 006204 002367      BGE      #-20 ;GO BACK TO CLR R2 AND WAIT SOME MORE
8619 006206 012777 006422 173522      MOV      #2$,@DURIV ;SET UP TRAPCATCHER
8620 006214 016777 010572 173516      MOV      DUPRT,@DURIS
8621 006222 012777 006514 173512      MOV      #3$,@DUTIV
8622 006230 016777 010556 173506      MOV      DUPRT,@DUTIS
8623 006236 106427 000000      MTPS    #0 ;ALLOW INTERRUPTS
8624 006242 016703 173450      MOV      RXDBUF,R3 ;SET UP FOR ERROR MSG
8625 006246 012700 000025      MOV      #25,RO ;EXPECTED CHAR
8626 006252 012767 000002 172644      MOV      #2,COUNT ;# OF SYNC CHARS TO GET INTO
8627      ;SYNCRONIZATION
8628 006260 105767 172662      TSTB    SYNCNO ;TEST TO SEE HOW MANY SYNC CHARS NEEDED
8629 006264 100402      BMI     9$
8630 006266 005367 172632      DEC     COUNT ;MAKE IT ONE LESS
8631 006272 052777 000100 173412 9$:      BIS     #RINTEN,@RXCSR ;SET INTERRUPT ENABLES
8632 006300 052777 000100 173420      BIS     #TXINTE,@TXCSR
8633 006306 000167 000012      JMP     8$
8634      ;THE FIRST XMIT INTERRUPT SHOULD COME
8635 006312 112777 000026 173412 1$:      MOVB   #26,@TXDBUF ;LOAD SYNC CHAR
8636 006320 005067 173164      CLR     $TMP5
8637 006324 005002      8$:    CLR     R2 ;WAIT FOR INTERRUPT
8638 006326 005202      INC     R2
8639 006330 001376      BNE     #-2 ;
8640 006332 005267 173152      INC     $TMP5
8641 006336 022767 000003 173144      CMP     #3,$TMP5
8642 006344 002367      BGE     8$
8643 006346 106427 000300      MTPS   #300 ;PREVENT INTERRUPTS

```



```

8644 006352 042777 000100 173346      BIC    #TXINTE,@TXCSR  ;CLR INTR ENABLES
8645 006360 042777 000100 173324      BIC    #RINTEN,@RXCSR  ;
8646 006366 016777 173346 173342      MOV    DURIS,@DURIV    ;RESTORE TRAPCATCHER
8647 006374 012777 000000 173336      MOV    #0,@DURIS      ;
8648 006402 016777 173336 173332      MOV    DUTIS,@DUTIV   ;
8649 006410 012777 000000 173326      MOV    #0,@DUTIS     ;
8650 006416 104004                ERROR  4                ;TXDONE INTERRUPT FAILED TO OCCUR
8651 006420 000540                BR     7$              ;GET OUT OF THE TEST
8652
8653
8654 006422 106427 000300                ;THE FOLLOWING IS THE RECEIVER INTERRUPT SVC ROUTINE
2$:  MTPS   #300          ;PREVENT INTERRUPTS
      MOV   @RXCSR,R4     ;SAVE
      MOV   @RXDBUF,R1    ;ACTUAL
      MOV   DURIS,@DURIV  ;RESTORE TRAPCATCHER
      MOV   #0,@DURIS     ;
      MOV   DUTIS,@DUTIV  ;
      MOV   #0,@DUTIS     ;
      MOV   #4$,(SP)      ;SET UP RETURN LOCATION
      BIC   #RINTEN,@RXCSR ;CLR INTERRUPT ENABLES
      BIC   #TXINTE,@TXCSR ;
      MOV   COUNT,R5      ;SAVE COUNT
      RTI
8655 006426 017704 173260                ;END OF RECEIVER INTERRUPT SVC ROUTINE
8656 006432 017701 173260                ;.....THE FOLLOWING IS THE XMITTER INTERRUPT SVC ROUTINE
8657 006436 016777 173276 173272      3$:  DEC    COUNT
      BMI   5$
      MOV   #1$,(SP)      ;SET UP RETURN LOCATION
                               ;(LOAD SYNC CHARACTER AGAIN)
8658 006444 012777 000000 173266      RTI
8659 006452 016777 173266 173262      5$:  MOV   #6$,(SP)      ;SET UP RETURN LOCATION
      RTI
8660 006460 012777 000000 173256      ;END OF XMITTER INTERRUPT SVC ROUTINE
8661 006466 012716 006646                6$:  MOVB   #25,@TXDBUF  ;LOAD CHARACTER
      BIC   #TXINTE,@TXCSR ;CLR INTR ENABLE
8662 006472 042777 000100 173212      CLR   $TMP5
8663 006500 042777 000100 173220      10$:  CLR   R2           ;WAIT FOR INTERRUPT(RECEIVER)
      INC   R2
      BNE   -2           ;
      INC   $TMP5
8664 006506 016705 172412                CMP   #3,$TMP5
8665 006512 000002                BGE   10$
8666
8667
8668 006514 005367 172404                MTPS   #300          ;PREVENT INTERRUPTS
8669 006520 100403                BIC   #RINTEN,@RXCSR  ;CLR INTR ENABLE
8670 006522 012716 006312                MOV   DURIS,@DURIV    ;RESTORE TRAPCATCHER
8671
8672 006526 000002                MOV   #0,@DURIS      ;
8673 006530 012716 006536                MOV   DUTIS,@DUTIV   ;
8674 006534 000002                MOV   #0,@DUTIS     ;
8675
8676 006536 112777 000025 173166      ERROR  4                ;RECEIVER INTR FAILED TO OCCUR
8677 006544 042777 000100 173154      BR     7$              ;GET OUT OF TEST
8678 006552 005067 172732                4$:  CMP   R0,R1
      BEQ   +4
8679 006556 005002                ERROR  2                ;CHARACTERS DID NOT MATCH
8680 006560 005202                MOV   RXCSR,R3        ;SET UP FOR ERROR MSG
8681 006562 001376                MOV   #200,R0        ;EXPECTED RXDONE
8682 006564 005267 172720                MOV   R4,R1          ;ACTUAL
8683 006570 022767 000003 172712      BIC   #177577,R1     ;SAVE ONLY RXDONE
8684 006576 002367
8685 006600 106427 000300
8686 006604 042777 000100 173100      BIC   #TXINTE,@TXCSR  ;CLR INTR ENABLE
8687 006612 016777 173122 173116      MOV   DURIS,@DURIV    ;RESTORE TRAPCATCHER
8688 006620 012777 000000 173112      MOV   #0,@DURIS      ;
8689 006626 016777 173112 173106      MOV   DUTIS,@DUTIV   ;
8690 006634 012777 000000 173102      MOV   #0,@DUTIS     ;
8691 006642 104004                ERROR  4                ;RECEIVER INTR FAILED TO OCCUR
8692 006644 000426                BR     7$              ;GET OUT OF TEST
8693 006646 020001                4$:  CMP   R0,R1
      BEQ   +4
8694 006650 001401                ERROR  2                ;CHARACTERS DID NOT MATCH
8695 006652 104002                MOV   RXCSR,R3        ;SET UP FOR ERROR MSG
8696 006654 016703 173032                MOV   #200,R0        ;EXPECTED RXDONE
8697 006660 012700 000200                MOV   R4,R1          ;ACTUAL
8698 006664 010401                BIC   #177577,R1     ;SAVE ONLY RXDONE
8699 006666 042701 177577

```



```

8700 006672 020001      CMP      R0,R1
8701 006674 001401      BEQ      .+4
8702 006676 104001      ERROR   1      ;FALSE INTERRUPT
8703 006700 020527 177777    CMP      R5,#-1 ;WAS COUNT =-1 WHEN RECEIVER
8704                                ;INTERRUPTED ?
8705 006704 001401      BEQ      .+4
8706 006706 104004      ERROR   4      ;IF R5 IS GREATER THAN -1.....
8707                                ;THEN EITHER THE # OF SYNC STRAP IS WRONG
8708                                ;OR RXDONE IS OCCURING TOO SOON
8709 006710 026727 172210 177777    CMP      COUNT,#-1
8710 006716 001401      BEQ      .+4
8711 006720 104004      ERROR   4      ;IF THIS TEST FAILS,BUT THE ABOVE TEST
8712                                ;DOESN'T.....IT MAY BE THAT CLEARING
8713                                ;TXINTE IN THE RECEIVER SVC ROUTINE
8714                                ;IS NOT STOPPING TXDONE INTERRUPTS
8715 006722 106427 000300 7$:      MTPS    #300 ;INHIBIT INTERRUPTS
8716
8717
8718                                ::THIS TEST VERIFYS MATCH DETECT & DATA RDY
(1)                                ::FLAGS FOR EVERY POSSIBLE MATCH CHARACTER
(1)                                ::BY OBSERVING RECACT BIT
(1)                                ::IT WILL TAKE TWO SYNC * CHARACTERS TO GET RECACT BIT
(1)                                ::* DEPENDENT ON MONITOR .....
(1)                                ::IF ONE SYNC STRAP IS SELECTED ,IT WILL
(1)                                ::ONLY TAKE ONE SYNC CHARACTER BEFORE RECACT TO
(1)                                ::ASSERT
(1)                                ::MODE: SYNC INTERNAL
(1)                                ::LENGTH: FIVE
(1)                                ::SYNC CHARACTER FOR MATCH: B/C
(1)                                ::THIS TEST USES THE TRANSMITTER AND RECEIVER CHIPS
(5)                                ::*****
(4) 006726 000004      TST12: SCOPE
(4)
(2) 006730 052777 000400 172770      BIS      #MRESET,@TXCSR ;MASTER RESET
(1) 006736 016703 172754      MOV      RXDBUF,R3      ;SET UP FOR ERROR MESSAGE
(1)                                ;SET SYNC INTERNAL,FIVE,NO PARITY,0 SYNC REGISTER
(1) 006742 012704 030000      MOV      #SYNINT!FIVE!NOPAR,R4 ;CREATE PARAMETERS
(1) 006746 012777 004020 172752 6$:      MOV      #MINT!SEND,@TXCSR ;SET SEND & MAINT INTER
(1) 006754 010477 172742      MOV      R4,@PARCSR ;LOAD CSR
(1) 006760 052777 000020 172724      BIS      #SYNSCH,@RXCSR ;SET SYNC SEARCH
(1)                                ;POKE CLK TO GET INTO SYNCRONIZATION
(1)                                ;BOTH THE LOGIC & RECEIVER
(2) 006766 052777 020000 172732      BIS      #CLK,@TXCSR ;POKE CLK UP
(2) 006774 042777 020000 172724      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(1) 007002 110477 172724      MOV      R4,@TXDBUF ;LOAD DATA CHARACTER
(1)                                ;POKE CLK TO GET TRANSMITTER & RECEIVER INTO SYNCRONIZATION
(2) 007006 052777 020000 172712      BIS      #CLK,@TXCSR ;POKE CLK UP
(2) 007014 042777 020000 172704      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(1) 007022 032777 004000 172662      BIT      #RECACT,@RXCSR ;RECACT ?
(1) 007030 001401      BEQ      .+4
(1) 007032 104004      ERROR   4      ;RECACT SHOULD NOT BE SET
(1) 007034 000404      BR      4$
(1) 007036 010477 172660 5$:      MOV      R4,@PARCSR ;LOAD PARCSR WITH PARAMETERS
(1) 007042 110477 172664      MOV      R4,@TXDBUF ;LOAD SYNC CHAR
(1) 007046 012767 000002 172050 4$:      MOV      #2,COUNT ;# OF SYNC CHARS

```

INITIALIZE THE COMMON TAGS

```

(1) 007054 005777 172646      2$:  TST   @TXCSR  ;DNA ?
(1) 007060 100001              BPL   .+4     ;BR IF NOT SET
(1) 007062 104004              ERROR  4      ;DNA SHOULD NOT BE SET OR....
(1)                                ;IT SHOULD BE CLEARED FROM PREVIOUS READ
(1) 007064 012767 000005 172030      MOV   #5,SHIFT ;# OF SHIFTS
(1) 007072                                1$:
(2) 007072 052777 020000 172626      BIS   #CLK,@TXCSR ;POKE CLK UP
(2) 007100 042777 020000 172620      BIC   #CLK,@TXCSR ;POKE CLK DOWN
(1) 007106 005367 172010      DEC   SHIFT    ;# OF SHIFTS
(1) 007112 001367              BNE   1$
(1) 007114 005367 172004      DEC   COUNT    ;# OF SYNC CHARS
(1) 007120 001403              BEQ   3$
(1)                                ;TEST SYNCNO TO SEE HOW MANY SYNC CHARACTERS NEEDED
(1) 007122 105767 172020      TSTB  SYNCNO
(1) 007126 100752              BMI   2$      ;TWO SYNC CHARACTERS..
(1) 007130 032777 004000 172554      3$:  BIT   #REACT,@RXCSR ;REACT ?
(1) 007136 001001              BNE   .+4
(1) 007140 104004              ERROR  4      ;REACT FAILED TO SET,POSSIBLE
(1)                                ;THAT THE RECEIVER FAILED TO MATCH
(1)                                ;THE SYNC CHARACTER
(1) 007142 017701 172550      MOV   @RXDBUF,R1 ;SAVE ACTUAL
(1) 007146 010400              MOV   R4,R0     ;SAVE EXPECTED
(1) 007150 042700 177400      BIC   #177400,R0 ;CLR UPPER BYTE
(1) 007154 020001              CMP   R0,R1    ;DO THEY COMPARE ?
(1) 007156 001401              BEQ   .+4
(1) 007160 104002              ERROR  2      ;IF REACT FAILED ALONG WITH THIS
(1)                                ;...IT PROBABLY IS A TRANSMITTER ERROR
(1)                                ;HOWEVER,...IF ONLY THIS FAILED IT
(1)                                ;PROBABLY IS A RECEIVER ERROR
(1) 007162 104405              SCOPI
(1)                                ;POKE CLK TO SEE DNA...DNA COMES UP ON THE FIRST
(1)                                ;BIT OF THE NEXT CHARACTER IF NO CHARACTER IS LOADED INTO
(1)                                ;TXDBUF
(1) 007164 052777 020000 172534      BIS   #CLK,@TXCSR ;POKE CLK UP
(1) 007172 005777 172530      TST   @TXCSR  ;DNA?
(1) 007176 100401              BMI   .+4
(1) 007200 104004              ERROR  4      ;DNA DID NOT ASSERT
(1)                                ;SET UP CONDITIONS FOR NEXT SYNC CHARACTER MATCH
(2) 007202 052777 000400 172516      BIS   #MRESET,@TXCSR ;MASTER RESET
(1) 007210 032777 000020 172474      BIT   #SYNSCH,@RXCSR ;SYNC SEARCH = 0 ?
(1) 007216 001401              BEQ   .+4
(1) 007220 104004              ERROR  4      ;SYNC SEARCH SHOULD BE NOT SET
(1) 007222 005204              INC   R4
(1) 007224 122704 000040      CMPB  #40,R4   ;IS THIS THE LAST CHARACTER ?
(1) 007230 001246              BNE   6$      ;NO
(1)                                ;: THIS TEST VERIFYS MATCH DETECT & DATA RDY
(1)                                ;: FLAGS FOR EVERY POSSIBLE MATCH CHARACTER
(1)                                ;: BY OBSERVING REACT BIT
(1)                                ;: IT WILL TAKE TWO SYNC * CHARACTERS TO GET REACT BIT
(1)                                ;: * DEPENDENT ON MONITOR .....
(1)                                ;: IF ONE SYNC STRAP IS SELECTED ,IT WILL
(1)                                ;: ONLY TAKE ONE SYNC CHARACTER BEFORE REACT TO
(1)                                ;: ASSERT
(1)                                ;: MODE: SYNC INTERNAL
(1)                                ;: LENGTH: SIX

```

8719





