

DUV11

OFLNE TMG INTR
CNDUUAO

AH-T458A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



The main body of the document is a microfiche card containing a grid of approximately 15 columns and 25 rows of data. Each cell in the grid contains a small, high-contrast image of a document page, which is a common format for microfiche storage. The data is too small to be legible in this image.



.REM *

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

PRODUCT NAME: CNDUUAO DUV11 OFLNE TMG INTR

PRODUCT CODE: AC-T457A-MC

PRODUCT DATE: DEC, 1982

MAINTAINER :DIAGNJSTICS 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 TIMING AND INTERRUPT TESTS VERIFY THAT THE TRANSMITTER AND RECEIVER CAN CORRECTLY TALK TO EACH OTHER. INTERRUPT LOGIC AND PRIORITY LEVEL /VECTOR ADDRESSES ARE ALSO VERIFIED

* .REM *

2. REQUIREMENTS

PDP-11/21 COMPUTER (LSI)

DUV11 SYNCHRONOUS/ISOCRONOUS OPTION

ONE CONSOLE TELETYPE OR EQUIVALENT

- 2.2 STORAGE
THE PROGRAM LOADS INTO 4K OF MEMORY WITH BOOTSTRAP

* .REM *

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 AVAILIBLITY OF THE M315 CONNECTOR, ONE BIT PASSES STATUS TO APT-11. BIT 0 IN \$USWR REFLECTS THIS STATUS, A 0 = CONNECTOR

PRESENT, A 1 = CONNECTOR NOT AVAILABLE.
THE USER CHANGES THE CONTENTS OF THIS LOCATION
WHEN BUILDING THE E TABLE, BY ANSWERING THE
PROMPT "SWITCH 2".

- 4.1.1 AFTER PROGRAM LOAD (INITIAL PROGRAM START)
ALL CONSOLE SWITCHES DOWN
- 4.1.2 TO MODIFY DEVICE VECTOR AND CONTROL REGISTER ADDRESSES
AFTER PROGRAM RESTART OR TO RUN MULTIPLE DEVICES
SW00=1
- 4.1.3 TO START PROGRAM AT SELECTED TEST AFTER A PROGRAM RESTART
(ONLY IN SINGLE DEVICE TESTS)
SW01=1
- 4.1.4 TO LOCK ON SELECTED TEST AFTER A PROGRAM RESTART
(ONLY IN SINGLE DEVICE TESTS)
SW14=1
NOTE1: IN GENERAL SW01 WILL BE USED WHEN SW14=1 IS USED
NOTE2: WITHOUT SW01=1 "LOCK ON TEST" WILL DEFAULT TO TEST 1
- 4.2 STARTING ADDRESS
THE STARTING ADDRESS FOR ALL TESTS IS 000200
THE RETARTING ADDRESS FOR ALL TESTS IS 000200
THE STARTING ADDRESS TO ENTER A SELECTED TEST IS 000200
THE STARTING ADDRESS TO LOCK ON TEST IS 000200
- 4.3 PROGRAM AND/OR OPERATOR ACTION
 - 4.3.1 INITIAL PROGRAM START
 - 4.3.1.1 LOAD PROGRAM INTO MEMORY WITH ABSOLUTE LOADER
 - 4.3.1.2 SET SWITCH REGISTER (LOC. 176) TO ZERO.
 - 4.3.1.3 TYPE 200G.
 - 4.3.1.4 PROGRAM WILL START.
 - 4.3.1.5 THE PROGRAM WILL TYPE "DUV11 CNDUU-A TAPE E" (ONCE ONLY)
 - 4.3.1.6 THE PROGRAM WILL TYPE "R" TO INDICATE THAT IT IS ABOUT
TO START TESTING ,AND THEN TESTING WILL BEGIN
 - 4.3.2 PROGRAM RESTART WITH ALL SWITCHES DOWN
 - 4.3.2.1 THE PROGRAM WILL TYPE "R" AND WILL COMMENCE TESTING

* .REM *

* .REM *

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 ? ANDDO 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 TYPEOUT IS MENTIONED HERE FOR CONVENIENCE
IT IS IN THE FORM:

END OF PASS TAPE Y
16XXXX = DEVICE

WHERE Y IS THE TAPE LOADED

WHERE 16XXXX IS THE DEVICE'S BASE REGISTER ADDRESS

TO INHIBIT THIS TYPEOUT - TURN TELETYPE OFF

7. RESTRICTIONS

7.1 MULTIPLE DEVICES

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

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

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

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

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

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

7.2.2.2ORSET 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 MCDIFIED 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)
TIMING & INTERRUPT TESTING OF THE DEVICE
SEE LISTING FOR DETAILS

* .REM *

* .REM *

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

11. REVISION HISTORY
DZDUUB1 DIAGNOSTIC WAS MODIFIED TO WORK IN 11/21 SBC-
PROCESSOR.
PRIORITY 340 WAS CHANGED TO 300 WHEREVER ENCOUNTERED.
DEFAULT CSR AND VECTOR ADDRESSES WERE CHANGED.
VECTOR 140 WAS INITIALIZED TO ODT ADDRESS AND ALSO
BEVNT HANDLED WITH A MESSAGE.
DIAGNOSTIC WAS RENAMED TO CNDUUA0.
12. LISTINGS

*

6489
 6583
 6584
 6585
 6596
 6610
 6611
 6612
 6673
 6674
 6879
 7007
 7008

.SBTTL APT COMMUNICATIONS ROUTINE

```

(1) (1) 000000 112767 000001 000236 *****
(1) (1) 000006 112767 000001 000226 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERROR
(1) (1) 000014 000403          000000 $ATY3:  MOV  #1,$MFLG     ;;TO TYPE A MESSAGE
(1) (1) 000016 112767 000001 000220 $ATY4:  MOV  #1,$FFLG     ;;TO ONLY REPORT FATAL ERROR
(1) (1) 000024          000000 $ATYC:
(3) (3) 000024 010046          000000 MOV  R0,-(SP)      ;;PUSH R0 ON STACK
(3) (3) 000026 010146          000000 MOV  R1,-(SP)      ;;PUSH R1 ON STACK
(1) (1) 000030 105767 000206          TSTB $MFLG        ;;SHOULD TYPE A MESSAGE?
(1) (1) 000034 001450          000000 BEQ  5$           ;;IF NOT: BR
(1) (1) 000036 122767 000001 001502 CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
(1) (1) 000044 001031          000000 BNE  3$           ;;IF NOT: BR
(1) (1) 000046 132767 000100 001473 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) (1) 000054 001425          000000 BEQ  3$           ;;IF NOT: BR
(1) (1) 000056 017600 000004          MOV  @4(SP),R0     ;;GET MESSAGE ADDR.
(1) (1) 000062 062766 000002 000004 ADD  #2,4(SP)      ;;BUMP RETURN ADDR.
(1) (1) 000070 005767 001432          1$: TST  $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
(1) (1) 000074 001375          000000 BNE  1$           ;;IF NOT: WAIT
(1) (1) 000076 010067 001440          MOV  R0,$MSGAD
(1) (1)          000000 ;;PUT ADDR IN MAILBOX
(1) (1) 000102 105720          000000 2$: TSTB (R0)+      ;;FIND END OF MESSAGE
(1) (1) 000104 001376          000000 BNE  2$           ;;
(1) (1) 000106 166700 001430          SUB  $MSGAD,R0     ;;SUB START OF MESSAGE
(1) (1) 000112 006200          000000 ASR  R0            ;;GET MESSAGE LNTH IN WORDS
(1) (1) 000114 010067 001424          MOV  R0,$MSGGLT   ;;PUT LENGTH IN MAILBOX
(1) (1) 000120 012767 000004 001400 MOV  #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
(1) (1) 000126 000413          000000 BR  5$
(1) (1) 000130 017667 000004 000016 3$: MOV  @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
(1) (1) 000136 062766 000002 000004 ADD  #2,4(SP)     ;;BUMP RETURN ADDRESS
(3) (3) 000144 016746 177626          MOV  177776,-(SP) ;;PUSH 177776 ON STACK
(1) (1) 000150 004767 012736          JSR  PC,$TYPE    ;;CALL TYPE MACRO
(1) (1) 000154 000000          000000 4$: .WORD 0
(1) (1) 000156          000000 5$:
(1) (1) 000156 105767 000062          10$: TSTB $FFLG      ;;SHOULD REPORT FATAL ERROR?
(1) (1) 000162 001416          000000 BFO  12$         ;;IF NOT: BR
(1) (1) 000164 005767 001356          TST  $ENV        ;;RUNNING UNDER APT?
(1) (1) 000170 001413          000000 BEQ  12$         ;;IF NOT: BR
(1) (1) 000172 005767 001330          11$: TST  $MSGTYPE   ;;FINISHED LAST MESSAGE?
(1) (1) 000176 001375          000000 BNE  11$        ;;IF NOT: WAIT
(1) (1) 000200 017667 000004 001322 MOV  @4(SP),$FATAL ;;GET ERROR #
(1) (1) 000206 062766 000002 000004 ADD  #2,4(SP)     ;;BUMP RETURN ADDR.
(1) (1) 000214 005267 001306          INC  $MSGTYPE    ;;TELL APT TO TAKE ERROR
  
```


CNDUU-A MACY11 30(1046) 14-DEC-82
CNDUU2.M11 29-OCT-82 13:17

10:02 PAGE 58-1
APT COMMUNICATIONS ROUTINE

M 1

SEQ 0012

(1) 000220 105067 000020
(1) 000224 105067 000013
(1) 000230 105067 000006
(3) 000234 012601
(3) 000236 012600
(1) 000240 000207
(1) 000242 000
(1) 000243 000
(1)
(1) 000244 000
(1) 000246
(1) 000200
(1) 000001
(1) 000100
(1) 000040
7247 000001
7373
7377
7412
7429
7442
7454
7467

```
12$: CLRB $FFLG      ;;CLEAR FATAL FLAG
      CLRB $LFLG     ;;CLEAR LOG FLAG
      CLRB $MFLG     ;;CLEAR MESSAGE FLAG
      MOV  (SP)+,R1  ;;POP STACK INTO R1
      MOV  (SP)+,R0  ;;POP STACK INTO R0
      RTS   PC       ;;RETURN
      $MFLG: .BYTE 0  ;;MESSG. FLAG
      $LFLG: .BYTE 0
      $FFLG: .BYTE 0  ;;LOG FLAG
      .EVEN
      APTSIZE=200
      APTENV=001
      APTSPOOL=100
      APTCSUP=040
      $TN=1
```

```
;;FATAL FLAG
```

CNDUU-A MACY11 30(1046) 14-DEC-82 10:02 PAGE 61
CNDUU2.M11 29-OCT-82 13:17 APT COMMUNICATIONS ROUTINE

N 1

SEQ 0013

7491
7557
7563
7699
7727
7760
7801
7832
7883
7931
7984
7999
8011
8113

8144
8145
8147
8148 000246
(1) 000246
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)

```
.LIST ME
.NLIST MC,MD,CND
.ENABLE ABS
PRGFRT ^/DUV11 CNDUU-A TAPE E /,^/CNDUU-A/,^/END OF PASS TAPE E/
$HEADER ^/1977/,<DUV11 CNDUU-A TAPE E >,<CNDUU-A>,<END OF PASS TAPE E>

:DUV11 CNDUU-A TAPE E
:COPYRIGHT 1977, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754

:STARTING PROCEDURE
:TYPE 200G
:PROGRAM WILL TYPE "DUV11 CNDUU-A TAPE E "
:PROGRAM WILL TYPE "R" TO INDICATE THAT TESTING HAS STARTED
:AT THE END OF A PASS, PROGRAM WILL TYPE "END OF PASS TAPE E"
:AND THEN RESUME TESTING

.TITLE CNDUU-A
```

;TRAP CATCHER FOR ILLEGAL INTERRUPTS.

```
(2)
(2)
(2)          000000          . = 0
(3) 000000 000002          .+2
(3) 000002 000000          HALT
(3) 000004 000006          .+2
(3) 000006 000000          HALT
(3) 000010 000012          .+2
(3) 000012 000000          HALT
(3) 000014 000016          .+2
(3) 000016 000000          HALT
(3) 000020 000022          .+2
(3) 000022 000000          HALT
(3) 000024 000026          .+2
(3) 000026 000000          HALT
(3) 000030 000032          .+2
(3) 000032 000000          HALT
(3) 000034 000036          .+2
(3) 000036 000000          HALT
(3) 000040 000042          .+2
(3) 000042 000000          HALT
(3) 000044 000046          .+2
(3) 000046 000000          HALT
(3) 000050 000052          .+2
(3) 000052 000000          HALT
(3) 000054 000056          .+2
(3) 000056 000000          HALT
(3) 000060 000062          .+2
(3) 000062 000000          HALT
(3) 000064 000066          .+2
(3) 000066 000000          HALT
(3) 000070 000072          .+2
(3) 000072 000000          HALT
(3) 000074 000076          .+2
(3) 000076 000000          HALT
(3) 000100 000102          .+2
(3) 000102 000000          HALT
(3) 000104 000106          .+2
(3) 000106 000000          HALT
(3) 000110 000112          .+2
(3) 000112 000000          HALT
(3) 000114 000116          .+2
(3) 000116 000000          HALT
(3) 000120 000122          .+2
(3) 000122 000000          HALT
(3) 000124 000126          .+2
(3) 000126 000000          HALT
(3) 000130 000132          .+2
(3) 000132 000000          HALT
(3) 000134 000136          .+2
(3) 000136 000000          HALT
(3) 000140 000142          .+2
(3) 000142 000000          HALT
(3) 000144 000146          .+2
(3) 000146 000000          HALT
(3) 000150 000152          .+2
```


3

| | | | |
|-----|--------|--------|------|
| (3) | 000152 | 000000 | HALT |
| (3) | 000154 | 000156 | .+2 |
| (3) | 000156 | 000000 | HALT |
| (3) | 000160 | 000162 | .+2 |
| (3) | 000162 | 000000 | HALT |
| (3) | 000164 | 000166 | .+2 |
| (3) | 000166 | 000000 | HALT |
| (3) | 000170 | 000172 | .+2 |
| (3) | 000172 | 000000 | HALT |
| (3) | 000174 | 000176 | .+2 |
| (3) | 000176 | 000000 | HALT |
| (3) | 000200 | 000202 | .+2 |
| (3) | 000202 | 000000 | HALT |
| (3) | 000204 | 000206 | .+2 |
| (3) | 000206 | 000000 | HALT |
| (3) | 000210 | 000212 | .+2 |
| (3) | 000212 | 000000 | HALT |
| (3) | 000214 | 000216 | .+2 |
| (3) | 000216 | 000000 | HALT |
| (3) | 000220 | 000222 | .+2 |
| (3) | 000222 | 000000 | HALT |
| (3) | 000224 | 000226 | .+2 |
| (3) | 000226 | 000000 | HALT |
| (3) | 000230 | 000232 | .+2 |
| (3) | 000232 | 000000 | HALT |
| (3) | 000234 | 000236 | .+2 |
| (3) | 000236 | 000000 | HALT |
| (3) | 000240 | 000242 | .+2 |
| (3) | 000242 | 000000 | HALT |
| (3) | 000244 | 000246 | .+2 |
| (3) | 000246 | 000000 | HALT |
| (3) | 000250 | 000252 | .+2 |
| (3) | 000252 | 000000 | HALT |
| (3) | 000254 | 000256 | .+2 |
| (3) | 000256 | 000000 | HALT |
| (3) | 000260 | 000262 | .+2 |
| (3) | 000262 | 000000 | HALT |
| (3) | 000264 | 000266 | .+2 |
| (3) | 000266 | 000000 | HALT |
| (3) | 000270 | 000272 | .+2 |
| (3) | 000272 | 000000 | HALT |
| (3) | 000274 | 000276 | .+2 |
| (3) | 000276 | 000000 | HALT |
| (3) | 000300 | 000302 | .+2 |
| (3) | 000302 | 000000 | HALT |
| (3) | 000304 | 000306 | .+2 |
| (3) | 000306 | 000000 | HALT |
| (3) | 000310 | 000312 | .+2 |
| (3) | 000312 | 000000 | HALT |
| (3) | 000314 | 000316 | .+2 |
| (3) | 000316 | 000000 | HALT |
| (3) | 000320 | 000322 | .+2 |
| (3) | 000322 | 000000 | HALT |
| (3) | 000324 | 000326 | .+2 |
| (3) | 000326 | 000000 | HALT |
| (3) | 000330 | 000332 | .+2 |

| | | | |
|-----|--------|--------|------|
| (3) | 000332 | 000000 | HALT |
| (3) | 000334 | 000336 | .+2 |
| (3) | 000336 | 000000 | HALT |
| (3) | 000340 | 000342 | .+2 |
| (3) | 000342 | 000000 | HALT |
| (3) | 000344 | 000346 | .+2 |
| (3) | 000346 | 000000 | HALT |
| (3) | 000350 | 000352 | .+2 |
| (3) | 000352 | 000000 | HALT |
| (3) | 000354 | 000356 | .+2 |
| (3) | 000356 | 000000 | HALT |
| (3) | 000360 | 000362 | .+2 |
| (3) | 000362 | 000000 | HALT |
| (3) | 000364 | 000366 | .+2 |
| (3) | 000366 | 000000 | HALT |
| (3) | 000370 | 000372 | .+2 |
| (3) | 000372 | 000000 | HALT |
| (3) | 000374 | 000376 | .+2 |
| (3) | 000376 | 000000 | HALT |
| (3) | 000400 | 000402 | .+2 |
| (3) | 000402 | 000000 | HALT |
| (3) | 000404 | 000406 | .+2 |
| (3) | 000406 | 000000 | HALT |
| (3) | 000410 | 000412 | .+2 |
| (3) | 000412 | 000000 | HALT |
| (3) | 000414 | 000416 | .+2 |
| (3) | 000416 | 000000 | HALT |
| (3) | 000420 | 000422 | .+2 |
| (3) | 000422 | 000000 | HALT |
| (3) | 000424 | 000426 | .+2 |
| (3) | 000426 | 000000 | HALT |
| (3) | 000430 | 000432 | .+2 |
| (3) | 000432 | 000000 | HALT |
| (3) | 000434 | 000436 | .+2 |
| (3) | 000436 | 000000 | HALT |
| (3) | 000440 | 000442 | .+2 |
| (3) | 000442 | 000000 | HALT |
| (3) | 000444 | 000446 | .+2 |
| (3) | 000446 | 000000 | HALT |
| (3) | 000450 | 000452 | .+2 |
| (3) | 000452 | 000000 | HALT |
| (3) | 000454 | 000456 | .+2 |
| (3) | 000456 | 000000 | HALT |
| (3) | 000460 | 000462 | .+2 |
| (3) | 000462 | 000000 | HALT |
| (3) | 000464 | 000466 | .+2 |
| (3) | 000466 | 000000 | HALT |
| (3) | 000470 | 000472 | .+2 |
| (3) | 000472 | 000000 | HALT |
| (3) | 000474 | 000476 | .+2 |
| (3) | 000476 | 000000 | HALT |
| (3) | 000500 | 000502 | .+2 |
| (3) | 000502 | 000000 | HALT |
| (3) | 000504 | 000506 | .+2 |
| (3) | 000506 | 000000 | HALT |
| (3) | 000510 | 000512 | .+2 |

5

| | | | |
|-----|--------|--------|------|
| (3) | 000512 | 000000 | HALT |
| (3) | 000514 | 000516 | .+2 |
| (3) | 000516 | 000000 | HALT |
| (3) | 000520 | 000522 | .+2 |
| (3) | 000522 | 000000 | HALT |
| (3) | 000524 | 000526 | .+2 |
| (3) | 000526 | 000000 | HALT |
| (3) | 000530 | 000532 | .+2 |
| (3) | 000532 | 000000 | HALT |
| (3) | 000534 | 000536 | .+2 |
| (3) | 000536 | 000000 | HALT |
| (3) | 000540 | 000542 | .+2 |
| (3) | 000542 | 000000 | HALT |
| (3) | 000544 | 000546 | .+2 |
| (3) | 000546 | 000000 | HALT |
| (3) | 000550 | 000552 | .+2 |
| (3) | 000552 | 000000 | HALT |
| (3) | 000554 | 000556 | .+2 |
| (3) | 000556 | 000000 | HALT |
| (3) | 000560 | 000562 | .+2 |
| (3) | 000562 | 000000 | HALT |
| (3) | 000564 | 000566 | .+2 |
| (3) | 000566 | 000000 | HALT |
| (3) | 000570 | 000572 | .+2 |
| (3) | 000572 | 000000 | HALT |
| (3) | 000574 | 000576 | .+2 |
| (3) | 000576 | 000000 | HALT |
| (3) | 000600 | 000602 | .+2 |
| (3) | 000602 | 000000 | HALT |
| (3) | 000604 | 000606 | .+2 |
| (3) | 000606 | 000000 | HALT |
| (3) | 000610 | 000612 | .+2 |
| (3) | 000612 | 000000 | HALT |
| (3) | 000614 | 000616 | .+2 |
| (3) | 000616 | 000000 | HALT |
| (3) | 000620 | 000622 | .+2 |
| (3) | 000622 | 000000 | HALT |
| (3) | 000624 | 000626 | .+2 |
| (3) | 000626 | 000000 | HALT |
| (3) | 000630 | 000632 | .+2 |
| (3) | 000632 | 000000 | HALT |
| (3) | 000634 | 000636 | .+2 |
| (3) | 000636 | 000000 | HALT |
| (3) | 000640 | 000642 | .+2 |
| (3) | 000642 | 000000 | HALT |
| (3) | 000644 | 000646 | .+2 |
| (3) | 000646 | 000000 | HALT |
| (3) | 000650 | 000652 | .+2 |
| (3) | 000652 | 000000 | HALT |
| (3) | 000654 | 000656 | .+2 |
| (3) | 000656 | 000000 | HALT |
| (3) | 000660 | 000662 | .+2 |
| (3) | 000662 | 000000 | HALT |
| (3) | 000664 | 000666 | .+2 |
| (3) | 000666 | 000000 | HALT |
| (3) | 000670 | 000672 | .+2 |

```

(3) 000672 000000 HALT
(3) 000674 000676 .+2
(3) 000676 000000 HALT
(3) 000700 000702 .+2
(3) 000702 000000 HALT
(3) 000704 000706 .+2
(3) 000706 000000 HALT
(3) 000710 000712 .+2
(3) 000712 000000 HALT
(3) 000714 000716 .+2
(3) 000716 000000 HALT
(3) 000720 000722 .+2
(3) 000722 000000 HALT
(3) 000724 000726 .+2
(3) 000726 000000 HALT
(3) 000730 000732 .+2
(3) 000732 000000 HALT
(3) 000734 000736 .+2
(3) 000736 000000 HALT
(3) 000740 000742 .+2
(3) 000742 000000 HALT
(3) 000744 000746 .+2
(3) 000746 000000 HALT
(3) 000750 000752 .+2
(3) 000752 000000 HALT
(3) 000754 000756 .+2
(3) 000756 000000 HALT
(3) 000760 000762 .+2
(3) 000762 000000 HALT
(3) 000764 000766 .+2
(3) 000766 000000 HALT
(3) 000770 000772 .+2
(3) 000772 000000 HALT
(3) 000774 000776 .+2
(3) 000776 000000 HALT
(1) 001000
(2) .EQUAT
(2) .SBTTL BASIC DEFINITIONS
(2)
(2) 001
(2) !F NB
(2) ;*INITIAL ADDRESS OF THE STACK POINTER *** **
(2) STACK=
(2) .IFF
(2) ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(2) STACK= 1100
(2) 001100
(2) 000
(2) .ENDC
(2) .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
(2) .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
(2)
(2) ;*MISCELLANEOUS DEFINITIONS
(2) HT= 11 ;;CODE FOR HORIZONTAL TAB
(2) LF= 12 ;;CODE FOR LINE FEED
(2) CR= 15 ;;CODE FOR CARRIAGE RETURN
(2) CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
(2) PS= 177776 ;;PROCESSOR STATUS WORD
(2) .EQUIV PS,PSW
(2) 177774
(2) STKLMT= 177774 ;;STACK LIMIT REGISTER
  
```



```
(2) 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(2) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(2) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(2) ***** THE FOLLOWING ODT START ADDRESS FOR SBC 11/21 IS ADDED
(2) 170000 ODTST= 170000
(2) ;*GENERAL PURPOSE REGISTER DEFINITIONS
(2) 000000 R0= %0 ;;GENERAL REGISTER
(2) 000001 R1= %1 ;;GENERAL REGISTER
(2) 000002 R2= %2 ;;GENERAL REGISTER
(2) 000003 R3= %3 ;;GENERAL REGISTER
(2) 000004 R4= %4 ;;GENERAL REGISTER
(2) 000005 R5= %5 ;;GENERAL REGISTER
(2) 000006 R6= %6 ;;GENERAL REGISTER
(2) 000007 R7= %7 ;;GENERAL REGISTER
(2) 000006 SP= %6 ;;STACK POINTER
(2) 000007 PC= %7 ;;PROGRAM COUNTER
(2) ;*PRIORITY LEVEL DEFINITIONS
(2) 000000 PR0= 0 ;;PRIORITY LEVEL 0
(2) 000040 PR1= 40 ;;PRIORITY LEVEL 1
(2) 000100 PR2= 100 ;;PRIORITY LEVEL 2
(2) 000140 PR3= 140 ;;PRIORITY LEVEL 3
(2) 000200 PR4= 200 ;;PRIORITY LEVEL 4
(2) 000240 PR5= 240 ;;PRIORITY LEVEL 5
(2) 000300 PR6= 300 ;;PRIORITY LEVEL 6
(2) 000340 PR7= 340 ;;PRIORITY LEVEL 7
(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) 001 .IF B <>
```

```

(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)          000 .ENDC
(2)          001 .IF B <>
(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)          000 .ENDC
(1) 001000 $SETVEC

```

```

(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) 000204                $VARIA
(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)

```

```

(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)
(2) 001172      000      INIFLG: .BYTE 0        ;PROGRAM INITIALIZATION FLAG
(2) 001173      000      STFLG:  .BYTE 0        ;TEST START FLAG
(2) 001174      000      LOKFLG: .BYTE 0        ;LOCK ON CURRENT TEST FLAG
(2)                                     .EVEN
(1)                                     .=1400
(1) 001400      001400   $SYMBOLS
(2)
(2)

```


012

```

(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
(1) 001400      .SCMTAG 6,6,$SYMBOLS,..,ENDTAB
(2)          .MACRO $$CMREG A,B
(2)          $REG'A: .WORD 0                 ;;CONTAINS (($REGAD)+'B)
(2)          .NLIST
(2)          $CM1=$CM1+1
(2)          $CM2=$CM2+2
(2)          .LIST
(2)          .ENDM $$CMREG
(2)          .MACRO $$CMTMP A
(2)          $TMP'A: .WORD 0                 ;;USER DEFINED
(2)          .NLIST
(2)          $CM4=$CM4+1
(2)          .LIST
(2)          .ENDM $$CMTMP

```

```

(2)          .SBTTL COMMON TAGS
(2)
(2) 001400   STARS
(3)          .IF B
(3)          ;*****
(3)          .IFF
(3)          .NLIST
(3)          .REPT
(3)          .LIST
(3)          ;*****
(3)          .NLIST
(3)          .ENDR
(3)          .LIST
(3)          .ENDC
(2)          000
(2)          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(2)          ;*USED IN THE PROGRAM.
(2)
(2)          001
(2) 001400   .IF NB .
(2)          .=.
(2)          .IFF
(2)          .=1100
(2)          .ENDC
(2) 001400   $CMTAG:                ;;START OF COMMON TAGS
(2)          .IF NB ENDTAB
(2) 001400   000000 .WORD 0
(2)          .IFF
(2)          $PASS: .WORD 0                ;;CONTAINS PASS COUNT
(2)          .ENDC
(2) 001402   000 $TSTNM: .BYTE 0                ;;CONTAINS THE TEST NUMBER
(2) 001403   000 $ERFLG: .BYTE 0                ;;CONTAINS ERROR FLAG
(2) 001404   000000 $ICNT: .WORD 0                ;;CONTAINS SUBTEST ITERATION COUNT
(2) 001406   000000 $LPADR: .WORD 0                ;;CONTAINS SCOPE LOOP ADDRESS
(2) 001410   000000 $LPERR: .WORD 0                ;;CONTAINS SCOPE RETURN FOR ERRORS
(2) 001412   000000 $ERTTL: .WORD 0                ;;CONTAINS TOTAL ERRORS DETECTED
(2) 001414   000 $ITEMB: .BYTE 0                ;;CONTAINS ITEM CONTROL BYTE
(2) 001415   001 $ERMAX: .BYTE 1                ;;CONTAINS MAX. ERRORS PER TEST
(2) 001416   000000 $ERRPC: .WORD 0                ;;CONTAINS PC OF LAST ERROR INSTRUCTION
(2) 001420   000000 $GDADR: .WORD 0                ;;CONTAINS ADDRESS OF 'GOOD' DATA
(2) 001422   000000 $BDR: .WORD 0                ;;CONTAINS ADDRESS OF 'BAD' DATA
(2) 001424   000000 $GDDAT: .WORD 0                ;;CONTAINS 'GOOD' DATA
(2) 001426   000000 $BDDAT: .WORD 0                ;;CONTAINS 'BAD' DATA
(2)          .WORD 0                ;;RESERVED--NOT TO BE USED
(2) 001432   000000
(2) 001434   000 $AUTB: .BYTE 0                ;;AUTOMATIC MODE INDICATOR
(2) 001435   000 $INTAG: .BYTE 0                ;;INTERRUPT MODE INDICATOR
(2) 001436   000000
(2) 001440   177570 $WR: .WORD DSWR                ;;ADDRESS OF SWITCH REGISTER
(2) 001442   177570 $DISPLAY: .WORD DDISP                ;;ADDRESS OF DISPLAY REGISTER
(2) 001444   177560 $TKS: 177560                ;;TTY KBD STATUS
(2) 001446   177562 $TKB: 177562                ;;TTY KBD BUFFER
(2) 001450   177564 $TPS: 177564                ;;TTY PRINTER STATUS REG. ADDRESS
(2) 001452   177566 $TPB: 177566                ;;TTY PRINTER BUFFER REG. ADDRESS
(2) 001454   000 $NULL: .BYTE 0                ;;CONTAINS NULL CHARACTER FOR FILLS
(2) 001455   002 $FILLS: .BYTE 2                ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
(2)          ;
(2) 001456   012 $FILLC: .BYTE 12                ;;INSERT FILL CHARS. AFTER A 'LINE FEED'

```

```

(2) 001457 000 STPFLG: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(2) 001 .IF B 6
(2) .NLIST
(2) $CM3=0
(2) .LIST
(2) .IFF
(2) $CM3=6
(2) 000006 .ENDC
(2) 000 .IF NE $CM3
(2) 001 $REGAD: .WORD 0 ;:CONTAINS THE ADDRESS FROM
(2) 001460 000000 ;:WHICH ($REG0) WAS OBTAINED
(2) 000000 $CM1=0
(2) 000000 $CM2=0
(2) 000006 .REPT $CM3
(2) $SCMREG \ $CM1, \ $CM2
(2) .ENDR
(2) $SCMREG \ $CM1, \ $CM2
(3) 001462 $REG0: .WORD 0 ;:CONTAINS (($REGAD)+0)
(4) 001462 000000 $CM1=$CM1+1
(4) 000001 $CM2=$CM2+2
(4) 000002 $SCMREG \ $CM1, \ $CM2
(3) 001464 $REG1: .WORD 0 ;:CONTAINS (($REGAD)+2)
(4) 001464 000000 $CM1=$CM1+1
(4) 000002 $CM2=$CM2+2
(4) 000004 $SCMREG \ $CM1, \ $CM2
(3) 001466 $REG2: .WORD 0 ;:CONTAINS (($REGAD)+4)
(4) 001466 000000 $CM1=$CM1+1
(4) 000003 $CM2=$CM2+2
(4) 000006 $SCMREG \ $CM1, \ $CM2
(3) 001470 $REG3: .WORD 0 ;:CONTAINS (($REGAD)+6)
(4) 001470 000000 $CM1=$CM1+1
(4) 000004 $CM2=$CM2+2
(4) 000010 $SCMREG \ $CM1, \ $CM2
(3) 001472 $REG4: .WORD 0 ;:CONTAINS (($REGAD)+10)
(4) 001472 000000 $CM1=$CM1+1
(4) 000005 $CM2=$CM2+2
(4) 000012 $SCMREG \ $CM1, \ $CM2
(3) 001474 $REG5: .WORD 0 ;:CONTAINS (($REGAD)+12)
(4) 001474 000000 $CM1=$CM1+1
(4) 000006 $CM2=$CM2+2
(4) 000014 .ENDC
(2) 000 .IF NB 6
(2) 001 $CM4=0
(2) 000000 .REPT 6
(2) 000006 $SCMTMP \ $CM4
(2) .ENDR
(2) $SCMTMP \ $CM4
(3) 001476 $TMP0: .WORD 0 ;:USER DEFINED
(4) 001476 000000 $CM4=$CM4+1
(4) 000001 $SCMTMP \ $CM4
(3) 001500 $TMP1: .WORD 0 ;:USER DEFINED
(4) 001500 000000 $CM4=$CM4+1
(4) 000002 $SCMTMP \ $CM4
(3) 001502 $TMP2: .WORD 0 ;:USER DEFINED
(4) 001502 000000 $CM4=$CM4+1
(4) 000003 $SCMTMP \ $CM4
(3) 001504

```

```
(4) 001504 000000 $TMP3: .WORD 0 ;;USER DEFINED
(4) 001504 000004 $CM4=$CM4+1
(3) 001506 $SCMTMP \ $CM4
(4) 001506 000000 $TMP4: .WORD 0 ;;USER DEFINED
(4) 001506 000005 $CM4=$CM4+1
(3) 001510 $SCMTMP \ $CM4
(4) 001510 000000 $TMP5: .WORD 0 ;;USER DEFINED
(4) 001510 000006 $CM4=$CM4+1
(2) 000
(2) 001 .ENDC
(2) 001512 000000 .IF NE $SWR&4000 ;;MAX. NUMBER OF ITERATIONS
(2) 000 $TIMES: 0
(2) 001 .ENDC
(2) 001514 000000 .IF NE 1000&$SWR ;;ESCAPE ON ERROR ADDRESS
(2) 000 $ESCAPE:0
(2) 001 .ENDC
(2) 001516 177607 000377 .IF NE 2000&$SWR
(2) 000 $BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
(2) 001 .ENDC
(2) 001522 077 $QUES: .ASCII /?/ ;;QUESTION MARK
(2) 001523 015 $CRLF: .ASCII <15> ;;CARRIAGE RETURN
(2) 001524 000012 $LF: .ASCIZ <12> ;;LINE FEED
(2) 001 .IIF NE 1&.. .EVEN
(2) 001526 STARS
(3) 001 .IF B
(3) :*****
(3) .IFF
(3) .NLIST
(3) .REPT
(3) .LIST
(3) :*****
(3) .NLIST
(3) .ENDR
(3) .LIST
(3) 000 .ENDC
(2) 001526 001 .IF NB ENDTAB
(3) .SAPTBL $APTBL <ENDTAB>
(3) .SBTTL APT MAILBOX-ETABLE
(3) 001526 STARS
(4) 002 .IF B
(4) :*****
(4) .IFF
(4) .NLIST
(4) .REPT
(4) .LIST
(4) :*****
(4) .NLIST
(4) .ENDR
(4) .LIST
(4) 001 .ENDC
(3) .NLIST ME
(3) :
(3) :
(3) .IIF NDF,AMSGTY,AMSGTY=0
(3) .IIF NDF,ATESTN,ATESTN=0
```

```
(3) .IIF NDF,APASS,APASS=0
(3) .IIF NDF,ADEVCT,ADEVCT=0
(3) .IIF NDF,AFATAL,AFATAL=0
(3)
(3)
(3) .IIF NDF,AUNIT,AUNIT=0
(3) .IIF NDF,AMSGAD,AMSGAD=0
(3) .IIF NDF,AMSGLG,AMSGLG=0
(3) .IIF NDF,AENV,AENV=0
(3) .IIF NDF,AENVM,AENVM=0
(3) .IIF NDF,ASWREG,ASWREG=0
(3) .IIF NDF,AUSWR,AUSWR=0
(3) .IIF NDF,ACPUOP,ACPUOP=0
(3) .IIF NDF,AMAMS1,AMAMS1=0
(3) .IIF NDF,AMTYP1,AMTYP1=0
(3) .IIF NDF,AMADR1,AMADR1=0
(3) .IIF NDF,AMAMS2,AMAMS2=0
(3) .IIF NDF,AMTYP2,AMTYP2=0
(3) .IIF NDF,AMADR2,AMADR2=0
(3) .IIF NDF,AMAMS3,AMAMS3=0
(3) .IIF NDF,AMTYP3,AMTYP3=0
(3) .IIF NDF,AMADR3,AMADR3=0
(3) .IIF NDF,AMAMS4,AMAMS4=0
(3) .IIF NDF,AMTYP4,AMTYP4=0
(3) .IIF NDF,AMADR4,AMADR4=0
(3) .IIF NDF,AVECT1,AVECT1=0
(3) .IIF NDF,AVECT2,AVECT2=0
(3) .IIF NDF,APRIOR,APRIOR=0
(3) .IIF NDF,ABASE,ABASE=0
(3) .IIF NDF,ADEVVM,ADEVVM=0
(3) .IIF NDF,ACDW1,ACDW1=0
(3) .IIF NDF,ACDW2,ACDW2=0
(3) .IIF NDF,ADDW0,ADDW0=0
(3) .IIF NDF,ADDW1,ADDW1=0
(3) .IIF NDF,ADDW2,ADDW2=0
(3) .IIF NDF,ADDW3,ADDW3=0
(3) .IIF NDF,ADDW4,ADDW4=0
(3) .IIF NDF,ADDW5,ADDW5=0
(3) .IIF NDF,ADDW6,ADDW6=0
(3) .IIF NDF,ADDW7,ADDW7=0
(3) .IIF NDF,ADDW8,ADDW8=0
(3) .IIF NDF,ADDW9,ADDW9=0
(3) .IIF NDF,ADDW10,ADDW10=0
(3) .IIF NDF,ADDW11,ADDW11=0
(3) .IIF NDF,ADDW12,ADDW12=0
(3) .IIF NDF,ADDW13,ADDW13=0
(3) .IIF NDF,ADDW14,ADDW14=0
(3) .IIF NDF,ADDW15,ADDW15=0
(3) .LIST ME
(3) .EVEN
(3) 001526 $MAIL: ::APT MAILBOX
(3) 001526 000000 $MSGTY: .WORD MSGTY ::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 000 $ETABLE: ;; APT ENVIRONMENT TABLE
(3) 001547 000 $ENV: .BYTE AENV ;; ENVIRONMENT BYTE
(3) $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 $CPUOP: .WORD ACPUOP ;; CPU TYPE, OPTIONS
(3) ;;
(3) ;; BITS 15-11=CPU TYPE
(3) ;; 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(3) ;; 11/70=06,PDQ=07,Q=10
(3) ;; BIT 10=REAL TIME CLOCK
(3) ;; BIT 9=FLOATING POINT PROCFOSSOR
(3) ;; BIT 8=MEMORY MANAGEMENT
(3) 002 .IF IDN <ENDTAB>,<$CPUOP>
(3) $ETEND:
(3) .MEXIT
(3) .ENDC
(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 $MADR1: .WORD AMADR1 ;; HIGH ADDRESS,BLK#1
(3) ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(3) 002 .IF IDN <ENDTAB>,<$MADR1>
(3) $ETEND:
(3) .MEXIT
(3) .ENDC
(3) 001562 000 $MAMS2: .BYTE AMAMS2 ;; HIGH ADDRESS,M.S. BYTE
(3) 001563 000 $MTYP2: .BYTE AMTYP2 ;; MEM. TYPE,BLK#2
(3) 001564 000000 $MADR2: .WORD AMADR2 ;; MEM.LAST ADDRESS,BLK#2
(3) 002 .IF IDN <ENDTAB>,<$MADR2>
(3) $ETEND:
(3) .MEXIT
(3) .ENDC
(3) 001566 000 $MAMS3: .BYTE AMAMS3 ;; HIGH ADDRESS,M.S.BYTE
(3) 001567 000 $MTYP3: .BYTE AMTYP3 ;; MEM. TYPE,BLK#3
(3) 001570 000000 $MADR3: .WORD AMADR3 ;; MEM.LAST ADDRESS,BLK#3
(3) 002 .IF IDN <ENDTAB>,<$MADR3>
(3) $ETEND:
(3) .MEXIT
(3) .ENDC
(3) 001572 000 $MAMS4: .BYTE AMAMS4 ;; HIGH ADDRESS,M.S.BYTE
(3) 001573 000 $MTYP4: .BYTE AMTYP4 ;; MEM. TYPE,BLK#4
(3) 001574 000000 $MADR4: .WORD AMADR4 ;; MEM.LAST ADDRESS,BLK#4
(3) 002 .IF IDN <ENDTAB>,<$MADR4>
(3) $ETEND:
(3) .MEXIT
(3) .ENDC
(3) 001576 000000 $VECT1: .WORD AVECT1 ;; INTERRUPT VECTOR#1,BUS PRIORITY#1
(3) 002 .IF IDN <ENDTAB>,<$VECT1>
(3) .EVEN
```



```

(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001600   000000
(3)          $VECT2: .WORD  AVECT2  ;; INTERRUPT VECTOR#2BUS PRIORITY#2
(3)          .IF IDN <ENDTAB>,<$VECT2>
(3)          .EVEN
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001602   000000
(3)          $BASE:  .WORD  ABASE
(3)          ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(3)          .IF IDN <ENDTAB>,<$BASE>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001604   000000
(3)          $DEVN:  .WORD  ADEVN   ;;DEVICE MAP
(3)          .IF IDN <ENDTAB>,<$DEVN>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001606   000000
(3)          $CDW1:  .WORD  ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(3)          .IF IDN <ENDTAB>,<$CDW1>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001610   000000
(3)          $CDW2:  .WORD  ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
(3)          .IF IDN <ENDTAB>,<$CDW2>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001612   000000
(3)          $DDW0:  .WORD  ADDW0   ;;DEVICE DESCRIPTOR WORD#0
(3)          .IF IDN <ENDTAB>,<$DDW0>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001614   000000
(3)          $DDW1:  .WORD  ADDW1   ;;DEVICE DESCRIPTOR WORD#1
(3)          .IF IDN <ENDTAB>,<$DDW1>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001616   000000
(3)          $DDW2:  .WORD  ADDW2   ;;DEVICE DESCRIPTOR WORD#2
(3)          .IF IDN <ENDTAB>,<$DDW2>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001620   000000
(3)          $DDW3:  .WORD  ADDW3   ;;DEVICE DESCRIPTOR WORD#3
(3)          .IF IDN <ENDTAB>,<$DDW3>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001622   000000
(3)          $DDW4:  .WORD  ADDW4   ;;DEVICE DESCRIPTOR WORD#4
(3)          .IF IDN <ENDTAB>,<$DDW4>
(3)          SETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001624   000000
(3)          $DDW5:  .WORD  ADDW5   ;;DEVICE DESCRIPTOR WORD#5

```

```

(3)          002          .IF IDN <ENDTAB>,<$DDW5>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001626 000000 $DDW6: .WORD  ADDW6  ;;DEVICE DESCRIPTOR WORD#6
(3)          002          .IF IDN <ENDTAB>,<$DDW6>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001630 000000 $DDW7: .WORD  ADDW7  ;;DEVICE DESCRIPTOR WORD#7
(3)          002          .IF IDN <ENDTAB>,<$DDW7>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001632 000000 $DDW8: .WORD  ADDW8  ;;DEVICE DESCRIPTOR WORD#8
(3)          002          .IF IDN <ENDTAB>,<$DDW8>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001634 000000 $DDW9: .WORD  ADDW9  ;;DEVICE DESCRIPTOR WORD#9
(3)          002          .IF IDN <ENDTAB>,<$DDW9>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001636 000000 $DDW10: .WORD  ADDW10 ;;DEVICE DESCRIPTOR WORD#10
(3)          002          .IF IDN <ENDTAB>,<$DDW10>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001640 000000 $DDW11: .WORD  ADDW11 ;;DEVICE DESCRIPTOR WORD#11
(3)          002          .IF IDN <ENDTAB>,<$DDW11>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001642 000000 $DDW12: .WORD  ADDW12 ;;DEVICE DESCRIPTOR WORD#12
(3)          002          .IF IDN <ENDTAB>,<$DDW12>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001644 000000 $DDW13: .WORD  ADDW13 ;;DEVICE DESCRIPTOR WORD#13
(3)          002          .IF IDN <ENDTAB>,<$DDW13>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001646 000000 $DDW14: .WORD  ADDW14 ;;DEVICE DESCRIPTOR WORD#14
(3)          002          .IF IDN <ENDTAB>,<$DDW14>
(3)          $ETEND:
(3)          .MEXIT
(3)          .ENDC
(3) 001650 000000 $DDW15: .WORD  ADDW15 ;;DEVICE DESCRIPTOR WORD#15
(3)          002          .IF IDN <ENDTAB>,<$DDW15>
(3)          $ETEND:
(3)          .MEXIT
(3)          001          .ENDC
(3)
(3)

```

20

CNDUU-A MACY11 30(1046) 14-DEC-82 10:02 PAGE 62-19
CNDUUA.M11 29-OCT-82 13:27 APT MAILBOX-ETABLE

H 3

SEQ 0033

(3) 001652

(3)

(2) 000

(2) 001

(2)

(2)

(3)

(3) 001652

(4)

(4)

\$ETEND:

.ENDC

.IF NB <\$SYMBOLS>

.IRP A,<\$SYMBOLS>

A

.ENDM

\$SYMBOLS

22

CNDUU-A MACY11 30(1046) 14-DEC-82 10:02 PAGE 62-21
CNDUUA.M11 29-OCT-82 13:27 APT MAILBOX-ETABLE

J 3

SEQ 0035

| | | | |
|-----|--------|------------------------|-------------------|
| (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) | 000 | .ENDC | |

```

(2) .SBTTL ERROR POINTER TABLE
(2)
(2) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(2) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(2) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(2) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(2) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(2)
(2) ;* EM ;:POINTS TO THE ERROR MESSAGE
(2) ;* DH ;:POINTS TO THE DATA HEADER
(2) ;* DT ;:POINTS TO THE DATA
(2) ;* DF ;:POINTS TO THE DATA FORMAT
(2)
(2) $ERRTB:
(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)
(1) ;DEFAULT DU ADDRESSES
(1) 001712 174300 RXCSR: 174300
(1) 001714 174301 HRXCSR: 174301
(1) 001716 174302 RXDBUF: 174302
(1) 001720 174303 HRXDBUF: 174303
(1) 001722 174302 PARCSR: 174302
(1) 001724 174303 HPARCSR: 174303
(1) 001726 174304 TXCSR: 174304
(1) 001730 174305 HTXCSR: 174305
(1) 001732 174306 TXDBUF: 174306
(1) 001734 174307 HTXDBUF: 174307
(1)
(1) ;DEFAULT DU VECTORS
(1) 001736 000330 DURIV: 330 ;REC INTR VECTOR
(1) 001740 000332 DURIS: 332 ;REC INTR STATUS
(1) 001742 000334 DUTIV: 334 ;XMIT INTR VECTOR
(1) 001744 000336 DUTIS: 336 ;XMIT INTR STATUS
(1)
(1) ;ERROR MESSAGES
(1) 001746 020040 051105 047522 EM4: .ASCIZ / ERROR PC /
(1) 001754 020122 041520 000040
(1) 001762 020040 047503 050115 EM1: .ASCIZ / COMPARISON ERROR ON REGISTERS/
(1) 001770 051101 051511 047117
(1) 001776 042440 051122 051117
(1) 002004 047440 020116 042522

```

```

(1) 002012 044507 052123 051105
(1) 002020 000123
(1) 002022 020040 042522 042503 EM2: .ASCIZ / RECEIVER ERROR/
(1) 002030 053111 051105 042440
(1) 002036 051122 051117 000
(1) 002043 040 052040 040522 EM3: .ASCIZ / TRANSMITTER ERROR/
(1) 002050 051516 044515 052124
(1) 002056 051105 042440 051122
(1) 002064 051117 000
(1)
(1) 002067 105 051122 041520 DM1: ;DATA HEADERS FOR ERROR MESSAGES
(1) 002074 020040 040527 052116 .ASCIZ /ERRPC WANTED ACTUAL/
(1) 002102 042105 020040 041501
(1) 002110 052524 046101 000
(1) 002116 001416 001130 001132 DT1: ;DATA TABLES FOR ERROR MESSAGES
(1) 002124 000000 .WORD $ERRPC,HLDO,HLDI,0
(1)
(1) 002126 001416 000000 DT4: .WORD $ERRPC,0
(1)
(1) 002132 000 000 000 DF1: .BYTE 0,0,0,0
(1) 002135 000
(1)
(1) 002136 .EVEN
(2) .SACT11
(2) .SBTTL ACT11 HOOKS
(2) 002136 STARS
(3) 001 .IF B
(3) ;*****
(3) .IFF
(3) .NLIST
(3) .REPT
(3) .LIST
(3) ;*****
(3) .NLIST
(3) .ENDR
(3) .LIST
(3) 000 .ENDC
(2) ;HOOKS REQUIRED BY ACT11
(2) 002136 $SVPC=. ;SAVE PC
(2) 001 .IF B
(2) 000046 =46
(2) 000046 012714 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(2) .IFF
(2) =46 ;;1)SET LOC.46 TO ADDRESS OF
(2) 000 .ENDC
(2) 001 .IF B
(2) 000052 =52
(2) 000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO
(2) .IFF
(2) =52
(2) .WORD ;;2)SET LOC.52 TO
(2) 000 .ENDC
(2) 002136 = $SVPC ;; RESTORE PC

```



```

(1) 002136      .SAPTHDR      10,10
(2)             .SBITL  APT PARAMETER BLOCK
(2) 002136      STARS
(3)             .IF B
(3)             ;*****
(3)             .IFF
(3)             .NLIST
(3)             .REPT
(3)             .LIST
(3)             ;*****
(3)             .NLIST
(3)             .ENDR
(3)             .LIST
(3)             .ENDC
(2)             000
(2) 002136      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(3)             STARS
(3)             .IF B
(3)             ;*****
(3)             .IFF
(3)             .NLIST
(3)             .REPT
(3)             .LIST
(3)             ;*****
(3)             .NLIST
(3)             .ENDR
(3)             .LIST
(3)             .ENDC
(2)             000
(2)             002136      .SX=.      ;;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      002136      $APTHDR  ;;POINT TO APT HEADER BLOCK
(2)             002136      =.SX      ;;RESET LOCATION COUNTER
(2) 002136      STARS
(3)             .IF B
(3)             ;*****
(3)             .IFF
(3)             .NLIST
(3)             .REPT
(3)             .LIST
(3)             ;*****
(3)             .NLIST
(3)             .ENDR
(3)             .LIST
(3)             .ENDC
(2)             000
(2)             ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(2)             ;INTERFACE SPEC.
(2) 002136      $APTHD:
(2) 002136      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(2) 002140      $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(2) 002142      $STMT: .WORD 10     ;;RUN TIM OF LONGEST TEST
(2) 002144      $PASTM: .WORD 10    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(2) 002146      $UNITM: .WORD      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(2) 002150      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

```

(1) 002152 .SETUP <.$SCOPE,.$ERROR,.$TRAP,.$POWER>
(2)          $SETUP= 0
(3)          $STUP= 0
(3)
(3)          000001 $SETUP= $SETUP!1      ;;BIT00
(3)          177777 $STUP= -1
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(3)          000000 $STUP= 0
(3)
(3)          000003 $SETUP= $SETUP!2      ;;31T01
(3)          177777 $STUP= -1
(3)
(3)
(3)
(3)
(3)          000000 $STUP= 0
(3)
(3)
(3)          000007 $SETUP= $SETUP!4      ;;BIT02
(3)          177777 $STUP= -1
(3)
(3)
(3)
(3)
(3)          000000 $STUP= 0
(3)
(3)
(3)          000017 $SETUP= $SETUP!10     ;;BIT03
(3)          177777 $STUP= -1
(3)
(3)
(3)
(1)
(1) 002152 $START
(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 COJNTS
(2)          ;TYPE TITLE MESSAGE
(2)

```

```

(2) 002152 .START:
(3) .SBTTL INITIALIZE THE COMMON TAGS
(3) 001 .IF DF $CMTAG
(3) ::CLEAR THE COMMON TAGS ($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) 000 .ENDC
(3) 001 .IF NB #STACK
(3) 002166 012706 001100 MOV ##STACK,SP ;;SETUP THE STACK POINTER
(3) .IFF
(3) MOV #STACK,SP ;;SETUP THE STACK POINTER
(3) 000 .ENDC
(3) .IIF NDF $SETUP,$SETUP= 0
(3) ::INITIALIZE A FEW VECTORS
(3) 001 .IF NE $SETUP&1 ;;BIT00
(3) 002172 012737 016340 000020 MOV # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(3) 002200 012737 000300 000022 MOV #PR6,@#IOTVEC+2 ;;LEVEL 6
(3) .IIF NDF $CMTAG, CLR $STNM ;;INITIALIZE THE TEST NUMBER
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&2 ;;BIT01
(3) 002206 012737 014230 000030 MOV # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(3) 002214 012737 000300 000032 MOV #PR6,@#EMTVEC+2 ;;LEVEL 6
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&4
(3) ::BIT02
(3) 002222 012737 016674 000034 MOV # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(3) 002230 012737 000300 000036 MOV #PR6,@#TRAPVEC+2;LEVEL 6
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&10 ;;BIT03
(3) 002236 012737 015032 000024 MOV # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(3) 002244 012737 000300 000026 MOV #PR6,@#PWRVEC+2 ;;LEVEL 6
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&20 ;;BIT04
(3) .IIF NDF $CMTAG, CLR $PASS ;;CLEAR THE PASS COUNT
(3) MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&40 ;;BIT05
(3) MOV #176543,$HINUM ;;PRIME THE RANDOM NUMBER GENERATOR
(3) MOV #123456,$LONUM ;;BOTH HIGH AND LOW WORDS
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&1 ;;BIT00
(3) .IF NE $SWR&4000
(3) 002252 005067 177234 .IIF NDF $CMTAG, MOV #1,$ICNT ;;INITIALIZE THE ITERATION COUNTER
(3) CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(3) 001 .ENDC
(3) 000 .ENDC
(3) 001 .IF NE $SETUP&2 ;;BIT01
(3) .IIF NDF $CMTAG, CLR $ERFLG ;;CLEAR THE ERROR FLAG
(3) 002 .IIF NE 1000&$SWR
(3) .IIF NDF $CMTAG, CLR $ERTTL ;;CLEAR THE ERROR COUNT
(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) 001 .ENDC
(3) 000 .ENDC

```

INITIALIZE THE COMMON TAGS

```
(3)          001          .IF NE $SETUP&20          ;;BIT04
(3)          002          .IF NE 10000&$SWR
(3)          .MACRO $$SETUP ?N1,?N2,?N3
(3)          ;;INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
(3)          ;;THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'
(3)          MOV          #RTRN,@#TBITVEC          ;;SET 'T' BIT VECTOR TO $RTRN
(3)          MOV          #PR6,@#TBITVEC+2          ;;LEVEL 6
(3)          MOV          #RTI,$RTRN          ;;SET $RTRN TO A RTI
(3)          MOV          #N2,@#RESVEC          ;;TRY TO DO A RTT
(3)          CLR          -(SP)          ;;DUMMY PS
(3)          MOV          #N1,-(SP)          ;;AND PC
(3)          RTT          ;;TRY THE RTT
(3)          N1:         MOV          #RTT,$RTRN          ;;RTT IS LEGAL--SET $RTRN TO A RTT
(3)          BR          N3
(3)          N2:         ADD          #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
(3)          N3:         MOV          #RESVEC+2,@#RESVEC          ;;RESTORE TRAP CATCHER
(3)          CLR          $TBIT          ;;CLEAR 'T' BIT SWITCH
(3)          .ENDM          $$SETUP
(3)          .ENDC          $$SETUP
(3)          001          .ENDC
(3)          000          .ENDC
(3)          001          .IF NE $SETUP&1
(3) 002270 012767 002270 177110 MOV          #.,$LPADR          ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(3) 002276 012767 002276 177104 MOV          #.,$LPERR          ;;SETUP THE ERROR LOOP ADDRESS
(3)          000          .ENDC
(3) 002304          SWRSU
(4)          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(4)          ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(4) 002304 013746 000004 MOV          @#ERRVEC,-(SP)          ;;SAVE ERROR VECTOR
(4) 002310 012737 002344 000004 MOV          #64$,@#ERRVEC          ;;SET UP ERROR VECTOR
(4) 002316 012767 177570 177114 MOV          #DSWR,SWR          ;;SETUP FOR A HARDWARE SWICH REGISTER
(4) 002324 012767 177570 177110 MOV          #DDISP,DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
(4) 002332 022777 177777 177100 CMP          #-1,@SWR          ;;TRY TO REFERENCE HARDWARE SWR
(4) 002340 001012 BNE          66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(4)          ;;AND THE HARDWARE SWR IS NOT = -1
(4) 002342 000403 BR          65$          ;;BRANCH IF NO TIMEOUT
(4) 002344 012716 002352 64$: MOV          #65$,(SP)          ;;SET UP FOR TRAP RETURN
(4) 002350 000002 RTI
(4) 002352 012767 000176 177060 65$: MOV          #SWREG,SWR          ;;POINT TO SOFTWARE SWR
(4) 002360 012767 000174 177054 MOV          #DISPREG,DISPLAY
(4) 002366 012637 000004 66$: MOV          (SP)+,@#ERRVEC          ;;RESTORE ERROR VECTOR
(3)          001          .IF DF $MAIL
(3)          .MACRO $$SETMAIL          ?$ARG1
(3)          CLR          $PASS          ;;CLEAR PASS COUNT
(3)          BITB          #APTSIZE,$ENVM          ;;TEST USER SIZE UNDER APT
(3)          BEQ          $ARG1          ;;YES,USE NON-APT SWITCH
(3)          MOV          #SSWREG,SWR          ;;NO,USE APT SWITCH REGISTER
(3)          $ARG1:
(3)          .ENDM          $$SETMAIL
(3)          001          .ENDC
(4) 002372 005067 177136 CLR          $PASS          ;;CLEAR PASS COUNT
(4) 002376 132767 000200 177143 BITB          #APTSIZE,$ENVM          ;;TEST USER SIZE UNDER APT
(4) 002404 001403 BEQ          67$          ;;YES,USE NON-APT SWITCH
(4) 002406 012767 001550 177024 MOV          #SSWREG,SWR          ;;NO,USE APT SWITCH REGISTER
(4) 002414 67$:
```

```

(3)          000          .ENDC
(2) 002414 012706 001100      MOV      #STACK,SP      ;SET STACK
(2) 002420 106427 000300      MTPS     #300          ;LOCK INTERRUPTS
(2) 002424 012737 015032 000024  MOV      #.PFAIL,@#24  ;SET UP POWER FAIL VECTOR
(2) 002432 105067 176535      CLRB     STFLG         ;CLEAR START FLAG
(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      $ERRPC       ;CLEAR LAST ERROR POINTER
(2) 002456 012767 000001 176716  MOV      #1,$TSTNM     ;SET UP FOR TEST !
(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$:  MOV      #SWREG,SWR
(2) 002524 012767 000174 176710  MOV      #DISPREG,DISPLAY
(2) 002532 022626              CMP      (SP)+,(SP)+
(2) 002534 012637 000004      MOV      (SP)+,@#4
(2) 002540 012637 000006      MOV      (SP)+,@#6
(2) 002544 022767 000176 176666  CMP      #SWREG,SWR
(2) 002552 001007              BNE     3$
(2) 002554 005737 000042      TST      @#4?          ;CHECK FOR CHAIN
(2) 002560 001402              BEQ     33$
(2) 002562 000167 000522      JMP      .BEGIN
(2) 002566 004767 010224 33$:  JSR      PC,CNTLU
(2) 002572 105767 176374 3$:  TSTB     INIFLG        ;HAS INITIALIZATION BEEN PERFORMED
(2) 002576 001004              BNE     ONCE
(2) 002600 104401 015172      TYPE     ,MTITLE      ;TYPE TITLE MESSAGE
(2) 002604 105167 176362      COMB     INIFLG        ;IF NOT SET FLAG AND DO
(1) 002610          .BEGIN
(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      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$
(1) 002704          $GETPAR MREGAD,DUBASE,1,1,174000,177776
(2) 002704 104406      INSTR   ;OUTPUT MESSAGE & GET INPUT STRING
(2) 002706 015240      MREGAD  ;MESSAGE
(2) 002710 104410      PARAM   ;CONVERT STRING
(2) 002712 174000      174000 ;LOW LIMIT
  
```

```

(2) 002714 177776          177776 ;HIGH LIMIT
(2) 002716 017170          DUBASE ;STORE AT THIS LOCATION
(2) 002720          001      .BYTE 1 ;MASK
(2) 002721          001      .BYTE 1 ;HOW MANY TIMES + 2
(1) 002722 016767 014242 176226 MOV DUBASE,KEEPADD ;SAVE
(1) 002730 004767 014102      JSR PC,DUADDR
(1) 002734 016767 176216 176212 MOV KEEPADD,BASEADD ;RESTORE FOR ROTATION
(1) 002742          $GETPAR MVECTO,DURIV,4,1,300,376
(2) 002742 104406          INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 002744 015225          MVECTO ;MESSAGE
(2) 002746 104410          PARAM ;CONVERT STRING
(2) 002750 000300          300 ;LOW LIMIT
(2) 002752 000376          376 ;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
(1) 002774          $GETFLG MMULT,MULTD
(2) 002774 104406          INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 002776 015270          MMULT ;MESSAGE
(2) 003000 104414          SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003002 001152          MULTD ;THIS FLAG
(1) 003004 105767 176142          TSTB MULTD ;ARE THERE MULTIPLE DEVICES
(1)          ;ON THE SYSTEM ?
(1) 003010 100406          BMI BBB ;YES,ASK NEXT QUESTION
(1) 003012 005067 176150          CLR ACTREG
(1) 003016 005067 176146          CLR ROTADD
(1) 003022 000167 000140          JMP OUTMUL ;JUMP AROUND NEXT QUESTION
(1) 003024          BBB:
(2) 003026 104406          INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003030 015317          MLASTD ;MESSAGE
(2) 003032 104410          PARAM ;CONVERT STRING
(2) 003034 174000          174000 ;LOW LIMIT
(2) 003036 177776          177776 ;HIGH LIMIT
(2) 003040 001160          LASTADD ;STORE AT THIS LOCATION
(2) 003042          001      .BYTE 1 ;MASK
(2) 003043          001      .BYTE 1 ;HOW MANY TIMES + 2
(1)          ;THE FOLLOWING ROUTINE SETS UP ACTREG FOR THE FIRST TIME
(1) 003044 012767 000001 176116 1$: 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
(1) 003144          $GETPAR MRANGE, LASTADD,1,1,174000,177776
  
```

```

(2) 003144 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003146 015413 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 013636 OUTMUL: MOV #300,DUPRT
(1) 003174 004767 013562 JSR PC,DULEV
(1) 003200 $GETSYN MSYNC,SYNCR,AAA:
(2) ;COMPARE THE FIRST CHARACTER IN THE TELETYPE INPUT
(2) ;BUFFER TO THE CHARACTERS '1' AND '2'.
(2) ;IF THE CHARACTER IS '1' CLEAR THE FLAG
(2) ;IF THE CHARACTER IS '2' SET THE FLAG
(2) 003200 AAA:
(2) 003200 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003202 015631 MSYNC ;MESSAGE
(2) 003204 122767 000061 012760 3$: CMPB #'1,INBUF ;IS IT '1' ?
(2) 003212 001003 BNE 1$
(2) 003214 105067 175726 CLRB SYNCR ;000
(2) 003220 000412 BR 4$
(2) 003222 122767 000062 012742 1$: CMPB #'2,INBUF ;IS IT '2' ?
(2) 003230 001004 BNE 2$
(2) 003232 112767 177777 175706 MOVB #-1,SYNCR ;377
(2) 003240 000402 BR 4$
(2) 003242 104407 2$: INSTER ;RETRY
(2) 003244 000757 BR 3$
(2) 003246 000240 4$: NOP
(1) 003250 $GETFLG MWIRE6,SEXMIT
(2) 003250 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003252 015677 MWIRE6 ;MESSAGE
(2) 003254 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003256 001147 SEXMIT ;THIS FLAG
(1) 003260 $GETFLG MWIRE5,SEREC
(2) 003260 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003262 015750 MWIRE5 ;MESSAGE
(2) 003264 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003266 001150 SEREC ;THIS FLAG
(1) 003270 $GETFLG MWIRE4,OPTCLR
(2) 003270 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003272 016020 MWIRE4 ;MESSAGE
(2) 003274 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003276 001151 OPTCLR ;THIS FLAG
(1) 003300 $GETFLG MEXTJ,JMRBY
(2) 003300 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(2) 003302 016077 MEXTJ ;MESSAGE
(2) 003304 104414 SETFLG ;SET FLAG BASED UPON INPUT STRING
(2) 003306 001153 JMRBY ;THIS FLAG
(1) 003310 $BEGIN
(2) ;TEST START AND RESTART
(2)
(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$
(2) 003330 $GETPAR MTSTPC,$TSTNM,1,1,TST1,17500
(3) 003330 104406 INSTR ;OUTPUT MESSAGE & GET INPUT STRING
(3) 003332 015563 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$: MOV #TST1,RETURN ;START AT TEST 1
(2) 003364 104401 015557 4$: TYPE ,MR ;TYPE R
(2) 003370 000177 175510 JMP @RETURN ;START TESTING

8149
8150 003374 $TRPAR SEVEN,10,,EVEPAR,ISYMOD,125,1252
(1) ;;THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1) ;;OF THE TRANSMITTER SECTION.
(1) ;;IT ALSO CHECKS DNA TIMING
(1) ;;MODE:ISYMOD
(1) ;;LENGTH:SEVEN PLUS PARITY
(1) ;;PARITY:EVEPAR
(1) ;;CHARACTER:125
(1) ;;
(1) 003374 $TSTNO
(2) 003374 NEWTST
(3) 000000 $NWTST=0
(3) 001 .IF B <>
(3) 003374 $$NEWTST \&TN,<>,SCOPE
(4) .IRP ASCII,<>
(4) .IF EQ $NWTST
(4) .NLIST
(4) $NWTST=1
(4) .SBTTL T1 ASCII
(4) .LIST
(4) STARS
(4) ;*TEST 1 ASCII
(4) .IFF
(4) ASCII
(4) .ENDC
(4) .ENDM
(4) 003374 STARS
(5) 002 .IF B
(5) ;*****
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) ;*****
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) .ENDC
(4) 003374 001 000004 TST1: SCOPE
```

```

(4)          000002      $TN=$TN+1
(4)          .IFF
(3)          $SNEWTEST      \ $TN,<>,<>
(3)          .ENDC
(3)          000          .IF NE 4000&$SWR
(3)          001          .IF NB
(3)          002          .IF LE <-1>
(3)          003          MOV      #1,$TIMES      ;;DO 1 ITERATION
(3)          .IFF
(3)          MOV      #,$TIMES      ;;DO ITERATIONS
(3)          .ENDC
(3)          002          .ENDC
(3)          001          .ENDC
(3)          002          .IF NB
(3)          MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(3)          .ENDC
(3)          001          .ENDC
(3)          000          .ENDC
(3)          001          .IF NB
(3)          002          .IF DF $MAIL
(3)          .IRP      $MAIL
(3)          TNM,<\$TN-1>
(3)          MOV      #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(3)          .ENDM
(3)          .ENDC
(3)          001          .ENDC
(3)          000          .ENDC
(2)          000000      .REPT 0
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP      $DN,<\$N>
(2)          NEWTST
(2)          :
(2)          MOV      # $DN,$STSTNM      ;SAVE THIS
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          .: .IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(2)          .: .IFF NB      <>,      MOV      #,'COUNT      ;SET UP THIS ICOUNT
(2)          .: .IF NB <>
(2)          .: MOV      #,NEXT      ;GO TO THIS TLST WHEN THRU
(2)          .IFF
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP      $DE,<\$E>
(2)          .IF DF      TST '$DE
(2)          .: MOV      #TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          .IFF
(2)          .: MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .ENDC
(2)          .ENDC
(2)          .: .IFF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(2)          .NLIST

```

```

(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 003376 TSETUP MINT,ISYMOD,SEVEN,EVEPAR,26
(2) 003376 $RESET
(3) 003376 052777 000400 176322 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 003404 012777 000000 176310 MOV #ISYMOD,@PARCSR ;SET THE MODE
(2) 003412 $RESET
(3) 003412 052777 000400 176306 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 003420 012777 004020 176300 MOV #MINT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 003426 012777 005426 176266 MOV #ISYMOD!SEVEN!EVEPAR!26,@PARCSR
(1) 003434 016703 176266 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 003440 112777 000125 176264 MOVB #125,@TXDBUF ;LOAD DATA CHAR
(1) 003446 012767 001252 176024 MOV #1252,$TMP1 ;TO BE SHIFTED CHAR
(1) 003454 012767 000012 175440 MOV #10,SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCHRONIZATION
(1) 003462 $POKE
(2) 003462 052777 020000 176236 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 003470 042777 020000 176230 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 003476 005000 1$: CLR R0
(1) 003500 006067 175774 ROR $TMP1 ;FORCE CARRY
(1) 003504 103002 BCC 2$ ;BR IF CARRY CLR
(1) 003506 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 003512 2$:
(2) 003512 052777 020000 176206 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 003520 042777 020000 176200 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 003526 017701 176174 MOV @TXCSR,R1 ;ACTUAL
(1) 003532 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 003536 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 003540 001401 BEQ +4
(1) 003542 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 003544 005367 175352 DEC SHIFT ;# OF SHIFTS
(1) 003550 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 003552 052777 020000 176146 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 IDN <ISYMOD>,<ISYMOD>
(1) .IFT
(1) 003560 012700 000000 MOV #0,R0 ;EXPECTED
(1) .IFF
(1) MOV #100000,R0 ;EXPECTED
(1) .ENDC
(1) 003564 017701 176136 MOV @TXCSR,R1 ;ACTUAL
(1) 003570 042701 077777 BIC #77777,R1 ;SAVE DNA ONLY
(1) 003574 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 003576 001401 BEQ +4
(1) 003600 104003 ERROR 3 ;DNA SHOULD BE SET
(1) ;IF DNA DID NOT SET
  
```

```

(1)                                     ;CHECK WORD LENGTH SELECT LOGIC
(1)
8151 003602      $TRPAR SEVEN,10,,ODDPAR,ISYMOD,125,1652
(1)                                     ::THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1)                                     ::OF THE TRANSMITTER SECTION.
(1)                                     ::IT ALSO CHECKS DNA TIMING
(1)                                     ::MODE:ISYMOD
(1)                                     ::LENGTH:SEVEN PLUS PARITY
(1)                                     ::PARITY:ODDPAR
(1)                                     ::CHARACTER:125
(1)                                     ::
(1) 003602      $TSTNO
(2) 003602      NEWTST
(3)           000000 $NWTST=0
(3)           001   .IF B <>
(3) 003602      $$NEWTST      \ $TN,<>,SCOPE
(4)           .IRP      ASCII,<>
(4)           .IF EQ $NWTST
(4)           .NLIST
(4)           $NWTST=1
(4)           .SBTTL T2      ASCII
(4)           .LIST
(4)           STARS
(4)           ;*TEST 2      ASCII
(4)           .IFF
(4)           ASCII
(4)           .ENDC
(4)           .ENDM
(4) 003602      STARS
(5)           002   .IF B
(5)           ;*****
(5)           .IFF
(5)           .NLIST
(5)           .REPT
(5)           .LIST
(5)           ;*****
(5)           .NLIST
(5)           .ENDR
(5)           .LIST
(5) 003602      001   .ENDC
(4)           000004 TST2: SCOPE
(4)           000003 $TN=$TN+1
(3)           .IFF
(3)           $$NEWTST      \ $TN,<>,<>
(3)           .ENDC
(3)           000   .IF NE 40:0B$$SWR
(3)           001   .IF NB
(3)           002   .IF LE <-1>
(3)           003   MOV      #1,$TIMES      ;;DO 1 ITERATION
(3)           .IFF
(3)           MOV      #,$TIMES      ;;DO ITERATIONS
(3)           .ENDC
(3)           002   .ENDC
(3)           001   .ENDC
(3)           002   .IF NB
(3)           MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS

```