```

(1)                                     ;PROBABLY IS A RECEIVER ERROR
(1) 007466 104405
(1)                                     SCOPE1
(1)                                     ;POKE CLK TO SEE DNA...DNA COMES UP ON THE FIRST
(1)                                     ;BIT OF THE NEXT CHARACTER IF NO CHARACTER IS LOADED INTO
(1) TXDBUF
(1) 007470 052777 020000 172230      BIS    #CLK,@TXCSR    ;POKE CLK UP
(1) 007476 005777 172224              TST    @TXCSR    ;DNA?
(1) 007502 100401                      BMI    .+4
(1) 007504 104004                      ERROR  4          ;DNA DID NOT ASSERT
(1)                                     ;SET UP CONDITIONS FOR NEXT SYNC CHARACTER MATCH
(2) 007506 052777 000400 172212      BIS    #MRESET,@TXCSR ;MASTER RESET
(1) 007514 032777 000020 172170      BIT    #SYNSCH,@RXCSR ;SYNC SEARCH = 0 ?
(1) 007522 001401                      BEQ    .+4
(1) 007524 104004                      ERROR  4          ;SYNC SEARCH SHOULD BE NOT SET
(1) 007526 005204                      INC    R4
(1) 007530 122704 000100              CMPB   #100,R4    ;IS THIS THE LAST CHARACTER ?
(1) 007534 001246                      BNE    6$        ;NO
(1)
8720
(1)                                     ::THIS TEST VERIFYS MATCH DETECT & DATA RDY
(1)                                     ::FLAGS FOR EVERY POSSIBLE MATCH CHARACTER
(1)                                     ::BY OBSERVING RECACT BIT
(1)                                     ::IT WILL TAKE TWO SYNC * CHARACTERS TO GET RECACT BIT
(1)                                     ::* DEPENDENT ON MONITOR .....
(1)                                     ::IF ONE SYNC STRAP IS SELECTED ,IT WILL
(1)                                     ::ONLY TAKE ONE SYNC CHARACTER BEFORE RECACT TO
(1)                                     ::ASSERT
(1)                                     ::MODE: SYNC INTERNAL
(1)                                     ::LENGTH: SEVEN
(1)                                     ::SYNC CHARACTER FOR MATCH: B/C
(1)                                     ::THIS TEST USES THE TRANSMITTER AND RECEIVER CHIPS
(1)
(5)
(4) 007536 000004
(4)
(2) 007540 052777 000400 172160      BIS    #MRESET,@TXCSR ;MASTER RESET
(1) 007546 016703 172144              MOV    RXDBUF,R3    ;SET UP FOR ERROR MESSAGE
(1)                                     ;SET SYNC INTERNAL,SEVEN,NO PARITY,0 SYNC REGISTER
(1) 007552 012704 034000              MOV    #SYNINT!SEVEN!NOPAR,R4 ;CREATE PARAMETERS
(1) 007556 012777 004020 172142      6$:   MOV    #MINT!SEND,@TXCSR    ;SET SEND & MAINT INTER
(1) 007564 010477 172132              MOV    R4,@PARCSR   ;LOAD CSR
(1) 007570 052777 000020 172114      BIS    #SYNSCH,@RXCSR ;SET SYNC SEARCH
(1)                                     ;POKE CLK TO GET INTO SYNCHRONIZATION
(1)                                     ;BOTH THE LOGIC & RECEIVER
(2) 007576 052777 020000 172122      BIS    #CLK,@TXCSR   ;POKE CLK UP
(2) 007604 042777 020000 172114      BIC    #CLK,@TXCSR   ;POKE CLK DOWN
(1) 007612 110477 172114              MOVB   R4,@TXDBUF   ;LOAD DATA CHARACTER
(1)                                     ;POKE CLK TO GET TRANSMITTER & RECEIVER INTO SYNCHRONIZATION
(2) 007616 052777 020000 172102      BIS    #CLK,@TXCSR   ;POKE CLK UP
(2) 007624 042777 020000 172074      BIC    #CLK,@TXCSR   ;POKE CLK DOWN
(1) 007632 032777 004000 172052      BIT    #RECACT,@RXCSR ;RECACT ?
(1) 007640 001401                      BEQ    .+4
(1) 007642 104004                      ERROR  4          ;RECACT SHOULD NOT BE SET
(1) 007644 000404                      BR     4$
(1) 007646 010477 172050              5$:   MOV    R4,@PARCSR   ;LOAD PARCSR WITH PARAMETERS
(1) 007652 110477 172054              MOVB   R4,@TXDBUF   ;LOAD SYNC CHAR
(1) 007656 012767 000002 171240      4$:   MOV    #2,COUNT    ;# OF SYNC CHARS

```



INITIALIZE THE COMMON TAGS

```

(1) 007664 005777 172036      2$:   TST   @TXCSR   ;DNA ?
(1) 007670 100001              BPL   .+4       ;BR IF NOT SET
(1) 007672 104004              ERROR  4         ;DNA SHOULD NOT BE SET OR....
(1)                                ;IT SHOULD BE CLEARED FROM PREVIOUS READ
(1) 007674 012767 000007 171220      MOV   #7,SHIFT   ;# OF SHIFTS
(1) 007702                      1$:
(2) 007702 052777 020000 172016      BIS   #CLK,@TXCSR ;POKE CLK UP
(2) 007710 042777 020000 172010      BIC   #CLK,@TXCSR ;POKE CLK DOWN
(1) 007716 005367 171200              DEC   SHIFT     ;# OF SHIFTS
(1) 007722 001367              BNE   1$
(1) 007724 005367 171174              DEC   COUNT     ;# OF SYNC CHARS
(1) 007730 001403              BEQ   3$
(1)                                ;TEST SYNCNO TO SEE HOW MANY SYNC CHARACTERS NEEDED
(1) 007732 105767 171210              TSTB  SYNCNO
(1) 007736 100752              BMI   2$       ;TWO SYNC CHARACTERS..
(1) 007740 032777 004000 171744      3$:   BIT   #REACT,@RXCSR ;REACT ?
(1) 007746 001001              BNE   .+4
(1) 007750 104004              ERROR  4         ;REACT FAILED TO SET,POSSIBLE
(1)                                ;THAT THE RECEIVER FAILED TO MATCH
(1)                                ;THE SYNC CHARACTER
(1) 007752 017701 171740              MOV   @RXDBUF,R1 ;SAVE ACTUAL
(1) 007756 010400              MOV   R4,R0      ;SAVE EXPECTED
(1) 007760 042700 177400              BIC   #177400,R0 ;CLR UPPER BYTE
(1) 007764 020001              CMP   R0,R1     ;DO THEY COMPARE ?
(1) 007766 001401              BEQ   .+4
(1) 007770 104002              ERROR  2         ;IF REACT FAILED ALONG WITH THIS
(1)                                ;...IT PROBABLY IS A TRANSMITTER ERROR
(1)                                ;HOWEVER,...IF ONLY THIS FAILED IT
(1)                                ;PROBABLY IS A RECEIVER ERROR
(1) 007772 104405
(1)                                SCOPI
(1)                                ;POKE CLK TO SEE DNA...DNA COMES UP ON THE FIRST
(1)                                ;BIT OF THE NEXT CHARACTER IF NO CHARACTER IS LOADED INTO
(1)                                ;TXDBUF
(1) 007774 052777 020000 171724      BIS   #CLK,@TXCSR ;POKE CLK UP
(1) 010002 005777 171720              TST   @TXCSR   ;DNA?
(1) 010006 100401              BMI   .+4
(1) 010010 104004              ERROR  4         ;DNA DID NOT ASSERT
(1)                                ;SET UP CONDITIONS FOR NEXT SYNC CHARACTER MATCH
(2) 010012 052777 000400 171706      BIS   #MRESET,@TXCSR ;MASTER RESET
(1) 010020 032777 000020 171664      BIT   #SYNSCH,@RXCSR ;SYNC SEARCH = 0 ?
(1) 010026 001401              BEQ   .+4
(1) 010030 104004              ERROR  4         ;SYNC SEARCH SHOULD BE NOT SET
(1) 010032 005204              INC   R4
(1) 010034 122704 000200      CMPB  #200,R4   ;IS THIS THE LAST CHARACTER ?
(1) 010040 001246              BNE   6$       ;NO
(1)
8721                                ;;THIS TEST VERIFYS MATCH DETECT & DATA RDY
(1)                                ;;FLAGS FOR EVERY POSSIBLE MATCH CHARACTER
(1)                                ;;BY OBSERVING REACT BIT
(1)                                ;;IT WILL TAKE TWO SYNC * CHARACTERS TO GET REACT BIT
(1)                                ;;* DEPENDENT ON MONITOR .....
(1)                                ;;IF ONE SYNC STRAP IS SELECTED ,IT WILL
(1)                                ;;ONLY TAKE ONE SYNC CHARACTER BEFORE REACT TO
(1)                                ;;ASSERT
(1)                                ;;MODE: SYNC INTERNAL
(1)                                ;;LENGTH: EIGHT
  
```





```

(1)                                     ;PROBABLY IS A RECEIVER ERROR
(1) 010276 104405                      SCOPE1
(1)                                     ;POKE CLK TO SEE DNA...DNA COMES UP ON THE FIRST
(1)                                     ;BIT OF THE NEXT CHARACTER IF NO CHARACTER IS LOADED INTO
(1) TXDBUF
(1) 010300 052777 020000 171420      BIS #CLK,@TXCSR ;POKE CLK UP
(1) 010306 005777 171414              TST @TXCSR ;DNA?
(1) 010312 100401                      BMI .+4
(1) 010314 104004                      ERROR 4 ;DNA DID NOT ASSERT
(1)                                     ;SET UP CONDITIONS FOR NEXT SYNC CHARACTER MATCH
(2) 010316 052777 000400 171402      BIS #MRESET,@TXCSR ;MASTER RESET
(1) 010324 032777 000020 171360      BIT #SYNSCH,@RXCSR ;SYNC SEARCH = 0 ?
(1) 010332 001401                      BEQ .+4
(1) 010334 104004                      ERROR 4 ;SYNC SEARCH SHOULD BE NOT SET
(1) 010336 005204                      INC R4
(1) 010340 122704 000000              CMPB #0,R4 ;IS THIS THE LAST CHARACTER ?
(1) 010344 001246                      BNE 6$ ;NO
(1)
8722                                     ;; THIS TEST PERFORMS A BINARY COUNT DATA PATTERN ON
(1)                                     ;; BOTH THE TRANSMITTER AND RECEIVER LOGIC
(1)                                     ;; MODE:SYNC EXTERNAL (SYNEXT)
(1)                                     ;; LENGTH:EIGHT PLUS PARITY
(1)                                     ;; PARITY:EVEPAR
(1)                                     ;; MAINT. MODE:MINT
(1)                                     ;;
(5) *****
(4) 010346 000004                      TST16: SCOPE
(4)
(3) 010350 052777 000400 171350      BIS #MRESET,@TXCSR ;MASTER RESET
(2) 010356 012777 020000 171336      MOV #SYNEXT,@PARCSR ;SET THE MODE
(3) 010364 052777 000400 171334      BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2)                                     ;SET MAINTENANCE MODE & SEND
(2)                                     ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 010372 012777 004020 171326      MOV #MINT!SEND,@TXCSR
(2)
(2)                                     ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 010400 012777 027426 171314      MOV #SYNEXT!EIGHT!EVEPAR!26,@PARCSR
(1) 010406 016703 171304              MOV RXDBUF,R3 ;SETUP FOR ERROR MSG
(1) 010412 005004                      CLR R4 ;FOR DATA CHAR CREATION
(1) 010414 110477 171312              MOVB R4,@TXDBUF ;LOAD CHARACTER
(1) 010420 052777 000020 171264      BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
(1)                                     ;GET INTO SYNCHRONIZATION
(2) 010426 052777 020000 171272      BIS #CLK,@TXCSR ;POKE CLK UP
(2) 010434 042777 020000 171264      BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 010442 012767 000011 170452      1$: MOV #9,SHIFT ;# OF SHIFTS
(1) 010450 010400                      MOV R4,R0 ;EXPECTED
(1) 010452
(2) 010452 052777 020000 171246      2$: BIS #CLK,@TXCSR ;POKE CLK UP
(2) 010460 042777 020000 171240      BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 010466 005367 170430              DEC SHIFT ;# OF SHIFTS
(1) 010472 022767 000003 170422      CMP #3,SHIFT ;TIME TO LOAD NEXT CHAR ?
(1) 010500 001003                      BNE 3$ ;NO ?
(1) 010502 005204                      INC R4 ;GENERATE NEXT CHAR
(1) 010504 110477 171222              MOVB R4,@TXDBUF ;LOAD NEXT CHARACTER
(1) 010510 005767 170406              3$: TST SHIFT ;IS IT 0 ?

```



```

(1) 010514 001356          BNE      2$
(1) 010516 105777 171170  TSTB    @RXCSR ;RXDONE = 1 ?
(1) 010522 100401          BMI     .+4
(1) 010524 104004          ERROR   4      ;RXDONE SHOULD BE SET
(1) 010526 017701 171164  MOV     @RXDBUF,R1 ;ACTUAL
(1) 010532 020001          CMP     R0,R1  ;COMPARE EXP VS ACT
(1) 010534 001401          BEQ     .+4
(1) 010536 104002          ERROR   2      ;CHARACTERS SHOULD COMPARE
(1) 010540 105704          TSTB    R4      ;LAST CHARACTER ?
(1) 010542 001337          BNE     1$      ;NO
(1) 010544
8723 4$:
(1)      ;:THIS TEST PERFORMS A BINARY COUNT DATA PATTERN ON
(1)      ;:BOTH THE TRANSMITTER AND RECEIVER LOGIC
(1)      ;:MODE:SYNC EXTERNAL (SYNEXT)
(1)      ;:LENGTH:EIGHT PLUS PARITY
(1)      ;:PARITY:ODDPAR
(1)      ;:MAINT. MODE:MINT
(1)      ;:
(5)      ;:*****
(4) 010544 000004  TST17: SCOPE
(4)
(3) 010546 052777 000400 171152  BIS     #MRESET,@TXCSR ;MASTER RESET
(2) 010554 012777 020000 171140  MOV     #SYNEXT,@PARCSR ;SET THE MODE
(3) 010562 052777 000400 171136  BIS     #MRESET,@TXCSR ;MASTER RESET
(2)
(2)      ;SET MAINTENANCE MODE & SEND
(2)      ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 010570 012777 004020 171130  MOV     #MINT!SEND,@TXCSR
(2)
(2)      ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 010576 012777 027026 171116  MOV     #SYNEXT!EIGHT!ODDPAR!26,@PARCSR
(1) 010604 016703 171106  MOV     RXDBUF,R3 ;SETUP FOR ERROR MSG
(1) 010610 005004          CLR     R4      ;FOR DATA CHAR CREATION
(1) 010612 110477 171114  MOVB   R4,@TXDBUF ;LOAD CHARACTER
(1) 010616 052777 000020 171066  BIS     #SYNSCH,@RXCSR ;SET SEARCH SYNC
(1)
(2) 010624 052777 020000 171074  ;GET INTO SYNCHRONIZATION
(2) 010632 042777 020000 171066  BIS     #CLK,@TXCSR ;POKE CLK UP
(1) 010640 012767 000011 170254 1$: BIC     #CLK,@TXCSR ;POKE CLK DOWN
(1) 010646 010400          MOV     #9,SHIFT ;# OF SHIFTS
(1) 010650          MOV     R4,R0 ;EXPECTED
(2) 010650 052777 020000 171050 2$: BIS     #CLK,@TXCSR ;POKE CLK UP
(2) 010656 042777 020000 171042  BIC     #CLK,@TXCSR ;POKE CLK DOWN
(1) 010664 005367 170232          DEC     SHIFT ;# OF SHIFTS
(1) 010670 022767 000003 170224  CMP     #3,SHIFT ;TIME TO LOAD NEXT CHAR ?
(1) 010676 001003          BNE     3$      ;NO ?
(1) 010700 005204          INC     R4      ;GENERATE NEXT CHAR
(1) 010702 110477 171024  MOVB   R4,@TXDBUF ;LOAD NEXT CHARACTER
(1) 010706 005767 170210 3$: TST     SHIFT ;IS IT 0 ?
(1) 010712 001356          BNE     2$
(1) 010714 105777 170772  TSTB    @RXCSR ;RXDONE = 1 ?
(1) 010720 100401          BMI     .+4
(1) 010722 104004          ERROR   4      ;RXDONE SHOULD BE SET
(1) 010724 017701 170766  MOV     @RXDBUF,R1 ;ACTUAL
(1) 010730 020001          CMP     R0,R1  ;COMPARE EXP VS ACT
(1) 010732 001401          BEQ     .+4
  
```

INITIALIZE THE COMMON TAGS

```
(1) 010734 104002          ERROR 2      ;CHARACTERS SHOULD COMPARE
(1) 010736 105704          TSTB  R4      ;LAST CHARACTER ?
(1) 010740 001337          BNE   1$      ;NO
(1) 010742
8724 4$:
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(5)
(4) 010742 000004          ;*****
(4) TST20: SCOPE
(1) 010744 105767 170203    TSTB  JMRBY   ;JUMP AROUND TEST ?
(1) 010750 100116          BPL   4$      ;YES ?
(3) 010752 052777 000400 170746  BIS   #MRESET,@TXCSR ;MASTER RESET
(2) 010760 012777 020000 170734  MOV   #SYNEXT,@PARCSR ;SET THE MODE
(3) 010766 052777 000400 170732  BIS   #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 010774 012777 010020 170724  MOV   #MEXT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 011002 012777 027426 170712  MOV   #SYNEXT!EIGHT!EVEPAR!26,@PARCSR
(1) 011010 016703 170702          MOV   RXDBUF,R3 ;SETUP FOR ERROR MSG
(1) 011014 005004          CLR   R4      ;FOR DATA CHAR CREATION
(1) 011016 110477 170710          MOVB  R4,@TXDBUF ;LOAD CHARACTER
(1) 011022 052777 000020 170662  BIS   #SYNSCH,@RXCSR ;SET SEARCH SYNC
(1)
(2) 011030 052777 020000 170670  ;GET INTO SYNCHRONIZATION
(2) BIS   #CLK,@TXCSR ;POKE CLK UP
(2) ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011036 016702 170056          MOV   HOLD,R2 ;WAIT THIS AMT
(2) 011042 005302          DEC   R2      ;WAIT
(2) 011044 001376          BNE   -2
(2) ;EXIT...
(2) 011046 042777 020000 170652  BIC   #CLK,@TXCSR ;POKE CLK DOWN
(2) ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011054 016702 170040          MOV   HOLD,R2 ;WAIT THIS AMT
(2) 011060 005302          DEC   R2      ;WAIT
(2) 011062 001376          BNE   -2
(2) ;EXIT...
(1) 011064 012767 000011 170030  1$: MOV   #9,SHIFT ;# OF SHIFTS
(1) 011072 010400          MOV   R4,R0   ;EXPECTED
(1) 011074
(2) 011074 052777 020000 170624  2$: BIS   #CLK,@TXCSR ;POKE CLK UP
(2) ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011102 016702 170012          MOV   HOLD,R2 ;WAIT THIS AMT
(2) 011106 005302          DEC   R2      ;WAIT
(2) 011110 001376          BNE   -2
(2) ;EXIT...
(2) 011112 042777 020000 170606  BIC   #CLK,@TXCSR ;POKE CLK DOWN
(2) ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011120 016702 167774          MOV   HOLD,R2 ;WAIT THIS AMT
(2) 011124 005302          DEC   R2      ;WAIT
```



```

(2) 011126 001376      BNE      .-2
(2)                   ;EXIT...
(1) 011130 005367 167766  DEC      SHIFT      ;# OF SHIFTS
(1) 011134 022767 000003 167760  CMP      #3,SHIFT    ;TIME TO LOAD NEXT CHAR ?
(1) 011142 001003      BNE      3$         ;NO ?
(1) 011144 005204      INC      R4         ;GENERATE NEXT CHAR
(1) 011146 110477 170560  MOVB     R4,@TXDBUF ;LOAD NEXT CHARACTER
(1) 011152 005767 167744 3$:      TST      SHIFT    ;IS IT 0 ?
(1) 011156 001346      BNE      2$
(1) 011160 105777 170526  TSTB    @RXCSR    ;RXDONE = 1 ?
(1) 011164 100401      BMI      .+4
(1) 011166 104004      ERROR    4         ;RXDONE SHOULD BE SET
(1) 011170 017701 170522  MOV      @RXDBUF,R1 ;ACTUAL
(1) 011174 020001      CMP      R0,R1     ;COMPARE EXP VS ACT
(1) 011176 001401      BEQ      .+4
(1) 011200 104002      ERROR    2         ;CHARACTERS SHOULD COMPARE
(1)                   ;CHECK OUT MODEM BYPASS JUMPER
(1) 011202 105704      TSTB    R4         ;LAST CHARACTER ?
(1) 011204 001327      BNE      1$         ;NO
(1) 011206
8725 4$:
(1)                   ;; THIS TEST PERFORMS A BINARY COUNT DATA PATTERN ON
(1)                   ;; BOTH THE TRANSMITTER AND RECEIVER LOGIC
(1)                   ;; MODE:SYNC EXTERNAL (SYNEXT)
(1)                   ;; LENGTH:EIGHT PLUS PARITY
(1)                   ;; PARITY:ODDPAR
(1)                   ;; MAINT. MODE:MEXT
(1)                   ;;
(5)                   ;*****
(4) 011206 000004      TST21: SCOPE
(4)
(1) 011210 105767 167737  TSTB    JMRBY     ;JUMP AROUND TEST ?
(1) 011214 100116      BPL     4$         ;YES ?
(3) 011216 052777 000400 170502  BIS     #MRESET,@TXCSR ;MASTER RESET
(2) 011224 012777 020000 170470  MOV     #SYNEXT,@PARCSR ;SET THE MODE
(3) 011232 052777 000400 170466  BIS     #MRESET,@TXCSR ;MASTER RESET
(2)
(2)                   ;SET MAINTENANCE MODE & SEND
(2)                   ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 011240 012777 010020 170460  MOV     #MEXT!SEND,@TXCSR
(2)
(2)                   ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 011246 012777 027026 170446  MOV     #SYNEXT!EIGHT!ODDPAR!26,@PARCSR
(1) 011254 016703 170436  MOV     RXDBUF,R3   ;SETUP FOR ERROR MSG
(1) 011260 005004      CLR     R4         ;FOR DATA CHAR CREATION
(1) 011262 110477 170444  MOVB    R4,@TXDBUF ;LOAD CHARACTER
(1) 011266 052777 000020 170416  BIS     #SYNSCH,@RXCSR ;SET SEARCH SYNC
(1)                   ;GET INTO SYNCHRONIZATION
(2) 011274 052777 020000 170424  BIS     #CLK,@TXCSR ;POKE CLK UP
(2)                   ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011302 016702 167612  MOV     HOLD,R2    ;WAIT THIS AMT
(2) 011306 005302      DEC     R2         ;WAIT
(2) 011310 001376      BNE     .-2
(2)                   ;EXIT...
(2) 011312 042777 020000 170406  BIC     #CLK,@TXCSR ;POKE CLK DOWN
(2)                   ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011320 016702 167574  MOV     HOLD,R2    ;WAIT THIS AMT
  
```

```

(2) 011324 005302      DEC      R2      ;WAIT
(2) 011326 001376      BNE      -2
(2)                   ;EXIT...
(1) 011330 012767 000011 167564 1$:  MOV      #9,SHIFT ;# OF SHIFTS
(1) 011336 010400      MOV      R4,R0 ;EXPECTED
(1) 011340
(2) 011340 052777 020000 170360 2$:  BIS      #CLK,@TXCSR ;POKE CLK UP
(2)                   ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011346 016702 167546      MOV      HOLD,R2 ;WAIT THIS AMT
(2) 011352 005302      DEC      R2      ;WAIT
(2) 011354 001376      BNE      -2
(2)                   ;EXIT...
(2) 011356 042777 020000 170342  BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2)                   ;WAIT FOR CABLE & DRIVER DELAYS
(2) 011364 016702 167530      MOV      HOLD,R2 ;WAIT THIS AMT
(2) 011370 005302      DEC      R2      ;WAIT
(2) 011372 001376      BNE      -2
(2)                   ;EXIT...
(1) 011374 005367 167522      DEC      SHIFT ;# OF SHIFTS
(1) 011400 022767 000003 167514  CMP      #3,SHIFT ;TIME TO LOAD NEXT CHAR ?
(1) 011406 001003      BNE      3$      ;NO ?
(1) 011410 005204      INC      R4      ;GENERATE NEXT CHAR
(1) 011412 110477 170314      MOV      R4,@TXDBUF ;LOAD NEXT CHARACTER
(1) 011416 005767 167500 3$:  TST      SHIFT ;IS IT 0 ?
(1) 011422 001346      BNE      2$
(1) 011424 105777 170262      TST      @RXCSR ;RXDONE = 1 ?
(1) 011430 100401      BMI      +4
(1) 011432 104004      ERROR   4 ;RXDONE SHOULD BE SET
(1) 011434 017701 170256      MOV      @RXDBUF,R1 ;ACTUAL
(1) 011440 020001      CMP      R0,R1 ;COMPARE EXP VS ACT
(1) 011442 001401      BEQ     +4
(1) 011444 104002      ERROR   2 ;CHARACTERS SHOULD COMPARE
(1)                   ;CHECK OUT MODEM BYPASS JUMPER
(1) 011446 105704      TST      R4      ;LAST CHARACTER ?
(1) 011450 001327      BNE      1$      ;NO
(1) 011452
8726 4$:
(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)                   ;; (OVRUN,RXERR)
(1)                   ;; MODE: ISYMOD
(1)                   ;; LENGTH: FIVE
(1)                   ;; CHAR: 12
(1)                   ;;
(5) *****
(4) 011452 000004      TST2:  SCOPE
(4)
(3) 011454 052777 000400 170244      BIS      #MRESET,@TXCSR ;MASTER RESET
(2) 011462 012777 000000 170232      MOV      #ISYMOD,@PARCSR ;SET THE MODE
(3) 011470 052777 000400 170230      BIS      #MRESET,@TXCSR ;MASTER RESET
(2)
(2)                   ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 011476 012777 064001 170222      MOV      #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)                   ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 011504 012777 000000 170210      MOV      #ISYMOD!FIVE!NOPAR!0,@PARCSR

```