```

(3)          001          .ENDC
(3)          000          .ENDC
(3)          001          .IF NB
(3)          002          .IF DF $MAIL
(3)          .IRP        TNM,<\$TN-1>
(3)          .MOV        #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(3)          .ENDM
(3)          001          .ENDC
(3)          000          .ENDC
(2)          000000      .REPT 0
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP        $DN,<\$N>
(2)          NEWST
(2)          :           MOV        #'$DN,$STNM      ;SAVE THIS
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          :           .IF NB      <>,      MOV        #,RETURN      ;SET UP THIS RETURN
(2)          :           .IF NB      <>,      MOV        #,ICOUNT      ;SET UP THIS ICOUNT
(2)          :           .IF NB      <>
(2)          :           .MOV        #,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .IFF
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP        $DE,<\$E>
(2)          .IF DF      TST '$DE
(2)          :           .MOV        #'TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          .IFF
(2)          :           .MOV        #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .ENDC
(2)          .ENDC
(2)          :           .IF NB      <>,      MOV        #,LOC      ;SET UP FOR SCOPE LOOP
(2)          .NLIST
(2)          .RADIX 10
(2)          $N=$N+1
(2)          $E=$E+1
(2)          .RADIX 8
(2)          .LIST
(2)          .ENDR
(1) 003604      TSETUP MINT,ISYMOD,SEVEN,ODDPAR,26
(2) 003604      $RESET
(3) 003604      052777 000400 176114      BIS        #MRESET,@TXCSR ;MASTER RESET
(2) 003612      012777 000000 176102      MOV        #ISYMOD,@PARCSR ;SET THE MODE
(2) 003620      $RESET
(3) 003620      052777 000400 176100      BIS        #MRESET,@TXCSR ;MASTER RESET
(2)
(2)          ;SET MAINTENANCE MODE & SEND
(2)          ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)

```



```

(3) 004010          $$NEWTEST      \STN,<>,SCOPE
(4)                .IRP          ASCII,<>
(4)                .IF EQ $NWTST
(4)                .NLIST
(4)                $NWTST=1
(4)                .SBTTL T3      ASCII
(4)                .LIST
(4)                STARS
(4)                :*TEST 3      ASCII
(4)                .IFF
(4)                ASCII
(4)                .ENDC
(4)                .ENDM
(4) 004010          STARS
(5)                002          .IF B
(5)                :*****
(5)                .IFF
(5)                .NLIST
(5)                .REPT
(5)                .LIST
(5)                :*****
(5)                .NLIST
(5)                .ENDR
(5)                .LIST
(5)                .ENDC
(4) 004010          001          TST3:  SCOPE
(4)                000004
(4)                000004
(3)                $TN=$TN+1
(3)                .IFF
(3)                $$NEWTEST      \STN,<>,<>
(3)                .ENDC
(3)                001          .IF NE 4000&$$SWR
(3)                002          .IF NB
(3)                003          .IF LE <-1>
(3)                .IFF          MOV #1,$TIMES      ;;DO 1 ITERATION
(3)                .IFF          MOV #,$TIMES      ;;DO ITERATIONS
(3)                .ENDC
(3)                002          .ENDC
(3)                001          .ENDC
(3)                002          .IF NB
(3)                .ENDC          MOV #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(3)                001          .ENDC
(3)                000          .ENDC
(3)                001          .IF NB
(3)                002          .IF DF $MAIL
(3)                .IRP          TNM,<\STN-1>
(3)                .IRP          MOV #'TNM,$TESTN  ;;SET TEST NUMBLR IN APT MAIL BOX
(3)                .ENDM
(3)                001          .ENDC
(3)                000          .ENDC
(2)                000000          .REPT 0
(2)                .NLIST
(2)                .RADIX 10
(2)                .LIST
(2)                .IRP          $DN,<\$N>
(2)                NEWTST

```

INITIALIZE THE COMMON TAGS

```

(2)      :      MOV      #SDN,$STNM      ;SAVE THIS
(2)      :.ENDM
(2)      :.NLIST
(2)      :.RADIX 8
(2)      :.LIST
(2)      :.IIF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(2)      :.IIF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(2)      :.IF NB <>
(2)      :      MOV      #,NEXT      ;GO TO THIS TEST WHEN THRU
(2)      :.IFF
(2)      :.NLIST
(2)      :.RADIX 10
(2)      :.LIST
(2)      :.IRP      $DE,<\$E>
(2)      :.IF DF      TST '$DE
(2)      :      MOV      #TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(2)      :.ENDM
(2)      :.NLIST
(2)      :.RADIX 8
(2)      :.LIST
(2)      :.IFF
(2)      :      MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(2)      :.ENDC
(2)      :.ENDC
(2)      :.IIF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(2)      :.NLIST
(2)      :.RADIX 10
(2)      :$N=$N+1
(2)      :$E=$E+1
(2)      :.RADIX 8
(2)      :.LIST
(2)      :.ENDR
(1) 004012 TSETUP MINT,SYNINT,EIGHT,EVEPAR,26
(2) 004012 $RESET
(3) 004012 052777 000400 175706      BIS      #MRESET,@TXCSR      ;MASTER RESET
(2) 004020 C12777 030000 175674      MOV      #SYNINT,@PARCSR      ;SET THE MODE
(2) 004026 $RESET
(3) 004026 052777 000400 175672      BIS      #MRESET,@TXCSR      ;MASTER RESET
(2)
(2)      :SET MAINTENANCE MODE & SEND
(2)      :NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 004034 012777 004020 175664      MOV      #MINT!SEND,@TXCSR
(2)
(2)      :SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2)      :MOV      #SYNINT!EIGHT!EVEPAR!26,@PARCSR
(1) 004050 C16703 175652      MOV      TXCSR,R3      ;SET UP FOR ERROR MSG
(1) 004054 112777 000125 175650      MOV      #125,@TXDBUF      ;LOAD DATA CHAR
(1) 004062 012767 000125 175410      MOV      #125,$TMP1      ;TO BE SHIFTED CHAR
(1) 004070 012767 000011 175024      MOV      #9,$SHIFT      ;# OF SHIFTS
(1)      :POKE CLK TO GET INTO SYNCRONIZATION
(1) 004076 $POKE
(2) 004076 052777 020000 175622      BIS      #CLK,@TXCSR      ;POKE CLK UP
(2) 004104 042777 020000 175614      BIC      #CLK,@TXCSR      ;POKE CLK DOWN
(1) 004112 005000      CLR      R0
(1) 004114 006067 175360      ROR      $TMP1      ;FORCE CARRY
(1) 004120 103002      BCC     2$      ;BR IF CARRY CLR

```



```

(1) 004122 052700 002000      BIS      #BITW,RO      ;EQUIV OF BITW
(1) 004126      2$:
(2) 004126 052777 020000 175572  BIS      #CLK,@TXCSR    ;POKE CLK UP
(2) 004134 042777 020000 175564  BIC      #CLK,@TXCSR    ;POKE CLK DOWN
(1) 004142 017701 175560      MOV      @TXCSR,R1      ;ACTUAL
(1) 004146 042701 075777      BIC      #075777,R1     ;SAVE BITW & DNA
(1) 004152 020001      CMP      RO,R1         ;COMPARE EXP VS ACT
(1) 004154 001401      BEQ     +4
(1) 004156 104003      ERROR   3             ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1)      ;BIT,...ALSO CHECK DNA
(1) 004160 005367 174736      DEC     SHIFT         ;# OF SHIFTS
(1) 004164 001322      BNE     1$           ;DO IT AGAIN ?
(1)      ;NOW POKE CLK TO SEE DNA
(1) 004166 052777 020000 175532  BIS      #CLK,@TXCSR    ;POKE CLK
(1)      001      IDN      <ISYMOD>,<SYNINT>
(1)      .IF
(1)      .IFT
(1)      MOV      #0,RO    ;EXPECTED
(1)      .IFF
(1) 004174 012700 100000      MOV      #10C000,RO    ;EXPECTED
(1)      000      .ENDC
(1) 004200 017701 175522      MOV      @TXCSR,R1     ;ACTUAL
(1) 004204 042701 077777      BIC      #77777,R1     ;SAVE DNA ONLY
(1) 004210 020001      CMP      RO,R1         ;COMPARE EXP VS ACT
(1) 004212 001401      BEQ     +4
(1) 004214 104003      ERROR   3             ;DNA SHOULD BE SET
(1)      ;IF DNA DID NOT SET
(1)      ;CHECK WORD LENGTH SELECT LOGIC
8153 004216      $TRPAR  EIGHT,9,ODDPAR,SYNINT,125,525
(1)      ;;THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1)      ;;OF THE TRANSMITTER SECTION.
(1)      ;;IT ALSO CHECKS DNA TIMING
(1)      ;;MODE:SYNINT
(1)      ;;LENGTH:EIGHT PLUS PARITY
(1)      ;;PARITY:ODDPAR
(1)      ;;CHARACTER:125
(1)      ;;
(1) 004216      $TSTNO
(2) 004216      NEWTST
(3)      000000      $NWTST=0
(3)      001      .IF B <>
(3) 004216      $$NEWTST      \ $TN,<> ,SCOPE
(4)      .IRP      ASCII,<>
(4)      .IF EQ $NWTST
(4)      .NLIST
(4)      $NWTST=1
(4)      .SBTTL  T4      ASCII
(4)      .LIST
(4)      STARS
(4)      :*TEST 4      ASCII
(4)      .IFF
(4)      ASCII
(4)      .ENDC
(4)      .ENDM
(4) 004216      STARS
(5)      002      .IF B

```

```

(5) .....
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) .....
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) .ENDC
(4) 004216 001 000004 TST4: SCOPE
(4) 000005 $TN=$TN+1
(3) .IFF
(3) $NEWTEST \STN,<>,<>
(3) .ENDC
(3) 000 .IF NE 4000&$SWR
(3) 001 .IF NB
(3) 002 .IF LE <-1>
(3) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(3) .IFF MOV #,$TIMES ;;DO ITERATIONS
(3) .ENDC
(3) 002 .ENDC
(3) 001 .ENDC
(3) 002 .IF NB
(3) MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(3) .ENDC
(3) 001 .ENDC
(3) 000 .IF NB
(3) 001 .IF DF $MAIL
(3) 002 .IRP TNM,<\STN-1>
(3) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
(3) .ENDC
(3) 001 .ENDC
(3) 000 .REPT 0
(2) 000000 .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DN,<\$N>
(2) NEWTST
(2) : MOV #,$DN,$STNM ;SAVE THIS
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) : .IFF NB <>, MOV #,RETURN ;SET UP THIS RETURN
(2) : .IFF NB <>, MOV #,ICOUNT ;SET UP THIS ICOUNT
(2) : .IF NB <>
(2) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(2) .IFF
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DE,<\$E>
(2) .IF DF TST '$DE

```

INITIALIZE THE COMMON TAGS

```

(2) ; MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) .IFF
(2) ; MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDC
(2) .ENDC
(2) ;.IIF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(2) .NLIST
(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 004220 TSETUP MINT,SYNINT,EIGHT,ODDPAR,26
(2) 004220 $RESET
(3) 004220 052777 000400 175500 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 004226 012777 030000 175466 MOV #SYNINT,@PARCSR ;SET THE MODE
(2) 004234 $RESET
(3) 004234 052777 000400 175464 BIS #MRESET,@TXCSR ;MASTER RESET
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MIDATA=0)
(2) 004242 012777 004020 175456 MOV #MINT!SEND,@TXCSR
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 004250 012777 037026 175444 MOV #SYNINT!EIGHT!ODDPAR!26,@PARCSR
(1) 004256 016703 175444 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 004262 112777 000125 175442 MOVB #125,@TXDBUF ;LOAD DATA CHAR
(1) 004270 012767 000525 175202 MOV #525,$TMP1 ;TO BE SHIFTED CHAR
(1) 004276 012767 000011 174616 MOV #9,$SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCHRONIZATION
(1) 004304 $POKE
(2) 004304 052777 020000 175414 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 004312 042777 020000 175406 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 004320 005000 1$: CLR R0
(1) 004322 006067 175152 ROR $TMP1 ;FORCE CARRY
(1) 004326 103002 BCC 2$ ;BR IF CARRY CLR
(1) 004330 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 004334 2$:
(2) 004334 052777 020000 175364 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 004342 042777 020000 175356 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 004350 017701 175352 MOV @TXCSR,R1 ;ACTUAL
(1) 004354 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 004360 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 004362 001401 BEQ +4
(1) 004364 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 004366 005367 174530 DEC SHIFT ;# OF SHIFTS
(1) 004372 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 004374 052777 020000 175324 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 .IF IDN <ISYMOD>,<SYNINT>

```

```
(1) .IFT
(1) .IFF      MOV      #0,R0      ;EXPECTED
(1) 004402   012700  100000
(1)          000
(1) 004406   017701  175314
(1) 004412   042701  077777
(1) 004416   020001
(1) 004420   001401
(1) 004422   104003
(1)
(1)
(1)
(1) 8154 004424      STRPAR  EIGHT,11,,EVEPAR,ISYMOD,125,2252
(1)          ::THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1)          ::OF THE TRANSMITTER SECTION.
(1)          ::IT ALSO CHECKS DNA TIMING
(1)          ::MODE:ISYMOD
(1)          ::LENGTH:EIGHT PLUS PARITY
(1)          ::PARITY:EVEPAR
(1)          ::CHARACTER:125
(1)          ::
(1) 004424      $TSTNO
(2) 004424      NEWTST
(3)          $NWTST=0
(3)          .IF B <>
(3) 004424          $$NEWTST      \ $TN,<>,SCOPE
(4)          .IRP      ASCII,<>
(4)          .IF EQ $NWTST
(4)          .NLIST
(4)          $NWTST=1
(4)          .SBTTL  T5      ASCII
(4)          .LIST
(4)          STARS
(4)          :*TEST 5      ASCII
(4)          .IFF
(4)          ASCII
(4)          .ENDC
(4)          .ENDM
(4) 004424      STARS
(5)          .IF B
(5)          ::*****
(5)          .IFF
(5)          .NLIST
(5)          .REPT
(5)          .LIST
(5)          ::*****
(5)          .NLIST
(5)          .ENDR
(5)          .LIST
(5)          .ENDC
(4) 004424      001
(4)          000004      TST5:  SCOPE
(4)          000006
(3)          $TN=$TN+1
(3)          .IFF
(3)          $$NEWTST      \ $TN,<>,<>
```

```
(3)          000          .ENDC
(3)          001          .IF NE 4000&$$SWR
(3)          002          .IF NB
(3)          003          .IF LE <-1>
(3)                        MOV          #1,$TIMES          ;;DO 1 ITERATION
(3)          .IFF
(3)                        MOV          #,$TIMES          ;;DO ITERATIONS
(3)          002          .ENDC
(3)          001          .ENDC
(3)          002          .IF NB
(3)                        MOV          #,$LPADR          ;;SET SCOPE LOOP ADDRESS
(3)          001          .ENDC
(3)          000          .ENDC
(3)          001          .IF NB
(3)          002          .IF DF $MAIL
(3)          .IRP          $MAIL
(3)                        TNM,<\$TN-1>
(3)                        MOV          #'TNM,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
(3)          .ENDM
(3)          001          .ENDC
(3)          000          .ENDC
(2)          000000      .REPT 0
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP          $DN,<\$N>
(2)          NEWTST
(2)          :
(2)                        MOV          #$DN,$TSTNM          ;SAVE THIS
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          :.IIF NB      <>,      MOV          #,RETURN          ;SET UP THIS RETURN
(2)          :.IIF NB      <>,      MOV          #,ICOUNT          ;SET UP THIS ICOUNT
(2)          :.IF NB      <>
(2)          :
(2)                        MOV          #,NEXT          ;GO TO THIS TEST WHEN THRU
(2)          .IFF
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP          $DE,<\$E>
(2)          .IF DF      TST '$DE
(2)          :
(2)                        MOV          #TST'$DE,NEXT          ;GO TO THIS TEST WHEN THRU
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          .IFF
(2)          :
(2)                        MOV          #.EOP,NEXT          ;GO TO THIS TEST WHEN THRU
(2)          .ENDC
(2)          .ENDC
(2)          :.IIF NB      <>,      MOV          #,LOCK          ;SET UP FOR SCOPE LOOP
(2)          .NLIST
(2)          .RADIX 10
(2)          $N=$N+1
(2)          $E=$E+1
(2)          .RADIX 8
```

INITIALIZE THE COMMON TAGS

```

(2) .LIST
(2) .ENDR
(1) 004426 ISETUP MINT,ISYMOD,EIGHT,EVEPAR,26
(2) 004426 $RESET
(3) 004426 052777 000400 175272 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 004434 012777 000000 175260 MOV #ISYMOD,@PARCSR ;SET THE MODE
(2) 004442 $RESET
(3) 004442 052777 000400 175256 BIS #MRESFT,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 004450 012777 004020 175250 MOV #MINT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 004456 012777 007426 175236 MOV #ISYMOD!EIGHT!EVEPAR!26,@PARCSR
(1) 004464 016703 175236 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 004470 112777 000125 175234 MOVB #125,@TXDBUF ;LOAD DATA CHAR
(1) 004476 012767 002252 174774 MOV #2252,$TMP1 ;TO BE SHIFTED CHAR
(1) 004504 012767 000013 174410 MOV #11,SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCHRONIZATION
(1) $POKE
(2) 004512 052777 020000 175206 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 004520 042777 020000 175200 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 004526 005000 1$: CLR R0
(1) 004530 006067 174744 ROR $TMP1 ;FORCE CARRY
(1) 004534 103002 BCC 2$ ;BR IF CARRY CLR
(1) 004536 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 004542 2$:
(2) 004542 052777 020000 175156 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 004550 042777 020000 175150 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 004556 017701 175144 MOV @TXCSR,R1 ;ACTUAL
(1) 004562 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 004566 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 004570 001401 BEQ +4
(1) 004572 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 004574 005367 174322 DEC SHIFT ;# OF SHIFTS
(1) 004600 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 004602 052777 020000 175116 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 IDN <ISYMOD>,<ISYMOD>
(1) .IFT
(1) 004610 012700 000000 MOV #0,R0 ;EXPECTED
(1) .IFF
(1) MOV #100000,R0 ;EXPECTED
(1) .ENDC
(1) 004614 017701 175106 MOV @TXCSR,R1 ;ACTUAL
(1) 004620 042701 077777 BIC #77777,R1 ;SAVE DNA ONLY
(1) 004624 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 004626 001401 BEQ +4
(1) 004630 104003 ERROR 3 ;DNA SHOULD BE SET
(1) ;IF DNA DID NOT SET
(1) ;CHECK WORD LENGTH SELECT LOGIC
(1)
(1)
(1) 8155 004632 $TRPAR EIGHT,11,ODDPAR,ISYMOD,125,3252
(1) ;:THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION

```

```
(1)                                     ;;OF THE TRANSMITTER SECTION.  
(1)                                     ;;IT ALSO CHECKS DNA TIMING  
(1)                                     ;;MODE:ISYMOD  
(1)                                     ;;LENGTH:EIGHT PLUS PARITY  
(1)                                     ;;PARITY:ODDPAR  
(1)                                     ;;CHARACTER:125  
(1)                                     ;;  
(1) 004632 $TSTNO  
(2) 004632 NEWTST  
(3) 000000 $NWTST=0  
(3) 001 .IF B <>  
(3) 004632 $$NEWTST \STN,<>,SCOPE  
(4) .IRP ASCII,<>  
(4) .IF EQ $NWTST  
(4) .NLIST  
(4) $NWTST=1  
(4) .SBTTL T6 ASCII  
(4) .LIST  
(4) STARS  
(4) ;*TEST 6 ASCII  
(4) .IFF  
(4) ASCII  
(4) .ENDC  
(4) .ENDM  
(4) 004632 STARS  
(5) 002 .IF B  
(5) ;*****  
(5) .IFF  
(5) .NLIST  
(5) .REPT  
(5) .LIST  
(5) ;*****  
(5) .NLIST  
(5) .ENDR  
(5) .LIST  
(5) 001 .ENDC  
(4) 004632 000004 TST6: SCOPE  
(4) 000007 $TN=$TN+1  
(3) .IFF  
(3) $$NEWTST \STN,<>,<>  
(3) .ENDC  
(3) 000 .IF NE 4000$$SWR  
(3) 001 .IF NB  
(3) 002 .IF LE <-1>  
(3) 003 MOV #1,$TIMES ;;DO 1 ITERATION  
(3) .IFF  
(3) MOV #,$TIMES ;;DO ITERATIONS  
(3) .ENDC  
(3) 002 .ENDC  
(3) 001 .ENDC  
(3) 002 .IF NB  
(3) MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS  
(3) .ENDC  
(3) 001 .ENDC  
(3) 001 .IF NB  
(3) 002 .IF DF $MAIL
```

```
(3) .IRP TNM,<\$TN-1>
(3) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
(3) .ENDC
(3) .ENDC
(2) 001 .REPT 0
(2) 000 .NLIST
(2) 000000 .RADIX 10
(2) .LIST
(2) .IRP $DN,<\$N>
(2) NEWTST
(2) : MOV #SDN,$STNM ;SAVE THIS
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) :.IIF NB <>, MOV #,RETURN ;SET UP THIS RETURN
(2) :.IIF NB <>, MOV #,ICOUNT ;SET UP THIS ICOUNT
(2) :.IF NB <>
(2) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(2) .IFF
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DE,<\$E>
(2) .IF DF TST '$DE
(2) : MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) .IFF
(2) : MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDC
(2) .ENDC
(2) :.IIF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(2) .NLIST
(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 004634 TSETUP MINT,ISYMOD,EIGHT,ODDPAR,26
(2) 004634 $RESET
(3) 004634 052777 000400 175064 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 004642 012777 000000 175052 MOV #ISYMOD,@PARCSR ;SET THE MODE
(2) 004650 $RESET
(3) 004650 052777 000400 175050 BIS #MRESET,@TXCSR ;MASTER RESET
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 004656 012777 004020 175042 MOV #MINT!SEND,@TXCSR
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 004664 012777 007026 175030 MOV #ISYMOD!EIGHT!ODDPAR!26,@PARCSR
```


8

CNDUU-A MACY11 30(1046) 14-DEC-82 10:02
CNDUUA.M11 29-OCT-82 13:27

PAGE 62-47
INITIALIZE THE COMMON TAGS

J 5

SEQ 0061

```

(1) 004672 016703 175030      MOV    TXCSR,R3      ;SET UP FOR ERROR MSG
(1) 004676 112777 000125 175026  MOVB   #125,@TXDBUF  ;LOAD DATA CHAR
(1) 004704 012767 003252 174566  MOV    #3252,$TMP1  ;TO BE SHIFTED CHAR
(1) 004712 012767 000013 174202  MOV    #11,SHIFT    ;# OF SHIFTS
(1)                                     ;POKE CLK TO GET INTO SYNCRONIZATION
(1) 004720                                     $POKE
(2) 004720 052777 020000 175000  BIS    #CLK,@TXCSR  ;POKE CLK UP
(2) 004726 042777 020000 174772  BIC    #CLK,@TXCSR  ;POKE CLK DOWN
(1) 004734 005000                                     1$: CLR    RO
(1) 004736 006067 174536                                     ROR    $TMP1 ;FORCE CARRY
(1) 004742 103002                                     BCC    2$ ;BR IF CARRY CLR
(1) 004744 052700 002000                                     BIS    #BITW,RO ;EQUIV OF BITW
(1) 004750                                     2$:
(2) 004750 052777 020000 174750  BIS    #CLK,@TXCSR  ;POKE CLK UP
(2) 004756 042777 020000 174742  BIC    #CLK,@TXCSR  ;POKE CLK DOWN
(1) 004764 017701 174736                                     MOV    @TXCSR,R1 ;ACTUAL
(1) 004770 042701 075777                                     BIC    #075777,R1 ;SAVE BITW & DNA
(1) 004774 020001                                     CMP    RO,R1 ;COMPARE EXP VS ACT
(1) 004776 001401                                     BEQ    +4
(1) 005000 104003                                     ERROR  3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1)                                     ;BIT...ALSO CHECK DNA
(1) 005002 005367 174114                                     DEC    SHIFT ;# OF SHIFTS
(1) 005006 001352                                     BNE    1$ ;DO IT AGAIN ?
(1)                                     ;NOW POKE CLK TO SEE DNA
(1) 005010 052777 020000 174710  BIS    #CLK,@TXCSR  ;POKE CLK
(1) 001                                     .IF   IDN    <ISYMOD>,<ISYMOD>
(1) 005016 012700 000000                                     .IFT
(1)                                     MOV    #0,P0 ;EXPECTED
(1)                                     .IFF
(1)                                     MOV    #100000,RO ;EXPECTED
(1) 000                                     .ENDC
(1) 005022 017701 174700                                     MOV    @TXCSR,R1 ;ACTUAL
(1) 005026 042701 077777                                     BIC    #77777,R1 ;SAVE DNA ONLY
(1) 005032 020001                                     CMP    RO,R1 ;COMPARE EXP VS ACT
(1) 005034 001401                                     BEQ    +4
(1) 005036 104003                                     ERROR  3 ;DNA SHOULD BE SET
(1)                                     ;IF DNA DID NOT SET
(1)                                     ;CHECK WORD LENGTH SELECT LOGIC
8156
8157 005040 $TRPAR EIGHT,9,EVEPAR,SYNINT,252,252
(1) ;:THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1) ;:OF THE TRANSMITTER SECTION.
(1) ;:IT ALSO CHECKS DNA TIMING
(1) ;:MODE:SYNINT
(1) ;:LENGTH:EIGHT PLUS PARITY
(1) ;:PARITY:EVEPAR
(1) ;:CHARACTER:252
(1) ;:
(1) 005040 $STNO
(2) 005040 NEWTST
(3) 000000 $NWTST=0
(3) 001 ;.IF B <>
(3) 005040 ;$$NEWTST \STN,<>,SCOPE
(4) ;.IRP ASCII,<>
(4) ;.IF EQ $NWTST

```

```
(4) .NLIST
(4) $NWTST=1
(4) .SBITL T7 ASCII
(4) .LIST
(4) STARS
(4) :*TEST 7 ASCII
(4) .IFF
(4) ASCII
(4) .ENDC
(4) .ENDM
(4) STARS
(4) 005040 002 .IF B
(5) :*****
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) :*****
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) 005040 001 .ENDC
(4) 000004 TST7: SCOPE
(4) 000010 $TN=$TN+1
(3) .IFF
(3) $$NEWTST \ $TN,<>,<>
(3) .ENDC
(3) 000 .IF NE 4000&$$SWR
(3) 001 .IF NB
(3) 002 .IF LE <-1>
(3) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(3) .IFF
(3) MOV #,$TIMES ;;DO ITERATIONS
(3) .ENDC
(3) 002 .ENDC
(3) 001 .ENDC
(3) 002 .IF NB
(3) MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(3) .ENDC
(3) 001 .ENDC
(3) 000 .IF NB
(3) 001 .IF DF $MAIL
(3) 002 .IRP
(3) TNM,<\$TN-1>
(3) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
(3) 001 .ENDC
(3) 000 .ENDC
(2) 000000 .REPT 0
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DN,<\$N>
(2) NEWTST
(2) : MOV # $DN,$TSTNM ;SAVE THIS
(2) .ENDM
(2) .NLIST
```

```

(2) .RADIX 8
(2) .LIST
(2) :.IF NB <>, MOV #,RETURN ;SET UP THIS RETURN
(2) :.IF NB <>, MOV #,!COUNT ;SET UP THIS !COUNT
(2) :.IF NB <>
(2) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(2) :.IFF
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DE,<\$E>
(2) :.IF DF TST'$DE
(2) : MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) :.IFF
(2) : MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDC
(2) .ENDC
(2) :.IF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(2) .NLIST
(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 005042 TSETUP MINT,SYNINT,EIGHT,EVEPAR,26
(2) 005042 $RESET
(3) 005042 052777 000400 174656 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 005050 012777 030000 174644 MOV #SYNINT,@PARCSR ;SET THE MODE
(2) 005056 $RESET
(3) 005056 052777 000400 174642 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 005064 012777 004020 174634 MOV #MINT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 005072 012777 037426 174622 MOV #SYNINT!EIGHT!EVEPAR!26,@PARCSR
(1) 005100 016703 174622 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 005104 112777 000252 174620 MOVB #252,@TXDBUF ;LOAD DATA CHAR
(1) 005112 012767 000252 174360 MOV #252,$TMP1 ;TO BE SHIFTED CHAR
(1) 005120 012767 000011 173774 MOV #9,$SHIFT ;# OF SHIFTS
(1)
(1) ;POKE CLK TO GET INTO SYNCRONIZATION
(1) 005126 $POKE
(2) 005126 052777 020000 174572 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 005134 042777 020000 174564 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 005142 005000 1$: CLR R0
(1) 005144 006067 174330 ROR $TMP1 ;FORCE CARRY
(1) 005150 103002 BCC 2$ ;BR IF CARRY CLR
(1) 005152 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 005156 2$:
(2) 005156 052777 020000 174542 BIS #CLK,@TXCSR ;POKE CLK UP
  
```

51

```

(2) 005164 042777 020000 174534 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 005172 017701 174530 MOV @TXCSR,R1 ;ACTJAL
(1) 005176 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 005202 020001 CMP RO,R1 ;COMPARE EXP VS ACT
(1) 005204 001401 BEQ +4
(1) 005206 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT...ALSO CHECK DNA
(1) 005210 005367 173706 DEC SHIFT ;# OF SHIFTS
(1) 005214 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 005216 052777 020000 174502 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 IDN <ISYMOD>,<SYNINT>
(1) .IFT
(1) MOV #0,RO ;EXPECTED
(1) .IFF
(1) 005224 012700 100000 MOV #100000,RO ;EXPECTED
(1) 000 .ENDC
(1) 005230 017701 174472 MOV @TXCSR,R1 ;ACTUAL
(1) 005234 042701 077777 BIC #77777,R1 ;SAVE DNA ONLY
(1) 005240 020001 CMP RO,R1 ;COMPARE EXP VS ACT
(1) 005242 001401 BEQ +4
(1) 005244 104003 ERROR 3 ;DNA SHOULD BE SET
(1) ;IF DNA DID NOT SET
(1) ;CHECK WORD LENGTH SELECT LOGIC
(1)
8158 005246 $TRPAR EIGHT,9,ODDPAR,SYNINT,252,652
(1) ;:THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1) ;:OF THE TRANSMITTER SECTION.
(1) ;:IT ALSO CHECKS DNA TIMING
(1) ;:MODE:SYNINT
(1) ;:LENGTH:EIGHT PLUS PARITY
(1) ;:PARITY:ODDPAR
(1) ;:CHARACTER:252
(1) ;:
(1) 005246 $TSTNO
(2) 005246 NEWTST
(3) 000000 $NWTST=0
(3) 001 .IF B <>
(3) 005246 $$NEWTST \STN,<>,SCOPE
(4) .IRP ASCII,<>
(4) .IF EQ $NWTST
(4) .NLIST
(4) $NWTST=1
(4) .SBTTL T10 ASCII
(4) .LIST
(4) STARS
(4) ;*TEST 10 ASCII
(4) .IFF
(4) ASCII
(4) .ENDC
(4) .ENDM
(4) 005246 STARS
(5) 002 .IF B
(5) ;:*****
(5) .IFF
(5) .NLIST

```

INITIALIZE THE COMMON TAGS

```

(5) .REPT
(5) .LIST
(5) :*****
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) .ENDC
(4) 005246 001
(4) 000004
(4) 000011
(3) $TN=$TN+1
(3) .IFF
(3)     $$NEWTST     \STN,<>,<>
(3) .ENDC
(3) 000
(3) 001 .IF NE 4000$$SWR
(3) 002 .IF NB
(3) 003 .IF LE <-1>
(3)     MOV #1,$TIMES     ;;DO 1 ITERATION
(3) .IFF
(3)     MOV #,$TIMES     ;;DO ITERATIONS
(3) .ENDC
(3) 002
(3) 001 .ENDC
(3) 002 .IF NB
(3)     MOV #,$LPADR     ;;SET SCOPE LOOP ADDRESS
(3) .ENDC
(3) 001 .ENDC
(3) 000 .IF NB
(3) 001 .IF DF $MAIL
(3) 002 .IRP
(3)     TNM,<\STN-1>
(3)     MOV #'TNM,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
(3) 001 .ENDC
(3) 000 .ENDC
(2) 000000
(2) .REPT 0
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DN,<\SN>
(2) NEWTST
(2) :
(2)     MOV #SDN,$STNM     ;SAVE THIS
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) :.IFF NB <>     MOV #,RETURN     ;SET UP THIS RETURN
(2) :.IFF NB <>     MOV #,ICOUNT     ;SET UP THIS ICOUNT
(2) :.IF NB <>
(2) :     MOV #,NEXT     ;GO TO THIS TEST WHEN THRU
(2) .IFF
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DE,<\SE>
(2) .IF DF TST '$DE
(2) :     MOV #TST'$DE,NEXT     ;GO TO THIS TEST WHEN THRU
(2) .ENDM
(2) .NLIST

```

```

(2) .RADIX 8
(2) .LIST
(2) .IFF
(2) : MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDC
(2) .ENDC
(2) :.IIF NB <>, MOV #.LOCK ;SET UP FOR SCOPE LOOP
(2) .NLIST
(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 005250 TSETUP MINT,SYNINT,EIGHT,ODDPAR,26
(2) 005250 $RESET
(3) 005250 052777 000400 174450 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 005256 012777 030000 174436 MOV #SYNINT,@PARCSR ;SET THE MODE
(2) 005264 $RESET
(3) 005264 052777 000400 174434 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 005272 012777 004020 174426 MOV #MINT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 005300 012777 037026 174414 MOV #SYNINT!EIGHT!ODDPAR!26,@PARCSR
(1) 005306 016703 174414 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 005312 112777 000252 174412 MOV#B #252,@TXDBUF ;LOAD DATA CHAR
(1) 005320 012767 000652 174152 MOV #652,$TMP1 ;TO BE SHIFTED CHAR
(1) 005326 012767 000011 173566 MOV #9,$SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCRONIZATION
(1) 005334 $POKE
(2) 005334 052777 020000 174364 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 005342 042777 020000 174356 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 005350 005000 1$: CLR R0
(1) 005352 006067 174122 ROR $TMP1 ;FORCE CARRY
(1) 005356 103002 BCC 2$ ;BR IF CARRY CLR
(1) 005360 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 005364 2$:
(2) 005364 052777 020000 174334 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 005372 042777 020000 174326 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 005400 017701 174322 MOV @TXCSR,R1 ;ACTUAL
(1) 005404 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 005410 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 005412 001401 BEQ +4
(1) 005414 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 005416 005367 173500 DEC SHIFT ;# OF SHIFTS
(1) 005422 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 005424 052777 020000 174274 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 IDN <ISYMOD>,<SYNINT>
(1) .IFT
(1) MOV #0,R0 ;EXPECTED
(1) .IFF

```

```
(1) 005432 012700 100000      MOV      #100000,R0      ;EXPECTED
(1) 005436 017701 174264      .ENDC
(1) 005442 042701 077777      MOV      @TXCSR,R1      ;ACTUAL
(1) 005446 020001              BIC      #77777,R1      ;SAVE DNA ONLY
(1) 005450 001401              CMP      R0,R1          ;COMPARE EXP VS ACT
(1) 005452 104003              BEQ      +4              ;DNA SHOULD BE SET
(1)                                     ERROR    3                ;IF DNA DID NOT SET
(1)                                     ;CHECK WORD LENGTH SELECT LOGIC
8159 005454      $TRPAR  EIGHT,9,,EVEPAR,SYNINT,0,0
(1)                                     ::THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1)                                     ::OF THE TRANSMITTER SECTION.
(1)                                     ::IT ALSO CHECKS DNA TIMING
(1)                                     ::MODE:SYNINT
(1)                                     ::LENGTH:EIGHT PLUS PARITY
(1)                                     ::PARITY:EVEPAR
(1)                                     ::CHARACTER:0
(1)                                     ::
(1) 005454      $STSTNO
(2) 005454      NEWTST
(3) 000000      $NWTST=0
(3) 001          .IF B <>
(3) 005454      $$NEWTST      \ $TN,<> ,SCOPE
(4) .IRP      ASCII,<>
(4) .IF EQ $NWTST
(4) .NLIST
(4) $NWTST=1
(4) .SBTTL  T11      ASCII
(4) .LIST
(4) STARS
(4) :*TEST 11      ASCII
(4) .IFF
(4) ASCII
(4) .ENDC
(4) .ENDM
(4) 005454      STARS
(5) 002          .IF B
(5) ::*****
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) ::*****
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) .ENDC
(4) 005454      001
(4) 000004      TST11:  SCOPE
(4) 000012      $TN=$TN+1
(3) .IFF
(3) $$NEWTST      \ $TN,<> ,<>
(3) .ENDC
(3) 000          .IF NE 4000$$SWR
(3) 001          .IF NB
(3) 002
```