```

(2) 011512 052777 000020 170172      BIS      #SYNSCH,@PXCSR ;SET SYNC SEARCH
(2)                                ;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
(2) 011520 042777 020000 170200      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 011526 052777 020000 170172      BIS      #CLK,@TXCSR ;POKE CLK UP
(2)                                ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 011534 042777 020000 170164      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 011542 052777 020000 170156      BIS      #CLK,@TXCSR ;POKE CLK UP
(1) 011550 016703 170142              MOV      RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 011554 012700 000012              MOV      #12,R0 ;EXPECTED
(1) 011560 012767 000007 167334      MOV      #7,SHIFT ;# OF SHIFTS
(1) 011566 012767 000124 167704      MOV      #124,$TMP1 ;DATA CHAR
(1) 011574 004767 005352              JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011600 105777 170106              TSTB    @RXCSR ;RXDONE ?
(1) 011604 100401                      BMI     .+4
(1) 011606 104004                      ERROR   4 ;RXDONE SHOULD BE SET
(1) 011610 017701 170102              MOV      @RXDBUF,R1 ;ACTUAL
(1) 011614 020001                      CMP     R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 011616 001401                      BEQ    .+4
(1) 011620 104002                      ERROR   2 ;RECEIVED DATA DID NOT MATCH
(1)                                ;EXPECTED DATA - CHECK MAINT DATA
(1)                                ;OR RECEIVER LOGIC
(1) 011622 012767 000007 167272      MOV      #7,SHIFT ;# OF SHIFTS
(1) 011630 012767 000124 167642      MOV      #124,$TMP1 ;DATA CHAR
(1) 011636 004767 005310              JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1)                                ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 011642 012767 000007 167252      MOV      #7,SHIFT ;# OF SHIFTS
(1) 011650 012767 000124 167622      MOV      #124,$TMP1 ;DATA CHAR
(1) 011656 004767 005270              JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 011662 012700 140012              MOV      #140000!12,R0 ;EXPECTED DATA PLUS
(1)                                ;RXERR & OVRRUN
(1) 011666 017701 170024              MOV      @RXDBUF,R1 ;ACTUAL
(1) 011672 020001                      CMP     R0,R1 ;COMPARE EXP VS. ACT
(1) 011674 001401                      BEQ    .+4
(1) 011676 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:ISYMOD
(1)                                ;; LENGTH:FIVE
(1)                                ;; CHAR:37
(1)                                ;;
(5)                                ;*****
(4) 011700 000004                      TST23: SCOPE
(4)
(3) 011702 052777 000400 170016      BIS      #MRESET,@TXCSR ;MASTER RESET
(2) 011710 012777 000000 170004      MOV      #ISYMOD,@PARCSR ;SET THE MODE
(3) 011716 052777 000400 170002      BIS      #MRESET,@TXCSR ;MASTER RESET
(2)
(2)                                ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 011724 012777 064001 167774      MOV      #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)                                ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 011732 012777 000000 167762      MOV      #ISYMOD!FIVE!NOPAR!0,@PARCSR

```

```

(2) 011740 052777 000020 167744      BIS      #SYNSCH,@RXCSR ;SET SYNC SEARCH
(2)                                ;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
(2) 011746 042777 020000 167752      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 011754 052777 020000 167744      BIS      #CLK,@TXCSR ;POKE CLK UP
(2)                                ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 011762 042777 020000 167736      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 011770 052777 020000 167730      BIS      #CLK,@TXCSR ;POKE CLK UP
(1) 011776 016703 167714                MOV      RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 012002 012700 000037                MOV      #37,R0 ;EXPECTED
(1) 012006 012767 000007 167106      MOV      #7,SHIFT ;# OF SHIFTS
(1) 012014 012767 000176 167456      MOV      #176,$TMP1 ;DATA CHAR
(1) 012022 004767 005124                JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 012026 105777 167660                TSTB    @RXCSR ;RXDONE ?
(1) 012032 100401                        BMI      .+4
(1) 012034 104004                        ERROR   4 ;RXDONE SHOULD BE SET
(1) 012036 017701 167654                MOV      @RXDBUF,R1 ;ACTUAL
(1) 012042 020001                        CMP      R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 012044 001401                        BEQ     .+4
(1) 012046 104002                        ERROR   2 ;RECEIVED DATA DID NOT MATCH
(1)                                ;EXPECTED DATA - CHECK MAINT DATA
(1)                                ;OR RECEIVER LOGIC
(1) 012050 012767 000007 167044      MOV      #7,SHIFT ;# OF SHIFTS
(1) 012056 012767 000176 167414      MOV      #176,$TMP1 ;DATA CHAR
(1) 012064 004767 005062                JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1)                                ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 012070 012767 000007 167024      MOV      #7,SHIFT ;# OF SHIFTS
(1) 012076 012767 000176 167374      MOV      #176,$TMP1 ;DATA CHAR
(1) 012104 004767 005042                JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 012110 012700 140037                MOV      #140000!37,R0 ;EXPECTED DATA PLUS
(1)                                ;RXERR & OVRRUN
(1) 012114 017701 167576                MOV      @RXDBUF,R1 ;ACTUAL
(1) 012120 020001                        CMP      R0,R1 ;COMPARE EXP VS. ACT
(1) 012122 001401                        BEQ     .+4
(1) 012124 104002                        ERROR   2 ;SPECIFICALLY LOOK AT RXERR &
(1)                                ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
8728                                ;;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:ISYMOD
(1)                                ;;LENGTH:FIVE
(1)                                ;;CHAR:0
(1)                                ;;
(5)                                ;*****
(4) 012126 000004                        TST24: SCOPE
(4)
(3) 012130 052777 000400 167570      BIS      #MRESET,@TXCSR ;MASTER RESET
(2) 012136 012777 000000 167556      MOV      #ISYMOD,@PARCSR ;SET THE MODE
(3) 012144 052777 000400 167554      BIS      #MRESET,@TXCSR ;MASTER RESET
(2)
(2)                                ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(2) 012152 012777 064001 167546      MOV      #MTDATA!CLK!MINT!BREAK,@TXCSR
(2)
(2)                                ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(2) 012160 012777 000000 167534      MOV      #ISYMOD!FIVE!NOPAR!0,@PARCSR

```



```

(2) 012166 052777 000020 167516      BIS      #SYNSCH,@RXCSR ;SET SYNC SEARCH
(2)                                ;POKE CLK TO GET RECEIVER INTO SYNCROIZATION....
(2) 012174 042777 020000 167524      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 012202 052777 020000 167516      BIS      #CLK,@TXCSR ;POKE CLK UP
(2)                                ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(2) 012210 042777 020000 167510      BIC      #CLK,@TXCSR ;POKE CLK DOWN
(2) 012216 052777 020000 167502      BIS      #CLK,@TXCSR ;POKE CLK UP
(1) 012224 016703 167466                MOV      RXDBUF,R3 ;SET UP FOR ERROR MESSAGE
(1) 012230 012700 000000                MOV      #0,R0 ;EXPECTED
(1) 012234 012767 000007 166660        MOV      #7,SHIFT ;# OF SHIFTS
(1) 012242 012767 000100 167230        MOV      #100,$TMP1 ;DATA CHAR
(1) 012250 004767 004676                JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 012254 105777 167432                TSTB    @RXCSR ;RXDONE ?
(1) 012260 100401                        BMI      .+4
(1) 012262 104004                        ERROR   4 ;RXDONE SHOULD BE SET
(1) 012264 017701 167426                MOV      @RXDBUF,R1 ;ACTUAL
(1) 012270 020001                        CMP      R0,R1 ;COMPARE EXPECTED VS. ACTUAL
(1) 012272 001401                        BEQ     .+4
(1) 012274 104002                        ERROR   2 ;RECEIVED DATA DID NOT MATCH
(1)                                ;EXPECTED DATA - CHECK MAINT DATA
(1)                                ;OR RECEIVER LOGIC
(1) 012276 012767 000007 166616        MOV      #7,SHIFT ;# OF SHIFTS
(1) 012304 012767 000100 167166        MOV      #100,$TMP1 ;DATA CHAR
(1) 012312 004767 004634                JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1)                                ;NOW SHIFT IN A SECOND CHARACTER WITHOUT READING RXDBUF
(1) 012316 012767 000007 166576        MOV      #7,SHIFT ;# OF SHIFTS
(1) 012324 012767 000100 167146        MOV      #100,$TMP1 ;DATA CHAR
(1) 012332 004767 004614                JSR      PC,RPOKE ;SHIFT IN THIS CHAR
(1) 012336 012700 140000                MOV      #140000!0,R0 ;EXPECTED DATA PLUS
(1)                                ;RXERR & OVRRUN
(1) 012342 017701 167350                MOV      @RXDBUF,R1 ;ACTUAL
(1) 012346 020001                        CMP      R0,R1 ;COMPARE EXP VS. ACT
(1) 012350 001401                        BEQ     .+4
(1) 012352 104002                        ERROR   2 ;SPECIFICALLY LOOK AT RXERR &
(1)                                ;OVRRUN BITS...THEY BOTH SHOULD BE SET
(1)
8729
(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) .EOP: SCOPE
(2) 012354 000004                        JSR      PC,CKSWR
(2) 012356 004767 000340                                ;TYPE NAME OF TEST
(2) 012362 104401                        TYPE
(2) 012364 015512                        MEPASS
(2) 012366 104413 012620                CONVRT  ,OUTCRY
(2) 012372 104401 015331                TYPE   ,DEVICE
(2) 012376 105767 166550                TSTB    MULTD ;ARE YOU RUNNING MULTIPLE DEVICES ?
(2) 012402 001511                        BEQ     CCC ;NO,JUMP AROUND
(2) 012404 005767 166556                TST     ACTREG ;ARE ANY DEVICES ACTIVE ?
(2) 012410 001007                        BNE     RUNIT ;YES
(2) 012412 104401 015343                TYPE   ,MCOW ;NO
(2) 012416 016700 166544                MOV      ACTREG,R0 ;DISPLAY ACTREG
(2) 012422 000000                        HALT    ;SELECT SOMETHING TO RUN @ ACTREG:

```

```

(2) ;SELECT SWITCHES & HIT CONTINUE (PUT SW00 =1)
(2) 012424 000167 167522 JMP .START ;START OVER AGAIN.....YOU DESELECTED EVERYTHING
(2) 012430 062767 000010 166516 RUNIT: ADD #10,BASEADD ;NEXT BLOCK (ADDRESSES)
(2) 012436 062767 000010 166516 ZERO: ADD #10,BASEIV ;NEXT BLOCK (VECTORS)
(2) 012444 000241 CLC
(2) 012446 006167 166516 ROL ROTADD ;UP DATE ROTATING POINTER
(2) 012452 103410 BCS 2$ ;IS IT THE LAST DEVICE
(2) ;TO BE TESTED IN THIS PASS ?
(2) 012454 036767 166510 166504 BIT ROTADD,ACTREG ;TEST THIS DEVICE FOR ACTIVE STATUS
(2) 012462 001762 BEQ RUNIT ;IF NOT ACTIVE, TRY NEXT ADDRESS
(2) 012464 004767 000034 JSR PC,REPLAY ;CALCULATE NEW PARAMETERS
(2) 012470 000167 000210 JMP RESTRT ;YES IT WAS ACTIVE,TEST THIS DEVICE
(2) 012474 012767 000001 166466 2$: MOV #1,ROTADD ;OK!,NOW SET UP ROTATING
(2) ;POINTER FOR NEXT MULTIPLE PASS
(2) 012502 016767 166450 166444 MOV KEEPADD,BASEADD ;RESTORE BASE ADDRESS
(2) 012510 016767 166450 166444 MOV KEEPIV,BASEIV ;RESTORE BASE INTERRUPT VECTORS
(2) 012516 004767 000002 JSR PC,REPLAY ;CALC NEW PARAMETERS
(2) 012522 000441 BR CCC ;JUMP AROUND REPLAY
(2) 012524 016767 166424 004416 REPLAY: MOV BASEADD,DUBASE ;SET UP FOR NEW ADDRESSES
(2) 012532 004767 004260 JSR PC,DUADDR ;CREATE NEW ADDRESSES
(2) 012536 016767 166420 167172 MOV BASEIV,DURIV ;CREATE DURIV
(2) 012544 062767 000002 166410 ADD #2,BASEIV
(2) 012552 016767 166404 167160 MOV BASEIV,DURIS ;CREATE DURIS
(2) 012560 062767 000002 166374 ADD #2,BASEIV
(2) 012566 016767 166370 167146 MOV BASEIV,DUTIV ;CREATE DUTIV
(2) 012574 062767 000002 166360 ADD #2,BASEIV
(2) 012602 016767 166354 167134 MOV BASEIV,DUTIS ;CREATE DUTIS
(2) 012610 016767 167122 166344 MOV DURIV,BASEIV ;RESTORE
(2) 012616 000207 RTS PC
(2)
(2) 012620 000001 OUTCRY: 1
(2) 012622 006 002 .BYTE 6,2
(2) 012624 001712 RXCSR
(2)
(2) 012626 CCC:
(2) 012626 005067 166550 CLR $TSTNM ;CLEAR TEST NUMBER
(2) 012632 005067 166560 CLR $ERRPC ;CLEAR LAST ERROR PC
(2) 012636 005067 166541 CLR $ERFLG ;CLEAR ERROR FLAG
(2) 012642 005267 166244 INC PASCNT ;UPDATE PASS COUNT
(2) 012646 016767 166240 166226 MOV PASCNT,LIGHTS ;DISPLAY PASS COUNT
(2) 012654 016767 166232 166652 MOV PASCNT,$PASS ;PASS COUNT TO APT
(2) 012662 013701 000042 MOV @#42,R1 ;CHECK FOR ACT-11 OR DDP
(2) 012666 001406 BEQ RESTRT ;IF NO CONTINUE TESTING
(2) 012670 000005 RESET
(2) 012672 000005 RESET
(2) 012674 004711 $ENDAD: JSR PC,(R1)
(2) 012676 000240 NOP
(2) 012700 000240 NOP
(2) 012702 000240 NOP
(2) 012704 RESTRT:
(2) 012704 012767 003376 166474 MOV #TST1+2,$LPADR ;LOAD LAST ADDR
(2) 012712 004767 000004 JSR PC,CKSWR
(2) 012716 000167 170366 JMP .BEGIN
(2)
(2) ;CHECK SWITCH REGISTER ROUTINE.
(2) ;CHECKS TO ALLOW FOR <^G> TO ALLOW

```



```
(2) ;THE CHANGING OF LOCATION 176
(2)
(2) 012722 005737 000042 CKSWR: TST @#42
(2) 012726 001040 BNE OUT
(2) 012730 022767 000176 166502 CMP #SWREG,SWR ;SOFTWARE SWR PRESENT?
(2) 012736 001034 BNE OUT ;NO--LEAVE
(2) 012740 105777 166500 TSTB @STKS ;CHECK TTY READY
(2) 012744 100031 BPL OUT ;NO--LEAVE
(2) 012746 017767 166474 000422 MOV @STKB,.MSG ;GET CHARACTER
(2) 012754 042767 177600 000414 BIC #177600,.MSG ;STRIP JUNK
(2) 012762 122767 000007 000406 CMPB #7,.MSG ;IS IT <^G> ?
(2) 012770 001017 BNE OUT ;NO
(2) 012772 104401 016117 TYPE ,MCNTG
(2) 012776 005137 013036 CNTLU: COM @#RDSW
(2) 013002 104401 016127 TYPE ,MMSWR
(2) 013006 104413 CONVRT
(2) 013010 013040 SWREGL
(2) 013012 104406 016140 INSTR,MMNEW
(2) 013016 104410 PARAM
(2) 013020 000000 0
(2) 013022 177777 177777
(2) 013024 000176 SWREG
(2) 013026 000 001 .BYTE 0,1
(2) 013030 005037 013036 OUT: CLR @#RDSW
(2) 013034 000207 RTS PC
(2) 013036 000000 RDSW: .WORD 0
(2) 013040 000001 SWREGL: 1
(2) 013042 006 002 .BYTE 6,2
(2) 013044 000176 SWREG
(2)
(1) 013046 000005 5
(2)
(2) ;CHECK FOR FREEZE ON CURRENT DATA
(2)
(2) 013050 004767 177646 .SCOP1: JSR PC,CKSWR
(2) 013054 032777 001000 166356 BIT #SW09,@SWR
(2) 013062 001402 BEQ 1$
(2) 013064 016716 166020 MOV LOCK,(SP)
(2) 013070 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
(2) ;* MESADR
(2) ;*
```

```

(2) 013072 105767 166361 $TYPE: TSTB $TFLG ::IS THERE A TERMINAL?
(2) 013076 100002 BPL 1$ ::BR IF YES
(2) 013100 000000 HALT ::HALT HERE IF NO TERMINAL
(2) 013102 000430 BR 3$ ::LEAVE
(2) 013104 010046 1$: MOV R0,-(SP) ::SAVE R0
(2) 013106 017600 000002 MOV @2(SP),R0 ::GET ADDRESS OF ASCIZ STRING
(2) 013112 122767 000001 166426 CMPB #APTENV,$ENV ::RUNNING IN APT MODE
(2) 013120 001011 BNE 62$ ::NO,GO CHECK FOR APT CONSOLE
(2) 013122 132767 000100 166417 BITB #APTSPOOL,$ENVM ::SPOOL MESSAGE TO APT
(2) 013130 001405 BEQ 62$ ::NO,GO CHECK FOR CONSOLE
(2) 013132 010067 000004 MOV R0,61$ ::SETUP MESSAGE ADDRESS FOR APT
(2) 013136 004767 164644 JSR PC,$ATY3 ::SPOOL MESSAGE TO APT
(2) 013142 000000 61$: .WORD 0 ::MESSAGE ADDRESS
(2) 013144 132767 000040 166375 62$: BITB #APTCSUP,$ENVM ::APT CONSOLE SUPPRESSED
(2) 013152 001003 BNE 60$ ::YES,SKIP TYPE OUT
(2) 013154 112046 2$: MOV (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
(2) 013156 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
(2) 013160 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
(2) 013162 012600 60$: MOV (SP)+,R0 ::RESTORE R0
(2) 013164 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
(2) 013170 000002 RTI ::RETURN
(2) 013172 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
(2) 013176 001430 BEQ 8$
(2) 013200 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
(2) 013204 001006 BNE 5$
(2) 013206 005726 TST (SP)+ ::POP <CR><LF> EQUIV
(2) 013210 104401 TYPE ::TYPE A CR AND LF
(2) 013212 001523 $CRLF
(2) 013214 105067 000130 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
(2) 013220 000755 BR 2$ ::GET NEXT CHARACTER
(2) 013222 004767 000056 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
(2) 013226 126726 166224 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
(2) 013232 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
(2) 013234 016746 166214 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
(2) 013240 105366 000001 7$: DECB 1(SP) ::AND THE NULL CHAR.
(2) 013244 002770 BLT 6$ ::DOES A NULL NEED TO BE TYPED?
(2) 013246 004767 000032 JSR PC,$TYPEC ::BR IF NO--GO POP THE NULL OFF OF STACK
(2) 013252 105367 000072 DECB $CHARCNT ::GO TYPE A NULL
(2) 013256 000770 BR 7$ ::DO NOT COUNT AS A COUNT
(2) ::LOOP
(2) ;HORIZONTAL TAB PROCESSOR
(2) 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
(2) 013264 004767 000014 9$: JSR PC,$TYPEC ::TYPE A SPACE
(2) 013270 132767 000007 000052 BITB #7,$CHARCNT ::BRANCH IF NOT AT
(2) 013276 001372 BNE 9$ ::TAB STOP
(2) 013300 005726 TST (SP)+ ::POP SPACE OFF STACK
(2) 013302 000724 BR 2$ ::GET NEXT CHARACTER
(2) 013304 105777 166140 $TYPEC: TSTB @STPS ::WAIT UNTIL PRINTER IS READY
(2) 013310 100375 BPL $TYPEC
(2) 013312 116677 000002 166132 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
(2) 013320 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
(2) 013326 001003 BNE 1$ ::BRANCH IF NO
(2) 013330 105067 000014 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT

```



```

(2) 013334 000406          BR          $TYPEX          ;;EXIT
(2) 013336 122766 000012 000002 1$:  CMPB    #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
(2) 013344 001402          BEQ     $TYPEX          ;;BRANCH IF YES
(2) 013346 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
(2) 013350 000000          $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
(2) 013352 000207          $TYPEX: RTS         PC

```

(2)  
(2)  
(2)  
(2)  
(2)  
(2)

:ASCII STRING INPUT ROUTINE

```

(2) 013354 017667 000000 000014 .INSTR: MOV    @(SP),.MSG      ;PICK UP MESSAGE
(2) 013362 062716 000002          ADD    #2,(SP)        ;JUMP AROUND MESSAGE FOR RTI
(2) 013366 105767 166154          TSTB  $ENV           ;APT CONTROL
(2) 013372 001036          BNE   INSTR2         ;YES NO TYPE
(2) 013374 104401          .INST1: TYPE
(2) 013376 000000          .MSG:  0
(2) 013400 012704 016152          MOV   #INBUF,R4      ;GET STARTING LOC OF INBUF
(2) 013404 012703 000007          MOV   #7,R3          ;MAX # OF CHARS
(2) 013410 105777 166030 1$:  TSTB  @$TKS          ;TTY FLAG
(2) 013414 100375          BPL   1$
(2) 013416 117714 166024          MOVB  @$TKB,(R4)     ;TAKE CHAR
(2) 013422 142714 000200          BICB  #200,(R4)     ;STRIP
(2) 013426 121427 000025          CMPB  (R4),#25      ;IS IT <^G>
(2) 013432 001760          BEQ   .INST1
(2) 013434 122427 000015          CMPB  (R4)+,#15     ;CHECK FOR CR
(2) 013440 001413          BEQ   INSTR2
(2) 013442 105777 166002 2$:  TSTB  @$TPS          ;TEST FLAG
(2) 013446 100375          BPL   2$
(2) 013450 117777 165772 165774          MOVB  @$TKB,$STPB   ;ECHO CHARACTER
(2) 013456 005303          DEC   R3             ;DID YOU TYPE TOO MANY CHARS ?
(2) 013460 001353          BNE   1$
(2) 013462 104401          .INSTE: TYPE
(2) 013464 015437          MQM   :?
(2) 013466 000742          BR    .INST1        ;RETRY
(2) 013470 000002          INSTR2: RTI

```

(2)  
(2)  
(2)

:CONVERT ASCII STRING TO OCTAL

```

(2) 013472 011605          .PARAM: MOV   (SP),R5 ;PUT CONTENTS OF SP INTO R5
(2) 013474 012567 000162          MOV   (R5)+,LOLIM   ;PUT LOW LIMIT INTO LOLIM
(2) 013500 012567 000160          MOV   (R5)+,HILIM   ;PUT HIGH LIMIT INTO HILIM
(2) 013504 012567 000156          MOV   (R5)+,DEVADR   ;PUT STORE LOC INTO DEVADR
(2) 013510 112567 000154          MOVB  (R5)+,LOBITS   ;PUT MASK INTO LOBITS
(2) 013514 112567 000151          MOVB  (R5)+,ADRCNT   ;PUT COUNT INTO ADRCNT
(2) 013520 010516          MOV   R5,(SP) ;RESTORE RETURN ADDR ON STACK FOR RTI
(2) 013522 005005          PARAM1: CLR   R5
(2) 013524 012704 016152          MOV   #INBUF,R4
(2) 013530 122714 000015          CMPB  #15,(R4)      ;CR ?
(2) 013534 001420          BEQ   PARERR        ;YOU TYPED CR TOO SOON !
(2) 013536 121427 000060 1$:  CMPB  (R4),#60      ;LOW LIMIT ASCII 0
(2) 013542 002415          BLT   PARERR
(2) 013544 121427 000067          CMPB  (R4),#67      ;HIGH LIMIT ASCII 7
(2) 013550 003012          BGT   PARERR
(2) 013552 142714 000060          BICB  #60,(R4)     ;CONVERT TO OCTAL

```

```

(2) 013556 152405            BISB      (R4)+,R5           ;STORE AWAY ITS AN OK CHAR
(2) 013560 122714 000015    CMPB      #15,(R4)         ;CR?
(2) 013564 001414            BEQ       LIMITS          ;NOW CHECK FOR HIGH &LOW LIMIT CONDS
(2) 013566 006305            ASL      R5                ;ALLOCATE ROOM FOR NEXT CHAR
(2) 013570 006305            ASL      R5
(2) 013572 006305            ASL      R5
(2) 013574 000760            BR       1$
(2) 013576 122714 000015    PARERR:  CMPB      #15,(R4)         ;CR?
(2) 013602 001003            BNE      120$
(2) 013604 005737 013036    TST      @#RDSW           ;CK SWR USED
(2) 013610 001023            BNE      PARTI
(2) 013612 104407            120$:    INSTER   ;RETRY
(2) 013614 000742            BR       PARAM1

(2)
(2) ;TEST TO SEE IF NUMBER IS WITHIN LIMITS
(2)
(2) 013616 020567 000042    LIMITS:  CMP      R5,HILIM
(2) 013622 101365            BHI      PARERR          ;THE # IS TOO HIGH
(2) 013624 020567 000032    CMP      R5,LOLIM
(2) 013630 103762            BLO      PARERR          ;THE # IS TOO LOW
(2) 013632 136705 000032    BITB    LOBITS,R5        ;TEST BY MASKINGTHE #
(2) 013636 001357            BNE      PARERR

(2) ;STORE NUMBER AT SPECIFIED ADDRESS
(2)
(2) 013640 016704 000022    1$:     MOV      DEVADR,R4        ;GET STARTING ADDR OF
(2) 013644 010524            MOV      R5,(R4)+        ;STORE AT THIS ADDR
(2) 013646 062705 000002    ADD      #2,R5
(2) 013652 105367 000013    DECB    ADCRNT           ;HOW MANY TIMES + 2 ?
(2) 013656 001372            BNE      1$
(2) 013660 000002            PARTI:  RTI
(2) 013662 000000            LOLIM:  0
(2) 013664 000000            HILIM:  0
(2) 013666 000000            DEVADR: 0
(2) 013670 000000            LOBITS: 0
(2)                    ADRCNT=L0BITS+1

(2) ;SAVE PC OF TEST THAT FAILED AND R0-R5
(2)
(2) 013672 016667 000004 165226 .SAV05: MOV    4(SP),SAVPC
(2)
(2) ;SAVE R0-R5
(2)
(2) 013700 010567 165570    SV05:   MOV      R5,$REG5
(2) 013704 010467 165562    MOV      R4,$REG4
(2) 013710 010367 165554    MOV      R3,$REG3
(2) 013714 010267 165546    MOV      R2,$REG2
(2) 013720 010167 165540    MOV      R1,$REG1
(2) 013724 010067 165532    MOV      R0,$REG0
(2) 013730 000002            RTI

(2) ;RESTORE R0-R5
(2)
(2) 013732 016700 165524    .RES05: MOV      $REG0,R0
(2) 013736 016701 165522    MOV      $REG1,R1
(2) 013742 016702 165520    MOV      $REG2,R2

```



```

(2) 013746 016703 165516      MOV      $REG3,R3
(2) 013752 016704 165514      MOV      $REG4,R4
(2) 013756 016705 165512      MOV      $REG5,R5
(2) 013762 000002              RTI

                                ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
(2)
(2)
(2) 013764 104401      .CONVR: TYPE
(2) 013766 015443      MCRLF      ;CR LF
(2) 013770 017601 000000      MOV      @(SP),R1      ;PICK UP DATA POINTER
(2) 013774 062716 000002      ADD      #2,(SP) ;SET UP SP FOR RTI
(2) 014000 012167 000130      MOV      (R1)+,WRDCNT ;PICK UP # OF WORDS FROM TABLE
(2) 014004 112167 000126      1$:      MOV      (R1)+,CHRCNT ;PICK UP # OF CHARS FROM TABLE
(2) 014010 112167 000123      MOV      (R1)+,SPACNT ;PICK UP # OF SPACES FROM TABLE
(2) 014014 013167 000120      MOV      @(R1)+,BINWRD ;PICK UP ADDRESS OF MSG
(2)                                ;FROM TABLE
(2) 014020 016704 000114      2$:      MOV      BINWRD,R4      ;SAVE
(2) 014024 116705 000106      MOV      CHRCNT,R5      ;SAVE
(2) 014030 012700 016214      MOV      #TEMP,R0      ;STARTING ADDRESS OF TEMP BLOCK
(2) 014034 010403      3$:      MOV      R4,R3      ;SAVE
(2) 014036 042703 177770      BIC      #177770,R3 ;CLR OUT UPPER BITS .. SAVE CHAR
(2) 014042 062703 000260      ADD      #260,R3 ;CONVERT TO ASCII
(2) 014046 110320      MOV      R3,(R0)+      ;STORE AWAY
(2) 014050 006204      ASR      R4      ;SHIFT FOR NEXT #
(2) 014052 006204      ASR      R4      ;DITTO
(2) 014054 006204      ASR      R4      ;DITTO
(2) 014056 005305      DEC      R5      ;DEC CHAR COUNT
(2) 014060 001365      BNE      3$      ;DO IT AGAIN ?
(2) 014062 012703 016256      MOV      #MDATA,R3 ;STARTING ADDRESS OF MDATA BLOCK
(2) 014066 114023      4$:      MOV      -(R0),(R3)+ ;REVERSE THE ORDER OF NUMBERS
(2) 014070 105367 000042      DECB    CHRCNT ;DEC CHAR COUNT
(2) 014074 001374      BNE      4$      ;DO IT AGAIN ?
(2) 014076 105767 000035      TSTB    SPACNT ;HOW MANY SPACES ?
(2) 014102 001405      BEQ      6$      ;TYPE # IF BR =0
(2) 014104 112723 000240      5$:      MOV      #240,(R3)+ ;"SPACE" IN ASCII
(2) 014110 105367 000023      DECB    SPACNT ;DEC # OF SPACE COUNT
(2) 014114 001373      BNE      5$      ;DO IT AGAIN ?
(2) 014116 105013      6$:      CLRB    (R3) ;INSERT '0' FOR TTY OUTPUT ROUTINE
(2) 014120 104401      TYPE
(2) 014122 016256      MDATA      ;THIS MESSAGE
(2) 014124 005367 000004      DEC      WRDCNT ;HOW MANY #'S ?
(2) 014130 001325      BNE      1$      ;DO THIS ROUTINE AGAIN IF NOT EQUAL TO 0
(2) 014132 000002      RTI      ;RETURN TO PROGRAM
(2) 014134 000000      WRDCNT: 0
(2) 014136 000000      CHRCNT: 0
(2)                                SPACNT=CHRCNT+1
(2) 014140 000000      BINWRD: 0
(2)
(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)
(2) 014142 017605 000000      .SETFLG:MOV  @(SP),R5
(2) 014146 122767 000116 001776  CMPB    #'N',INBJF ;IS IT 'N' ?
(2) 014154 001002      BNE      1$

```