```
(3)          003      .IF LE <-1>
(3)              MOV      #1,$TIMES      ;;DO 1 ITERATION
(3)              .IFF
(3)              MOV      #,$TIMES      ;;DO ITERATIONS
(3)          002      .ENDC
(3)          001      .ENDC
(3)          002      .IF NB
(3)              MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(3)          001      .ENDC
(3)          000      .ENDC
(3)          001      .IF NB
(3)          002      .IF DF $MAIL
(3)              .IRP      TNM,<\$TN-1>
(3)                  MOV      #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(3)              .ENDM
(3)          001      .ENDC
(3)          000      .ENDC
(2)          000000    .REPT 0
(2)              .NLIST
(2)              .RADIX 10
(2)              .LIST
(2)              .IRP      $DN,<\$N>
(2)          NEWTST
(2)              :
(2)              MOV      #,$DN,$STSTNM      ;SAVE THIS
(2)              .ENDM
(2)              .NLIST
(2)              .RADIX 8
(2)              .LIST
(2)              :.IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(2)              :.IFF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(2)              :.IF NB <>
(2)              :
(2)              MOV      #,NEXT      ;GO TO THIS TEST WHEN THRU
(2)              .IFF
(2)              .NLIST
(2)              .RADIX 10
(2)              .LIST
(2)              .IRP      $DE,<\$E>
(2)              .IF DF      TST '$DE
(2)              :
(2)              MOV      #TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(2)              .ENDM
(2)              .NLIST
(2)              .RADIX 8
(2)              .LIST
(2)              .IFF
(2)              :
(2)              MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(2)              .ENDC
(2)              .ENDC
(2)              :.IFF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(2)              .NLIST
(2)              .RADIX 10
(2)              $N=$N+1
(2)              $E=$E+1
(2)              .RADIX 8
(2)              .LIST
(2)              .ENDR
(1)          005456    TSETUP MINT,SYNINT,EIGHT,EVEPAR,26
```



```

(2) 005456          $RESET
(3) 005456 052777 000400 174242   BIS    #MRESET,@TXCSR ;MASTER RESET
(2) 005464 012777 030000 174230   MOV    #SYNINT,@PARCSR ;SET THE MODE
(2) 005472          $RESET
(3) 005472 052777 000400 174226   BIS    #MRESET,@TXCSR ;MASTER RESET
(2)
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 005500 012777 004020 174220   MOV    #MINT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 005506 012777 037426 174206   MOV    #SYNINT!EIGHT!EVEPAR!26,@PARCSR
(1) 005514 016703 174206          MOV    TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 005520 112777 000000 174204   MOV    #0,@TXDBUF ;LOAD DATA CHAR
(1) 005526 012767 000000 173744   MOV    #0,$TMP1 ;TO BE SHIFTED CHAR
(1) 005534 012767 000011 173360   MOV    #9,SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCRONIZATION
(1) 005542          $POKE
(2) 005542 052777 020000 174156   BIS    #CLK,@TXCSR ;POKE CLK UP
(2) 005550 042777 020000 174150   BIC    #CLK,@TXCSR ;POKE CLK DOWN
(1) 005556 005000          1$: CLR    R0
(1) 005560 006067 173714          ROR    $TMP1 ;FORCE CARRY
(1) 005564 103002          BCC    2$ ;BR IF CARRY CLR
(1) 005566 052700 002000          BIS    #BITW,R0 ;EQUIV OF BITW
(1) 005572          2$:
(2) 005572 052777 020000 174126   BIS    #CLK,@TXCSR ;POKE CLK UP
(2) 005600 042777 020000 174120   BIC    #CLK,@TXCSR ;POKE CLK DOWN
(1) 005606 017701 174114          MOV    @TXCSR,R1 ;ACTUAL
(1) 005612 042701 075777          BIC    #075777,R1 ;SAVE BITW & DNA
(1) 005616 020001          CMP    R0,R1 ;COMPARE EXP VS ACT
(1) 005620 001401          BEQ    +4
(1) 005622 104003          ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 005624 005367 173272          DEC    SHIFT ;# OF SHIFTS
(1) 005630 001352          BNE    1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 005632 052777 020000 174066   BIS    #CLK,@TXCSR ;POKE CLK
(1) ;IF IDN <ISYMOD>,<SYNINT>
(1) ;IFT
(1) ;IFF
(1) ;IFFF
(1) 005640 012700 100000          MOV    #100000,R0 ;EXPECTED
(1) ;IF
(1) 005644 017701 174056          MOV    @TXCSR,R1 ;ACTUAL
(1) 005650 042701 077777          BIC    #77777,R1 ;SAVE DNA ONLY
(1) 005654 020001          CMP    R0,R1 ;COMPARE EXP VS ACT
(1) 005656 001401          BEQ    +4
(1) 005660 104003          ERROR 3 ;DNA SHOULD BE SET
(1) ;IF DNA DID NOT SET
(1) ;CHECK WORD LENGTH SELECT LOGIC
8160 005662          $TRPAR EIGHT,9,ODDPAR,SYNINT,0,400
(1) ;:THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1) ;:OF THE TRANSMITTER SECTION.
(1) ;:IT ALSO CHECKS DNA TIMING
(1) ;:MODE:SYNINT
  
```

```
(1)                                     ::LENGTH:EIGHT PLUS PARITY
(1)                                     ::PARITY:ODDPAR
(1)                                     ::CHARACTER:0
(1)                                     ::
(1) 005662 $TSTNO
(2) 005662 NEWTST
(3) 000000 $NWTST=0
(3) 001 .IF B <>
(3) 005662 $$NEWTEST \STN,<>,SCOPE
(4) .IRP ASCII,<>
(4) .IF EQ $NWTST
(4) .NLIST
(4) $NWTST=1
(4) .SBTTL T12 ASCII
(4) .LIST
(4) STARS
(4) :*TEST 12 ASCII
(4) .IFF
(4) ASCII
(4) .ENDC
(4) .ENDM
(4) 005662 STARS
(5) 002 .IF B
(5) ::*****
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) ::*****
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) 001 .ENDC
(4) 005662 000004 TST12: SCOPE
(4) 000013 $TN=$TN+1
(3) .IFF
(3) $$NEWTEST \STN,<>,<>
(3) .ENDC
(3) 000 .IF NE 4000&$$SWR
(3) 001 .IF NB
(3) 002 .IF LE <-1>
(3) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(3) .IFF
(3) MOV #,$TIMES ;;DO ITERATIONS
(3) .ENDC
(3) 002 .ENDC
(3) 001 .IF NB
(3) 002 MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(3) .ENDC
(3) 000 .ENDC
(3) 001 .IF NB
(3) 002 .IF DF $MAIL
(3) .IRP TNM,<\STN-1>
(3) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
```

```
(3)          001          .ENDC
(3)          000          .ENDC
(2)          000000      .REPT 0
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP $DN,<\$N>
(2)          NEWTST
(2)          :          MOV      # $DN,$STNM      ;SAVE THIS
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          :.IIF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(2)          :.IIF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(2)          :.IF NB      <>
(2)          :          MOV      #,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .IFF
(2)          .NLIST
(2)          .RADIX 10
(2)          .LIST
(2)          .IRP $DE,<\$E>
(2)          .IF DF      TST $DE
(2)          :          MOV      #TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .ENDM
(2)          .NLIST
(2)          .RADIX 8
(2)          .LIST
(2)          .IFF
(2)          :          MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(2)          .ENDC
(2)          .ENDC
(2)          :.IIF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(2)          .NLIST
(2)          .RADIX 10
(2)          $N=$N+1
(2)          $E=$E+1
(2)          .RADIX 8
(2)          .LIST
(2)          .ENDR
(1) 005664      TSETUP MINT,SYNINT,EIGHT,ODDPAR,26
(2) 005664      $RESET
(3) 005664      052777 000400 174034      BIS      #MRESET,@TXCSR      ;MASTER RESET
(2) 005672      012777 030000 174022      MOV      #SYNINT,@PARCSR      ;SET THE MODE
(2) 005700      $RESET
(3) 005700      052777 000400 174020      BIS      #MRESET,@TXCSR      ;MASTER RESET
(2)          :SET MAINTENANCE MODE & SEND
(2)          ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 005706      012777 004020 174012      MOV      #MINT!SEND,@TXCSR
(2)          :SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 005714      012777 037026 174000      MOV      #SYNINT!EIGHT!ODDPAR!26,@PARCSR
(1) 005722      016703 174000      MOV      TXCSR,R3      ;SET UP FOR ERROR MSG
(1) 005726      112777 000000 173776      MOV      #0,@TXDBUF      ;LOAD DATA CHAR
(1) 005734      012767 000400 173536      MOV      #400,$TMP1      ;TO BE SHIFTED CHAR
```

```

(1) 005742 012767 000011 173152      MOV    #9,SHIFT      ;# OF SHIFTS
(1)                                     ;POKE CLK TO GET INTO SYNCRONIZATION
(1) 005750                                     $POKE
(2) 005750 052777 020000 173750      BIS    #CLK,@TXCSR   ;POKE CLK UP
(2) 005756 042777 020000 173742      BIC    #CLK,@TXCSR   ;POKE CLK DOWN
(1) 005764 005000                                     1$: CLR    RO
(1) 005766 006067 173506               ROR    $TMP1        ;FORCE CARRY
(1) 005772 103002                       BCC    2$           ;BR IF CARRY CLR
(1) 005774 052700 002000               BIS    #BITW,RO     ;EQUIV OF BITW
(1) 006000                                     2$:
(2) 006000 052777 020000 173720      BIS    #CLK,@TXCSR   ;POKE CLK UP
(2) 006006 042777 020000 173712      BIC    #CLK,@TXCSR   ;POKE CLK DOWN
(1) 006014 017701 173706               MOV    @TXCSR,R1     ;ACTUAL
(1) 006020 042701 075777               BIC    #075777,R1    ;SAVE BITW & DNA
(1) 006024 020001                       CMP    RO,R1        ;COMPARE EXP VS ACT
(1) 006026 001401                       BEQ    +4
(1) 006030 104003                       ERROR  3            ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1)                                     ;BIT,...ALSO CHECK DNA
(1) 006032 005367 173064               DEC    SHIFT        ;# OF SHIFTS
(1) 006036 001352                       BNE    1$           ;DO IT AGAIN ?
(1)                                     ;NOW POKE CLK TO SEE DNA
(1) 006040 052777 020000 173660      BIS    #CLK,@TXCSR   ;POKE CLK
(1)                                     .IF
(1)                                     .IFT
(1)                                     .IFF
(1)                                     .ENDC
(1) 006046 012700 100000               MOV    #100000,RO    ;EXPECTED
(1)                                     .ENDC
(1) 006052 017701 173650               MOV    @TXCSR,R1     ;ACTUAL
(1) 006056 042701 077777               BIC    #77777,R1     ;SAVE DNA ONLY
(1) 006062 020001                       CMP    RO,R1        ;COMPARE EXP VS ACT
(1) 006064 001401                       BEQ    +4
(1) 006066 104003                       ERROR  3            ;DNA SHOULD BE SET
(1)                                     ;IF DNA DID NOT SET
(1)                                     ;CHECK WORD LENGTH SELECT LOGIC
8161 006070      $TRPAR EIGHT,9,EVEPAR,SYNINT,377,377
(1)                                     ;;THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1)                                     ;;OF THE TRANSMITTER SECTION.
(1)                                     ;;IT ALSO CHECKS DNA TIMING
(1)                                     ;;MODE:SYNINT
(1)                                     ;;LENGTH:EIGHT PLUS PARITY
(1)                                     ;;PARITY:EVEPAR
(1)                                     ;;CHARACTER:377
(1)                                     ;;
(1) 006070      $TSTNO
(2) 006070      NEWTST
(3)                                     $NWTST=0
(3) 000000      .IF B <>
(3) 006070      $$NEWTST \ $TN,<>,SCOPE
(4) .IRP ASCII,<>
(4) .IF EQ $NWTST
(4) .NLIST
(4) $NWTST=1
(4) .SBTTL T13 ASCII
(4) .LIST
  
```

```
(4) STARS
(4) :*TEST 13      ASCII
(4) .IFF
(4)          ASCII
(4) .ENDC
(4) .ENDM
(4) STARS
(4) 006070      002
(5) .IF B
(5) :*****
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) :*****
(5) .NLIST
(5) .ENDR
(5) .LIST
(5) 006070      001
(4) 006070      000004
(4)          000014
(3) $TN=$TN+1
(3) .IFF
(3)          $$NEWTEST      \STN,<>,<>
(3) .ENDC
(3)          000
(3)          001
(3)          002
(3)          003
(3) .IF NE 4000$$SWR
(3) .IF NB
(3) .IF LE <-1>
(3)          MOV      #1,$TIMES      ;;DO 1 ITERATION
(3) .IFF
(3)          MOV      #,$TIMES      ;;DO ITERATIONS
(3) .ENDC
(3)          002
(3)          001
(3)          002
(3) .IF NB
(3)          MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(3) .ENDC
(3)          001
(3)          000
(3)          001
(3)          002
(3) .IF DF $MAIL
(3) .IRP      TNM,<\STN-1>
(3)          MOV      #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
(3)          001
(3)          000
(2)          000000
(2) .REPT 0
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP      $DN,<\$N>
(2) NEWTST
(2)          MOV      #,$DN,$STNM      ;SAVE THIS
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) .:IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(2) .:IFF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
```

```

(2) ;.IF NB <>
(2) ; MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(2) .IFF
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DE,<\$E>
(2) .IF DF TST'$DE
(2) ; MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) .IFF
(2) ; MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDC
(2) .ENDC
(2) ;.IIF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(2) .NLIST
(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 006072 TSETUP MINT,SYNINT,EIGHT,EVEPAR,26
(2) 006072 $RESET
(3) 006072 052777 000400 173626 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 006100 012777 030000 173614 MOV #SYNINT,@PARCSR ;SET THE MODE
(2) 006106 $RESET
(3) 006106 052777 000400 173612 BIS #MRESET,@TXCSR ;MASTER RESET
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 006114 012777 004020 173604 MOV #MINT!SEND,@TXCSR
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 006122 012777 037426 173572 MOV #SYNINT!EIGHT!EVEPAR!26,@PARCSR
(1) 006130 016703 173572 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 006134 112777 000377 173570 MOV #377,@TXDBUF ;LOAD DATA CHAR
(1) 006142 012767 000377 173330 MOV #377,$TMP1 ;TO BE SHIFTED CHAR
(1) 006150 012767 000011 172744 MOV #9,$SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCHRONIZATION
(1) 006156 $POKE
(2) 006156 052777 020000 173542 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 006164 042777 020000 173534 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006172 005000 1$: CLR R0
(1) 006174 006067 173300 ROR $TMP1 ;FORCE CARRY
(1) 006200 103002 2$: BCC 2$ ;BR IF CARRY CLR
(1) 006202 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 006206 2$:
(2) 006206 052777 020000 173512 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 006214 042777 020000 173504 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006222 017701 173500 MOV @TXCSR,R1 ;ACTUAL
(1) 006226 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 006232 020001 CMP R0,R1 ;COMPARE EXP VS ACT
  
```

```
(1) 006234 001401 BEQ +4
(1) 006236 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 006240 005367 172656 DEC SHIFT ;# OF SHIFTS
(1) 006244 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 006246 052777 020000 173452 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 IDN <ISYMOD>,<SYNINT>
(1) .IF
(1) .IFT MOV #0,R0 ;EXPECTED
(1) .IFF
(1) 006254 012700 100000 MOV #100000,R0 ;EXPECTED
(1) 000 .ENDC
(1) 006260 017701 173442 MOV @TXCSR,R1 ;ACTUAL
(1) 006264 042701 077777 BIC #77777,R1 ;SAVE DNA ONLY
(1) 006270 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 006272 001401 BEQ +4
(1) 006274 104003 ERROR 3 ;DNA SHOULD BE SET
(1) ;IF DNA DID NOT SET
(1) ;CHECK WORD LENGTH SELECT LOGIC
8162 006276 $TRPAR EIGHT,9,ODDPAR,SYNINT,377,777
(1) ;:THIS TEST VERIFYS CHARACTER PLUS PARITY GENERATION
(1) ;:OF THE TRANSMITTER SECTION.
(1) ;:IT ALSO CHECKS DNA TIMING
(1) ;:MODE:SYNINT
(1) ;:LENGTH:EIGHT PLUS PARITY
(1) ;:PARITY:ODDPAR
(1) ;:CHARACTER:377
(1) ;:
(1) 006276 $STSTNO
(2) 006276 NEWTST
(3) 000000 $NWTST=0
(3) 001 .IF B <>
(3) 006276 $$NEWTST \ $TN,<> ,SCOPE
(4) .IRP ASCII,<>
(4) .IF EQ $NWTST
(4) .NLIST
(4) $NWTST=1
(4) .SBTTL T14 ASCII
(4) .LIST
(4) STARS
(4) ;*TEST 14 ASCII
(4) .IFF
(4) ASCII
(4) .ENDC
(4) .ENDM
(4) 006276 STARS
(5) 002 .IF B
(5) ;:*****
(5) .IFF
(5) .NLIST
(5) .REPT
(5) .LIST
(5) ;:*****
(5) .NLIST
```

```
(5) .ENDR
(5) .LIST
(5) .ENDC
(4) 006276 000004 TST14: SCOPE
(4) 000015 $TN=$TN+1
(3) .IFF
(3) $NEWTEST \ $TN,<>,<>
(3) .ENDC
(3) 000 .IF NE 4000&$SWR
(3) 001 .IF NB
(3) 002 .IF LE <-1>
(3) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(3) .IFF MOV #,$TIMES ;;DO ITERATIONS
(3) .ENDC
(3) 002 .ENDC
(3) 001 .ENDC
(3) 002 .IF NB
(3) MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(3) .ENDC
(3) 001 .ENDC
(3) 000 .ENDC
(3) 001 .IF NB
(3) 002 .IF DF $MAIL
(3) .IRP TNM,<\ $TN-1>
(3) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(3) .ENDM
(3) 001 .ENDC
(3) 000 .ENDC
(2) 000000 .REPT 0
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DN,<\ $N>
(2) NEWTST
(2) : MOV # $DN,$STNM ;SAVE THIS
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) : .IF NB <> MOV #,RETURN ;SET UP THIS RETURN
(2) : .IF NB <> MOV #,ICOUNT ;SET UP THIS ICOUNT
(2) : .IF NB <>
(2) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(2) .IFF
(2) .NLIST
(2) .RADIX 10
(2) .LIST
(2) .IRP $DE,<\ $E>
(2) .IF DF TST $DE
(2) : MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(2) .ENDM
(2) .NLIST
(2) .RADIX 8
(2) .LIST
(2) .IFF
(2) : MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
```



```

(2) .ENDC
(2) .ENDC
(2) :.IIF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(2) .NLIST
(2) .RADIX 10
(2) $N=$N+1
(2) $E=$E+1
(2) .RADIX 8
(2) .LIST
(2) .ENDR
(1) 006300 TSETUP MINT,SYNINT,EIGHT,ODDPAR,26
(2) 006300 $RESET
(3) 006300 052777 000400 173420 BIS #MRESET,@TXCSR ;MASTER RESET
(2) 006306 012777 030000 173406 MOV #SYNINT,@PARCSR ;SET THE MODE
(2) 006314 $RESET
(3) 006314 052777 000400 173404 BIS #MRESET,@TXCSR ;MASTER RESET
(2)
(2) ;SET MAINTENANCE MODE & SEND
(2) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(2) 006322 012777 004020 173376 MOV #MINT!SEND,@TXCSR
(2)
(2) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(2) 006330 012777 037026 173364 MOV #SYNINT!EIGHT!ODDPAR!26,@PARCSR
(1) 006336 016703 173364 MOV TXCSR,R3 ;SET UP FOR ERROR MSG
(1) 006342 112777 000377 173362 MOVB #377,@TXDBUF ;LOAD DATA CHAR
(1) 006350 012767 000777 173122 MOV #777,$TMP1 ;TO BE SHIFTED CHAR
(1) 006356 012767 000011 172536 MOV #9,SHIFT ;# OF SHIFTS
(1) ;POKE CLK TO GET INTO SYNCHRONIZATION
(1) 006364 $POKE
(2) 006364 052777 020000 173334 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 006372 042777 020000 173326 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006400 005000 1$: CLR R0
(1) 006402 006067 173072 ROR $TMP1 ;FORCE CARRY
(1) 006406 103002 BCC 2$ ;BR IF CARRY CLR
(1) 006410 052700 002000 BIS #BITW,R0 ;EQUIV OF BITW
(1) 006414 2$:
(2) 006414 052777 020000 173304 BIS #CLK,@TXCSR ;POKE CLK UP
(2) 006422 042777 020000 173276 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006430 017701 173272 MOV @TXCSR,R1 ;ACTUAL
(1) 006434 042701 075777 BIC #075777,R1 ;SAVE BITW & DNA
(1) 006440 020001 CMP R0,R1 ;COMPARE EXP VS ACT
(1) 006442 001401 BEQ +4
(1) 006444 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH ACTUAL DATA
(1) ;BIT,...ALSO CHECK DNA
(1) 006446 005367 172450 DEC SHIFT ;# OF SHIFTS
(1) 006452 001352 BNE 1$ ;DO IT AGAIN ?
(1) ;NOW POKE CLK TO SEE DNA
(1) 006454 052777 020000 173244 BIS #CLK,@TXCSR ;POKE CLK
(1) 001 IDN <ISYMOD>,<SYNINT>
(1) .IFT
(1) .IFF MOV #0,R0 ;EXPECTED
(1) 006462 012700 100000 .IFF MOV #100000,R0 ;EXPECTED
(1) 000
(1) 006466 017701 173234 .ENDC MOV @TXCSR,R1 ;ACTUAL
(1) 006472 042701 077777 BIC #77777,R1 ;SAVE DNA ONLY
  
```

```
(1) 006476 020001      CMP      R0,R1      ;COMPARE EXP VS ACT
(1) 006500 001401      BEQ      +4
(1) 006502 104003      ERROR    3          ;DNA SHOULD BE SET
(1)                                     ;IF DNA DID NOT SET
(1)                                     ;CHECK WORD LENGTH SELECT LOGIC
(1)
8163
8164
8165      ::THIS TEST VERIFYS THAT BY SENDING ONLY ONE SYNC
8166      ::CHARACTER (TWO SELECTED BY STRAPPING ) RECACT =0
8167      ::THEN SEND ONE ORDINARY CHARACTER (TO BREAK UP THE SEQUENCE)
8168      ::RECACT =0.....IT WILL TAKE TWO MORE SYNC CHARS
8169      ::BEFORE RECACT =1
8170      ::NOTE: THIS TEST WILL ONLY WORK WHEN TWO SYNC CHARS
8171      ::HAS BEEN BEEN SELECTED ...OTHERWISE JUMP AROUND THIS TEST
8172      ::MODE:SYNC INTERNAL (SYNINT)
8173      ::PARITY: NOPAR
8174      ::LENGTH: EIGHT
8175
8176      ::THIS TEST CHECKS ONLY THE RECEIVER SECTION
8177
8178 006504 $TSTNO
(1) 006504 NEWTST
(2) 000000 $NWTST=0
(2) 001 .IF B <>
(2) 006504 $$NEWTST \ $TN,<>,SCOPE
(3) .IRP ASCII,<>
(3) .IF EQ $NWTST
(3) .NLIST
(3) $NWTST=1
(3) .SBTTL T15 ASCII
(3) .LIST
(3) STARS
(3) :*TEST 15 ASCII
(3) .IFF
(3) ASCII
(3) .ENDC
(3) .ENDM
(3) 006504 STARS
(4) 002 .IF B
(4) :*****
(4) .IFF
(4) .NLIST
(4) .REPT
(4) .LIST
(4) :*****
(4) .NLIST
(4) .ENDR
(4) .LIST
(4) 001 .ENDC
(3) 006504 000004 TST15: SCOPE
(3)
(3) $TN=$TN+1
(2) .IFF
(2) $$NEWTST \ $TN,<>,<>
(2) 000 .ENDC
```

```
(2)          001      .IF NE 4000BSSWR
(2)          002      .IF NB
(2)          003      .IF LE <-1>
(2)              MOV      #1,$TIMES      ;;DO 1 ITERATION
(2)          .IFF
(2)              MOV      #,$TIMES      ;;DO ITERATIONS
(2)          002      .ENDC
(2)          001      .ENDC
(2)          002      .IF NB
(2)              MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(2)          001      .ENDC
(2)          000      .ENDC
(2)          001      .IF NB
(2)          002      .IF DF $MAIL
(2)              .IRP      TNM,<\$TN-1>
(2)                  MOV      #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(2)              .ENDM
(2)          .ENDC
(2)          001      .ENDC
(2)          000      .ENDC
(1)          000000  .REPT 0
(1)          .NLIST
(1)          .RADIX 10
(1)          .LIST
(1)          .IRP      $DN,<\$N>
(1)          NEWTST
(1)              MOV      # $DN,$STSTNM      ;SAVE THIS
(1)          .ENDM
(1)          .NLIST
(1)          .RADIX 8
(1)          .LIST
(1)          .: .IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(1)          .: .IFF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(1)          .: .IF NB <>
(1)          .:              MOV      #,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .: .IFF
(1)          .: .NLIST
(1)          .: .RADIX 10
(1)          .: .LIST
(1)          .: .IRP      $DE,<\$E>
(1)          .: .IF DF      TST '$DE
(1)          .:              MOV      #TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .: .ENDM
(1)          .: .NLIST
(1)          .: .RADIX 8
(1)          .: .LIST
(1)          .: .IFF
(1)          .:              MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .: .ENDC
(1)          .: .ENDC
(1)          .: .IFF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(1)          .: .NLIST
(1)          .: .RADIX 10
(1)          .: $N=$N+1
(1)          .: $E=$E+1
(1)          .: .RADIX 8
(1)          .: .LIST
```

```

(1)
8179 006506 105767 172434 .ENDR
8180 006512 100127 TSTB SYNCNO ;TEST FOR # OF SYNC CHARS REQUIRED
8181 006514 RSETUP BPL 2$ ;IF NOT TWO GET OUT OF TEST
(1) 006514 $RESET MINT,SYNINT,EIGHT,NOPAR,26
(2) 006514 052777 000400 173204 BIS #MRESET,@TXCSR ;MASTER RESET
(1) 006522 012777 030000 173172 MOV #SYNINT,@PARCSR ;SET THE MODE
(1) 006530 $RESET BIS #MRESET,@TXCSR ;MASTER RESET
(2) 006530 052777 000400 173170
(1)
(1) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(1) 006536 012777 064001 173162 MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
(1)
(1) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(1) 006544 012777 036026 173150 MOV #SYNINT!EIGHT!NOPAR!26,@PARCSR
8182 006552 $SYNCR
(1) 006552 052777 000020 173132 BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
(1) ;POKE CLK TO GET RECEIVER INTO SYNCRIZATION....
(1) 006560 042777 020000 173140 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006566 052777 020000 173132 BIS #CLK,@TXCSR ;POKE CLK UP
8183 006574 $POKER
(1) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 006574 042777 020000 173124 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006602 052777 020000 173116 BIS #CLK,@TXCSR ;POKE CLK UP
8184 006610 012767 000010 172304 MOV #8,,SHIFT ;# OF SHIFTS
8185 006616 012767 000026 172654 MOV #26,$TMP1 ;SYNC CHAR TO BE SHIFTED IN
8186 006624 004767 010342 JSR PC,RPOKE ;SHIFT IN THIS SYNC CHAR
8187 006630 032777 004000 173054 BIT #REACT,@RXCSR ;REACT = 0 ?
8188 006636 001401 BEQ .+4
8189 006640 104004 ERROR 4 ;REACT SHOULD BE 0
8190 006642 012767 000010 172252 MOV #8,,SHIFT ;# OF SHIFTS
8191 006650 012767 000025 172622 MOV #25,$TMP1 ;ANY CHARACTER
8192 006656 004767 010310 JSR PC,RPOKE ;SHIFT IN THIS CHARACTER
8193 ;YOU HAVE JUST LOST SYNCRONIZATION.....
8194 ;POKE THE CLK TWICE TO GET INTO SYNCRONIZATION
8195 006662 $POKER
(1) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 006662 042777 020000 173036 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006670 052777 020000 173030 BIS #CLK,@TXCSR ;POKE CLK UP
8196 006676 $POKER
(1) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 006676 042777 020000 173022 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 006704 052777 020000 173014 BIS #CLK,@TXCSR ;POKE CLK UP
8197 006712 012767 000002 172204 MOV #2,COUNT ;# OF SYNC CHARS
8198 006720 032777 004000 172764 1$: BIT #REACT,@RXCSR ;REACT = 0 ?
8199 006726 001401 BEQ .+4
8200 006730 104004 ERROR 4 ;REACT SHOULD BE 0
8201 006732 012767 000010 172162 MOV #8,,SHIFT ;# OF SHIFTS
8202 006740 012767 000026 172532 MOV #26,$TMP1 ;SYNC CHAR
8203 006746 004767 010220 JSR PC,RPOKE ;SHIFT IN THIS SYNC CHAR
8204 006752 005367 172146 DEC COUNT
8205 006756 001360 BNE 1$ ;IS COUNT = 0 ? NO GO AGAIN
8206 006760 032777 004000 172724 BIT #REACT,@RXCSR ;REACT = 1 ?
8207 006766 001001 BNE .+4
8208 006770 104004 ERROR 4 ;REACT SHOULD BE ASSERTED
8209 006772 2$:

```

```
8210      ;;THIS TEST VERIFYS MODE SELECT.....
8211      ;;SYNC EXTERNAL VS. SYNC INTERNAL
8212      ;;
8213      ;;BASICALLY THE TEST CHECKS THAT THE RECEIVED
8214      ;;DATA FREEZES IN SYNC INTERNAL
8215      ;;IN SYNC EXTERNAL THIS DATA IS TRANSPARENT
8216      ;;THIS TEST ONLY APPLIES TO THE RECEIVER SECTION
8217      ;;LENGTH: EIGHT
8218      ;;NOTE:SEARCH SYNC IS NOT SET
8219      $STNO
      (1) 006772 NEWTST
      (2) 006772 $NWTST=0
      (2)           000000 .IF B <>
      (2)           001    $$NEWTST      \STN,<>,SCOPE
      (3)           006772 .IRP      ASCII,<>
      (3)           .IF EQ $NWTST
      (3)           .NLIST
      (3)           $NWTST=1
      (3)           .SBTTL  T16      ASCII
      (3)           .LIST
      (3)           STARS
      (3)           :*TEST 16      ASCII
      (3)           .IFF
      (3)           ASCII
      (3)           .ENDC
      (3)           .ENDM
      (3) 006772 STARS
      (4)           002 .IF B
      (4)           :*****
      (4)           .IFF
      (4)           .NLIST
      (4)           .REPT
      (4)           .LIST
      (4)           :*****
      (4)           .NLIST
      (4)           .ENDR
      (4)           .LIST
      (4) 006772 001 .ENDC
      (3) 006772 000004 TST16: SCOPE
      (3)           000017 $TN=$TN+1
      (2)           .IFF
      (2)           $$NEWTST      \STN,<>,<>
      (2)           .ENDC
      (2)           000 .IF NE 4000&$$SWR
      (2)           001 .IF NB
      (2)           002 .IF LE <-1>
      (2)           003      MOV      #1,$TIMES      ;;DO 1 ITERATION
      (2)           .IFF
      (2)           MOV      #,$TIMES      ;;DO ITERATIONS
      (2)           .ENDC
      (2)           002 .ENDC
      (2)           001 .ENDC
      (2)           002 .IF NB
      (2)           MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
      (2)           .ENDC
      (2)           001 .ENDC
      (2)           000 .ENDC
```

```
(2)          001      .IF NB
(2)          002      .IF DF $MAIL
(2)           .IRP    TNM,<\$TN-1>
(2)           .MOV    #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(2)           .ENDM
(2)          001      .ENDC
(2)          000      .ENDC
(1)          000000   .REPT 0
(1)           .NLIST
(1)           .RADIX 10
(1)           .LIST
(1)           .IRP    $DN,<\$N>
(1)           NEWTST
(1)           :      .MOV    #'$DN,$TSTNM      ;SAVE THIS
(1)           .ENDM
(1)           .NLIST
(1)           .RADIX 8
(1)           .LIST
(1)           :      .IF NB      <>,      .MOV    #,RETURN      ;SET UP THIS RETURN
(1)           :      .IF NB      <>,      .MOV    #,ICOUNT      ;SET UP THIS ICOUNT
(1)           :      .IF NB <>
(1)           :      .MOV    #,NEXT      ;GO TO THIS TEST WHEN THRU
(1)           .IFF
(1)           .NLIST
(1)           .RADIX 10
(1)           .LIST
(1)           .IRP    $DE,<\$E>
(1)           .IF DF  TST '$DE
(1)           :      .MOV    #'TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(1)           .ENDM
(1)           .NLIST
(1)           .RADIX 8
(1)           .LIST
(1)           .IFF
(1)           :      .MOV    #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(1)           .ENDC
(1)           .ENDC
(1)           :      .IF NB      <>,      .MOV    #,LOCK      ;SET UP FOR SCOPE LOOP
(1)           .NLIST
(1)           .RADIX 10
(1)           $N=$N+1
(1)           $E=$E+1
(1)           .RADIX 8
(1)           .LIST
(1)           .ENDR
8220 006774  RSETUP MINT,SYNINT,EIGHT,NOPAR,26
(1) 006774  $RESET
(2) 006774  052777 000400 172724  .BIS    #MRESET,@TXCSR ;MASTER RESET
(1) 007002  012777 030000 172712  .MOV    #SYNINT,@PARCSR ;SET THE MODE
(1) 007010  $RESET
(2) 007010  052777 000400 172710  .BIS    #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(1) 007016  012777 064001 172702  .MOV    #MTDATA!CLK!MINT!BREAK,@TXCSR
(1)
(1) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
```

INITIALIZE THE COMMON TAGS

```

(1) 007024 012777 036026 172670      MOV      #SYNINT!EIGHT!NOPAR!26,@PARCSR
8221 007032                               $POKER
(1)                               ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 007032 042777 020000 172666      BIC      #CLK,@TXCSR      ;POKE CLK DOWN
(1) 007040 052777 020000 172660      BIS      #CLK,@TXCSR      ;POKE CLK UP
8222 007046                               $POKER
(1)                               ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 007046 042777 020000 172652      BIC      #CLK,@TXCSR      ;POKE CLK DOWN
(1) 007054 052777 020000 172644      BIS      #CLK,@TXCSR      ;POKE CLK UP
8223 007062 012767 000010 172032      MOV      #8,SHIFT        ;# OF SHIFTS
8224 007070 012767 000125 172402      MOV      #125,$TMP1      ;DATA CHARACTER
8225 007076 016703 172614             MOV      RXDBUF,R3        ;FOR ERROR MESSAGE
8226 007102 042777 040000 172616 1$:  BIC      #MTDATA,@TXCSR    ;CLEAR MAINT DATA
8227 007110 000241                   CLC
8228 007112 006067 172362             ROR      $TMP1 ;FORCE CARRY
8229 007116 103003                   BCC      2$
8230 007120 052777 040000 172600      BIS      #MTDATA,@TXCSR    ;SET MTDATA
8231 007126 042777 020000 172572 2$:  BIC      #CLK,@TXCSR      ;POKE CLK
8232 007134 052777 020000 172564      BIS      #CLK,@TXCSR      ;
8233 007142 012700 000377             MOV      #377,R0 ;EXPECTED
8234 007146 017701 172544             MOV      @RXDBUF,R1      ;ACTUAL
8235 007152 020001                   CMP      R0,R1
8236 007154 001401                   BEQ      +4
8237 007156 104002                   ERROR    2
8238                               ;DATA CHARACTER SHOULD COMPARE.....
8239 007160 005367 171736             DEC      SHIFT ;IS IT THE LAST SHIFT?
8240 007164 001346                   BNE      1$ ;NO ...SHIFT SOME MORE
8241 007166                               RSETUP  MINT,SYNEXT,EIGHT,NOPAR,26
(1) 007166                               $RESET
(2) 007166 052777 000400 172532      BIS      #MRESET,@TXCSR    ;MASTER RESET
(1) 007174 012777 020000 172520      MOV      #SYNEXT,@PARCSR   ;SET THE MODE
(1) 007202                               $RESET
(2) 007202 052777 000400 172516      BIS      #MRESET,@TXCSR    ;MASTER RESET
(1)
(1)                               ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(1) 007210 012777 064001 172510      MOV      #MTDATA!CLK!MINT!BREAK,@TXCSR
(1)
(1)                               ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(1) 007216 012777 026026 172476      MOV      #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8242 007224                               $POKER
(1)                               ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 007224 042777 020000 172474      BIC      #CLK,@TXCSR      ;POKE CLK DOWN
(1) 007232 052777 020000 172466      BIS      #CLK,@TXCSR      ;POKE CLK UP
8243 007240 012767 000010 171654      MOV      #8,SHIFT        ;# OF SHIFTS
8244 007246 012767 000125 172224      MOV      #125,$TMP1      ;DATA CHARACTER
8245 007254 005067 172222             CLR      $TMP2
8246 007260 105167 172216             COMB    $TMP2 ;MAKE LOW BYTE ALL 1'S
8247                               ;TO MATCH RXDBUF'S CONTENTS AFTER A MASTER RESET
8248 007264 042777 040000 172434 5$:  BIC      #MTDATA,@TXCSR    ;CLR MAINT DATA
8249 007272 000241                   CLC
8250 007274 006067 172200             ROR      $TMP1 ;FORCE CARRY
8251 007300 103003                   BCC      6$
8252 007302 052777 040000 172416      BIS      #MTDATA,@TXCSR
8253 007310 106067 172166             RORB    $TMP2 ;PICK UP CARRY BIT
8254 007314 042777 020000 172404      BIC      #CLK,@TXCSR
8255 007322 052777 020000 172376      BIS      #CLK,@TXCSR

```

| | | | | | | | |
|------|--------|--------|--------|-------------|------------|----------------|--|
| 8256 | 007330 | 016700 | 172146 | MOV | \$TMP2,R0 | | :EXPECTED |
| 8257 | 007334 | 017701 | 172356 | MOV | @RXDBUF,R1 | | :ACTUAL |
| 8258 | 007340 | 020001 | | CMP | R0,R1 | | |
| 8259 | 007342 | 001401 | | BEQ | ,+4 | | |
| 8260 | 007344 | 104002 | | ERROR | 2 | | :DATA CHARACTER SHOULD COMPARE... |
| 8261 | | | | | | | :THE DATA CHARACTER SHOULD BE SEEN AS IT |
| 8262 | | | | | | | :SHIFTS ACROSS THE RECEIVER DATA OUTPUT |
| 8263 | 007346 | 005367 | 171550 | DEC | SHIFT | | |
| 8264 | 007352 | 001344 | | BNE | \$\$ | | |
| 8265 | | | | | | | |
| 8266 | | | | | | | ::THIS TEST VERIFYS TX DONE FUNCTION, DONE = 1 |
| 8267 | | | | | | | ::MODE: SYNC INTERNAL |
| 8268 | | | | | | | ::PARITY: NO PARITY (NOPAR) |
| 8269 | | | | | | | ::LENGTH: EIGHT |
| 8270 | | | | | | | :: |
| 8271 | 007354 | | | \$STNO | | | |
| (1) | 007354 | | | NEWST | | | |
| (2) | | 000000 | | \$NWTST=0 | | | |
| (2) | | 001 | | .IF B <> | | | |
| (2) | 007354 | | | | \$NEWTEST | \\$TN,<>,SCOPE | |
| (3) | | | | .IRP | ASCII,<> | | |
| (3) | | | | .IF EQ | \$NWTST | | |
| (3) | | | | .NLIST | | | |
| (3) | | | | \$NWTST=1 | | | |
| (3) | | | | .SBTTL | T17 | ASCII | |
| (3) | | | | .LIST | | | |
| (3) | | | | STARS | | | |
| (3) | | | | ;*TEST | 17 | ASCII | |
| (3) | | | | .IFF | | | |
| (3) | | | | | ASCII | | |
| (3) | | | | .ENDC | | | |
| (3) | | | | .ENDM | | | |
| (3) | 007354 | | | STARS | | | |
| (4) | | 002 | | .IF B | | | |
| (4) | | | | ::***** | | | |
| (4) | | | | .IFF | | | |
| (4) | | | | .NLIST | | | |
| (4) | | | | .REPT | | | |
| (4) | | | | .LIST | | | |
| (4) | | | | ::***** | | | |
| (4) | | | | .NLIST | | | |
| (4) | | | | .ENDR | | | |
| (4) | | | | .LIST | | | |
| (4) | | 001 | | .ENDC | | | |
| (3) | 007354 | 000004 | | TST17: | SCOPE | | |
| (3) | | | | | | | |
| (3) | | 000020 | | \$TN=\$TN+1 | | | |
| (2) | | | | .IFF | | | |
| (2) | | | | | \$NEWTEST | \\$TN,<>,<> | |
| (2) | | 000 | | .ENDC | | | |
| (2) | | 001 | | .IF NE | 4000BSSWR | | |
| (2) | | 002 | | .IF NB | | | |
| (2) | | 003 | | .IF LE | <-1> | | |
| (2) | | | | MOV | #1,\$TIMES | | ::DC 1 ITERATION |
| (2) | | | | .IFF | | | |
| (2) | | | | MOV | #, \$TIMES | | ::DO ITERATIONS |


```
(2)          002          .ENDC
(2)          001          .ENDC
(2)          002          .IF NB
(2)          .           MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(2)          001          .ENDC
(2)          000          .ENDC
(2)          001          .IF NB
(2)          002          .IF DF $MAIL
(2)          .           .IRP    TNM,<\$TN-1>
(2)          .           MOV    #'TNM,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
(2)          .           .ENDM
(2)          001          .ENDC
(2)          000          .ENDC
(1)          000000      .REPT 0
(1)          .           .NLIST
(1)          .           .RADIX 10
(1)          .           .LIST
(1)          .           .IRP    $DN,<\$N>
(1)          NEWTST
(1)          .           MOV    #,$DN,$TSTNM    ;SAVE THIS
(1)          .           .ENDM
(1)          .           .NLIST
(1)          .           .RADIX 8
(1)          .           .LIST
(1)          .           .IIF NB <>, MOV    #,RETURN    ;SET UP THIS RETURN
(1)          .           .IIF NB <>, MOV    #,ICOUNT    ;SET UP THIS ICOUNT
(1)          .           .IF NB <>
(1)          .           MOV    #,NEXT    ;GO TO THIS TEST WHEN THRU
(1)          .           .IFF
(1)          .           .NLIST
(1)          .           .RADIX 10
(1)          .           .LIST
(1)          .           .IRP    $DE,<\$E>
(1)          .           .IF DF TST '$DE
(1)          .           MOV    #TST'$DE,NEXT    ;GO TO THIS TEST WHEN THRU
(1)          .           .ENDM
(1)          .           .NLIST
(1)          .           .RADIX 8
(1)          .           .LIST
(1)          .           .IFF
(1)          .           MOV    #.EOP,NEXT    ;GO TO THIS TEST WHEN THRU
(1)          .           .ENDC
(1)          .           .ENDC
(1)          .           .IIF NB <>, MOV    #,LOCK    ;SET UP FOR SCOPE LOOP
(1)          .           .NLIST
(1)          .           .RADIX 10
(1)          .           $N=$N+1
(1)          .           $E=$E+1
(1)          .           .RADIX 8
(1)          .           .LIST
(1)          .           .ENDR
8272
8273 007356      TSETUP MINT,SYNINT,EIGHT,NOPAR,26
(1) 007356      $RESET
(2) 007356 052777 000400 172342  BIS    #MRESET,@TXCSR ;MASTER RESET
(1) 007364 012777 030000 ;72330  MOV    #SYNINT,@PARCSR ;SET THE MODE
```

```

(1) 007372 $RESET
(2) 007372 052777 000400 172326 BIS #MRESET,@TXCSR ;MASTER RESET
(1)
(1)
(1) ;SET MAINTENANCE MODE & SEND
(1) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 007400 012777 004020 172320 MOV #MINT!SEND,@TXCSR
(1)
(1) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 007406 012777 036026 172306 MOV #SYNINT!EIGHT!NOPAR!26,@PARCSR
8274
8275 007414 105777 172306 TSTB @TXCSR ;TXDONE?
8276 007420 100401 RMI ;+4
8277 007422 104004 ERROR 4 ;TXDONE SHOULD BE SET
8278 007424 112777 000021 172300 MOVB #21,@TXDBUF ;LOAD ANY CHAR
8279 007432 $POKE ;POKE CLK TO GET INTO SYNCIRONIZATION
(1) 007432 052777 020000 172266 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 007440 042777 020000 172260 BIC #CLK,@TXCSR ;POKE CLK DOWN
8280 007446 105777 172254 TSTB @TXCSR
8281 007452 100001 BPL ;+4
8282 007454 104004 ERROR 4 ;TXDONE SHOULD BE CLR
8283
8284 $POKE
(1) 007456 052777 020000 172242 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 007464 042777 020000 172234 BIC #CLK,@TXCSR ;POKE CLK DOWN
8285 007472 105777 172230 TSTB @TXCSR
8286 007476 100401 BMI ;+4
8287 007500 104004 ERROR 4 ;TXDONE SHOULD BE SET
8288
8289
8290
8291 ;;THIS TEST PROVES THAT RXERR FREEZES THE "RECEIVER
8292 ;;RESET" WHILE IN STRIP SYNC MODE
8293 ;;THIS TEST FIRST PROVES THAT AUTOMATIC RESETS OCCUR
8294 ;;WHEN STRIP SYNC IS SET AND SYNC CHARACTERS ARE SENT
8295 ;;BUT IF AN ERROR SHOULD OCCUR...THIS AUTOMATIC RESET
8296 ;;IS DISCOMBOBULATED
8297 ;;IE: FORCE OVERRUN (OVERRUN) WHILE STRIP SYNC IS SET
8298 ;;BY TRANSMITTING A DATA CHARACTER THEN TRANSMIT A SYNC CHARACTER
8299 ;;AND DON'T READ THAT DATA CHARACTER. NOTE: NORMALLY THE LOGIC
8300 ;;RESETS THE RXDONE & ERROR FLAGS PROVIDING THAT ONLY SYNC CHARACTERS ARE
8301 ;;STRIPPED
8302 ;;MODE: SYNC EXTERNAL (SYNEXT)
8303 ;;LENGTH: EIGHT
8304 ;;NOTE: THIS TEST USES BOTH RECEIVER AND TRANSMITTER LOGIC
8305 ;;
8306
8307 007502 $STNO
(1) 007502 NEWTST
(2) 000000 $NWTST=0
(2) 001 .IF B <>
(2) 007502 $$NEWTST \STN,<>,SCOPE
(3) .IRP ASCII,<>
(3) .IF EQ $NWTST
(3) .NLST
(3) $NWTST=1
(3) .SBTTL 120 ASCII
  
```

```
(3) .LIST
(3) STARS
(3) :*TEST 20      ASCI
(3) .IFF
(3)           ASCI
(3) .ENDC
(3) .ENDM
(3) 007502      002
(4) STARS
(4) .IF B
(4) :*****
(4) .IFF
(4) .NLIST
(4) .REPT
(4) .LIST
(4) :*****
(4) .NLIST
(4) .ENDR
(4) .LIST
(4) .ENDC
(3) 007502      001
(3) 000004      000004
(3) 000021
(2) $TN=$TN+1
(2) .IFF
(2)           $$NEWTST      \ $TN,<>,<>
(2) .ENDC
(2) 000           .IF NE 4000$$SWR
(2) 001           .IF NB
(2) 002           .IF LE <-1>
(2) 003           MOV      #1,$TIMES      ;;DO 1 ITERATION
(2) .IFF           MOV      #,$TIMES      ;;DO ITERATIONS
(2) .ENDC
(2) 002           .ENDC
(2) 001           .ENDC
(2) 002           .IF NB
(2) .ENDC           MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(2) 001           .ENDC
(2) 000           .ENDC
(2) 001           .IF NB
(2) 002           .IF DF $MAIL
(2) .IRP           TNM,<\$TN-1>
(2) .IRP           MOV      #'TNM,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
(2) .ENDM
(2) 001           .ENDC
(2) 000           .ENDC
(1) 000000      .REPT 0
(1) .NLIST
(1) .RADIX 10
(1) .LIST
(1) .IRP $DN,<\$N>
(1) NEWTST
(1) :
(1) :
(1) .ENDM
(1) .NLIST
(1) .RADIX 8
(1) .LIST
(1) .IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
```

```

(1)      :.IIF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(1)      :.IF NB <>
(1)      :      MOV      #,NEXT      ;GO TO THIS TEST WHEN THRU
(1)      :.IFF
(1)      :.NLIST
(1)      :.RADIX      10
(1)      :.LIST
(1)      :.IRP      $DE,<\$E>
(1)      :.IF DF      TST '$DE'
(1)      :      MOV      #TST'$DE',NEXT      ;GO TO THIS TEST WHEN THRU
(1)      :.ENDM
(1)      :.NLIST
(1)      :.RADIX      8
(1)      :.LIST
(1)      :.IFF
(1)      :      MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(1)      :.ENDC
(1)      :.ENDC
(1)      :.IIF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(1)      :.NLIST
(1)      :.RADIX      10
(1)      :$N=$N+1
(1)      :$E=$E+1
(1)      :.RADIX      8
(1)      :.LIST
(1)      :.ENDR
8308
8309 007504      TSETUP      MINT,SYNEXT,EIGHT,NOPAR,26
(1) 007504      $RESET
(2) 007504 052777 000400 172214      BIS      #MRESET,@TXCSR      ;MASTER RESET
(1) 007512 012777 020000 172202      MOV      #SYNEXT,@PARCSR      ;SET THE MODE
(1) 007520      $RESET
(2) 007520 052777 000400 172200      BIS      #MRESET,@TXCSR      ;MASTER RESET
(1)
(1)      ;SET MAINTENANCE MODE & SEND
(1)      ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 007526 012777 004020 172172      MOV      #MINT!SEND,@TXCSR
(1)
(1)      ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 007534 012777 026026 172160      MOV      #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8310 007542 112777 000026 172162      MOV      #26,@TXDBUF      ;LOAD SYNC CHAR
8311 007550 052777 000420 172134      BIS      #SYNSCH!STPSYN,@RXCSR      ;SET SYNC SEARCH & STRIP SYNC
8312 007556 016703 172134      MOV      RXDBUF,R3      ;SET UP FOR ERROR MESSAGE
8313 007562 012767 000003 171334      MOV      #3,COUNT      ;# OF TIMES SYNC WILL BE SENT
8314 007570      $POKE
(1) 007570 052777 020000 172130      BIS      #CLK,@TXCSR      ;POKE CLK UP
(1) 007576 042777 020000 172122      BIC      #CLK,@TXCSR      ;POKE CLK DOWN
8315 007604 012767 000010 171310      2$:      MOV      #8.,SHIFT      ;# OF SHIFTS
8316 007612      1$:
(1) 007612 052777 020000 172106      BIS      #CLK,@TXCSR      ;POKE CLK UP
(1) 007620 042777 020000 172100      BIC      #CLK,@TXCSR      ;POKE CLK DOWN
8317 007626 005367 171270      DEC      SHIFT
8318 007632 001367      BNE      1$
8319 007634 105777 172052      TSTB     @RXCSR      ;RXDONE?
8320 007640 100001      BPL
8321 007642 104004      ERROR      4      ;RXDONE SHOULD NOT ASSERT
  
```

```

8322 007644 005367 171254          DEC    COUNT
8323 007650 001355          BNE    2$
8324 007652 012700 000026        MOV    #26,R0          ;EXPECTED
8325 007656 017701 172034        MOV    @RXDBUF,R1     ;ACTUAL
8326 007662 020001          CMP    R0,R1
8327 007664 001401          BEQ    .+4
8328 007666 104002          ERROR  2
8329                                     ;NOTE THAT OVERRUN SHOULD NOT OCCUR, ALSO
8330                                     ;SECOND & 3RD SYNC CHARACTER CAME FROM
8331 007670 012767 000003 171226    MOV    #3,COUNT       ;# OF TIMES
8332 007676 112777 000025 172026    MOVB   #25,@TXDBUF    ;LOAD ANY CHAR....HOWEVER...
8333                                     ;ONE MORE SYNC CHAR WILL BE SENT BEFORE
8334                                     ;THE "25" CHAR IS SENT (THE DINA BIT IS
8335                                     ;ALREADY UP)
8336
8337 007704 012767 000010 171210 4$:  MOV    #8.,SHIFT      ;# OF SHIFTS
8338
8339 007712 052777 020000 172006 3$:  BIS    #CLK,@TXCSR    ;POKE CLK UP
      (1) 007712 042777 020000 172000  BIC    #CLK,@TXCSR    ;POKE CLK DOWN
      (1) 007720 005367 171170          DEC    SHIFT
8340 007726 005367 171170          BNE    3$
8341 007732 001367          DEC    COUNT
8342 007734 005367 171164          BNE    4$
8343 007740 001361          TSTB   @RXCSR ;RXDONE = 1 ?
8344 007742 105777 171744          BMI    .+4
8345 007746 100401          ERROR  4 ;RXDONE SHOULD BE SET
8346 007750 104004          MOV    #RXERR!OVRUN!26,R0 ;EXPECTED
8347 007752 012700 140026        MOV    @RXDBUF,R1     ;ACTUAL
8348 007756 017701 171734        CMP    R0,R1
8349 007762 020001          BEQ    .+4
8350 007764 001401          ERROR  2
8351 007766 104002                                     ;NOTE THAT OVRUN SHOULD OCCUR,
8352                                     ;ALSO SECOND SYNC CHARACTER CAME
8353                                     ;FROM SYNC HOLDING REGISTER
8354                                     ;SUMMARY: THE OVRUN STOPPED
8355                                     ;THE AUTOMATIC RESETTING OF
8356                                     ;RXDONE & ERROR FLAGS.....CHECK THIS
8357
8358                                     ;; THIS TEST VERIFYS THAT EITHER SEQUENCE OF
8359                                     ;; LOADING TXDBUF AND SETTING SEND
8360                                     ;; DOES CAUSE TRANSMISSION
8361                                     ;; MODE: SYNC EXT
8362                                     ;; LENGTH: EIGHT
8363                                     ;;
8364 007770          $TSTNO
      (1) 007770          NEWTST
      (2)          $NWTST=0
      (2)          .IF B <>
      (2) 007770          $$NEWTST          \ $TN,<> ,SCOPE
      (3)          .IRP   ASCII,<>
      (3)          .IF EQ $NWTST
      (3)          .NLIST
      (3)          $NWTST=1
      (3)          .SBTTL 121          ASCII
      (3)          .LIST
      (3)          STARS
  
```

INITIALIZE THE COMMON TAGS

```
(3)      ;*TEST 21      ASCII
(3)      .IFF
(3)      ASCII
(3)      .ENDC
(3)      .ENDM
(3) 007770 STARS
(4)      .IF B
(4)      ;*****
(4)      .IFF
(4)      .NLIST
(4)      .REPT
(4)      .LIST
(4)      ;*****
(4)      .NLIST
(4)      .ENDR
(4)      .LIST
(4)      .ENDC
(3) 007770 001
(3) 000004 TST21: SCOPE
(3)      000022
(2)      $TN=$TN+1
(2)      .IFF
(2)      $$NEWTEST      \ $TN,<>,<>
(2)      .ENDC
(2)      000
(2)      001      .IF NE 4000&$$SWR
(2)      002      .IF NB
(2)      003      .IF LE <-1>
(2)      .IFF      MOV      #1,$TIMES      ;;DO 1 ITERATION
(2)      .IFF      MOV      #,$TIMES      ;;DO ITERATIONS
(2)      .ENDC
(2)      002      .ENDC
(2)      001      .ENDC
(2)      002      .IF NB
(2)      .ENDC      MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(2)      001      .ENDC
(2)      000      .ENDC
(2)      001      .IF NB
(2)      002      .IF DF $MAIL
(2)      .IRP      TNM,<\$TN-1>
(2)      .IRP      MOV      #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(2)      .ENDM
(2)      001      .ENDC
(2)      000      .ENDC
(1) 000000 .REPT 0
(1)      .NLIST
(1)      .RADIX 10
(1)      .LIST
(1)      .IRP      $DN,<\$N>
(1)      NEWTST
(1)      :      MOV      # $DN,$TSTNM      ;SAVE THIS
(1)      .ENDM
(1)      .NLIST
(1)      .RADIX 8
(1)      .LIST
(1)      ;.IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(1)      ;.IFF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(1)      ;.IF NB <>
```

```

(1)      ;      MOV      #,NEXT ;GO TO THIS TEST WHEN THRU
(1)      .IFF
(1)      .NLIST
(1)      .RADIX 10
(1)      .LIST
(1)      .IRP      $DE,<\$E>
(1)      .IF DF   TST'$DE
(1)      ;      MOV      #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(1)      .ENDM
(1)      .NLIST
(1)      .RADIX 8
(1)      .LIST
(1)      .IFF
(1)      ;      MOV      #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(1)      .ENDC
(1)      .ENDC
(1)      ;.IFF NB   <>,      MOV      #,LOCK ;SET UP FOR SCOPE LOOP
(1)      .NLIST
(1)      .RADIX 10
(1)      $N=$N+1
(1)      $E=$E+1
(1)      .RADIX 8
(1)      .LIST
(1)      .ENDR
8365
8366 007772 TSETUP MINT,SYNEXT,EIGHT,NOPAR,26
(1) 007772 $RESET
(2) 007772 052777 000400 171726 BIS #MRESET,@TXCSR ;MASTER RESET
(1) 010000 012777 020000 171714 MOV #SYNEXT,@PARCSR ;SET THE MODE
(1) 010006 $RESET
(2) 010006 052777 000400 171712 BIS #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINTENANCE MODE & SEND
(1) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 010014 012777 004020 171704 MOV #MINT!SEND,@TXCSR
(1)
(1) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 010022 012777 026026 171672 :MOV #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8367 010030 016703 171672 MOV TXCSR,R3 ;SET UP FOR ERROR MESSAGE
8368 010034 042777 000020 171664 BIC #SEND,@TXCSR ;DROP SEND
8369 010042 012767 000025 171430 MOV #25,$TMP1
8370 010050 112777 000025 171654 MOVB #25,@TXDBUF ;LOAD CHARACTER
8371 010056 052777 000020 171642 BIS #SEND,@TXCSR ;SET SEND
8372 ;GET INTO SYNCRONIZATION
8373 010064 $POKE
(1) 010064 052777 020000 171634 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 010072 042777 020000 171626 BIC #CLK,@TXCSR ;POKE CLK DOWN
8374 010100 012767 000010 171014 MOV #8.,SHIFT ;# OF SHIFTS
8375 010106 005000 1$: CLR R0
8376 010110 006067 171364 ROR $TMP1
8377 010114 103002 BCC 2$
8378 010116 052700 002000 BIS #BITW,R0 ;EQUIV OF BIT WINDOW
8379
8380 010122 2$:
(1) 010122 052777 020000 171576 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 010130 042777 020000 171570 BIC #CLK,@TXCSR ;POKE CLK DOWN

```

```
8381 010136 017701 171564      MOV    @TXCSR,R1      ;ACTUAL
8382 010142 042701 075777      BIC    #075777,R1    ;SAVE BIT WINDOW & DNA
8383 010146 020001                CMP    R0,R1
8384 010150 001401                BEQ    +4
8385 010152 104003                ERROR  3              ;BIT WINDOW DID NOT MATCH ACTUAL DATA BIT
8386                                ;ALSO CHECK DNA
8387 010154 005367 170742      DEC    SHIFT
8388 010160 001352                BNE    1$
8389
8390
8391
8392                                ;THIS TEST VERIFYS THAT DROPPING OF SEND IN THE
8393                                ;MIDDLE OF TRANSMITTING A CHARACTER DOES INDEED
8394                                ;FINISH TRANSMITTING THAT CHARACTER
8395                                ;MODE: SYNC EXT
8396                                ;LENGTH: EIGHT
8397                                ;
8398                                ;
8399 010162                                $STNO
(1) 010162                                NEWTST
(2)                                $NWTST=0
(2)                                .IF B <>
(2) 010162                                .SSNEWTEST          \STN,<>,SCOPE
(3)                                .IRP    ASCII,<>
(3)                                .IF EQ $NWTST
(3)                                .NLIST
(3)                                $NWTST=1
(3)                                .SBTTL  T22      ASCII
(3)                                .LIST
(3)                                STARS
(3)                                ;*TEST 22      ASCII
(3)                                .IFF
(3)                                ASCII
(3)                                .ENDC
(3)                                .ENDM
(3) 010162                                STARS
(4)                                .IF B
(4)                                ;*****
(4)                                .IFF
(4)                                .NLIST
(4)                                .REPT
(4)                                .LIST
(4)                                ;*****
(4)                                .NLIST
(4)                                .ENDR
(4)                                .LIST
(4)                                .ENDC
(3) 010162 000004                TST22:  SCOPE
(3)                                $TN=$TN+1
(2)                                .IFF
(2)                                .SSNEWTEST          \STN,<>,<>
(2)                                .ENDC
(2)                                .IF NE 4000&$SWR
(2)                                .IF NB
(2)                                .IF LE <-1>
```



```
(2)          MOV      #1,$TIMES      ;;DO 1 ITERATION
(2)          .IFF
(2)          MOV      #,$TIMES      ;;DO ITERATIONS
(2)          .ENDC
(2)          002      .ENDC
(2)          001      .ENDC
(2)          002      .IF NB
(2)          MOV      #,$LPADR      ;;SET SCOPE LOOP ADDRESS
(2)          .ENDC
(2)          001      .ENDC
(2)          000      .ENDC
(2)          001      .IF NB
(2)          002      .IF DF $MAIL
(2)          .IRP      TNM,<\$TN-1>
(2)          MOV      #'TNM,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
(2)          .ENDM
(2)          .ENDC
(2)          001      .ENDC
(2)          000      .ENDC
(1)          000000 .REPT 0
(1)          .NLIST
(1)          .RADIX 10
(1)          .LIST
(1)          .IRP      $DN,<\$N>
(1)          NEWTST
(1)          :      MOV      #,$DN,$TSTNM  ;SAVE THIS
(1)          .ENDM
(1)          .NLIST
(1)          .RADIX 8
(1)          .LIST
(1)          :.IFF NB      <>,      MOV      #,RETURN      ;SET UP THIS RETURN
(1)          :.IFF NB      <>,      MOV      #,ICOUNT      ;SET UP THIS ICOUNT
(1)          :.IF NB <>
(1)          :      MOV      #,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .IFF
(1)          .NLIST
(1)          .RADIX 10
(1)          .LIST
(1)          .IRP      $DE,<\$E>
(1)          .IF DF      TST '$DE
(1)          :      MOV      #TST'$DE,NEXT  ;GO TO THIS TEST WHEN THRU
(1)          .ENDM
(1)          .NLIST
(1)          .RADIX 8
(1)          .LIST
(1)          .IFF
(1)          :      MOV      #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .ENDC
(1)          .ENDC
(1)          :.IFF NB      <>,      MOV      #,LOCK      ;SET UP FOR SCOPE LOOP
(1)          .NLIST
(1)          .RADIX 10
(1)          $N=$N+1
(1)          $E=$E+1
(1)          .RADIX 8
(1)          .LIST
(1)          .ENDR
8400
8401 010164 TSETUP MINT,SYNEXT,EIGHT,NOPAR,26
```

```

(1) 010164 $RESET
(2) 010164 052777 000400 171534 BIS #MRESET,@TXCSR ;MASTER RESET
(1) 010172 012777 020000 171522 MOV #SYNEXT,@PARCSR ;SET THE MODE
(1) 010200 $RESET
(2) 010200 052777 000400 171520 BIS #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINTENANCE MODE & SEND
(1) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 010206 012777 004020 171512 MOV #MINT!SEND,@TXCSR
(1)
(1) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 010214 012777 026026 171500 MOV #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8402 010222 016703 171500 MOV TXCSR,R3 ;SETUP FOR ERROR MESSAGE
8403 010226 112777 000252 171476 MOVB #252,@TXDBUF ;LOAD DATA CHAR.
8404 010234 012767 000252 171236 MOV #252,$TMP1 ;SHIFTED CHAR
8405 010242 012767 000010 170652 MOV #8,SHIFT ;# OF SHIFTS
8406 ;GET INTO SYNCRONIZATION
8407 010250 $POKE
(1) 010250 052777 020000 171450 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 010256 042777 020000 171442 BIC #CLK,@TXCSR ;POKE CLK DOWN
8408
8409 010264 005000 1$: CLR R0
8410 010266 006067 171206 ROR $TMP1 ;FORCE CARRY
8411 010272 103002 BCC 2$
8412 010274 052700 002000 BIC #BITW,R0 ;EQUIV OF BIT WINDOW
8413
8414 010300 2$:
(1) 010300 052777 020000 171420 B'S #CLK,@TXCSR ;POKE CLK UP
(1) 010306 042777 020000 171412 BIC #CLK,@TXCSR ;POKE CLK DOWN
8415 010314 017701 171406 MOV @TXCSR,R1 ;ACTUAL
8416 010320 042701 075777 JIC #075777,R1 ;SAVE ONLY BIT WINDOW & DNA
8417 010324 020001 MOV R0,R1
8418 010326 001401 BEQ +4
8419 010330 104003 ERROR 3 ;BIT WINDOW DID NOT MATCH
8420 ;ACTUAL DATA BIT
8421 010332 005367 170564 DEC SHIFT
8422 010336 022767 000003 170556 CMP #,SHIFT
8423 010344 001003 BNE 3$
8424 010346 042777 000020 171352 BIC #SEND,@TXCSR ;DROP SEND
8425 010354 005767 170542 3$: TST SHIFT
8426 010360 001341 BNE 1$ ;DO IT AGAIN?
8427
8428
8429
8430 ::THIS TEST VERIFYS THAT RXDONE ASSERTS WHEN STRIP SYNC IS SET
8431 ::MODE: SYNC INTERNAL
8432 ::LENGTH: EIGHT
8433
8434 010362 $STNO
(1) 010362 NEWTST
(2) 000000 $NWTST=0
(2) 001 .IF B <>
(2) 010362 $$NEWTST \STN,<>,SCOPE
(3) .IRP ASCII,<>
(3) .IF EQ $NWTST
(3) .NLIST
  
```

```
(3) $NWTST=1
(3) .SBTTL 123 ASCII
(3) .LIST
(3) STARS
(3) :*TEST 23 ASCII
(3) .IFF
(3) ASCII
(3) .ENDC
(3) .ENDM
(3) 010362 STARS
(4) 002 .IF B
(4) :*****
(4) .IFF
(4) .NLIST
(4) .REPT
(4) .LIST
(4) :*****
(4) .NLIST
(4) .ENDR
(4) .LIST
(4) 010362 001 .ENDC
(3) 000004 TST23: SCOPE
(3) 000024 $TN=$TN+1
(2) .IFF
(2) $$NEWTST \ $TN,<>,<>
(2) .ENDC
(2) 000 .IF NE 4000&$$SWR
(2) 001 .IF NB
(2) 002 .IF LE <-1>
(2) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(2) .IFF
(2) MOV #,$TIMES ;;DO ITERATIONS
(2) .ENDC
(2) 002 .ENDC
(2) 001 .IF NB
(2) 002 MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(2) .ENDC
(2) 001 .ENDC
(2) 000 .IF NB
(2) 001 .IF DF $MAIL
(2) 002 .IRP TNM,<\$TN-1>
(2) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(2) .ENDM
(2) .ENDC
(2) 001 .ENDC
(2) 000 .REPT 0
(1) 000000 .NLIST
(1) .RADIX 10
(1) .LIST
(1) .IRP $DN,<\$N>
(1) NEWTST
(1) : MOV # $DN,$STNM ;SAVE THIS
(1) .ENDM
(1) .NLIST
(1) .RADIX 8
```

```

(1) .LIST
(1) :.IIF NB <>, MOV #,RETURN ;SET UP THIS RETURN
(1) :.IIF NB <>, MOV #,ICOUNT ;SET UP THIS ICOUNT
(1) :.IF NB <>
(1) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(1) :.IFF
(1) :.NLIST
(1) :.RADIX 10
(1) :.LIST
(1) :.IRP $DE,<\$E>
(1) :.IF DF TST'$DE
(1) : MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(1) :.ENDM
(1) :.NLIST
(1) :.RADIX 8
(1) :.LIST
(1) :.IFF
(1) : MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(1) :.ENDC
(1) :.ENDC
(1) :.IIF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(1) :.NLIST
(1) :.RADIX 10
(1) :$N=$N+1
(1) :$E=$E+1
(1) :.RADIX 8
(1) :.LIST
(1) :.ENDR
8435 010364 RSETUP MINT,SYNINT,EIGHT,NOPAR,26
(1) 010364 $RESET
(2) 010364 052777 000400 171334 BIS #MRESET,@TXCSR ;MASTER RESET
(1) 010372 012777 030000 171322 MOV #SYNINT,@PARCSR ;SET THE MODE
(1) 010400 $RESET
(2) 010400 052777 000400 171320 BIS #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(1) 010406 012777 064001 171312 MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
(1)
(1) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(1) 010414 012777 036026 171300 MOV #SYNINT!EIGHT!NOPAR!26,@PARCSR
8436
8437 010422 052777 000420 171262 BIS #SYNSCH!STPSYN,@RXCSR ;SET SYNC SEARCH &
8438 ;STRIP SYNC
8439 ;POKE CLK TO GET RECEIVER INTO SYNCHRONIZATION.....
8440 010430 042777 020000 171270 BIC #CLK,@TXCSR ;POKE CLK
8441 010436 052777 020000 171262 BIS #CLK,@TXCSR ;
8442 010444 $POKER
(1) ;POKE CLK TO GET LOGIC INTO SYNCHRONIZATION
(1) 010444 042777 020000 171254 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 010452 052777 020000 171246 BIS #CLK,@TXCSR ;POKE CLK UP
8443 010460 016703 171232 MOV RXDBUF,R3 ;FOR ERROR MESSAGE
8444 010464 012767 000002 170432 MOV #2,COUNT ;# OF TIMES OF SYNC CHARS.
8445 ;TEST TO SEE HOW MANY SYNC CHARS NEEDED
8446 010472 105767 170450 TSTB SYNCNO
8447 010476 100402 BMI 3$ ;WILL IT BE ONE OR TWO ?
8448 010500 005367 170420 DEC COUNT ;MAKE IT ONE LESS

```

```

8449 010504 012767 000010 170410 3$: MOV #8,SHIFT ;#OF SHIFTS
8450 010512 012767 000026 170760 MOV #26,$TMP1 ;SYNC CHAR
8451 010520 004767 006446 JSR PC,RPOKE
8452 010524 005367 170374 DEC COUNT ;IS IT THE LAST SYNC CHAR ?
8453 010530 001365 BNE 3$ ;GO AGAIN AND SHIFT IN ANOTHER SYNC CHAR
8454 010532 032777 004000 171152 BIT #REACT,@RXCSR ;REACT=1?
8455 010540 001001 BNE .+4
8456 010542 104004 ERROR 4 ;REACT SHOULD BE ASSERTED
8457 010544 105777 171142 TSTB @RXCSR ;RXDONE=0?
8458 010550 100001 BPL .+4
8459 010552 104004 ERROR 4 ;RXDONE SHOULD NOT BE ASSERTED
8460 010554 012767 000010 170340 MOV #8,SHIFT ;#OF SHIFTS
8461 010562 012767 000025 170710 MOV #25,$TMP1 ;ANY CHARACTER
8462 010570 004767 006376 JSR PC,RPOKE
8463 010574 105777 171112 TSTB @RXCSR ;RXDONE=1?
8464 010600 100401 BMI .+4
8465 010602 104004 ERROR 4 ;RXDONE SHOULD NOW BE ASSERTED
8466 010604 012700 000025 MOV #25,R0 ;EXPECTED
8467 010610 017701 171102 MOV @RXDBUF,R1 ;ACTUAL
8468 010614 020001 CMP R0,R1
8469 010616 001401 BEQ .+4
8470 010620 104002 ERROR 2 ;CHARACTERS SHOULD BE MATCHED

```

8471
8472
8473
8474
8475
8476
8477
8478
8479
8480
8481
8482
8483
8484
8485

```

::THIS TEST VERIFYS THAT BY DROPPING SYNSCH
::IN THE MIDDLE OF A CHARACTER, SYNC CHARACTER SEQUENCE
::IS NEEDED BEFORE REACT, RXDONE ASSERT AGAIN.
::ALSO NOTE: SINCE REACT IS DEPENDENT ON MATCH DETECT,
::AND IF SYNSCH IS DROPPED IN THE MIDDLE OF
::A SYNC CHARACTER AND THEN RAISED AGAIN ;RXDONE SHOULD
::NOT ASSERT UNTIL NEW SYNC CHARACTER SEQUENCE.....HOWEVER
::ONE "PAD" CHARACTER MUST PRECEED THIS JUGGLING OF SEARCH SYNC (SYNSCH)
::MODE: SYNC INTERNAL (SYNINT)
::LENGTH: EIGHT
::

```

```

8486 010622 $TSTNO
(1) 010622 NEWTST
(2) 000000 $NWTST=0
(2) 001 .IF B <>
(2) 010622 $$NEWTST \STN,<>,SCOPE
(3) .IRP ASCII,<>
(3) .IF EQ $NWTST
(3) .NLIST
(3) $NWTST=1
(3) .SBTTL 124 ASCII
(3) .LIST
(3) STARS
(3) :*TEST 24 ASCII
(3) .IFF
(3) ASCII
(3) .ENDC
(3) .ENDM
(3) 010622 STARS
(4) 002 .IF B

```