```

(2) 014156 105015          CLRB   (R5)   ;000
(2) 014160 000406          BR     2$
(2) 014162 122767 000131 001762 1$:  CMPB   #'Y',INBUF   ;IS IT 'Y' ?
(2) 014170 001005          BNE    3$
(2) 014172 112715 177777          MOVB   #-1,(R5)     ;377
(2) 014176 062716 000002          ADD    #2,(SP)
(2) 014202 000002          RTI
(2) 014204 104407          3$:   INSTER  ;RETRY
(2) 014206 000755          BR     .SETFLG
(2)                                     .SBTTL  ERROR HANDLER ROUTINE
(2)
(2)                                     ::*****
(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)
(2) 014210          $ERROR:
(2) 014210 105267 165167 7$:   INCB   $ERFLG      ;;SET THE ERROR FLAG
(2) 014214 001775          BEQ    7$           ;;DON'T LET THE FLAG GO TO ZERO
(2) 014216 016777 165160 165216  MOV    $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(2) 014224 032777 002000 165206  BIT    #BIT10,@SWR    ;;BELL ON ERROR?
(2) 014232 001402          BEQ    1$           ;;NO - SKIP
(2) 014234 104401 001516          TYPE   $BELL        ;;RING BELL
(2) 014240 005267 165146          1$:   INC    $ERTTL     ;;COUNT THE NUMBER OF ERRORS
(2) 014244 011667 165146          MOV    (SP),$ERRPC   ;;GET ADDRESS OF ERROR INSTRUCTION
(2) 014250 162767 000002 165140  SUB    #2,$ERRPC
(2) 014256 117767 165134 165130  MOVB   @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(2) 014264 032777 020000 165146  BIT    #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
(2) 014272 001004          BNE    20$          ;;SKIP TYPEOUTS
(2) 014274 004767 000072          JSR    PC,SAVIT     ;;GO TO USER ERROR ROUTINE
(2) 014300 104401 001523          TYPE   .$CRLF
(2) 014304          20$:
(2) 014304 122767 000001 165234  CMPB   #APTENV,$ENV  ;;RUNNING IN APT MODE
(2) 014312 001007          BNE    2$           ;;NO,SKIP APT ERROR REPORT
(2) 014314 116767 165074 000004  MOVB   $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER
(2) 014322 004767 163470          JSR    PC,$ATY4     ;;REPORT FATAL ERROR TO APT
(2) 014326 000          21$:  .BYTE  0
(2) 014327 000          .BYTE  0
(2) 014330 000777          22$:  BR     22$          ;;APT ERROR LOOP
(2) 014332 005777 165102 2$:   TST    @SWR         ;;HALT ON ERROR
(2) 014336 100001          BPL    3$           ;;SKIP IF CONTINUE
(2) 014340 000000          HALT
(2) 014342 032777 001000 165070 3$:   BIT    #BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
(2) 014350 001402          BEQ    4$           ;;BR IF NO
(2) 014352 016716 165032          MOV    $LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
(2) 014356 005767 165132          4$:   TST    $ESCAPE     ;;CHECK FOR AN ESCAPE ADDRESS
(2) 014362 001402          BEQ    5$           ;;BR IF NONE
(2) 014364 016716 165124          MOV    $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(2) 014370          5$:

```



```

(2) 014370 000002          RTI          ;;RETURN
(2) 014372 010067 164532  SAVIT:  MOV      R0,HLD0
(2) 014376 010167 164530      MOV      R1,HLD1
(2) 014402 010267 164526      MOV      R2,HLD2
(2) 014406 010367 164524      MOV      R3,HLD3
(2) 014412 010467 164522      MOV      R4,HLD4
(2) 014416 010567 164520      MOV      R5,HLD5
(2) 014422 016767 164754 164514  MOV      $TSTNM,HLD6
(3)
(3)
(4)
(3)
(3)
(3)
(3)
(3)
(3)
(3) 014430          $ERRTYP:
(3) 014430 104401 001523      TYPE      , $CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(3) 014434 010046          MOV      R0,-(SP)          ;;SAVE R0
(3) 014436 005000          CLR      R0              ;;PICKUP THE ITEM INDEX
(3) 014440 153700 001414      BISB     @#$ITEMB,R0
(3) 014444 001004          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(4) 014446 016746 164744      MOV      $ERRPC,-(SP)    ;;TYPE THE PC OF THE ERROR
(4)
(4) 014452 104402          TYPOC     ;;SAVE $ERRPC FOR TYPEOUT
(3) 014454 000426          BR      6$              ;;ERROR ADDRESS
(3) 014456 005300 1$:      DEC      R0              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(3) 014460 006300          ASL     R0              ;;GET OUT
(3) 014462 006300          ASL     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(3) 014464 006300          ASL     R0              ;;WORK FOR THE ERROR TABLE
(3) 014466 062700 001652      ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(3) 014472 012067 000004      MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
(3) 014476 001404          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(3) 014500 104401          TYPE     ;;TYPE THE "ERROR MESSAGE"
(3) 014502 000000 2$:      .WORD   0              ;;"ERROR MESSAGE" POINTER GOES HERE
(3) 014504 104401 001523      TYPE     , $CRLF        ;;"CARRIAGE RETURN" & "LINE FEED"
(3) 014510 012067 000004 3$:      MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(3) 014514 001404          BEQ     5$              ;;SKIP TYPEOUT IF 0
(3) 014516 104401          TYPE     ;;TYPE THE "DATA HEADER"
(3) 014520 000000 4$:      .WORD   0              ;;"DATA HEADER" POINTER GOES HERE
(3) 014522 104401 001523      TYPE     , $CRLF        ;;"CARRIAGE RETURN" & "LINE FEED"
(3) 014526 011000 5$:      MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
(3) 014530 001004          BNE     7$              ;;GO TYPE THE DATA
(3) 014532 012600 6$:      MOV     (SP)+,R0        ;;RESTORE R0
(3)
(3) 014534 104401 001523      TYPE     , $CRLF        ;;"CARRIAGE RETURN" & "LINE FEED"
(3) 014540 000207          RTS     PC              ;;RETURN
(3) 014542 7$:
(4) 014542 013046          MOV     @ (R0)+,-(SP)    ;;SAVE @ (R0)+ FOR TYPEOUT
(4) 014544 104402          TYPOC     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(3) 014546 005710          TST     (R0)           ;;IS THERE ANOTHER NUMBER?
(3) 014550 001770          BEQ     6$              ;;BR IF NO
(3) 014552 104401 014560      TYPE     ,8$            ;;TYPE TWO(2) SPACES
(3) 014556 000771          BR      7$              ;;LOOP
(3) 014560 020040 000      8$:      .ASCIZ  / /            ;;TWO(2) SPACES
  
```

```

(3)          014564
(2)
(2)          .EVEN
(2)          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(3)          *****
(2)          *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(2)          *OCTAL (ASCII) NUMBER AND TYPE IT.
(2)          *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(2)          *CALL:
(2)          *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(2)          *      TYPOS   N                  ;;CALL FOR TYPEOUT
(2)          *      .BYTE  N                  ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(2)          *      .BYTE  M                  ;;M=1 OR 0
(2)          *                                          ;;1=TYPE LEADING ZEROS
(2)          *                                          ;;0=SUPPRESS LEADING ZEROS
(2)          *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(2)          *$TYPOS OR $TYPOC
(2)          *CALL:
(2)          *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(2)          *      TYPON   N                  ;;CALL FOR TYPEOUT
(2)          *
(2)          *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(2)          *CALL:
(2)          *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(2)          *      TYPOC   N                  ;;CALL FOR TYPEOUT
(2)
(2) 014564 017646 000000          $TYPOS: MOV      @ (SP),-(SP)          ;;PICKUP THE MODE
(2) 014570 116667 000001 000211 MOVB   1(SP), $OFILL          ;;LOAD ZERO FILL SWITCH
(2) 014576 112667 000207          MOVB   (SP)+, $OMODE+1        ;;NUMBER OF DIGITS TO TYPE
(2) 014602 062716 000002          ADD    #2, (SP)             ;;ADJUST RETURN ADDRESS
(2) 014606 000406          BR     $TYPON
(2) 014610 112767 000001 000171 $TYPOC: MOVB   #1, $OFILL          ;;SET THE ZERO FILL SWITCH
(2) 014616 112767 000006 000165 MOVB   #6, $OMODE+1        ;;SET FOR SIX(6) DIGITS
(2) 014624 112767 000005 000154 $TYPON: MOVB   #5, $OCNT          ;;SET THE ITERATION COUNT
(2) 014632 010346          MOV    R3, -(SP)           ;;SAVE R3
(2) 014634 010446          MOV    R4, -(SP)           ;;SAVE R4
(2) 014636 010546          MOV    R5, -(SP)           ;;SAVE R5
(2) 014640 116704 000145          MOVB   $OMODE+1, R4        ;;GET THE NUMBER OF DIGITS TO TYPE
(2) 014644 005404          NEG    R4
(2) 014646 062704 000006          ADD    #6, R4              ;;SUBTRACT IT FOR MAX. ALLOWED
(2) 014652 110467 000132          MOVB   R4, $OMODE          ;;SAVE IT FOR USE
(2) 014656 116704 000125          MOVB   $OFILL, R4          ;;GET THE ZERO FILL SWITCH
(2) 014662 016605 000012          MOV    12(SP), R5         ;;PICKUP THE INPUT NUMBER
(2) 014666 005003          CLR    R3                  ;;CLEAR THE OUTPUT WORD
(2) 014670 006105          1$:   ROL    R5              ;;ROTATE MSB INTO "C"
(2) 014672 000404          BR     3$                  ;;GO DO MSB
(2) 014674 006105          2$:   ROL    R5              ;;FORM THIS DIGIT
(2) 014676 006105          ROL    R5
(2) 014700 006105          ROL    R5
(2) 014702 010503          MOV    R5, R3
(2) 014704 006103          3$:   ROL    R3              ;;GET LSB OF THIS DIGIT
(2) 014706 105367 000076          DECB   $OMODE              ;;TYPE THIS DIGIT?
(2) 014712 100016          BPL    7$                  ;;BR IF NO
(2) 014714 042703 177770          BIC    #177770, R3         ;;GET RID OF JUNK
(2) 014720 001002          BNE    4$                  ;;TEST FOR 0
(2) 014722 005704          TST    R4                  ;;SUPPRESS THIS 0?
  
```



```

(2) 014724 001403      BEQ      5$      ;;BR IF YES
(2) 014726 005204      4$: INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
(2) 014730 052703 000060  BIS      #'0,R3    ;;MAKE THIS DIGIT ASCII
(2) 014734 052703 000040  5$: BIS      #' ,R3    ;;MAKE ASCII IF NOT ALREADY
(2) 014740 110367 000040  MOVB     R3,8$     ;;SAVE FOR TYPING
(2) 014744 104401 015004  TYPE     ,8$     ;;GO TYPE THIS DIGIT
(2) 014750 105367 000032  7$: DECB    $OCNT  ;;COUNT BY 1
(2) 014754 003347      BGT      2$      ;;BR IF MORE TO DO
(2) 014756 002402      BLT      6$      ;;BR IF DONE
(2) 014760 005204      INC      R4      ;;INSURE LAST DIGIT ISN'T A BLANK
(2) 014762 000744      BR       2$      ;;GO DO THE LAST DIGIT
(2) 014764 012605      6$: MOV     (SP)+,R5  ;;RESTORE R5
(2) 014766 012604      MOV     (SP)+,R4  ;;RESTORE R4
(2) 014770 012603      MOV     (SP)+,R3  ;;RESTORE R3
(2) 014772 016666 000002 000004  MOV     2(SP),4(SP) ;;SET THE STACK FOR RETURNING
(2) 015000 012616      MOV     (SP)+,(SP)
(2) 015002 000002      RTI                      ;;RETURN
(2) 015004      000      8$: .BYTE   0      ;;STORAGE FOR ASCII DIGIT
(2) 015005      000      .BYTE   0      ;;TERMINATOR FOR TYPE ROUTINE
(2) 015006      000      $OCNT: .BYTE  0      ;;OCTAL DIGIT COUNTER
(2) 015007      000      $OFILL: .BYTE  0      ;;ZERO FILL SWITCH
(2) 015010 000000      $OMODE: .WORD  0      ;;NUMBER OF DIGITS TO TYPE
(2)                      ;ENTER HERE ON POWER FAILURE
(2)
(2)
(2) 015012      $PWRDN:
(2) 015012 010046      .PFAIL: MOV     R0,-(SP)      ;SAVE R0-R5 ON PROCESSOR STACK
(2) 015014 010146      MOV     R1,-(SP)
(2) 015016 010246      MOV     R2,-(SP)
(2) 015020 010346      MOV     R3,-(SP)
(2) 015022 010446      MOV     R4,-(SP)
(2) 015024 010546      MOV     R5,-(SP)
(2) 015026 016746 162772  MOV     24,-(SP)
(2) 015032 010667 164060  MOV     SP,SAVSP      ;SAVE STACK POINTER
(2) 015036 012767 015050 162760  MOV     #RESTART,24   ;SET UP FOR POWER UP TRAP
(2) 015044 000000      HALT                      ;HALT ON POWER DOWN NORMAL
(2) 015046 000777      BR       .
(2)
(2)                      ;PROCESSOR WILL TRAP HERE WHEN POWER IS RESTORED
(2)
(2) 015050 016706 164042  RESTAR: MOV     SAVSP,SP      ;RESTORE STACK POINTER
(2) 015054 012605      MOV     (SP)+,R5      ;RESTORE R0-R5
(2) 015056 012604      MOV     (SP)+,R4
(2) 015060 012603      MOV     (SP)+,R3
(2) 015062 012602      MOV     (SP)+,R2
(2) 015064 012601      MOV     (SP)+,R1
(2) 015066 012600      MOV     (SP)+,R0
(2) 015070 012767 015012 162726  MOV     #.PFAIL,24    ;SET UP FOR POWER FAILURE
(2) 015076 106427 000300  MTPS    #300
(2) 015102 012706 001100  MOV     #STACK,SP
(2) 015106 005067 001102  CLR     TEMP
(2) 015112 005267 001076  INC     TEMP
(2) 015116 001375      BNE     .-4
(2) 015120 104413      CONVRT
(2) 015122 015144      PFTAB
(2) 015124 104401      TYPE
  
```