```

(4) ;*****
(4) .IFF
(4) .NLIST
(4) .REPT
(4) .LIST
(4) ;*****
(4) .NLIST
(4) .ENDR
(4) .LIST
(4) .ENDC
(3) 010622 001
(3) 000004 TST24: SCOPE
(3) 000025 $TN=$TN+1
(2) .IFF
(2) $NEWTEST \STN,<>,<>
(2) .ENDC
(2) 000 .IF NE 4000&$SWR
(2) 001 .IF NB
(2) 002 .IF LE <-1>
(2) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(2) .IFF
(2) MOV #,$TIMES ;;DO ITERATIONS
(2) .ENDC
(2) 002 .ENDC
(2) 001 .ENDC
(2) 002 .IF NB
(2) MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(2) .ENDC
(2) 001 .ENDC
(2) 000 .IF NB
(2) 001 .IF DF $MAIL
(2) 002 .IRP TNM,<\STN-1>
(2) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(2) .ENDM
(2) .ENDC
(2) 001 .ENDC
(2) 000 .ENDC
(1) 000000 .REPT 0
(1) .NLIST
(1) .RADIX 10
(1) .LIST
(1) .IRP $DN,<\$N>
(1) NEWTST
(1) : MOV #,$DN,$STNM ;SAVE THIS
(1) .ENDM
(1) .NLIST
(1) .RADIX 8
(1) .LIST
(1) :.IFF NB <>, MOV #,RETURN ;SET UP THIS RETURN
(1) :.IFF NB <>, MOV #,ICOUNT ;SET UP THIS ICOUNT
(1) :.IF NB <>
(1) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU
(1) .IFF
(1) .NLIST
(1) .RADIX 10
(1) .LIST
(1) .IRP $DE,<\$E>
(1) .IF DF TST $DE

```

```

(1) ; MOV #TST'SDE,NEXT ;GO TO THIS TEST WHEN THRU
(1) ;.ENDM
(1) ;.NLIST
(1) ;.RADIX 8
(1) ;.LIST
(1) ;.IFF
(1) ; MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(1) ;.ENDC
(1) ;.ENDC
(1) ;.IIF NB <>, MOV #.LOCK ;SET UP FOR SCOPE LOOP
(1) ;.NLIST
(1) ;.RADIX 10
(1) ;$N=$N+1
(1) ;$E=$E+1
(1) ;.RADIX 8
(1) ;.LIST
(1) ;.ENDR
8487
8488 010624 ;.E TUP MINT,SYNINT,EIGHT,NOPAR,26
(1) 010624 ;.RESET
(2) 010624 052777 000400 171074 ; BIS #MRESET,@TXCSR ;MASTER RESET
(1) 010632 012777 030000 171062 ; MOV #SYNINT,@PARCSR ;SET THE MODE
(1) 010640 $RESET
(2) 010640 052777 000400 171060 ; BIS #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINT DATA,CLK,BREAK,&MAINTENANCE MODE
(1) 010646 012777 064001 171052 ; MOV #MTDATA!CLK!MINT!BREAK,@TXCSR
(1)
(1) ;SET MODE ,# OF BITS,PARITY SENSE,&LOAD SYNC REG
(1) 010654 012777 036026 171040 ; MOV #SYNINT!EIGHT!NOPAR!26,@PARCSR
8489
8490 010662 $SYNCR
(1) 010662 052777 000020 171022 ; BIS #SYNSCH,@RXCSR ;SET SYNC SEARCH
(1) ;POKE CLK TO GET RECEIVER INTO SYNCRIZATION....
(1) 010670 042777 020000 171030 ; BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 010676 052777 020000 171022 ; BIS #CLK,@TXCSR ;POKE CLK UP
8491 010704 $POKER
(1) ;POKE CLK TO GET LOGIC INTO SYNCRONIZATION
(1) 010704 042777 020000 171014 ; BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 010712 052777 020000 171006 ; BIS #CLK,@TXCSR ;POKE CLK UP
8492 010720 012767 000002 170176 ; MOV #2,COUNT ;# OF TIMES
8493 010726 016703 170764 ; MOV RXDBUF,R3 ;FOR ERROR MESSAGE
8494 010732 012767 000010 170162 1$: ; MOV #8,SHIFT ;# OF SHIFTS
8495 010740 012767 000026 170532 ; MOV #26,$TMP1 ;SYNC CHAR.
8496 010746 004767 006220 ; JSR PC,RPOKE
8497 010752 005367 170146 ; DEC COUNT
8498 010756 001403 ; BEQ 2$
8499 ;TEST TO SEE HOW MANY SYNC CHARACTERS NEEDED
8500 010760 105767 170162 ; TSTB SYNCNO
8501 010764 100762 ; BMI 1$
8502
8503 010766 032777 004000 170716 2$: ; BIT #REACT,@RXCSR ;REACT=1?
8504 010774 001001 ; BNE .+4
8505 010776 104004 ; ERROR 4 ;REACT SHOULD BE SET
8506 011000 105777 170706 ; TSTB @RXCSR ;RXDONE = 0?
8507 011004 100001 ; BPL .+4

```

```

8508 011006 104004
8509
8510
8511
8512
8513
8514
8515 011010 012767 000010 170104
8516 011016 012767 000025 170454
8517 011024 004767 006142
8518 011030 105777 170656
8519 011034 100401
8520 011036 104004
8521 011040 017701 170652
8522 011044 012700 000025
8523 011050 020001
8524 011052 001401
8525 011054 104002
8526
8527
8528
8529 011056 012767 000004 170036
8530 011064 012767 000026 170406
8531 011072 004767 006074
8532 011076 032777 004000 170606
8533 011104 001001
8534 011106 104004
8535 011110 105777 170576
8536 011114 100001
8537 011116 104004
8538 011120 042777 000020 170564
8539 011126 032777 004000 170556
8540 011134 001401
8541 011136 104004
8542
8543
8544
8545 011140 012767 000002 167754
8546 011146 004767 006020
8547 011152 052777 000020 170532
8548 011160 032777 004000 170524
8549 011166 001401
8550 011170 104004
8551 011172 105777 170514
8552 011176 100001
8553 011200 104004
8554 011202 012767 000002 167712
8555 011210 004767 005756
8556 011214 032777 004000 170470
8557 011222 001401
8558 011224 104004
8559 011226 105777 170460
8560 011232 100001
8561 011234 104004
8562 011236 012700 000025
8563 011242 017701 170450

```

```

      ERROR 4 ;RXDONE SHOULD = 0
;THE FOLLOWING ALLOWS THE RECEIVER "CHIP" TO RECOGNIZE THE DROPPING OF SYNSCH
;...BASICALLY IT SIMULATES THE "PAD" CHARACTER REQUIREMENT AT THE END
;OF DATA TRANSFER.
;ONE "PAD" CHARACTER IS REQUIRED TO FLUSH OUT SYNC CHARACTER
;AND YOU HAVE TO DROP SYNSCH ON THE NEXT OR FOLLOWING CHARACTER
;FOR A "CLEAN" RESTART
      MOV #8,SHIFT ;# OF SHIFTS
      MOV #25,$TMP1 ;"PAD" CHARACTER
      JSR PC,RPOKE ;SHIFT IN "PAD" CHARACTER
      TSTB @RXCSR ;RXDONE = 1 ?
      BMI +4
      ERROR 4 ;RXDONE SHOULD = 1
      MOV @RXDBUF,R1 ;STORE ACTUAL & CLEAR RXDONE
      MOV #25,R0 ;EXPECTED
      CMP R0,R1 ;COMPARE EXPECTED DATA VS ACTUAL DATA
      BEQ +4
      ERROR 2 ;DATA SHOULD COMPARE WITHOUT ERROR FLAGS
;THE FOLLOWING IS THE SYNC CHARACTER AFTER THE "PAD" CHARACTER
;IN WHICH SEARCH SYNC (SYNSCH) IS JUGGLED TO CREATE AN ABORT SITUATION
;AT THE END OF A DATA TRANSFER SEQUENCE
      MOV #4,SHIFT ;# OF SHIFTS
      MOV #26,$TMP1 ;SYNC CHAR.
      JSR PC,RPOKE
      BIT #REACT,@RXCSR ;REACT=1?
      BNE +4
      ERROR 4 ;REACT SHOULD STILL BE SET
      TSTB @RXCSR ;RXDONE = 0 ?
      BPL +4
      ERROR 4 ;RXDONE SHOULD = 0 FROM PREVIOUS READING OF RXDBUF
      BIC #SYNSCH,@RXCSR ;DROP SEARCH SYNC
      BIT #REACT,@RXCSR ;REACT=0?
      BEQ +4
      ERROR 4 ;REACT SHOULD NOT BE SET
;NOW SHIFT TWO BITS TO ALLOW SEARCH SYNC =0 TO TAKE
;EFFECT IN THE LOGIC(THIS ALLOWS THE RECEIVER CHIP TO SEE
;THE DROPPING OF SEARCH SYNC)
      MOV #2,SHIFT ;# OF SHIFTS
      JSR PC,RPOKE
      BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
      BIT #REACT,@RXCSR
      BEQ +4
      ERROR 4 ;REACT =0 ?
      TSTB @RXCSR ;RXDONE = 0 ?
      BPL +4
      ERROR 4 ;RXDONE = 0 ?
      MOV #2,SHIFT ;# OF SHIFTS TO FINISH UP THE SYNC CHARACTER
      JSR PC,RPOKE
      BIT #REACT,@RXCSR ;REACT=0?
      BEQ +4
      ERROR 4 ;REACT SHOULD NOT BE SET
      TSTB @RXCSR ;RXDONE=0?
      BPL +4
      ERROR 4 ;RXDONE SHOULD NOT BE ASSERTED
      MOV #25,R0 ;EXPECTED
      MOV @RXDBUF,R1 ;ACTUAL

```



```

8564 011246 020001      CMP      R0,R1      ;COMPARE EXPECTED VS ACTUAL
8565 011250 001401      BEQ      +4
8566 011252 104002      ERROR    2          ;CHARACTERS SHOULD BE MATCHED.....
8567                          ;REMBEMBER THAT THE "25" FROZE WHEN SYN SCH WAS DROPPED
8568                          ;THEREFORE ....25 WILL BE READ AND NOT 26
8569                          ;THE FOLLOWING VERIFYS THAT THE RE SYNCHRONIZATION COMES UP "CLEAN"
8570 011254 012767 000002 167642  MOV      #2,COUNT    ;# OF TIMES OF SYNC CHARS.
8571                          ;
8572                          ;TEST TO SEE HOW MANY SYNC CHARS NEEDED
8573 011262 105767 167660  TSTB     SYNCNO     ;HOW MANY SYNCs ARE YOU STRAPPED FOR ?
8574 011266 100402      BMI      3$         ;WILL IT BE ONE OR TWO ?
8575 011270 005367 167630  DEC      COUNT     ;IT WAS ONLY ONE NEEDED
8576 011274 012767 000010 167620 3$:  MOV      #8,SHIFT   ;#OF SHIFTS
8577 011302 012767 000026 170170  MOV      #26,$TMP1  ;SYNC CHAR
8578 011310 004767 005656  JSR      PC,RPOKE
8579 011314 005367 167604  DEC      COUNT     ;IS IT THE LAST SYNC CHAR ?
8580 011320 001365      BNE      3$         ;GO AGAIN AND SHIFT IN ANOTHER SYNC CHAR
8581 011322 032777 004000 170362  BIT      #RECACT,@RXCSR ;RECACT=1?
8582 011330 001001      BNE      +4
8583 011332 104004      ERROR    4          ;RECACT SHOULD BE ASSERTED
8584 011334 105777 170352  TSTB     @RXCSR     ;RXDONE=0?
8585 011340 100001      BPL      +4
8586 011342 104004      ERROR    4          ;RXDONE SHOULD NOT BE ASSERTED
8587 011344 012767 000010 167550  MOV      #8,SHIFT   ;#OF SHIFTS
8588 011352 012767 000025 170120  MOV      #25,$TMP1  ;ANY CHARACTER
8589 011360 004767 005606  JSR      PC,RPOKE
8590 011364 105777 170322  TSTB     @RXCSR     ;RXDONE=1?
8591 011370 100401      BMI      +4
8592 011372 104004      ERROR    4          ;RXDONE SHOULD NOW BE ASSERTED
8593 011374 012700 000025  MOV      #25,R0     ;EXPECTED
8594 011400 017701 170312  MOV      @RXDBUF,R1 ;ACTUAL
8595 011404 020001      CMP      R0,R1
8596 011406 001401      BEQ      +4
8597 011410 104002      ERROR    2          ;CHARACTERS SHOULD BE MATCHED

```

```

8598
8599
8600
8601      ;;THIS TEST VERIFYS THAT HDX MODE DISQUALIFIES THE
8602      ;;RECEIVER WHEN SEND IS ASSERTED
8603      ;;MODE: SYNC EXT
8604      ;;LENGTH: EIGHT
8605      ;;NOTE: THIS TEST WORKS ONLY IN MAINT. EXTERNAL MODE
8606      ;;THIS TEST USES BOTH RECEIVER & TRANSMITTER LOGIC
8607 011412 $TSTNO
      (1) 011412 NEWTST
      (2) 000000 $NWTST=0
      (2) 001 .IF B <>
      (2) 011412 $$NEWTST \STN,<>,SCOPE
      (3) .IRP ASCII,<>
      (3) .IF EQ $NWTST
      (3) .NLIST
      (3) $NWTST=1
      (3) .SBTTL 125 ASCII
      (3) .LIST
      (3) STARS
      (3) ;*TEST 25 ASCII

```

```

(3) .IFF
(3) .ENDC ASCII
(3) .ENDM
(3) 011412 STARS
(4) 002 .IF B
(4) ;*****
(4) .IFF
(4) .NLIST
(4) .REPT
(4) .LIST
(4) ;*****
(4) .NLIST
(4) .ENDR
(4) .LIST
(4) 011412 001 .ENDC
(3) 000004 TST25: SCOPE
(3)
(3) 000026 $TN=$TN+1
(2) .IFF
(2) $NEWTEST \ $TN,<>,<>
(2) .ENDC
(2) 000 .IF NE 4000&$SWR
(2) 001 .IF NB
(2) 002 .IF LE <-1>
(2) 003 MOV #1,$TIMES ;;DO 1 ITERATION
(2) .IFF
(2) MOV #,$TIMES ;;DO ITERATIONS
(2) .ENDC
(2) 002 .ENDC
(2) 001 .IF NB
(2) 002 MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS
(2) .ENDC
(2) 001 .ENDC
(2) 000 .IF NB
(2) 001 .IF DF $MAIL
(2) 002 .IRP TNM,<\$TN-1>
(2) MOV #'TNM,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
(2) .ENDM
(2) .ENDC
(2) 001 .ENDC
(2) 000 .REPT 0
(1) 000000 .NLIST
(1) .RADIX 10
(1) .LIST
(1) .IRP $DN,<\$N>
(1) NEWTST
(1) : MOV # $DN,$TSTNM ;SAVE THIS
(1) .ENDM
(1) .NLIST
(1) .RADIX 8
(1) .LIST
(1) .: .IFF NB <> MOV #,RETURN ;SET UP THIS RETURN
(1) .: .IFF NB <> MOV #,ICOUNT ;SET UP THIS ICOUNT
(1) .: .IF NB <>
(1) : MOV #,NEXT ;GO TO THIS TEST WHEN THRU

```

```

(1) .IFF
(1) .NLIST
(1) .RADIX 10
(1) .LIST
(1) .IRP $DE,<\$E>
(1) .IF DF TST,$DE
(1) : MOV #TST'$DE,NEXT ;GO TO THIS TEST WHEN THRU
(1) .ENDM
(1) .NLIST
(1) .RADIX 8
(1) .LIST
(1) .IFF
(1) : MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(1) .ENDC
(1) .ENDC
(1) : .IIF NB <>, MOV #,LOCK ;SET UP FOR SCOPE LOOP
(1) .NLIST
(1) .RADIX 10
(1) $N=$N+1
(1) $E=$E+1
(1) .RADIX 8
(1) .LIST
(1) .ENDR
8608 011414 105767 167533 TSTB JMRBY
8609 011420 100155 BPL 1$ ;GET OUT OF THIS TEST IF 'NO'
8610 011422 016703 170270 MOV RXDBUF,R3 ;FOR ERROR MESSAGE
8611 011426 TSETUP MEXT,SYNEXT,EIGHT,NOPAR,26
(1) 011426 $RESET
(2) 011426 052777 000400 170272 BIS #MRESET,@TXCSR ;MASTER RESET
(1) 011434 012777 020000 170260 MOV #SYNEXT,@PARCSR ;SET THE MODE
(1) 011442 $RESET
(2) 011442 052777 000400 170256 BIS #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINTENANCE MODE & SEND
(1) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 011450 012777 010020 170250 MOV #MEXT!SEND,@TXCSR
(1)
(1) ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
(1) 011456 012777 026026 170236 MOV #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8612 011464 052777 000020 170220 BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
8613 011472 112777 000025 170232 MOVB #25,@TXDBUF ;ANY CHARACTER
8614 ;POKE CLK FOR SYNCRONIZATION
8615 011500 PUSSYFOOT
(1) 011500 052777 020000 170220 BIS #CLK,@TXCSR ;POKE CLK UP
(1) ;WAIT FOR CABLE & DRIVER DELAYS
(1) 011506 016702 167406 MOV HOLD,R2 ;WAIT THIS AMT
(1) 011512 005302 DEC R2 ;WAIT
(1) 011514 001376 BNE .-2
(1) ;EXIT...
(1) 011516 042777 020000 170202 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) ;WAIT FOR CABLE & DRIVER DELAYS
(1) 011524 016702 167370 MOV HOLD,R2 ;WAIT THIS AMT
(1) 011530 005302 DEC R2 ;WAIT
(1) 011532 001376 BNE .-2
(1) ;EXIT...
8616 011534 012767 000010 167360 MOV #8.,SHIFT ;# OF SHIFTS

```

```

8617 011542 052777 020000 170156 2$: BIS #CLK,@TXCSR ;POKE CLK UP
(1) 011542 052777 020000 170156 ;WAIT FOR CABLE & DRIVER DELAYS
(1) 011550 016702 167344 MOV HOLD,R2 ;WAIT THIS AMT
(1) 011554 005302 DEC R2 ;WAIT
(1) 011556 001376 BNE .-2
(1) ;EXIT...
(1) 011560 042777 020000 170140 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 011560 042777 020000 170140 ;WAIT FOR CABLE & DRIVER DELAYS
(1) 011566 016702 167326 MOV HOLD,R2 ;WAIT THIS AMT
(1) 011572 005302 DEC R2 ;WAIT
(1) 011574 001376 BNE .-2
(1) ;EXIT...
8618 011576 005367 167320 DEC SHIFT ;# OF SHIFTS
8619 011602 022767 000003 167312 CMP #3,SHIFT ;IS IT TIME TO LOAD NEXT CHAR ?
8620 011610 001003 BNE 3$
8621 011612 112777 000024 170112 MOV#B #24,@RXDBUF ;LOAD NEXT CHAR.
8622 011620 005767 167276 3$: TST SHIFT
8623 011624 001346 BNE 2$
8624 011626 105777 170060 TSTB @RXCSR ;RXDONE=1?
8625 011632 100401 BMI .+4
8626 011634 104004 ERROR 4 ;RXDONE SHOULD BE SET
8627 011636 012700 000025 MOV #25,R0 ;EXPECTED
8628 011642 017701 170050 MOV @RXDBUF,R1 ;ACTUAL
8629 011646 020001 CMP R0,R1
8630 011650 001401 BEQ .+4
8631 011652 104002 ERROR 2 ;CHARACTERS SHOULD COMPARE
8632 011654 052777 000010 170044 BIS #HDXEN,@TXCSR ;SET HALF DUPLEX HDX
8633 011662 012767 000010 167232 MOV #8.,SHIFT ;# OF SHIFTS
8634 011670 052777 020000 170030 4$: BIS #CLK,@TXCSR ;POKE CLK UP
(1) 011670 052777 020000 170030 ;WAIT FOR CABLE & DRIVER DELAYS
(1) 011676 016702 167216 MOV HOLD,R2 ;WAIT THIS AMT
(1) 011702 005302 DEC R2 ;WAIT
(1) 011704 001376 BNE .-2
(1) ;EXIT...
(1) 011706 042777 020000 170012 BIC #CLK,@TXCSR ;POKE CLK DOWN
(1) 011706 042777 020000 170012 ;WAIT FOR CABLE & DRIVER DELAYS
(1) 011714 016702 167200 MOV HOLD,R2 ;WAIT THIS AMT
(1) 011720 005302 DEC R2 ;WAIT
(1) 011722 001376 BNE .-2
(1) ;EXIT...
8635 011724 005367 167172 DEC SHIFT
8636 011730 001357 BNE 4$
8637 011732 105777 167754 TSTB @RXCSR ;RXDONE=0?
8638 011736 100001 BPL .+4
8639 011740 104004 ERROR 4 ;RXDONE SHOULD NOT BE ASSERTED
8640 ;CHECK OUT HDX LOGIC
8641 011742 017701 167750 MOV @RXDBUF,R1 ;ACTUAL
8642 011746 020001 CMP R0,R1
8643 011750 001401 BEQ .+4
8644 011752 104002 ERROR 2 ;CHARACTERS SHOULD COMPARE
8645 ;NOTE THAT CHARACTER 25 WILL BE FROZEN
8646 ;IN THE RXDBUF EVEN THOUGH CHARACTER 24 WAS
8647 ;SENT TO THE RECEIVER
8648

```

```
8649 011754 1$:  
8650  
8651  
8652 ::THIS TEST VERIFYS THAT BREAK FORCES A SPACE CONDITION  
8653 ::ON THE LINE WHILE TRANSMITTING  
8654 ::THIS TEST USES BOTH THE RECEIVER AND TRANSMITTER LOGIC  
8655 ::MODE: SYNC EXT (SYNEXT)  
8656 ::LENGTH: EIGHT  
8657 ::  
8658  
8659 011754 $TSTNO  
(1) 011754 NEWTST  
(2) 000000 $NWTST=0  
(2) 001 .IF B <>  
(2) 011754 $$NEWTST \STN,<>,SCOPE  
(3) .IRP ASCII,<>  
(3) .IF EQ $NWTST  
(3) .NLIST  
(3) $NWTST=1  
(3) .SBTTL T26 ASCII  
(3) .LIST  
(3) STARS  
(3) :*TEST 26 ASCII  
(3) .IFF  
(3) ASCII  
(3) .ENDC  
(3) .ENDP  
(3) 011754 STARS  
(4) 002 .IF B  
(4) ::*****  
(4) .IFF  
(4) .NLIST  
(4) .REPT  
(4) .LIST  
(4) ::*****  
(4) .NLIST  
(4) .ENDR  
(4) .LIST  
(4) 001 .ENDC  
(3) 011754 000004 TST26: SCOPE  
(3) 000027 $TN=$TN+1  
(2) .IFF  
(2) $$NEWTST \STN,<>,<>  
(2) .ENDC  
(2) 000 .IF NE 4000$$SWR  
(2) 001 .IF NB  
(2) 002 .IF LE <-1>  
(2) MOV #1,$TIMES ;;DO 1 ITERATION  
(2) .IFF  
(2) MOV #,$TIMES ;;DO ITERATIONS  
(2) .ENDC  
(2) 002 .ENDC  
(2) 001 .ENDC  
(2) 002 .IF NB  
(2) MOV #,$LPADR ;;SET SCOPE LOOP ADDRESS  
(2) 001 .ENDC
```

```
(2)          000          .ENDC
(2)          001          .IF NB
(2)          002          .IF DF $MAIL
(2)          .IRP        TNM,<\$TN-1>
(2)          .MOV        #'TNM,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
(2)          .ENDM
(2)          001          .ENDC
(2)          000          .ENDC
(1)          000000      .REPT 0
(1)          .NLIST
(1)          .RADIX 10
(1)          .LIST
(1)          .IRP        $DN,<\$N>
(1)          NEWTST
(1)          :           .MOV        # $DN,$STSTNM      ;SAVE THIS
(1)          .ENDM
(1)          .NLIST
(1)          .RADIX 8
(1)          .LIST
(1)          .:IF NB      <>,      .MOV        #,RETURN      ;SET UP THIS RETURN
(1)          .:IF NB      <>,      .MOV        #,ICOUNT      ;SET UP THIS ICOUNT
(1)          .:IF NB      <>
(1)          .:           .MOV        #,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .:           .IFF
(1)          .NLIST
(1)          .RADIX 10
(1)          .LIST
(1)          .IRP        $DE,<\$E>
(1)          .IF DF      TST,$DE
(1)          .:           .MOV        #TST'$DE,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .ENDM
(1)          .NLIST
(1)          .RADIX 8
(1)          .LIST
(1)          .:           .IFF
(1)          .:           .MOV        #.EOP,NEXT      ;GO TO THIS TEST WHEN THRU
(1)          .ENDC
(1)          .ENDC
(1)          .:IF NB      <>,      .MOV        #,LOCK      ;SET UP FOR SCOPE LOOP
(1)          .NLIST
(1)          .RADIX 10
(1)          $N=$N+1
(1)          $E=$E+1
(1)          .RADIX 8
(1)          .LIST
(1)          .ENDR
8660 011756  TSETUP MINT,SYNEXT,EIGHT,NOPAR,26
(1) 011756  $RESET
(2) 011756  052777 000400 167742  BIS      #MRESET,@TXCSR ;MASTER RESET
(1) 011764  012777 020000 167730  MOV      #SYNEXT,@PARCSR ;SET THE MODE
(1) 011772  $RESET
(2) 011772  052777 000400 167726  BIS      #MRESET,@TXCSR ;MASTER RESET
(1)
(1) ;SET MAINTENANCE MODE & SEND
(1) ;NOTE:BIT WINDOW&CLK ARE CLEARED (MTDATA=0)
(1) 012000  012777 004020 167720  MOV      #MINT!SEND,@TXCSR
```

```

(1)
(1)
(1) C12006 012777 026026 167706 ;SET MODE,# OF BITS,PARITY SENSE,& LOAD SYNC REG
8661 012014 052777 000020 167670 MOV #SYNEXT!EIGHT!NOPAR!26,@PARCSR
8662 012022 016703 167670 BIS #SYNSCH,@RXCSR ;SET SEARCH SYNC
8663 012026 012767 000002 167070 MOV RXDBUF,R3 ;FOR ERROR MESSAGE
8664 012034 112777 000025 167670 MOV #2,COUNT ;# OF TIMES
8665 MOVB #25,@TXDBUF ;ANY CHARACTER
8666 012042 $POKE ;POKE CLK FOR SYNCRONIZATION
(1) 012042 052777 020000 167656 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 012050 042777 020000 167650 BIC #CLK,@TXCSR ;POKE CLK DOWN
8667 012056 012700 000025 MOV #25,R0 ;EXPECTED
8668 012062 012767 000010 167032 1$: MOV #8.,SHIFT ;# OF SHIFTS
8669
8670 012070 2$:
(1) 012070 052777 020000 167630 BIS #CLK,@TXCSR ;POKE CLK UP
(1) 012076 042777 020000 167622 BIC #CLK,@TXCSR ;POKE CLK DOWN
8671 012104 005367 167012 DEC SHIFT
8672 012110 022767 000003 167004 CMP #3,SHIFT
8673 012116 001003 BNE 3$
8674 012120 112777 000024 167604 MOVB #24,@TXDBUF ;LOAD NEXT CHAR
8675 012126 005767 166770 3$: TST SHIFT
8676 012132 001356 BNE 2$
8677 012134 105777 167552 TSTB @RXCSR ;RXDONE=1?
8678 012140 100401 BMI .+4
8679 012142 104004 ERROR 4
8680 012144 017701 167546 MOV @RXDBUF,R1 ;ACTUAL
8681 012150 020001 CMP R0,R1
8682 012152 001401 BEQ .+4
8683 012154 104003 ERROR 3
8684 012156 052777 000001 167542 BIS #BREAK,@TXCSR ;SET BREAK
8685 012164 012700 000000 MOV #0,R0 ;EXPECTED
8686 012170 005367 166730 DEC COUNT
8687 012174 001332 BNE 1$
8688
8689
8690
8691
8692
8693 012176
(1) 012176 $TSTNO
(2) 000000 NEWTST
(2) 001 $NWTST=0
(2) 012176 .IF B <> $$NEWTST \ $TN,<>,SCOPE
(3) .IRP ASCII,<>
(3) .IF EQ $NWTST
(3) .NLIST
(3) $NWTST=1
(3) .SBTTL T27 ASCII
(3) .LIST
(3) STARS
(3) :*TEST 27 ASCII
(3) .IFF
(3) ASCII
(3) .ENDC
(3) .ENDM

```


INITIALIZE THE COMMON TAGS

```

(1) .IRP $DE,<\$E>
(1) .IF DF TST '$DE'
(1) .MOV #TST'$DE',NEXT ;GO TO THIS TEST WHEN THRU
(1) .ENDM
(1) .NLIST
(1) .RADIX 8
(1) .LIST
(1) .IFF
(1) .MOV #.EOP,NEXT ;GO TO THIS TEST WHEN THRU
(1) .ENDC
(1) .ENDC
(1) .IIF NB <>, .MOV #.LOCK ;SET UP FOR SCOPE LOOP
(1) .NLIST
(1) .RADIX 10
(1) $N=$N+1
(1) $E=$E+1
(1) .RADIX 8
(1) .LIST
(1) .ENDR
8694 012200 105767 166747 TSTB JMRBY ;IN MAINT EXTERNAL?
8695 012204 100073 BPL 3$ ;IF ANSWER NO JUMP AROUND TEST
8696 012206 $RESET
(1) 012206 052777 000400 167512 BIS #MRESET,@RXCSR ;MASTER RESET
8697 012214 105767 166731 TSTB OPTCLR ;IS THE OPTIONAL CLR JUMPER IN ?
8698 012220 100405 BMI 5$ ;YES
8699 012222 012777 000000 167462 MOV #0,@RXCSR ;CLR THE UNRESETTABLE BITS
8700 012230 005777 167456 TST @RXCSR ;GET RID OF DSC BY READING RXCSR
8701 012234 012777 012256 167474 5$: MOV #4,@DURIV ;SET UP TRAPCATCHER
8702 012242 016777 004564 167470 MOV DUPRT,@DURIS ;
8703 012250 106427 000000 MTPS #0 ;ALLOW INTERRUPTS
8704 012254 000422 BR 1$ ;JUMP AROUND INTERRUPT SVC ROUTINE
8705 ;THE FOLLOWING IS THE INTERRUPT SVC ROUTINE
8706 012256 106427 000300 4$: MTPS #300 ;DON'T ALLOW ANYMORE INTERRUPTS
8707 012262 005777 167424 TST @RXCSR ;DSC=1?
8708 012266 100401 BMI .+4
8709 012270 104004 ERROR 4 ;FALSE INTERRUPT
8710 012272 042777 000040 167412 BIC #DSINTE,@RXCSR ;CLEAR INTERRUPT ENABLE
8711 012300 012716 012370 MOV #2$(SP) ;SET UP RETURN LOCATION
8712 012304 016777 167430 167424 MOV DURIS,@DURIV ;RESTORE TRAPCATCHER
8713 012312 012777 000000 167420 MOV #0,@DURIS ;
8714 012320 000002 RTI
8715
8716 012322 052777 000040 167362 1$: BIS #DSINTE,@RXCSR ;SET INTERRUPT ENABLE
8717 012330 052777 000002 167354 BIS #DTR,@RXCSR ;TRY TO CAUSE INTERRUPT
8718 012336 005000 CLR RO
8719 012340 005200 INC RO ;WAIT FOR INTERRUPT
8720 012342 001376 BNE .-2
8721 012344 016777 167370 167364 MOV DURIS,@DURIV ;RESTORE TRAPCATCHER
8722 012352 012777 000000 167360 MOV #0,@DURIS ;
8723
8724 012360 042777 000040 167324 BIC #DSINTE,@RXCSR ;CLEAR INTERRUPT ENABLE
8725 012366 104004 ERROR 4 ;INTERRUPT FAILED TO OCCUR
8726 012370 106427 000300 2$: MTPS #300
8727 012374 3$:
8728
8729 012374 PRGEND <DUV11 CNDUU-A TAPE E>,<END OF PASS TAPE E>

```

```

(1) 012374 $EOP .BEGIN
(2)
(2) 000012 .RADIX 10
(2) 000000 $N=$N-1
(2) 000010 .RADIX 8
(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) 012374 000004 .EOP: SCOPE
(2) 012376 004767 000340 JSR PC,CKSWR
(2) 012402 104401 TYPE ;TYPE NAME OF TEST
(2) 012404 015532 MEPASS
(2) 012406 104413 012640 CONVRT .OUTCRY
(2) 012412 104401 015351 TYPE .DEVICE
(2) 012416 105767 166530 TSTB MULTD ;ARE YOU RUNNING MULTIPLE DEVICES ?
(2) 012422 001511 BEQ CCC ;NO, JUMP AROUND
(2) 012424 005767 166536 TST ACTREG ;ARE ANY DEVICES ACTIVE ?
(2) 012430 001007 BNE RUNIT ;YES
(2) 012432 104401 015363 TYPE .MCOV ;NO
(2) 012436 016700 166524 MOV ACTREG,R0 ;DISPLAY ACTREG
(2) 012442 000000 HALT ;SELECT SOMETHING TO RUN @ ACTREG:
(2) ;SELECT SWITCHES & HIT CONTINUE (PUT SW00 =1)
(2) 012444 000167 167502 JMP .START ;START OVER AGAIN.....YOU DESELECTED EVERYTHING
(2) 012450 062767 000010 166476 RUNIT: ADD #10,BASEADD ;NEXT BLOCK (ADDRESSES)
(2) 012456 062767 000010 166476 ZERO: ADD #10,BASEIV ;NEXT BLOCK (VECTORS)
(2) 012464 000241 CLC
(2) 012466 006167 166476 ROL ROTADD ;UP DATE ROTATING POINTER
(2) 012472 103410 BCS 2$ ;IS IT THE LAST DEVICE
(2) ;TO BE TESTED IN THIS PASS ?
(2) 012474 036767 166470 166464 BIT ROTADD,ACTREG ;TEST THIS DEVICE FOR ACTIVE STATUS
(2) 012502 001762 BEQ RUNIT ;IF NOT ACTIVE, TRY NEXT ADDRESS
(2) 012504 004767 000034 JSR PC,REPLAY ;CALCULATE NEW PARAMETERS
(2) 012510 000167 000210 JMP RESTRT ;YES IT WAS ACTIVE,TEST THIS DEVICE
(2) 012514 012767 000001 166446 2$: MOV #1,ROTADD ;OK!,NOW SET UP ROTATING
(2) ;POINTER FOR NEXT MULTIPLE PASS
(2) 012522 016767 166430 166424 MOV KEEPADD,BASEADD ;RESTORE BASE ADDRESS
(2) 012530 016767 166430 166424 MOV KEEPIV,BASEIV ;RESTORE BASE INTERRUPT VECTORS
(2) 012536 004767 000002 JSR PC,REPLAY ;CALC NEW PARAMETERS
(2) 012542 000441 BR CCC ;JUMP AROUND REPLAY
(2) 012544 016767 166404 004416 REPLAY: MOV BASEADD,DUBASE ;SET UP FOR NEW ADDRESSES
(2) 012552 004767 004260 JSR PC,DUADDR ;CREATE NEW ADDRESSES
(2) 012556 016767 166400 167152 MOV BASEIV,DURIV ;CREATE DURIV
(2) 012564 062767 000002 166370 ADD #2,BASEIV
(2) 012572 016767 166364 167140 MOV BASEIV,DURIS ;CREATE DURIS
(2) 012600 062767 000002 166354 ADD #2,BASEIV
(2) 012606 016767 166350 167126 MOV BASEIV,DUTIV ;CREATE DUTIV
(2) 012614 062767 000002 166340 ADD #2,BASEIV
(2) 012622 016767 166334 167114 MOV BASEIV,DUTIS ;CREATE DUTIS
(2) 012630 016767 167102 166324 MOV DURIV,BASEIV ;RESTORE
(2) 012636 000207 RTS PC
(2)
(2) 012640 000001 OUTCRY: 1
(2) 012642 006 002 .BYTE 6,2

```

```

(2) 012644 (01712 RXCSR
(2)
(2) 012646 CCC:
(2) 012646 005067 166530 CLR $STNM ;CLEAR TEST NUMBER
(2) 012652 005067 166540 CLR $ERRPC ;CLEAR LAST ERROR PC
(2) 012656 005067 166521 CLR $ERFLG ;CLEAR ERROR FLAG
(2) 012662 005267 166224 INC PASCNT ;UPDATE PASS COUNT
(2) 012666 016767 166220 165206 MOV PASCNT,LIGHTS ;DISPLAY PASS COUNT
(2) 012674 016767 166212 166632 MOV PASCNT,$PASS ;PASS COUNT TO APT
(2) 012702 013701 00004? MOV @#42,R1 ;CHECK FOR ACT-11 OR DDP
(2) 012706 001406 BEQ RESTRT ;IF NO CONTINUE TESTING
(2) 012710 000005 RESFT
(2) 012712 000005 RESET
(2) 012714 004711 $ENDAD: JSR PC,(R1)
(2) 012716 000240 NOP
(2) 012720 000240 NOP
(2) 012722 000240 NOP
(2) 012724 RESTRT:
(2) 012724 012767 003376 166454 MOV #TST1+2,$LPADR ;LOAD LAST ADDR
(2) 012732 004767 000004 JSR PC,CKSWR
(2) 012736 000167 170346 JMP .BEGIN
(1) 012742 $CK
(2)
(2) ;CHECK SWITCH REGISTER ROUTINE.
(2) ;CHECKS TO ALLOW FOR <^G> TO ALLOW
(2) ;THE CHANGING OF LOCATION 176
(2)
(2) 012742 005737 000042 CKSWR: TST @#42
(2) 012746 001040 BNE OUT
(2) 012750 022767 000176 166462 CMP #SWREG,SWR ;SOFTWARE SWR PRESENT?
(2) 012756 001034 BNE OUT ;NO--LEAVE
(2) 012760 105777 166460 TSTB @$TKS ;CHECK TTY READY
(2) 012764 100031 BPL OUT ;NO--LEAVE
(2) 012766 017767 166454 000422 MOV @$TKB,.MSG ;GET CHARACTER
(2) 012774 042767 177600 000414 BIC #177600,.MSG ;STRIP JUNK
(2) 013002 122767 000007 000406 CMPB #7,.MSG ;IS IT <^G> ?
(2) 013010 001017 BNE OUT ;NO
(2) 013012 104401 016137 TYPE .MCNTG
(2) 013016 005137 013056 CNTLU: COM @#RDSW
(2) 013022 104401 016147 TYPE .MMSWR
(2) 013026 104413 CONVRT
(2) 013030 013060 SWREGL
(2) 013032 104406 016160 INSTR,MMNEW
(2) 013036 104410 PARAM
(2) 013040 000000 0
(2) 013042 177777 177777
(2) 013044 000176 SWREG
(2) 013046 000 001 .BYTE 0,1
(2) 013050 005037 013056 OUT: CLR @#RDSW
(2) 013054 000207 RTS PC
(2) 013056 000000 RDSW: .WORD 0
(2) 013060 000001 SWREGL: 1
(2) 013062 006 002 .BYTE 6,2
(2) 013064 000176 SWREG
(1) 013066 000005 5

```

```

(1) 013070          $SCOP1
(2)
(2)                ;CHECK FOR FREEZE ON CURRENT DATA
(2)
(2) 013070 004767 177646          .SCOP1: JSR    PC,CKSWR
(2) 013074 032777 001000 166336  BIT    #SW09,@SWR
(2) 013102 001402                BEQ    1$
(2) 013104 016716 166000          MOV    LOCK,(SP)
(2) 013110 000002          1$:    RTI
(1) 013112          .STYPE
(2)                .SBTTL  TYPE ROUTINE
(2)
(2) 013112          STARS
(3)                .IF B
(3)                :*****
(3)                .IFF
(3)                .NLIST
(3)                .REPT
(3)                .LIST
(3)                :*****
(3)                .NLIST
(3)                .ENDR
(3)                .LIST
(3)                .ENDC
(2)                000
(2)                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(2)                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS FOR 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) 013112 105767 166341          $TYPE:  TSTB   $TPFLG      ;;IS THERE A TERMINAL?
(2) 013116 100002                BPL     1$          ;;BR IF YES
(2) 013120 000000                HALT                    ;;HALT HERE IF NO TERMINAL
(2) 013122 000430                BR      3$          ;;LEAVE
(2) 013124 010046          1$:    MOV    R0,-(SP)      ;;SAVE R0
(2) 013126 017600 000002          MOV    @2(SP),R0    ;;GET ADDRESS OF ASCIZ STRING
(2)                001
(2) 013132 122767 000001 166406  .IF DF $MAIL
(2) 013140 001011                CMPB   #APTENV,$ENV  ;;RUNNING IN APT MODE
(2) 013142 132767 000100 166377  BNE    62$          ;;NO,GO CHECK FOR APT CONSOLE
(2) 013150 001405                BITB   #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(2) 013152 010067 000004          BEQ    62$          ;;NO,GO CHECK FOR CONSOLE
(2) 013156 004767 164624          MOV    R0,61$      ;;SETUP MESSAGE ADDRESS FOR APT
(2) 013162 000000          JSR    PC,$ATY3    ;;SPOOL MESSAGE TO APT
(2) 013164 132767 000040 166355 61$:   .WORD   0          ;;MESSAGE ADDRESS
(2) 013172 001003          62$:   BITB   #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(2)                000          BNE    60$          ;;YES,SKIP TYPE OUT
(2) 013174 112046          .ENDC
(2)                2$:   MOVB   (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK

```



```

(2) .IIF EQ .-$NULL,$NULL: .BYTE 0 ;;CONTAINS NULL CHARACTER FOR FILLS
(2) .IIF NDF $FILLS,$FILLS:
(2) .IIF EQ .-$FILLS,$FILLS: .BYTE 2 ;;CONTAINS # OF FILLER CHARACTER
(2) .IIF NDF $FILLC,$FILLC:
(2) .IIF EQ .-$FILLC,$FILLC: .BYTE 12 ;;INSERT FILL CHARS. AFTER A 'LI
(2) .IIF NDF $TPFLG,$TPFLG:
(2)
(2) .IIF EQ .-$TPFLG,$TPFLG: .BYTE 0 ;;"TERMINAL AVAILABLE" FLAG (BIT
(2) .IIF NDF $QUES,$QUES:
(2) .IIF EQ .-$QUES,$QUES: .ASCII "?" ;;QUESTION MARK
(2) .IIF NDF $CRLF,$CRLF:
(2) .IIF EQ .-$CRLF,$CRLF: .ASCII <15> ;;CARRIAGE RETURN
(2) .IIF NDF $LF,$LF:
(2) .IIF EQ .-$LF,$LF: .ASCII <12> ;;LINEFEED
(2) .IIF NE 18., .EVEN
(1) 013374 $INSTR

```

;ASCII STRING INPUT ROUTINE

```

(2) 013374 017667 000000 000014 .INSTR: MOV @($P),.MSG ;PICK UP MESSAGE
(2) 013402 062716 000002 ;ADD #2,($P) ;JUMP AROUND MESSAGE FOR RTI
(2) 013406 105767 166134 ;TSTB $ENV ;APT CONTROL
(2) 013412 001036 ;BNE INSTR2 ;YES NO TYPE
(2) 013414 104401 .INST1: TYPE
(2) 013416 000000 .MSG: 0
(2) 013420 012704 016172 ;MOV #INBUF,R4 ;GET STARTING LOC OF INBUF
(2) 013424 012703 000007 ;MOV #7,R3 ;MAX # OF CHARS
(2) 013430 105777 166015 1$: ;TSTB @$TKS ;TTY FLAG
(2) 013434 100375 ;BPL 1$
(2) 013436 117714 166004 ;MOVB @$TKB,(R4) ;TAKE CHAR
(2) 013442 142714 000200 ;BICB #200,(R4) ;STRIP
(2) 013446 121427 000025 ;CMPB (R4),#25 ;IS IT <^G>
(2) 013452 001760 ;BEQ .INST1
(2) 013454 122427 000015 ;CMPB (R4)+,#15 ;CHECK FOR CR
(2) 013460 001413 ;BEQ INSTR2
(2) 013462 105777 165762 2$: ;TSTB @$TPS ;TEST FLAG
(2) 013466 100375 ;BPL 2$
(2) 013470 117777 165752 165754 ;MOVB @$TKB,$STPB ;ECHO CHARACTER
(2) 013476 005303 ;DEC R3 ;DID YOU TYPE TOO MANY CHARS ?
(2) 013500 001353 ;BNE 1$
(2) 013502 104401 .INSTE: TYPE
(2) 013504 015457 ;MOVM :?
(2) 013506 000742 ;BR .INST1 ;RETRY
(2) 013510 000002 INSTR2: RTI
(1) 013512 $PARAM

```

;CONVERT ASCII STRING TO OCTAL

```

(2) 013512 011605 .PARAM: MOV ($P),R5 ;PUT CONTENTS OF SP INTO R5
(2) 013514 012567 000162 ;MOV (R5)+,LOLIM ;PUT LOW LIMIT INTO LOLIM
(2) 013520 012567 000160 ;MOV (R5)+,HILIM ;PUT HIGH LIMIT INTO HILIM
(2) 013524 012567 000156 ;MOV (R5)+,DEVADR ;PUT STORE LOC INTO DEVADR
(2) 013530 112567 000154 ;MOVB (R5)+,LOBITS ;PUT MASK INTO LOBITS
(2) 013534 112567 000151 ;MOVB (R5)+,ADRCNT ;PUT COUNT INTO ADRCNT
(2) 013540 010516 ;MOV R5,($P) ;RESTORE RETURN ADDR ON STACK FOR RTI

```

```

(2) 013542 005005          PARAM1: CLR      R5
(2) 013544 012704 016172  MOV     #INBUF,R4
(2) 013550 122714 000015  CMPB   #15,(R4)          ;CR ?
(2) 013554 001420          BEQ    PARERR ;YOU TYPED CR TOO SOON !
(2) 013556 121427 000060 1$:    CMPB   (R4),#60      ;LOW LIMIT ASCII 0
(2) 013562 002415          BLT    PARERR
(2) 013564 121427 000067  CMPB   (R4),#67          ;HIGH LIMIT ASCII 7
(2) 013570 003012          BGT    PARERR
(2) 013572 142714 000060  BICB   #60,(R4)          ;CONVERT TO OCTAL
(2) 013576 152405          BISB   (R4)+,R5          ;STORE AWAY ITS AN OK CHAR
(2) 013600 122714 000015  CMPB   #15,(R4)          ;CR ?
(2) 013604 001414          BEQ    LIMITS ;NOW CHECK FOR HIGH & LOW LIMIT CONDS
(2) 013606 006305          ASL    R5              ;ALLOCATE ROOM FOR NEXT CHAR
(2) 013610 006305          ASL    R5
(2) 013612 006305          ASL    R5
(2) 013614 000760          BR     1$
(2) 013616 122714 000015  PARERR: CMPB  #15,(R4)      ;CR?
(2) 013622 001003          BNE    120$
(2) 013624 005737 013056  TST    @#RDSW          ;CK SWR USED
(2) 013630 001023          BNE    PARTI
(2) 013632 104407          120$: INSTER ;RETRY
(2) 013634 000742          BR     PARAM1

;TEST TO SEE IF NUMBER IS WITHIN LIMITS

(2) 013636 020567 000042  LIMITS: CMP    R5,HILIM
(2) 013642 101365          BHI    PARERR ;THE # IS TOO HIGH
(2) 013644 020567 000032  CMP    R5,LOLIM
(2) 013650 103762          BLO    PARERR ;THE # IS TOO LOW
(2) 013652 136705 000032  BITB   LOBITS,R5        ;TEST BY MASKINGTHE #
(2) 013656 001357          BNE    PARERR

;STORE NUMBER AT SPECIFIED ADDRESS

(2) 013660 016704 000022  1$:    MOV    DEVADR,R4        ;GET STARTING ADDR OF
(2) 013664 010524          MOV    R5,(R4)+        ;STORE AT THIS ADDR
(2) 013666 062705 000002  ADD    #2,R5
(2) 013672 105367 000013  DECB   ADRCNT ;HOW MANY TIMES + 2 ?
(2) 013676 001372          BNE    1$
(2) 013700 000002          PARTI: RTI
(2) 013702 000000          LOLIM: 0
(2) 013704 000000          HILIM: 0
(2) 013706 000000          DEVADR: 0
(2) 013710 000000          LOBITS: 0
(2) 013712 013711          ADRCNT=LOBITS+1
(1) 013712          $REG

;SAVE PC OF TEST THAT FAILED AND R0-R5
(2) 013712 016667 000004 165206 .SAV05: MOV    4(SP),SAVPC
(2) 013720 010567 165550          ;SAVE R0-R5
(2) 013724 010467 165542          SV05:  MOV    R5,$REG5
(2) 013730 010367 165534          MOV    R4,$REG4
          MOV    R3,$REG3

```

```

(2) 013734 010267 165526      MOV      R2,$REG2
(2) 013740 010167 165520      MOV      R1,$REG1
(2) 013744 010067 165512      MOV      R0,$REG0
(2) 013750 000002      RTI

(2)
(2)
(2)
(2)
(2)
(2)
(2) 013752 016700 165504      .RES05: MOV      $REG0,R0
(2) 013756 016701 165502      MOV      $REG1,R1
(2) 013762 016702 165500      MOV      $REG2,R2
(2) 013766 016703 165476      MOV      $REG3,R3
(2) 013772 016704 165474      MOV      $REG4,R4
(2) 013776 016705 165472      MOV      $REG5,R5
(2) 014002 000002      RTI
(1) 014004      $CONVRT

(2)
(2)
(2)
(2)
(2) 014004 104401      .CONVR: TYPE
(2) 014006 015463      MCRLF      ;CR LF
(2) 014010 017601 000000      MOV      @($P),R1      ;PICK UP DATA POINTER
(2) 014014 062716 000002      ADD      #2,$P      ;SET UP $P FOR RTI
(2) 014020 012167 000130      MOV      (R1)+,WRDCNT      ;PICK UP # OF WORDS FROM TABLE
(2) 014024 112167 000126      1$:      MOV      (R1)+,CHRCNT      ;PICK UP # OF CHARS FROM TABLE
(2) 014030 112167 000123      MOV      (R1)+,SPACNT      ;PICK UP # OF SPACES FROM TABLE
(2) 014034 013167 000120      MOV      @($R1)+,BINWRD      ;PICK UP ADDRESS OF MSG
(2)
(2)
(2) 014040 016704 000114      2$:      MOV      BINWRD,R4      ;FROM TABLE
(2) 014044 116705 000106      MOV      CHRCNT,R5      ;SAVE
(2) 014050 012700 016234      MOV      #TEMP,R0      ;STARTING ADDRESS OF TEMP BLOCK
(2) 014054 010403      3$:      MOV      R4,R3      ;SAVE
(2) 014056 042703 177770      BIC      #177770,R3      ;CLR OUT UPPER BITS .. SAVE CHAR
(2) 014062 062703 000260      ADD      #260,R3      ;CONVERT TO ASCII
(2) 014066 110320      MOV      R3,(R0)+      ;STORE AWAY
(2) 014070 006204      ASR      R4      ;SHIFT FOR NEXT #
(2) 014072 006204      ASR      R4      ;DITTO
(2) 014074 006204      ASR      R4      ;DITTO
(2) 014076 005305      DEC      R5      ;DEC CHAR COUNT
(2) 014100 001365      BNE      3$      ;DO IT AGAIN ?
(2) 014102 012703 016276      MOV      #MDATA,R3      ;STARTING ADDRESS OF MDATA BLOCK
(2) 014106 114023      4$:      MOV      -(R0),(R3)+      ;REVERSE THE ORDER OF NUMBERS
(2) 014110 105367 000042      DECB     CHRCNT      ;DEC CHAR COUNT
(2) 014114 001374      BNE      4$      ;DO IT AGAIN ?
(2) 014116 105767 000035      TSTB     SPACNT      ;HOW MANY SPACES ?
(2) 014122 001405      BEQ      6$      ;TYPE # IF BR =0
(2) 014124 112723 000240      5$:      MOV      #240,(R3)+      ;"SPACE" IN ASCII
(2) 014130 105367 000023      DECB     SPACNT      ;DEC # OF SPACE COUNT
(2) 014134 001373      BNE      5$      ;DO IT AGAIN ?
(2) 014136 105013      6$:      CLRB     (R3)      ;INSERT '0' FOR TTY OUTPUT ROUTINE
(2) 014140 104401      TYPE
(2) 014142 016276      MDATA      ;THIS MESSAGE
(2) 014144 005367 000004      DEC      WRDCNT      ;HOW MANY #'S ?
(2) 014150 001325      BNE      1$      ;DO THIS ROUTINE AGAIN IF NOT EQUAL TO 0
(2) 014152 000002      RTI      ;RETURN TO PROGRAM
(2) 014154 000000      WRDCNT: 0
(2) 014156 000000      CHRCNT: 0

```



```

(2)          014157          SPACNT=CHRCNT+1
(2) 014160 000000          BINWRD: 0
(1) 014162          $$SETFLG
(2)
(2)          ;COMPARE THE FIRST CHARACTER IN THE TELETYPE INPUT
(2)          ;BUFFER TO THE CHARACTERS 'N' AND 'Y'.
(2)          ;IF THE CHARACTER IS 'N' CLEAR THE FLAG
(2)          ;IF THE CHARACTER IS 'Y' SET THE FLAG
(2)
(2) 014162 017605 000000          .SETFLG:MOV @ (SP),R5
(2) 014166 122767 000116 001776          CMPB #'N',INBUF ;IS IT 'N' ?
(2) 014174 001002          BNE 1$
(2) 014176 105015          CLRB (R5) ;000
(2) 014200 000406          BR 2$
(2) 014202 122767 000131 001762 1$: CMPB #'Y',INBUF ;IS IT 'Y' ?
(2) 014210 001005          BNE 3$
(2) 014212 112715 177777          MOVB #-1,(R5) ;377
(2) 014216 062716 000002          2$: ADD #2,(SP)
(2) 014222 000002          RTI
(2) 014224 104407          3$: INSTER ;RETRY
(2) 014226 000755          BR .SETFLG
(1) 014230          $.ERROR SAVIT
(2)          $.SBTTL ERROR HANDLER ROUTINE
(2)
(2) 014230          STARS
(3)          .IF B
(3)          ;*****
(3)          .IFF
(3)          .NLIST
(3)          .REPT
(3)          .LIST
(3)          ;*****
(3)          .NLIST
(3)          .ENDR
(3)          .LIST
(3)          .ENDC
(2)          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(2)          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(2)          .IF B SAVIT
(2)          .IFF NE 20000&$$SWR,;*AND TYPE OUT THE PC OF THE ERROR INSTRUCTION
(2)          .IFF
(2)          ;*AND GO TO SAVIT ON ERROR
(2)          .ENDC
(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) 014230          $.ERROR:
(2)          .IFF NE $SETUP&100, CKSWR ;;TEST FOR CHANGE IN SOFT-SW?
(2)          .IF NB <>
(2)          .IRP NEWINS,<>
(2)          NEWINS

```

```

(2) .ENDM
(2) .ENDC
(2) 014230 105267 165147 7$: INCB $ERFLG ;;SET THE ERROR FLAG
(2) 014234 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
(2) 014236 016777 165140 165176 MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(2) 001 .IF EQ 2000&$$SWR ;SW10
(2) INC $ERTTL ;;INC THE ERROR COUNT
(2) .IF+
(2) 014244 032777 002000 165166 BIT #BIT10,@SWR ;;BELL ON ERROR?
(2) 014252 001402 BEQ 1$ ;;NO - SKIP
(2) 014254 104401 001516 TYPE $BELL ;;RING BELL
(2) 014260 005267 165126 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
(2) .ENDC
(2) 014264 011667 165126 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(2) 014270 162767 000002 165120 SUB #2,$ERRPC
(2) 014276 117767 165114 165110 MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(2) 001 .IF NE 20000&$$SWR ;SW13
(2) 014304 032777 020000 165126 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
(2) 014312 001004 BNE 20$ ;;SKIP TYPEOUTS
(2) .IFTF
(2) 002 .IF DF $40CAT
(2) CMP (SP),#1002 ;;IF RETURN PC LESS THAN 1002
(2) BHI 12$ ;;ERROR IS ILLEGAL TRAP
(2) ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
(2) MOV 4(SP),$ERRPC ;;GET PC AT TIME OF FALSE TRAP
(2) SUB #2,$ERRPC ;;ADJUST PC
(2) TYPE 10$ ;;TYPE HEADER
(2) TYPOCT $ERRPC ;;TYPE PC AT TIME OF ERROR
(2) TYPE 11$
(2) SUB #4,(SP) ;;GET FALSE TRAP VECTOR ADDR
(2) MOV (SP),$ERRPC
(2) TYPOCT $ERRPC
(2) TYPE $CRLF
(2) CMP (SP)+,(SP)+ ;;POP FALSE TRAP VECTOR PC&ADDR
(2) BR 20$
(2) 10$: .ASCIZ <200>'PC= '
(2) 11$: .ASCIZ ' UNEXPECTED TRAP TO '
(2) .EVEN
(2) 12$:
(2) .ENDC
(2) 001 .IF NB SAVIT
(2) 002 JSR PC,SAVIT ;;GO TO USER ERROR ROUTINE
(2) 014314 004767 000072 001 .ENDC
(2) .IFT
(2) 002 .IF B SAVIT
(2) TYPE $CRLF
(2) TYPOCT $ERRPC,<ERROR ADDRESS>
(2) .ENDC
(2) TYPE $CRLF
(2) .ENDC
(2) 014320 104401 001523 000
(2) 014324 001
(2) 014324 122767 000001 165214 .IF DF $MAIL
(2) 014332 001007 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(2) 014334 116767 165054 000004 BNE 2$ ;;NO SKIP APT ERROR REPORT
(2) 014342 004767 163450 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(2) JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT

```

```

(2) 014346 000 21$: .BYTE 0
(2) 014347 000 .BYTE 0
(2) 014350 000,77 22$: BR 22$ ;;APT ERROR LOOP
(2) 000 .ENDC
(2) 001 .IF NE $SWR&100000 ;SW15
(2) 014352 005777 1650b2 2$: TST @SWR ;;HALT ON ERROR
(2) 014356 100001 BPL 3$ ;;SKIP IF CONTINUE
(2) 014360 000000 HALT ;;HALT ON ERROR!
(2) .IIF NE $SETUP&100, CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(2) 002 .IF NE $SWR&1000 ;SW09
(2) 014362 032777 001000 165050 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(2) 014370 001402 BEQ 4$ ;;BR IF NO
(2) 014372 016716 165012 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(2) 014376 005767 165112 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(2) 014402 001402 BEQ 5$ ;;BR IF NONE
(2) 014404 016716 165104 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(2) 014410 5$:
(2) .IFF
(2) 3$:
(2) .ENDC
(2) .IFF
(2) 001 .IF NE $SWR&1000 ;SW09
(2) 002 2$: BIT #BIT09,@SWR ;;LOOP ON ERROR?
(2) BEQ 3$ ;;BR IF NO
(2) MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(2) 3$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(2) BEQ 4$ ;;BR IF NONE
(2) MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(2) 4$:
(2) .IFF
(2) 2$:
(2) .ENDC
(2) 001 .ENDC
(2) 000 .ENDC
(2) 001 .IF NE $SETUP&20 ;BIT04--(.$EOP)
(2) CMP #SENDAD,@#42 . ;;ACT-11 AUTO-ACCEPT?
(2) BNE 6$ ;;BRANCH IF NO
(2) HALT . ;;YES
(2) 6$:
(2) .ENDC
(2) 000 .IF B <>
(2) 001 RTI ;;RETURN
(2) 014410 000002 .IFF
(2) .IRP LASTIN,<>
(2) LASTIN
(2) .ENDM
(2) .ENDC
(2) 000 .IIF NDF $ERTTL,$ERTTL:
(2) .IIF EQ .-$ERTTL,$ERTTL: .WORD 0 ;;ERROR COUNT
(2) .IIF NDF $STSTM,$STSTM:
(2) .IIF EQ .-$STSTM,$STSTM: .BYTE 0 ;;TEST NUMBER
(2) .IIF NDF $ERFLG,$ERFLG:
(2) .IIF EQ .-$ERFLG,$ERFLG: .BYTE 0 ;;ERROR FLAG
(2) .IIF NE .&1, .EVEN
(2) 001 .IF NE $SWR&1000;SW09
(2) .IIF NDF $ESCAPE,$ESCAPE:
(2) .IIF EQ .-$ESCAPE,$ESCAPE: .WORD 0 ;;ESCAPE ON ERROR ADDRESS:

```

07

```

(2)          000      .ENDC
(2)          (2)      .IIF NDF $ERRPC,$ERRPC:
(2)          (2)      .IIF EQ .-$ERRPC,$ERRPC:          .WORD 0          ;;LAST ERROR INSTRUCTION EXECUTE
(2)          001      .IF NE $$SWR2000;SW10
(2)          (2)      .IIF NDF $BELL,$BELL:
(2)          (2)      .IIF EQ .-$BELL,$BELL:          .ASCIZ <207><377><377> ;;ASCII CODE FOR BELL
(2)          000      .ENDC
(2)          (2)      .IIF NDF $ITEMB,$ITEMB:
(2)          (2)      .IIF EQ .-$ITEMB,$ITEMB:          .WORD 0          ;;ITEM BYTE
(2)          (2)      .IIF NDF $QUES,$QUES:
(2)          (2)      .IIF EQ .-$QUES,$QUES:          .ASCII "?"          ;;QUESTION MARK
(2)          (2)      .IIF NDF $CRLF,$CRLF:
(2)          (2)      .IIF EQ .-$CRLF,$CRLF:          .ASCII <15>          ;;CARRAIGE RETURN
(2)          (2)      .IIF NDF $LF,$LF:
(2)          (2)      .IIF EQ .-$LF,$LF:          .ASCIZ <12>          ;;LINEFEED
(2)          (2)      .IIF NE 18.,          .EVEN
(1) 014412          $NEED
(2) 014412 010067 164512 SAVIT: MOV R0,HLDO
(2) 014416 010167 164510      MOV R1,HLD1
(2) 014422 010267 164506      MOV R2,HLD2
(2) 014426 010367 164504      MOV R3,HLD3
(2) 014432 010467 164502      MOV R4,HLD4
(2) 014436 010567 164500      MOV R5,HLD5
(2) 014442 016767 164734 164474      MOV $TSTNM,HLD6
(2) 014450          .SERRTYP
(3)          .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
(3) 014450          STARS
(4)          001      .IF B
(4)          (4)      :*****
(4)          (4)      .IFF
(4)          (4)      .NLIST
(4)          (4)      .REPT
(4)          (4)      .LIST
(4)          (4)      :*****
(4)          (4)      .NLIST
(4)          (4)      .ENDR
(4)          (4)      .LIST
(4)          000      .ENDC
(3)          (3)      :*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(3)          (3)      :*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(3)          (3)      :*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(3) 014450          $SERRTYP:
(3) 014450 104401 001523      TYPE          $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
(3) 014454 010046          MOV R0,-(SP)          ;; SAVE R0
(3) 014456 005000          CLR R0          ;; PICKUP THE ITEM INDEX
(3) 014460 153700 001414      BISB @#$ITEMB,R0
(3) 014464 001004          BNE 1$          ;; IF ITEM NUMBER IS ZERO, JUST
(3)          (3)          ;; TYPE THE PC OF THE ERROR
(3) 014466          TYPOCT $ERRPC,<ERROR ADDRESS>
(4) 014464 016746 164724      MOV $ERRPC,-(SP)          ;; SAVE $ERRPC FOR TYPEOUT
(4)          (4)          ;; ERROR ADDRESS
(4) 014472 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(3)          001      .IF B

```

```

(3) 014474 000426          BR      6$      ::GET OUT
(3)                      .IFF
(3)                      BR      10$     ::GET OUT
(3)                      .ENDC
(3) 014476 005300          1$: DEC   RO      ::ADJUST THE INDEX SO THAT IT WILL
(3) 014500 006300          ASL   RO      ::      WORK FOR THE ERROR TABLE
(3) 014502 006300          ASL   RO
(3) 014504 006300          ASL   RO
(3) 014506 062700 001652  ADD   #ERRTB,RO  ::FORM TABLE POINTER
(3) 014512 012067 000004  MOV   (RO)+,2$  ::PICKUP "ERROR MESSAGE" POINTER
(3) 014516 001404          BEQ   3$      ::SKIP TYPEOUT IF NO POINTER
(3) 014520 104401          TYPE  ::TYPE THE "ERROR MESSAGE"
(3) 014522 000000          2$: .WORD 0      ::"ERROR MESSAGE" POINTER GOES HERE
(3) 014524 104401 001523  TYPE  , $CRLF   ::"CARRIAGE RETURN" & "LINE FEED"
(3) 014530 012067 000004  3$: MOV   (RO)+,4$  ::PICKUP "DATA HEADER" POINTER
(3) 014534 001404          BEQ   5$      ::SKIP TYPEOUT IF 0
(3) 014536 104401          TYPE  ::TYPE THE "DATA HEADER"
(3) 014540 000000          4$: .WORD 0      ::"DATA HEADER" POINTER GOES HEPE
(3) 014542 104401 001523  TYPE  , $CRLF   ::"CARRIAGE RETURN" & "LINE FEED"
(3) 014546 011000          .IF B
(3) 014550 001004          5$: MOV   (RO),RO  ::PICKUP "DATA TABLE" POINTER
(3) 014552 012600          BNE   7$      ::GO TYPE THE DATA
(3)                      6$: MOV   (SP)+,RO  ::RESTORE RO
(3) 014554 104401 001523  TYPE  , $CRLF   ::"CARRIAGE RETURN" & "LINE FEED"
(3) 014560 000207          RTS   PC      ::RETURN
(3) 014562
(4) 014562 013046          7$: MOV   @ (RO)+, -(SP)  ::SAVE @ (RO)+ FOR TYPEOUT
(4)                      .IIF NB <>
(4) 014564 104402          TYPOC  ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(3) 014566 005710          TST   (RO)    ::IS THERE ANOTHER NUMBER?
(3) 014570 001770          BEQ   6$      ::BR IF NO
(3) 014572 104401 014600  TYPE  , 8$     ::TYPE TWO(2) SPACES
(3) 014576 000771          BR    7$      ::LOOP
(3) 014600 020040 000      8$: .ASCIZ / /      ::TWO(2) SPACES
(3) 014604 .EVEN
(3)                      .IFF
(3) 5$: MOV   R1, -(SP)  ::SAVE R1
(3)                      MOV   (RO)+, R1  ::PICKUP "DATA TABLE" POINTER
(3)                      BEQ   9$      ::BR IF NO DATA TO BE TYPED
(3)                      MOV   (RO)+, RO  ::PICKUP "DATA FORMAT" POINTER
(3) 6$: TSTB  (RO)+    ::"OCTAL" OR "DECIMAL"
(3)                      BNE   7$      ::BR IF DECIMAL
(3)                      TYPOCT @ (R1)+  ::TYPE AN OCTAL NUMBER
(3)                      BR    8$
(3) 7$: TYPDEC @ (R1)+  ::TYPE DECIMAL NUMBER
(3) 8$: TST   (R1)    ::IS THERE ANOTHER NUMBER?
(3)                      BEQ   9$      ::BR IF NO
(3)                      TYPE  , 11$    ::TYPE TWO(2) SPACES
(3)                      BR    6$      ::LOOP
(3) 9$: MOV   (SP)+, R1  ::RESTORE R1
(3) 10$: MOV  (SP)+, RO  ::RESTORE RO
(3)                      TYPE  , $CRLF  ::"CARRIAGE RETURN" & "LINE FEED"
(3)                      RTS   PC      ::RETURN
(3) 11$: .ASCIZ / /      ::TWO(2) SPACES

```

ERROR MESSAGE TYPEOUT ROUTINE

```

(3)                                .EVEN
(3)                                .ENDC
(1) 014604 000                    .STYPOCT
(2)                                .SBITL  BINARY TO OCTAL (ASCII) AND TYPE
(2) 014604 001                    STARS
(3)                                .IF B
(3)                                .....
(3)                                .IFF
(3)                                .NLIST
(3)                                .REPT
(3)                                .LIST
(3)                                .....
(3)                                .NLIST
(3)                                .ENDR
(3)                                .LIST
(3) 000                            .ENDC
(2)                                *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(2)                                *OCTAL (ASCII) NUMBER AND TYPE IT.
(2)                                *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(2)                                *CALL:
(2)                                *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(2)                                *      TYPOS    ;;CALL FOR TYPEOUT
(2)                                *      .BYTE   N            ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(2)                                *      .BYTE   M            ;;M=1 OR 0
(2)                                *                                ;;1=TYPE LEADING ZEROS
(2)                                *                                ;;0=SUPPRESS LEADING ZEROS
(2)                                *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(2)                                *$TYPOS OR $TYPOC
(2)                                *CALL:
(2)                                *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(2)                                *      TYPON    ;;CALL FOR TYPEOUT
(2)                                *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(2)                                *CALL:
(2)                                *      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(2)                                *      TYPOC    ;;CALL FOR TYPEOUT
(2) 014604 017646 000000           $TYPOS: MOV      @(SP),-(SP)      ;;PICKUP THE MODE
(2) 014610 116667 000001 000211   MOVVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(2) 014616 112667 000207           MOVVB    (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
(2) 014622 062716 000002           ADD      #2,(SP)       ;;ADJUST RETURN ADDRESS
(2) 014626 000406                   BR      $TYPON
(2) 014630 112767 000001 000171   $TYPOC: MOVVB    #1,$OFILL    ;;SET THE ZERO FILL SWITCH
(2) 014636 112767 000006 000165   MOVVB    #6,$OMODE+1    ;;SET FOR SIX(6) DIGITS
(2) 014644 112767 000005 000154   $TYPON: MOVVB    #5,$OCNT    ;;SET THE ITERATION COUNT
(2) 014652 010346                   MOV      R3,-(SP)      ;;SAVE R3
(2) 014654 010446                   MOV      R4,-(SP)      ;;SAVE R4
(2) 014656 010546                   MOV      R5,-(SP)      ;;SAVE R5
(2) 014660 116704 000145           MOVVB    $OMODE+1,R4   ;;GET THE NUMBER OF DIGITS TO TYPE
(2) 014664 005404                   NEG      R4
(2) 014666 062704 000006           ADD      #6,R4         ;;SUBTRACT IT FOR MAX. ALLOWED
(2) 014672 110467 000132           MOVVB    R4,$OMODE     ;;SAVE IT FOR USE
(2) 014676 116704 000125           MOVVB    $OFILL,R4    ;;GET THE ZERO FILL SWITCH
(2) 014702 016605 000012           MOV      12(SP),R5    ;;PICKUP THE INPUT NUMBER

```

```

(2) 014706 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
(2) 014710 006105          1$:    ROL      R5          ;;ROTATE MSB INTO 'C'
(2) 014712 000404          BR       3$          ;;GO DO MSB
(2) 014714 006105          2$:    ROL      R5          ;;FORM THIS DIGIT
(2) 014716 006105          ROL      R5
(2) 014720 006105          ROL      R5
(2) 014722 010503          MOV      R5,R3
(2) 014724 006103          3$:    ROL      R3          ;;GET LSB OF THIS DIGIT
(2) 014726 105367 000076  DECB    $OMODE      ;;TYPE THIS DIGIT?
(2) 014732 100016          BPL     7$          ;;BR IF NO
(2) 014734 042703 177770  BIC     #177770,R3  ;;GET RID OF JUNK
(2) 014740 001002          BNE     4$          ;;TEST FOR 0
(2) 014742 005704          TST     R4          ;;SUPPRESS THIS 0?
(2) 014744 001403          BEQ     5$          ;;BR IF YES
(2) 014746 005204          4$:    INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S
(2) 014750 052703 000060  BIS     #'0,R3      ;;MAKE THIS DIGIT ASCII
(2) 014754 052703 000040  5$:    BIS     #' ,R3  ;;MAKE ASCII IF NOT ALREADY
(2) 014760 110367 000040  MOVVB   R3,8$       ;;SAVE FOR TYPING
(2) 014764 104401 015024  TYPE    8$          ;;GO TYPE THIS DIGIT
(2) 014770 105367 000032  7$:    DECB    $OCNT   ;;COUNT BY 1
(2) 014774 003347          BGT     2$          ;;BR IF MORE TO DO
(2) 014776 002402          BLT     6$          ;;BR IF DONE
(2) 015000 005204          INC     R4          ;;INSURE LAST DIGIT ISN'T A BLANK
(2) 015002 000744          BR      2$          ;;GO DO THE LAST DIGIT
(2) 015004 012605          6$:    MOV     (SP)+,R5  ;;RESTORE R5
(2) 015006 012604          MOV     (SP)+,R4      ;;RESTORE R4
(2) 015010 012603          MOV     (SP)+,R3      ;;RESTORE R3
(2) 015012 016666 000002 000004  MOV     2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
(2) 015020 012616          MOV     (SP)+,(SP)
(2) 015022 000002          RTI                    ;;RETURN
(2) 015024 000          8$:    .BYTE   0          ;;STORAGE FOR ASCII DIGIT
(2) 015025 000          .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
(2) 015026 000          $OCNT:  .BYTE   0          ;;OCTAL DIGIT COUNTER
(2) 015027 000          $OFILL: .BYTE   0          ;;ZERO FILL SWITCH
(2) 015030 000000          $OMODE: .WORD   0          ;;NUMBER OF DIGITS TO TYPE
(1) 015032          $PFAIL

(2)          ;ENTER HERE ON POWER FAILURE

(2)
(2)
(2) 015032          $PWRDN:
(2) 015032 010046          .PFAIL: MOV     R0,-(SP)          ;SAVE R0-R5 ON PROCESSOR STACK
(2) 015034 010146          MOV     R1,-(SP)
(2) 015036 010246          MOV     R2,-(SP)
(2) 015040 010346          MOV     R3,-(SP)
(2) 015042 010446          MOV     R4,-(SP)
(2) 015044 010546          MOV     R5,-(SP)
(2) 015046 016746 162752  MOV     24,-(SP)
(2) 015052 010667 164040  MOV     SP,SAVSP      ;SAVE STACK POINTER
(2) 015056 012767 015070 162740  MOV     #RESTART,24  ;SET UP FOR POWER UP TRAP
(2) 015064 000000          HALT                    ;HALT ON POWER DOWN NORMAL
(2) 015066 000777          BR      .