```

(2) 015126 015446
(2) 015130 005067 164247
(2) 015134 005067 164256
(2) 015140 000177 163740
(2) 015144 000001
(2) 015146 006 002
(2) 015150 000207
(2) 015152 005015 042012 053125
(2) 015160 030461 041440 042116
(2) 015166 053125 040455 052040
(2) 015174 050101 020105 020106
(2) 015202 005015 000
(2) 015205 015 053012 041505
(2) 015212 040440 042104 000055
(2) 015220 005015 051461 020124
(2) 015226 042504 035126 051040
(2) 015234 041505 041440 051123
(2) 015242 040440 042104 000055
(2) 015250 005015 052515 052114
(2) 015256 042040 053105 037440
(2) 015264 024040 020131 051117
(2) 015272 047040 026451 000
(2) 015277 015 046012 051501
(2) 015304 020124 042504 035126
(2) 015312 051040 041505 041440
(2) 015320 051123 040440 042104
(2) 015326 026522 000
(2) 015331 075 042504 044526
(2) 015336 042503 020040 000
(2) 015343 015 051412 046105
(2) 015350 041505 020124 047524
(2) 015356 051040 047125 040040
(2) 015364 041501 051124 043505
(2) 015372 000
(2) 015373 015 047412 043126
(2) 015400 047514 051072 052105
(2) 015406 050131 020105 040514
(2) 015414 052123 042040 053105
(2) 015422 051040 041530 051123
(2) 015430 040440 042104 026523
(2) 015436 000
(2) 015437 040 037440 000
(2) 015443 015 000012
(2) 015446 043120 044501 026114
(2) 015454 020040 042522 052123
(2) 015462 051101 020124 052101
(2) 015470 052040 051505 020124
(2) 015476 047111 050040 047522
(2) 015504 051107 051505 000123
(2) 015512 005015 047105 020104
(2) 015520 043117 050040 051501
(2) 015526 020123 040524 042520
(2) 015534 043040 000
(2) 015537 015 051012 000
(2) 015543 015 052012 051505
(2) 015550 020124 041520 000055
  
```

MPFAIL  
 CLR \$ERFLG  
 CLR \$ERRPC  
 JMP @RETURN  
 PFTAB: 1  
 .BYTE 6,2  
 RETURN  
 MTITLE: .ASCIZ <15><12><12>/DUV11 CNDUV-A TAPE F /<15><12>  
 MVECTO: .ASCIZ <15><12>/VEC ADD-/  
 MREGAD: .ASCIZ <15><12>/1ST DEV: REC CSR ADD-/  
 MMULT: .ASCIZ <15><12>/MULT DEV ? (Y OR N)-/  
 MLASTD: .ASCIZ <15><12>/LAST DEV: REC CSR ADDR-/  
 DEVICE: .ASCIZ /=DEVICE /  
 MCOW: .ASCIZ <15><12>/SELECT TO RUN @ACTREG/  
 MRANGE: .ASCIZ <15><12>/OVFLO:RETYPE LAST DEV RXCSR ADDS-/  
 MQM: .ASCIZ / ?/  
 MCRLF: .ASCIZ <15><12>  
 MPFAIL: .ASCIZ /PFAIL, RESTART AT TEST IN PROGRESS/  
 MEPASS: .ASCIZ <15><12>/END OF PASS TAPE F/  
 MR: .ASCIZ <15><12>/R/  
 MTSTPC: .ASCIZ <15><12>/TEST PC-/



(2)	015556	005015	047514	045503	MLOCK: .ASCIZ <15><12>/LOCK ON TEST? (Y OR N)-/
(2)	015564	047440	020116	052040	
(2)	015572	051505	037524	024040	
(2)	015600	020131	051117	047040	
(2)	015606	026451	000		
(2)	015611	015	021412	047440	MSYNC: .ASCIZ <15><12>/# OF SYNC CHARS SELECTED ( 1 OR 2)-/
(2)	015616	020106	054523	041516	
(2)	015624	041440	040510	051522	
(2)	015632	051440	046105	041505	
(2)	015640	042524	020104	020050	
(2)	015646	020061	051117	031040	
(2)	015654	026451	000		
(2)	015657	015	044412	020123	MWIRE6: .ASCIZ <15><12>/IS SEC XMIT SWITCH E55-2 IN? (Y OR N)-/
(2)	015664	042523	020103	046530	
(2)	015672	052111	051440	044527	
(2)	015700	041524	020110	032505	
(2)	015706	026465	020062	047111	
(2)	015714	020077	054450	047440	
(2)	015722	020122	024516	000055	
(2)	015730	005015	051511	051440	MWIRES: .ASCIZ <15><12>/IS SEC REC SWITCH E55-3 IN? (Y OR N)-/
(2)	015736	041505	051040	041505	
(2)	015744	051440	044527	041524	
(2)	015752	020110	032505	026465	
(2)	015760	020063	047111	020077	
(2)	015766	054450	047440	020122	
(2)	015774	024516	000055		
(2)	016000	005015	051511	047440	MWIRE4: .ASCIZ <15><12>/IS OPT CLR ENABLE SWITCH E55-1 IN? (Y OR N)-/
(2)	016006	052120	041440	051114	
(2)	016014	042440	040516	046102	
(2)	016022	020105	053523	052111	
(2)	016030	044103	042440	032465	
(2)	016036	030455	044440	037516	
(2)	016044	024040	020131	051117	
(2)	016052	047040	026451	000	
(2)	016057	015	005012	031510	MEXTJ: .ASCIZ <15><12><12>/H315 CONNECTOR ON ?(Y OR N)-/
(2)	016064	032461	041440	047117	
(2)	016072	042516	052103	051117	
(2)	016100	047440	020116	024077	
(2)	016106	020131	051117	047040	
(2)	016114	026451	000		
(2)	016117	015	020012	043536	MCNTG: .ASCIZ <15><12>/ ^G /
(2)	016124	020040	000		
(2)	016127	040	053523	036522	MMSWR: .ASCIZ / SWR= /
(2)	016134	020040	000040		
(2)	016140	020040	047040	053505	MMNEW: .ASCIZ / NEW= /
(2)	016146	020075	000040		
(2)					.EVEN
(2)					
(2)					:BUFFERS FOR INPUT-OUTPUT
(2)	016152	000000			INBUF: 0
(2)		016214			.=. +40
(2)	016214	000000			TEMP: 0
(2)		016256			.=. +40
(2)	016256	000000			MDATA: 0
(2)		016320			.=. +40

```

(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)
(3) 016320 $SCOPE:
(5) ;SCOPE LOOP AND INTERATION HANDLER
(5)
(5) 016320 .SCOPE:
(5) 016320 004767 174376 JSR PC,CKSWR
(5) 016324 005067 163066 CLR $ERRPC ;CLEAR LAST ERROR PC
(5) 016330 022716 003376 CMP #TST1+2,(SP) ;IS SCOPE AT BEGINING OF TEST 1?
(5) 016334 001422 BEQ $XTSTR ;YES NO LOOP.
(5)
(5) 016336 032777 040000 163074 TTST: BIT #BIT14,@SWR ;THIS CODE IS FOR TESTING FOR BIT 14
(5) 016344 001412 BEQ 1$ ;ON LSI WHICH SYSMAC CANNOT HANDLE
(5) 016346 016767 163030 163032 MOV $TSTNM,$LPADR
(5) 016354 000406 BR 1$
(5) 016356 105777 163062 TSTB @STKS ;KEYBOARD DONE?
(5) 016362 100123 BPL $OVER ;BR IF NO
(5) 016364 017766 163056 177776 MOV @STKB,-2(SP) ;CLEAR DONE BIT
(3) 016372 032777 040000 163040 1$: BIT #BIT14,@SWR ;LOOP ON PRESENT TEST?
(3) 016400 001114 BNE $OVER ;YES IF SW14=1
(3) ;#####START OF CODE FOR THE XOR TESTER#####
(3) 016402 000416 $XTSTR: BR 6$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
(3) ;THIS INSTRUCTION TO A "NOP" (NOP=240)
(3) 016404 013746 000004 MOV @#ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
(3) 016410 012737 016430 000004 MOV #5$,@#ERRVEC ;SET FOR TIMEOUT
(3) 016416 005737 177060 TST @#177060 ;TIME OUT ON XOR?
(3) 016422 012637 000004 MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
(3) 016426 000463 BR $SVLAD ;GO TO THE NEXT TEST
(3) 016430 022626 5$: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
(3) 016432 012637 000004 MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
(3) 016436 000423 BR 7$ ;LOOP ON THE PRESENT TEST
(3) 016440 6$:;#####END OF CODE FOR THE XOR TESTER#####
(3) 016440 032777 000400 162772 BIT #BIT08,@SWR ;LOOP ON SPEC. TEST?
(3) 016446 001404 BEQ 2$ ;BR IF NO
(3) 016450 127767 162764 162724 CMPB @SWR,$TSTNM ;ON THE RIGHT TEST? SWR<7:0>
(3) 016456 001465 BEQ $OVER ;BR IF YES
(3) 016460 105767 162717 2$: TSTB $ERFLG ;HAS AN ERROR OCCURRED?
(3) 016464 001421 BEQ 3$ ;BR IF NO
(3) 016466 126767 162723 162707 CMPB $ERMAX,$ERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
(3) 016474 101015 BHI 3$ ;BR IF NO
(3) 016476 032777 001000 162734 BIT #BIT09,@SWR ;LOOP ON ERROR?
(3) 016504 001404 BEQ 4$ ;BR IF NO
(3) 016506 016767 162676 162672 7$: MOV $LPERR,$LPADR ;SET LOOP ADDRESS TO LAST SCOPE
(3) 016514 000446 BR $OVER
(3) 016516 105067 162661 4$: CLRB $ERFLG ;ZERO THE ERROR FLAG

```



```

(3) 016522 005067 162764 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(3) 016526 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
(3) 016530 032777 004000 162702 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(3) 016536 001011 BNE 1$ ;;BR IF YES
(3) 016540 005767 162770 TST $PASS ;;IF FIRST PASS OF PROGRAM
(3) 016544 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(3) 016546 005267 162632 INC $ICNT ;;INCREMENT ITERATION COUNT
(3) 016552 026767 162734 162624 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(3) 016560 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(3) 016562 012767 000001 162614 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(3) 016570 016767 000056 162714 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(3) 016576 105267 162600 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
(3) 016602 116767 162574 162722 MOV $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(3) 016610 011667 162572 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(3) 016614 011667 162570 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(3) 016620 005067 162670 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(3) 016624 112767 000001 162563 MOV #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(3) 016632 016777 162544 162602 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(3) 016640 016716 162542 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(5) 016644 000002 4$: RTI
(5) 016646 001407 BRW: 1407
(5) 016650 000432 BRX: 432
(3) 016652 000005 $MXCNT: 5 ;;MAX. NUMBER OF ITERATIONS
(2) .SBTTL TRAP DECODER
(2)
(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)
(2) $TRAP: MOV R0,-(SP) ;;SAVE R0
(2) 016654 010046 MOV 2(SP),R0 ;;GET TRAP ADDRESS
(2) 016656 016600 000002 TST -(R0) ;;BACKUP BY 2
(2) 016662 005740 MOV (R0),R0 ;;GET RIGHT BYTE OF TRAP
(2) 016664 111000 ASL R0 ;;POSITION FOR INDEXING
(2) 016666 006300 MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
(2) 016670 016000 016710 RTS R0 ;;GO TO ROUTINE
(2)
(2)
(2) ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(2)
(2) $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
(2) 016676 011646 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
(2) 016700 016666 000004 000002 RTI ;;RESTORE THE PSW
(2) 016706 000002
(4) .SBTTL TRAP TABLE
(4)
(4) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(4) ;*BY THE "TRAP" INSTRUCTION.
(4)
(4) : ROUTINE
(4) : -----
(4) $TRPAD: .WORD $TRAP2
(4) 016710 016676 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
(4) 016712 013072 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(4) 016714 014610

```

(4)	016716	014564			\$TYPOS	::CALL=TYPOS	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
(4)	016720	014624			\$TYPON	::CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
(2)								
(2)								
(3)	016722	013050			.SCOPI	::CALL=SCOPI	TRAP+5(104405)	
(3)	016724	013354			.INSTR	::CALL=INSTR	TRAP+6(104406)	
(3)	016726	013462			.INSTER	::CALL=INSTER	TRAP+7(104407)	
(3)	016730	013472			.PARAM	::CALL=PARAM	TRAP+10(104410)	
(3)	016732	013672			.SAV05	::CALL=SAV05	TRAP+11(104411)	
(3)	016734	013732			.RES05	::CALL=RES05	TRAP+12(104412)	
(3)	016736	013764			.CONVRT	::CALL=CONVRT	TRAP+13(104413)	
(3)	016740	014142			.SETFLG	::CALL=SETFLG	TRAP+14(104414)	
8730					:*****			
8731					:UTILITIES			
8732					:*****			
8733					:THIS UTILITY CALCULATES PRIORITY LEVEL			
8734					DULEV:	ASL	DUPRT	:SHIFT LEFT
8735	016742	006367	000044			ASL	DUPRT	:
8736	016746	006367	000040			ASL	DUPRT	:
8737	016752	006367	000034			ASL	DUPRT	:
8738	016756	006367	000030			ASL	DUPRT	:
8739	016762	006367	000024			ASL	DUPRT	:
8740	016766	016767	000020	000020		MOV	DUPRT,LESS1	:MOVE THIS TO LESS1
8741	016774	162767	000001	000012		SUB	#1,LESS1	:CREATE LESS1
8742	017002	042767	000037	000004		BIC	#37,LESS1	:CLEAR TNZVC
8743	017010	000207				RTS	PC	
8744	017012	000240			DUPRT:	PR5		
8745	017014	000200			LESS1:	PR4		:LEVEL TO ALLOW INTERRUPTS
8746								
8747								
8748	017016	016767	000126	162666	DUADDR:	MOV	DUBASE,RXCSR	:XXX0
8749	017024	005267	000120			INC	DUBASE	
8750	017030	016767	000114	162656		MOV	DUBASE,HRXCSR	:XXX1
8751	017036	005267	000106			INC	DUBASE	
8752	017042	016767	000102	162646		MOV	DUBASE,RXDBUF	:XXX2
8753	017050	016767	000074	162644		MOV	DUBASE,PARCSR	:XXX2
8754	017056	005267	000066			INC	DUBASE	
8755	017062	016767	000062	162630		MOV	DUBASE,HRXDBUF	:XXX3
8756	017070	016767	000054	162626		MOV	DUBASE,HPARCSR	:XXX3
8757	017076	005267	000046			INC	DUBASE	
8758	017102	016767	000042	162616		MOV	DUBASE,TXCSR	:XXX4
8759	017110	005267	000034			INC	DUBASE	
8760	017114	016767	000030	162606		MOV	DUBASE,HTXCSR	:XXX5
8761	017122	005267	000022			INC	DUBASE	
8762	017126	016767	000016	162576		MOV	DUBASE,TXDBUF	:XXX6
8763	017134	005267	000010			INC	DUBASE	
8764	017140	016767	000004	162566		MOV	DUBASE,HTXDBUF	:XXX7
8765	017146	000207				RTS	PC	
8766	017150	000000			DUBASE:	0		
8767								
8768								
8769								
8770								
8771	017152	042777	040000	162546	RPOKE:	BIC	#MTDATA,@TXCSR	
8772	017160	005067	162316			CLR	\$TMP2	
8773	017164	006067	162310			ROR	\$TMP1	:FORCE CARRY



```

8774 017170 006067 162306 ROR $TMP2 ;PICK UP CARRY IN BIT 15
8775 017174 006267 162302 ASR $TMP2 ;SHIFT INTO BIT 14
8776 017200 042767 100000 162274 BIC #BIT15,$TMP2 ;CLR BIT 15
8777 017206 056777 162270 162512 BIS $TMP2,@TXCSR ;POKE MAINT DATA
8778 017214 042777 020000 162504 BIC #CLK,@TXCSR ;POKE CLK
8779 017222 052777 020000 162476 BIS #CLK,@TXCSR ;
8780 017230 005367 161666 DEC SHIFT
8781 017234 001346 BNE RPOKE
8782 017236 000207 RTS PC
8783
8784 ;THIS ROUTINE CALCULATES ODD PARITY FOR AN 8 BIT CHAR
8785 017240 016767 162234 162234 ODD8: MOV $TMP1,$TMP2 ;SAVE TEMP1
8786 017246 005067 162232 CLR $TMP3
8787 017252 012727 000010 MOV #8.,(PC)+
8788 017256 000000 4$: 0
8789 017260 006067 162216 1$: ROR $TMP2
8790 017264 005567 162214 ADC $TMP3
8791 017270 005367 177762 DEC 4$
8792 017274 001371 BNE 1$
8793 017276 006067 162202 ROR $TMP3
8794 017302 103404 BCS 2$
8795 017304 052767 000400 162166 BIS #BIT8,$TMP1 ;SET ODD PARITY
8796 017312 000403 BR 3$
8797 017314 042767 000400 162156 2$: BIC #BIT8,$TMP1 ;CLR EVEN PARITY
8798 ;$TMP1 NOW HAS ODD PARITY CHARACTER
8799 017322 000207 3$: RTS PC
8800
8801 ;THIS ROUTINE CALCULATES EVEN PARITY FOR AN 8 BIT CHARACTER
8802 017324 016767 162150 162150 EVEN8: MOV $TMP1,$TMP2 ;SAVE TEMP1
8803 017332 005067 162146 CLR $TMP3
8804 017336 012727 000010 MOV #8.,(PC)+
8805 017342 000000 4$: 0
8806 017344 006067 162132 1$: ROR $TMP2
8807 017350 005567 162130 ADC $TMP3
8808 017354 005367 177762 DEC 4$
8809 017360 001371 BNE 1$
8810 017362 006067 162116 ROR $TMP3
8811 017366 103004 BCC 2$
8812 017370 052767 000400 162102 BIS #BIT8,$TMP1 ;SET EVEN PARITY
8813 017376 000403 BR 3$
8814 017400 042767 000400 162072 2$: BIC #BIT8,$TMP1 ;CLR ODD PARITY
8815 ;$TMP1 NOW HAS EVEN PARITY CHARACTER
8816 017406 000207 3$: RTS PC
8817 017410 062716 000002 TRPREG: ADD #2,(SP) ;ALLOW IT TO "CRUNCH" INTO HLT BACK
8818 ;IN MAIN PART OF THE PROGRAM
8819 017414 000002 RTI
8820 POINT=. ;SAVE POINTER
(1) 000100 017416 .=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) 017416 017416 .=POINT ;PRIORITY
(1) 017416 104401 017424 $CLKVEC: ;RESTORE POINTER
(1) 017422 000000 TYPE,CLKMES
HALT

```

CNDUV-A MACY11 30(1046) 14-DEC-82 10:03 PAGE 61-59  
CNDUVA.M11 30-OCT-82 12:45 TRAP TABLE

SEQ 0073

(1)	017424	005015	045514	042526	CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
(1)	017432	020103	047111	042524	
(1)	017440	051122	050125	020124	
(1)	017446	020055	044504	041523	
(1)	017454	047117	042516	052103	
(1)	017462	046040	041524	000040	
8821		000001			.END



AAA	003200	8154#	
ABASE	= 000000	8154	
ACDW1	= 000000	8154	
ACDW2	= 000000	8154	
ACPUOP	= 000000	8154	
ACTREG	001166	8154#*	8729
ADDW0	= 000000	8154	
ADDW1	= 000000	8154	
ADDW10	= 000000	8154	
ADDW11	= 000000	8154	
ADDW12	= 000000	8154	
ADDW13	= 000000	8154	
ADDW14	= 000000	8154	
ADDW15	= 000000	8154	
ADDW2	= 000000	8154	
ADDW3	= 000000	8154	
ADDW4	= 000000	8154	
ADDW5	= 000000	8154	
ADDW6	= 000000	8154	
ADDW7	= 000000	8154	
ADDW8	= 000000	8154	
ADDW9	= 000000	8154	
ADEVCT	= 000000	8154	
ADEVN	= 000000	8154	
ADRCNT	= 013671	8729#*	
AENV	= 000000	8154	
AENVN	= 000000	8154	
AFATAL	= 000000	8154	
AMADR1	= 000000	8154	
AMADR2	= 000000	8154	
AMADR3	= 000000	8154	
AMADR4	= 000000	8154	
AMAMS1	= 000000	8154	
AMAMS2	= 000000	8154	
AMAMS3	= 000000	8154	
AMAMS4	= 000000	8154	
AMSGAD	= 000000	8154	
AMSGLG	= 000000	8154	
AMSGTY	= 000000	8154	
AMTYP1	= 000000	8154	
AMTYP2	= 000000	8154	
AMTYP3	= 000000	8154	
AMTYP4	= 000000	8154	
APASS	= 000000	8154	
APRIOR	= 000000	8154	
APTCSU	= 000040	7013#	8729
APTENV	= 000001	7013#	8729
APTSIZ	= 000200	7013#	8154
APTSP0	= 000100	7013#	8729
ASWREG	= 000000	8154	
ATESTN	= 000000	8154	
AUNIT	= 000000	8154	
AUSWR	= 000000	8154	
AVECT1	= 000000	8154	
AVECT2	= 000000	8154	
BASEAD	001154	8154#*	8729*









OUT	013030	8729#												
OUTCRY	012620	8729#												
OUTMUL	003166	8154#												
OVRRUN=	040000	8154#												
PARAM =	104410	8154#	8729#											
PARAM1	013522	8729#												
PARCSR	001722	8154#	8163*	8225*	8303*	8469*	8543*	8545*	8605*	8718*	8719*	8720*	8721*	8722*
		8723*	8724*	8725*	8726*	8727*	8728*	8753*						
PAREN =	001000	8154#												
PARER =	010000	8154#												
PARERR	013576	8729#												
PARTI	013660	8729#												
PASCNT	001112	8154#*	8729*											
PFTAB	015144	8729#												
PIRQ =	177772	8154#												
PIRQVE=	000240	8154#												
POINT =	017416	8820#												
POPRO =	012600	8154#												
POP1SP=	005726	8154#												
POP2SP=	022626	8154#												
PRO =	000000	8154#												
PR1 =	000040	8154#												
PR2 =	000100	8154#												
PR3 =	000140	8154#												
PR4 =	000200	8154#	8745											
PR5 =	000240	8154#	8744											
PR6 =	000300	8154#												
PR7 =	000340	8154#												
PS =	177776	8154#												
PSW =	177776	8154#												
PUSHRO=	010046	8154#												
PUSH1S=	005746	8154#												
PUSH2S=	024646	8154#												
PWRVEC=	000024	8154#*												
RDSW	013036	8729#*												
REACT=	004000	8154#	8718	8719	8720	8721								
REPLAY	012524	8729#												
RESTAR	015050	8729#												
RESTRT	012704	8729#												
RESVEC=	000010	8154#												
RES05 =	104412	8729#												
RETURN	001104	8154#*	8729											
RING =	040000	8154#												
RINTEN=	000100	8154#	8172	8182	8206	8248	8260	8264	8288	8556	8576	8579	8631	8645
		8662	8686											
ROTADD	001170	8154#*	8729*											
RPOKE	017152	8726	8727	8728	8771#	8781								
RTS =	000004	8154#												
RUNA =	*****	5678	5862	6003	6417									
RUNB =	*****	5703	5871	6010	6426									
RUNC =	*****	5728	5879	6017	6435									
RUND =	*****	5753	5888	6024	6444									
RUNE =	*****	5778	5898	6031	6453									
RUNF =	000000	5676#	5803	5908	6038	6462								
RUNIT	012430	8729#												
RXCSR	001712	8154#	8164*	8172*	8173	8182*	8206*	8226*	8235	8241*	8248*	8257	8260*	8264*

U  
U  
U  
U  
U







SATY1	000000	7013#		
SATY3	000006	7013#	8729	
SATY4	000016	7013#	8729	
SAUTOB	001434	8154#		
SBASE	001602	8154#		
SBDADR	001422	8154#		
SBDDAT	001426	8154#		
SBELL	001516	8154#	8729	
SCDW1	001606	8154#		
SCDW2	001610	8154#		
SCHARC	013350	8729#*		
SCKSWR=	***** U	8729		
SCLKVE	017416	8820#		
SCMTAG	001400	8154#		
SCM1 =	000006	8154#		
SCM2 =	000014	8154#		
SCM3 =	000006	8154#		
SCM4 =	000006	8154#		
SCPUOP	001554	8154#		
SCRLF	001523	8154#	8729	
SDDW0	001612	8154#		
SDDW1	001614	8154#		
SDDW10	001636	8154#		
SDDW11	001640	8154#		
SDDW12	001642	8154#		
SDDW13	001644	8154#		
SDDW14	001646	8154#		
SDDW15	001650	8154#		
SDDW2	001616	8154#		
SDDW3	001620	8154#		
SDDW4	001622	8154#		
SDDW5	001624	8154#		
SDDW6	001626	8154#		
SDDW7	001630	8154#		
SDDW8	001632	8154#		
SDDW9	001634	8154#		
SDEVCT	001536	8154#		
SDEVM	001604	8154#		
SE =	000002	6841#		
SENDAD	012674	8154	8729#	
SENV	001546	7013	8154#	8729
SENVN	001547	7013	8154#	8729
SERFLG	001403	8154#*	8729*	
SERMAX	001415	8154#*	8729*	
SERROR	014210	8154	8729#	
SERRPC	001416	8154#*	8729*	
SERRTB	001652	8154#	8729	
SERRTY	014430	8729#		
SERTTL	001412	8154#*	8729*	
SESCAP	001514	8154#*	8729*	
SETABL	001546	8154#		
SETEND	001652	8154#		
SFATAL	001530	7013*	8154#	
SFFLG	000244	7013#*		
SFILLC	001456	8154#	8729	
SFILLS	001455	8154#	8729	







.PARAM	013472	8729#	
.PFAIL	015012	8154	8729#
.RESOS	013732	8729#	
.SAVOS	013672	8729#	
.SCOPE	016320	8729#	
.SCOPI	013050	8729#	
.SETFL	014142	8729#	
.START	002152	8154#	8729
.\$ASTA=	***** U	7013	
.\$X =	002136	8154#	







\$.ERRO	2700#	6493#	8729
\$.ERRT	2896#	6493#	8729
\$.SMULT	4523#		
\$.SPOWE	4229#	6493#	
\$.SRAND	4307#		
\$.SRDDE	3891#		
\$.SRDOC	3797#		
\$.SREAD	3395#		
\$.SR2AZ	4958#		
\$.SSAVE	3969#		
\$.SSB2D	4771#		
\$.SSB2O	4874#		
\$.SSCOP	2454#	6493#	8729
\$.SSIZE	4361#		
\$.SSUPR	4913#		
\$.STRAP	4073#	6493#	8729
\$.STYPB	3287#		
\$.STYPD	3209#		
\$.STYPE	2985#	6492#	8729
\$.STYPO	3112#	6493#	8729
\$.S4OCA	972#		

. ABS. 017470 000

ERRORS DETECTED: 0

CNDUVA,CNDUVA/CRF/NL:TOC=CNMAC2.SML,CNDUV1.M11,CNDUV2.M11,CNDUVA.M11  
RUN-TIME: 17 16 1 SECONDS  
RUN-TIME RATIO: 89/34=2.5  
CORE USED: 43K (86 PAGES)