(2)          ;PROCESSOR WILL TRAP HERE WHEN POWER IS RESTORED
(2)
(2) 015070 016706 164022  RESTAR: MOV     SAVSP,SP      ;RESTORE STACK POINTER
(2) 015074 012605          MOV     (SP)+,R5      ;RESTORE R0-R5

```

```

(2) 015076 012604      MOV      (SP)+,R4
(2) 015100 012603      MOV      (SP)+,R3
(2) 015102 012602      MOV      (SP)+,R2
(2) 015104 012601      MOV      (SP)+,R1
(2) 015106 012600      MOV      (SP)+,R0
(2) 015110 012767 015032 162706  MOV      #,PFAIL,24      ;SET UP FOR POWER FAILURE
(2) 015116 106427 000300      MTPS     #300
(2) 015122 012706 001100      MOV      #STACK,SP
(2) 015126 005067 001102      CLR      TEMP
(2) 015132 005267 001076      INC      TEMP
(2) 015136 001375      BNE      .-4
(2) 015140 104413      CONVRT
(2) 015142 015164      PFTAB
(2) 015144 104401      TYPE
(2) 015146 015466      MPFAIL
(2) 015150 005067 164227      CLR      $ERFLG
(2) 015154 005067 164236      CLR      $ERRPC
(2) 015160 000177 163720      JMP      @RETURN
(2) 015164 000001      PFTAB: 1
(2) 015166 006 002      .BYTE 6,2
(2) 015170 000207      RETURN
(1) 015172      $MSC <DUV11 CNDUU-A TAPE E>,<END OF PASS TAPE E>
(2) 015172 005015 042012 053125  MTITLE: .ASCIZ <15><12><12>/DUV11 CNDUU-A TAPE E /<15><12>
(2) 015200 030461 041440 042116
(2) 015206 052525 040455 052040
(2) 015214 050101 020105 020105
(2) 015222 005015 000
(2) 015225 015 053012 041505  MVECTO: .ASCIZ <15><12>/VEC ADD-/
(2) 015232 040440 042104 000055
(2) 015240 005015 051461 020124  MREGAD: .ASCIZ <15><12>/1ST DEV: REC CSR ADD-/
(2) 015246 042504 035126 051040
(2) 015254 041505 041440 051123
(2) 015262 040440 042104 000055
(2) 015270 005015 052515 052114  MMULT: .ASCIZ <15><12>/MULT DEV ? (Y OR N)-/
(2) 015276 042040 053105 037440
(2) 015304 024040 020131 051117
(2) 015312 047040 026451 000
(2) 015317 015 046012 051501  MLASTD: .ASCIZ <15><12>/LAST DEV: REC CSR ADDR-/
(2) 015324 020124 042504 035126
(2) 015332 051040 041505 041440
(2) 015340 051123 040440 042104
(2) 015346 026522 000
(2) 015351 075 042504 044526  DEVICE: .ASCIZ /=DEVICE /
(2) 015356 042503 020040 000
(2) 015363 015 051412 046105  MCOV: .ASCIZ <15><12>/SELECT TO RUN @ACTREG/
(2) 015370 041505 020124 047524
(2) 015376 051040 047123 040040
(2) 015404 041501 051124 043505
(2) 015412 000
(2) 015413 015 047412 043126  MRANGE: .ASCIZ <15><12>/OVFLO:RETYPE LAST DEV RXCSR ADDS-/
(2) 015420 047514 051072 052105
(2) 015426 050131 020105 040514
(2) 015434 052123 042040 053105
(2) 015442 051040 041530 051123
(2) 015450 040440 042104 026523
(2) 015456 000

```


BINARY TO OCTAL (ASCII) AND TYPE

| | | | | | |
|-----|--------|--------|--------|--------|--|
| (2) | 015457 | 040 | 037440 | 000 | MQM: .ASCIIZ / ?/ |
| (2) | 015463 | 015 | 000012 | | MCRLF: .ASCIIZ <15><12> |
| (2) | 015466 | 043120 | 044501 | 026114 | MPFAIL: .ASCIIZ /PFAIL, RESTART AT TEST IN PROGRESS/ |
| (2) | 015474 | 020040 | 04252 | 052123 | |
| (2) | 015502 | 051101 | 020124 | 052101 | |
| (2) | 015510 | 052040 | 051505 | 020124 | |
| (2) | 015516 | 047111 | 050040 | 047522 | |
| (2) | 015524 | 051107 | 051505 | 000123 | |
| (2) | 015532 | 005015 | 047105 | 020104 | MEPASS: .ASCIIZ <15><12>/END OF PASS TAPE E/ |
| (2) | 015540 | 043117 | 050040 | 051501 | |
| (2) | 015546 | 020123 | 040524 | 042520 | |
| (2) | 015554 | 042440 | 000 | | |
| (2) | 015557 | 015 | 051012 | 000 | MR: .ASCIIZ <15><12>/R/ |
| (2) | 015563 | 015 | 052012 | 051505 | MTSTPC: .ASCIIZ <15><12>/TEST PC-/ |
| (2) | 015570 | 020124 | 041520 | 000055 | |
| (2) | 015576 | 005015 | 047514 | 045503 | MLOCK: .ASCIIZ <15><12>/LOCK ON TEST? (Y OR N)-/ |
| (2) | 015604 | 047440 | 020116 | 052040 | |
| (2) | 015612 | 051505 | 037524 | 024040 | |
| (2) | 015620 | 020131 | 051117 | 047040 | |
| (2) | 015626 | 026451 | 000 | | |
| (2) | 015631 | 015 | 021412 | 047440 | MSYNC: .ASCIIZ <15><12>/# OF SYNC CHARS SELECTED (1 OR 2)-/ |
| (2) | 015636 | 020106 | 054523 | 041516 | |
| (2) | 015644 | 041440 | 040510 | 051522 | |
| (2) | 015652 | 051440 | 046105 | 041505 | |
| (2) | 015660 | 042524 | 020104 | 020050 | |
| (2) | 015666 | 020061 | 051117 | 031040 | |
| (2) | 015674 | 026451 | 000 | | |
| (2) | 015677 | 015 | 044412 | 020123 | MWIRE6: .ASCIIZ <15><12>/IS SEC XMIT SWITCH E55-2 IN? (Y OR N)-/ |
| (2) | 015704 | 042523 | 020103 | 046530 | |
| (2) | 015712 | 052111 | 051440 | 044527 | |
| (2) | 015720 | 041524 | 020110 | 032505 | |
| (2) | 015726 | 026465 | 020062 | 047111 | |
| (2) | 015734 | 020077 | 054450 | 047440 | |
| (2) | 015742 | 020122 | 024516 | 000055 | |
| (2) | 015750 | 005015 | 051511 | 051440 | MWIRE5: .ASCIIZ <15><12>/IS SEC REC SWITCH E55-3 IN? (Y OR N)-/ |
| (2) | 015756 | 041505 | 051040 | 041505 | |
| (2) | 015764 | 051440 | 044527 | 041524 | |
| (2) | 015772 | 020110 | 032505 | 026465 | |
| (2) | 016000 | 020063 | 047111 | 020077 | |
| (2) | 016006 | 054450 | 047440 | 020122 | |
| (2) | 016014 | 024516 | 000055 | | |
| (2) | 016020 | 005015 | 051511 | 047440 | MWIRE4: .ASCIIZ <15><12>/IS OPT CLR ENABLE SWITCH E55-1 IN? (Y OR N)-/ |
| (2) | 016026 | 052120 | 041440 | 051114 | |
| (2) | 016034 | 042440 | 040516 | 046102 | |
| (2) | 016042 | 020105 | 053523 | 052111 | |
| (2) | 016050 | 044103 | 042440 | 032465 | |
| (2) | 016056 | 030455 | 044440 | 037516 | |
| (2) | 016064 | 024040 | 020131 | 051117 | |
| (2) | 016072 | 047040 | 026451 | 000 | |
| (2) | 016077 | 015 | 005012 | 031510 | MEXTJ: .ASCIIZ <15><12><12>/H315 CONNECTOR ON ?(Y OR N)-/ |
| (2) | 016104 | 032461 | 041440 | 047117 | |
| (2) | 016112 | 042516 | 052103 | 051117 | |
| (2) | 016120 | 047440 | 020116 | 024077 | |
| (2) | 016126 | 020131 | 051117 | 047040 | |
| (2) | 016134 | 026451 | 000 | | |
| (2) | 016137 | 015 | 020012 | 043536 | MCNTG: .ASCIIZ <15><12>/ *G / |

```

(2) 016144 020040 000
(2) 016147 040 053523 036522 MMSWR: .ASCIZ / SWR= /
(2) 016154 020040 000040
(2) 016160 020040 047040 053505 MMNEW: .ASCIZ / NEW= /
(2) 016166 020075 000040
(2)
(1) 016172 .EVEN
(2) $BUFFER
(2)
(2) ;BUFFERS FOR INPUT-OUTPUT
(2)
(2) 016172 000000 INBUF: 0
(2) 016234 016234 .=.+40
(2) 016234 000000 TEMP: 0
(2) 016276 016276 .=.+40
(2) 016276 000000 MDATA: 0
(2) 016340 016340 .=.+40
(1) 016340 $SCOPE
(2) 016340 . $SCOPE 5,SC,,SC1
(3) .SBTTL SCOPE HANDLER ROUTINESTARTS
(3) ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(3) ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(3) .IF EQ $STNM+1-$ERFLG
(3) ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(3) .ENDC
(3) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(3) ;*SW14=1 LOOP ON TEST
(3) ;*SW11=1 INHIBIT ITERATIONS
(3) ;*SW09=1 LOOP ON ERROR
(3) 001 .IF NE 400&$SWR
(3) ;*SW08=1 LOOP ON TEST IN SW<7:0>
(3) .IIF EQ <$SWRMK&377>-200,;*SW08=1 LOOP ON TEST IN SWR<6:0>
(3) .IIF EQ <$SWRMK&377>-300,;*SW08=1 LOOP ON TEST IN SWR<5:0>
(3) .IIF EQ <$SWRMK&377>-340,;*SW08=1 LOOP ON TEST IN SWR<4:0>
(3) .IIF EQ <$SWRMK&377>-360,;*SW08=1 LOOP ON TEST IN SWR<3:0>
(3) .IIF EQ <$SWRMK&377>-370,;*SW08=1 LOOP ON TEST IN SWR<2:0>
(3) .IIF EQ <$SWRMK&377>-374,;*SW08=1 LOOP ON TEST IN SWR<1:0>
(3) .IIF EQ <$SWRMK&377>-376,;*SW08=1 LOOP ON TEST IN SWR<0>
(3) 000 .ENDC
(3) ;*CALL
(3) ;* SCOPE ;;SCOPE=IOT
(3)
(3) 016340 $SCOPE:
(3) .IIF NE $SETUP&100, CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(3) 001 .IF NB <SC>
(3) .IRP NEWINS,<SC>
(3) NEWINS
(4) .ENDM
(4) 016340 SC
(5)
(5) ;SCOPE LOOP AND INTERATION HANDLER
(5)
(5) 016340 .SCOPE:
(5) 016340 004767 174376 JSR PC,CKSWR
(5) 016344 005067 163046 CLR $ERRPC ;CLEAR LAST ERROR PC
(5) 016350 022716 003376 CMP #TST1+2,(SP) ;IS SCOPE AT BEGINING OF TEST 1?
(5) 016354 001422 BEQ $XTSTR ;YES NO LOOP.

```

0114

SEQ 0127

```

(5) 016356 032777 040000 163054 TTST: BIT #BIT14,@SWR ;THIS CODE IS FOR TESTING FOR BIT 14
(5) 016364 001412 BEQ 1$ ;ON LSI WHICH SYSMAC CANNOT HANDLE
(5) 016366 016767 163010 163012 MOV $TSTNM,$LPADR
(5) 016374 000406 BR 1$
(5) 016376 105777 163042 TSTB @$TKS ;KEYBOARD DONE?
(5) 016402 100123 BPL $OVER ;BR IF NO
(5) 016404 017766 163036 177776 MOV @$TKB,-2(SP) ;CLEAR DONE BIT
(3) 000 .ENDC
(3) 001 .IF NE $SWR&40000
(3) 002 .IF DF $40CAT
(3) ;GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002
(3) ;OTHERWISE CONTINUE
(3) CMP (SP),#1002 ;UNEXPECTED TRAP OR INTERRUPT?
(3) BHI 1$ ;ARE TRAPPED HERE VIA IOT
(3) JMP $ERROR ;GO PROCESS UNEXPECTED TRAP
(3) 001 .ENDC
(3) 016412 032777 040000 163020 1$: BIT #BIT14,@SWR ;LOOP ON PRESENT TEST?
(3) 016420 001114 BNE $OVER ;YES IF SW14=1
(3) 000 .ENDC
(3) ;*****START OF CODE FOR THE XOR TESTER*****
(3) 016422 000416 $XTSTR: BR 6$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
(3) ;THIS INSTRUCTION TO A "NOP" (NOP=240)
(3) 016424 013746 000004 MOV @#ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
(3) 016430 012737 016450 000004 MOV #5$,@#ERRVEC ;SET FOR TIMEOUT
(3) 016436 005737 177060 TST @#177060 ;TIME OUT ON XOR?
(3) 016442 012637 000004 MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
(3) 016446 000463 BR $SVLAD ;GO TO THE NEXT TEST
(3) 016450 022626 5$: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
(3) 016452 012637 000004 MOV (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
(3) 001 .IF NE $SWR&1000
(3) 016456 000423 BR 7$ ;LOOP ON THE PRESENT TEST
(3) .IFF
(3) BR $OVER ;LOOP ON THE PRESENT TEST
(3) .ENDC
(3) 016460 000 6$;*****END OF CODE FOR THE XOR TESTER*****
(3) 001 .IF NE $SWR&400
(3) 016460 032777 000400 162752 BIT #BIT08,@SWR ;LOOP ON SPEC. TEST?
(3) 002 .IF NE $SWR&5000:SW11 OR SW09
(3) 016466 001404 BEQ 2$ ;BR IF NO
(3) .IFF
(3) BEQ $SVLAD ;BR IF NO
(3) .ENDC
(3) 001 .IF NB
(3) 002 .IF DIF <SW08TBL>,<>
(3) 003 .ERROR ;ILLEGAL ARGUMENT IN .SCOPE -
(3) .IFF
(3) CLR -(SP) ;CLEAR A TEMP. LOCATION
(3) MOV @SWR,(SP) ;PICKUP THE DESIRED TEST NUMBER
(3) .IF NE $SWRMK, BIC #SWRMK,(SP) ;MASK OUT UNDESIRED BITS
(3) BEQ 8$ ;BRANCH IF BAD TEST NUMBER IN SWR
(3) .IRP X,<\$TN-1>
(3) CMP #'X',(SP) ;CHECK THE NU
(3) MBER IN THE SWR
(3) .ENDM
(3) BLT 8$ ;BRANCH IF TEST NUMBER IS OUT OF RANGE

```

```

(3)                                MOV      (SP), $TSTNM      ;; UPDATE THE TEST NUMBER
(3)                                DEC      (SP)              ;; BACKUP BY ONE
(3)                                ASL      (SP)              ;; SCALE THE TEST NUMBER AS AN INDEX
(3)                                ADD      #$$SW08TBL, (SP)      ;; FORM THE ADDRESS OF TEST POINTER
(3)                                MOV      @ (SP)+, $LPADR      ;; SET LOOP ADDRESS TO DESIRED TEST
(3)                                BR       $OVER            ;; GO LOOP ON THE TEST
(3)                                8$:      TST      (SP)+        ;; CLEAN THE BAD TEST NUMBER OFF OF THE STACK
(3)                                .ENDC
(3)                                .IFF
(3) 016470 127767 162744 162704 .IF EQ  $$SWRMK      ;; ON THE RIGHT TEST?   SWR<7:0>
(3)                                CMPB    @SWR, $TSTNM
(3)                                .IFF
(3)                                MOV      @SWR, -(SP)        ;; SET DESIRED TEST NUM. FROM SWR
(3)                                BIC     #$$SWRMK, (SP)      ;; STRIP AWAY UNDESIRED BITS
(3)                                CMPB    (SP)+, $TSTNM      ;; ON THE RIGHT TEST?
(3)                                .ENDC
(3) 016476 001465                BEQ     $OVER            ;; BR IF YES
(3)                                .ENDC
(3)                                .ENDC
(3)                                .IF NE  $$SWR&5000:SW11!SW09
(3) 016500 105767 162677        2$:      TSTB    $ERFLG      ;; HAS AN ERROR OCCURRED?
(3)                                .IF NE  $$SWR&4000:SW11
(3) 016504 001421                BEQ     3$              ;; BR IF NO
(3)                                .IFF
(3)                                BEQ     $$SVLAD          ;; BR IF NO
(3)                                .ENDC
(3)                                .IF NE  $$SWR&1000:SW09
(3)                                .IF NE  $$SWR&4000:SW11
(3) 016506 126767 162703 162667 CMPB    $ERMAX, $ERFLG  ;; MAX. ERRORS FOR THIS TEST OCCURRED?
(3) 016514 101015                BHI    3$              ;; BR IF NO
(3)                                .ENDC
(3) 016516 032777 001000 162714 BIT     #BIT09, @SWR    ;; LOOP ON ERROR?
(3) 016524 001404                BEQ    4$              ;; BR IF NO
(3) 016526 016767 162656 162652 7$:      MOV     $LPERR, $LPADR  ;; SET LOOP ADDRESS TO LAST SCOPE
(3) 016534 000446                BR     $OVER
(3)                                .IFTF
(3) 016536 105067 162641        4$:      CLR    $ERFLG      ;; ZERO THE ERROR FLAG
(3) 016542 005067 162744        CLR    $TIMES        ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
(3)                                .IFT
(3) 016546 000415                BR     1$              ;; ESCAPE TO THE NEXT TEST
(3)                                .ENDC
(3)                                .ENDC
(3)                                .IF NE  $$SWR&4000:SW11
(3) 016550 032777 004000 162662 3$:      BIT     #BIT11, @SWR  ;; INHIBIT ITERATIONS?
(3) 016556 001011                BNE    1$              ;; BR IF YES
(3)                                .IF B
(3) 016560 005767 162750        TST    $PASS          ;; IF FIRST PASS OF PROGRAM
(3) 016564 001406                BEQ    1$              ;; INHIBIT ITERATIONS
(3)                                .ENDC
(3) 016566 005267 162612        INC    $ICNT          ;; INCREMENT ITERATION COUNT
(3) 016572 026767 162714 162604 CMP     $TIMES, $ICNT  ;; CHECK THE NUMBER OF ITERATIONS MADE
(3) 016600 002024                BGE    $OVER          ;; BR IF MORE ITERATION REQUIRED
(3) 016602 012767 000001 162574 1$:      MOV     #1, $ICNT     ;; REINITIALIZE THE ITERATION COUNTER
(3) 016610 016767 000056 162674 MOV     $MXCNT, $TIMES ;; SET NUMBER OF ITERATIONS TO DO
(3)                                .ENDC
(3) 016616 105207 162560        $SVLAD: INCB    $TSTNM  ;; COUNT TEST NUMBERS

```



```

(3) .LIST
(3) $SW08TBL:
(3) .REPT $TN-1
(3) .IRP X,<\$SW08>
(3) .WORD TST'X+2 ;:STARTING ADDRESS OF TEST X
(3) .ENDM
(3) .NLIST
(3) $$$SW08=$$SW08+1
(3) .LIST
(3) .ENDR
(3) 001
(3) .ENDC
(3) 000
(1) 016674
(2) .STRAP
(2) .SBTTL TRAP DECODER
(2) 016674
(3) STARS
(3) .IF B
(3) ;:*****
(3) .IFF
(3) .NLIST
(3) .REPT
(3) .LIST
(3) ;:*****
(3) .NLIST
(3) .ENDR
(3) .LIST
(3) 000
(3) .ENDC
(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) 001
(2) 016674 010046
(2) .IF B
(2) $TRAP: MOV R0,-(SP) ;:SAVE R0
(2) .IFF
(2) $TRAP: MOV 2(SP),-(SP) ;
(2) ;ASSUME THE STATUS OF
(2) BIC #20,(SP) ;: THE CALLER--DO NOT ALLOW
(2) MOV #1$,-(SP) ;: T-BIT TRAPS
(2) RTI ;:SET THE NEW STATUS
(2) 1$: MOV R0,-(SP) ;:SAVE R0
(2) .ENDC
(2) 000
(2) 016676 016600 000002
(2) MOV 2(SP),R0 ;:GET TRAP ADDRESS
(2) TST -(R0) ;:BACKUP BY 2
(2) 016702 005740
(2) MOV (R0),R0 ;:GET RIGHT BYTE OF TRAP
(2) 016704 111000
(2) 001
(2) .IF NB
(2) BPL $TRAP1 ;:NON-USER TRAP,BELOW 200
(2) BIC #^C177,R0 ;:STRIP AWAY THE JUNK
(2) JMP (PC) ;:USER TRAP,ABOVE 177, GO TO
(2) .WORD ;: USER TRAP HANDLER-
(2) $TRAP1:
(2) .ENDC
(2) 000
(2) 001
(2) .IF NB
(2) CMP # $TERM,R0 ;:CHECK FOR OUT OF BOUNDS
(2) BGT .+6 ;:BR IF OK
(2) HALT ;:OUT OF BOUNDS

```



```

(2)          001          .IF DF $TYPOC
(2) 016734          SETTRAP TYPOC,$TYPOC,^/TYPE OCTAL NUMBER (WITH LEADING ZEROS)/
(3) 016734          $$SET TYPOC,$TYPOC,\<TRAP+$TRP>,\$TRP,<TYPE OCTAL NUMBER (WITH LEADING ZEROS)>
(4)          002          .IF EQ $TRP-1
(4)          .SBTTL TRAP TABLE
(4)
(4)          : *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(4)          : *BY THE 'TRAP' INSTRUCTION.
(4)
(4)          :          ROUTINE
(4)          :          -----
(4) $TRPAD: .WORD $TRAP2
(4) .ENDC
(4)          TYPOC= 104402
(4) 016734 014630 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)          000003 $TRP=$TRP+1
(2) 016736          SETTRAP TYPOS,$TYPOS,^/TYPE OCTAL NUMBER (NO LEADING ZEROS)/
(3) 016736          $$SET TYPOS,$TYPOS,\<TRAP+$TRP>,\$TRP,<TYPE OCTAL NUMBER (NO LEADING ZEROS)>
(4)          002          .IF EQ $TRP-1
(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) .ENDC
(4)          TYPOS= 104403
(4) 016736 014604 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)          000004 $TRP=$TRP+1
(2) 016740          SETTRAP TYPON,$TYPON,^/TYPE OCTAL NUMBER (AS PER LAST CALL)/
(3) 016740          $$SET TYPON,$TYPON,\<TRAP+$TRP>,\$TRP,<TYPE OCTAL NUMBER (AS PER LAST CALL)>
(4)          002          .IF EQ $TRP-1
(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) .ENDC
(4)          TYPON= 104404
(4) 016740 014644 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)          000005 $TRP=$TRP+1
(2)          000          .ENDC
(2)          001          .IF DF $TYPDS
(2)          SETTRAP TYPDS,$TYPDS,^/TYPE DECIMAL NUMBER (WITH SIGN)/
(2)          .ENDC
(2)          000          .IF DF $TYPBN
(2)          SETTRAP TYPBN,$TYPBN,^/TYPE BINARY (ASCII) NUMBER/
(2)          .ENDC
(2)          000
(2)          001          .IF DF $GTSWR
(2)          SETTRAP GTSWR,$GTSWR,^/GET SOFT-SWR SETTING/

```



```

(2)          000          .ENDC
(2)
(2)          001          .IF DF $CKSWR
(2)                                SETTRAP CKSWR,$CKSWR,^/TEST FOR CHANGE IN SOFT-SWR/
(2)          000          .ENDC
(2)          001          .IF DF $RDCHR
(2)                                SETTRAP RDCHR,$RDCHR,^/TTY TYPEIN CHARACTER ROUTINE/
(2)          000          .ENDC
(2)          001          .IF DF $RDLIN
(2)                                SETTRAP RDLIN,$RDLIN,^/TTY TYPEIN STRING ROUTINE/
(2)          000          .ENDC
(2)          001          .IF DF $RDOCT
(2)                                SETTRAP RDOCT,$RDOCT,^/READ AN OCTAL NUMBER FROM TTY/
(2)          000          .ENDC
(2)          001          .IF DF $RDDEC
(2)                                SETTRAP RDDEC,$RDDEC,^/READ A DECIMAL NUMBER FROM TTY/
(2)          000          .ENDC
(2)          001          .IF DF $SAVREG
(2)                                SETTRAP SAVREG,$SAVREG,^/SAVE R0-R5 ROUTINE/
(2)                                SETTRAP RESREG,$RESREG,^/RESTORE R0-R5 ROUTINE/
(2)          000          .ENDC
(2)          001          .IF DF $R2A
(2)                                SETTRAP R2AZ,$R2AZ
(2)                                SETTRAP R2AZ.,$R2AZ.
(2)                                SETTRAP R2AZQ,$R2AZQ
(2)          000          .ENDC
(1) 016742   000          SETTRAP SCOP1,.$SCOP1      ;CALL TO LOOP ON CURRENT DATA HANDLER
(2) 016742   001          $$SET SCOP1,.$SCOP1,\<TRAP+$TRP>,\$TRP,<>
(3)
(3)          .IF EQ $TRP-1
(3)          .SBTTL TRAP TABLE
(3)
(3)          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)          ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)          :          ROUTINE
(3)          :          -----
(3)          $TRPAD: .WORD  $TRAP2
(3)          .ENDC
(3)          SCOP1= 104405
(3)          .SCOP1  ;;CALL=SCOP1      TRAP+5(104405)
(2) 016742   000006   $TRP=$TRP+1
(1) 016744   000          SETTRAP INSTR,.$INSTR      ;ACLL TO ASCII STRING OUTPUT ROUTINE
(2) 016744   001          $$SET INSTR,.$INSTR,\<TRAP+$TRP>,\$TRP,<>
(3)
(3)          .IF EQ $TRP-1
(3)          .SBTTL TRAP TABLE
(3)
(3)          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)          ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)          :          ROUTINE
(3)          :          -----
(3)          $TRPAD: .WORD  $TRAP2
(3)          .ENDC
(3)          INSTR= 104406
(3)          .INSTR  ;;CALL=INSTR      TRAP+6(104406)
(2) 016744   000007   $TRP=$TRP+1

```

```

(1) 016746 SETTRAP INSTER,..INSTER ;CALL TO INPUT ERROR HANDLER
(2) 016746 $$$SET INSTER,..INSTER,\<TRAP+$TRP>,\$TRP,<>
(3) 001 .IF EQ $TRP-1
(3) .SBTTL TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3) ;*BY THE "TRAP" INSTRUCTION.
(3)
(3) : ROUTINE
(3) :-----
(3) $TRPAD: .WORD $TRAP2
(3) .ENDC
(3) 000 INSTER= 104407
(3) 104407 .INSTER ;;CALL=INSTER TRAP+7(104407)
(3) 016746 013502 $TRP=$TRP+1
(2) 000010 SETTRAP PARAM,..PARAM ;CALL TO NUMERICAL DATA INPUT ROUTINE
(1) 016750 $$$SET PARAM,..PARAM,\<TRAP+$TRP>,\$TRP,<>
(2) 016750 001 .IF EQ $TRP-1
(3) .SBTTL TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3) ;*BY THE "TRAP" INSTRUCTION.
(3)
(3) : ROUTINE
(3) :-----
(3) $TRPAD: .WORD $TRAP2
(3) .ENDC
(3) 000 PARAM= 104410
(3) 104410 .PARAM ;;CALL=PARAM TRAP+10(104410)
(3) 016750 013512 $TRP=$TRP+1
(2) 000011 SETTRAP SAV05,..SAV05 ;CALL TO REGISTER SAVE ROUTINE
(1) 016752 $$$SET SAV05,..SAV05,\<TRAP+$TRP>,\$TRP,<>
(2) 016752 001 .IF EQ $TRP-1
(3) .SBTTL TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3) ;*BY THE "TRAP" INSTRUCTION.
(3)
(3) : ROUTINE
(3) :-----
(3) $TRPAD: .WORD $TRAP2
(3) .ENDC
(3) 000 SAV05= 104411
(3) 104411 .SAV05 ;;CALL=SAV05 TRAP+11(104411)
(3) 016752 013712 $TRP=$TRP+1
(2) 000012 SETTRAP RES05,..RES05 ;CALL TO REGISTER RESTORE ROUTINE
(1) 016754 $$$SET RES05,..RES05,\<TRAP+$TRP>,\$TRP,<>
(2) 016754 001 .IF EQ $TRP-1
(3) .SBTTL TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3) ;*BY THE "TRAP" INSTRUCTION.
(3)
(3) : ROUTINE
(3) :-----
(3) $TRPAD: .WORD $TRAP2

```


| | | | | | | | | |
|------|--------|--------|--------|--------|--------|-----|-------------------------------------|--------------------------|
| 8752 | 017062 | 016767 | 000102 | 162626 | | MOV | DUBASE,RXDBUF | :XXX2 |
| 8753 | 017070 | 016767 | 000074 | 162624 | | MOV | DUBASE,PARCSR | :XXX2 |
| 8754 | 017076 | 005267 | 000066 | | | INC | DUBASE | |
| 8755 | 017102 | 016767 | 000062 | 162610 | | MOV | DUBASE,HRXDBUF | :XXX3 |
| 8756 | 017110 | 016767 | 000054 | 162606 | | MOV | DUBASE,HPARCSR | :XXX3 |
| 8757 | 017116 | 005267 | 000046 | | | INC | DUBASE | |
| 8758 | 017122 | 016767 | 000042 | 162576 | | MOV | DUBASE,TXCSR | :XXX4 |
| 8759 | 017130 | 005267 | 000034 | | | INC | DUBASE | |
| 8760 | 017134 | 016767 | 000030 | 162566 | | MOV | DUBASE,HTXCSR | :XXX5 |
| 8761 | 017142 | 005267 | 000022 | | | INC | DUBASE | |
| 8762 | 017146 | 016767 | 000016 | 162556 | | MOV | DUBASE,TXDBUF | :XXX6 |
| 8763 | 017154 | 005267 | 000010 | | | INC | DUBASE | |
| 8764 | 017160 | 016767 | 000004 | 162546 | | MOV | DUBASE,HTXDBUF | :XXX7 |
| 8765 | 017166 | 000207 | | | | RTS | PC | |
| 8766 | 017170 | 000000 | | | | | DUBASE: 0 | |
| 8767 | | | | | | | | |
| 8768 | | | | | | | | |
| 8769 | | | | | | | | |
| 8770 | | | | | | | | |
| 8771 | 017172 | 042777 | 040000 | 162526 | RPOKE: | BIC | #MTDATA,@TXCSR | |
| 8772 | 017200 | 005067 | 162276 | | | CLR | \$TMP2 | |
| 8773 | 017204 | 006067 | 162270 | | | ROR | \$TMP1 | :FORCE CARRY |
| 8774 | 017210 | 006067 | 162266 | | | ROR | \$TMP2 | :PICK UP CARRY IN BIT 15 |
| 8775 | 017214 | 006267 | 162262 | | | ASR | \$TMP2 | :SHIFT INTO BIT 14 |
| 8776 | 017220 | 042767 | 100000 | 162254 | | BIC | #BIT15,\$TMP2 | :CLR BIT 15 |
| 8777 | 017226 | 056777 | 162250 | 162472 | | BIS | \$TMP2,@TXCSR | :POKE MAINT DATA |
| 8778 | 017234 | 042777 | 020000 | 162464 | | BIC | #CLK,@TXCSR | :POKE CLK |
| 8779 | 017242 | 052777 | 020000 | 162456 | | BIS | #CLK,@TXCSR | : |
| 8780 | 017250 | 005367 | 161646 | | | DEC | SHIFT | |
| 8781 | 017254 | 001346 | | | | BNE | RPOKE | |
| 8782 | 017256 | 000207 | | | | RTS | PC | |
| 8783 | | | | | | | | |
| 8784 | | | | | | | | |
| 8785 | 017260 | 016767 | 162214 | 162214 | ODD8: | MOV | \$TMP1,\$TMP2 | :SAVE TEMP1 |
| 8786 | 017266 | 005067 | 162212 | | | CLR | \$TMP3 | |
| 8787 | 017272 | 012727 | 000010 | | | MOV | #8.,(PC)+ | |
| 8788 | 017276 | 000000 | | | 4\$: | 0 | | |
| 8789 | 017300 | 006067 | 162176 | | 1\$: | ROR | \$TMP2 | |
| 8790 | 017304 | 005567 | 162174 | | | ADC | \$TMP3 | |
| 8791 | 017310 | 005367 | 177762 | | | DEC | 4\$ | |
| 8792 | 017314 | 001371 | | | | BNE | 1\$ | |
| 8793 | 017316 | 006067 | 162162 | | | ROR | \$TMP3 | |
| 8794 | 017322 | 103404 | | | | BCS | 2\$ | |
| 8795 | 017324 | 052767 | 000400 | 162146 | | BIS | #BIT8,\$TMP1 | :SET ODD PARITY |
| 8796 | 017332 | 000403 | | | | BR | 3\$ | |
| 8797 | 017334 | 042767 | 000400 | 162136 | 2\$: | BIC | #BIT8,\$TMP1 | :CLR EVEN PARITY |
| 8798 | | | | | | | \$TMP1 NOW HAS ODD PARITY CHARACTER | |
| 8799 | 017342 | 000207 | | | 3\$: | RTS | PC | |
| 8800 | | | | | | | | |
| 8801 | | | | | | | | |
| 8802 | 017344 | 016767 | 162130 | 162130 | EVEN8: | MOV | \$TMP1,\$TMP2 | :SAVE TEMP1 |
| 8803 | 017352 | 005067 | 162126 | | | CLR | \$TMP3 | |
| 8804 | 017356 | 012727 | 000010 | | | MOV | #8.,(PC)+ | |
| 8805 | 017362 | 000000 | | | 4\$: | 0 | | |
| 8806 | 017364 | 006067 | 162112 | | 1\$: | ROR | \$TMP2 | |
| 8807 | 017370 | 005567 | 162110 | | | ADC | \$TMP3 | |

```

8808 017374 005367 177762          DEC      4$
8809 017400 001371          BNE      1$
8810 017402 006067 162076          ROR      $TMP3
8811 017406 103004          BCC      2$
8812 017410 052767 000400 162062    BIS      #BIT8,$TMP1      ;SET EVEN PARITY
8813 017416 000403          BR       3$
8814 017420 042767 000400 162052    2$:    BIC      #BIT8,$TMP1      ;CLR ODD PARITY
8815          ;$TMP1 NOW HAS EVEN PARITY CHARACTER
8816 017426 000207          3$:    RTS      PC
8817 017430 062716 000002    TRPREG: ADD    #2,(SP) ;ALLOW IT TO "CRUNCH" INTO HLT BACK
8818          ;IN MAIN PART OF THE PROGRAM
8819 017434 000002          RTI
8820          ;INITIALIZE SBC 11/21 PROCESSOR ODT AND LKVEC
8821 017436          .INIT
(1)          017436          POINT=.      ;SAVE POINTER
(1)          000100          .=100
(1) 000100 017436          $CLKVEC      ;LKVEC HANDLER
(1) 000102 000300          300          ;INTERRUPT HANDLER PRI
(1)          000140          .=140          ;BRKVEC
(1) 000140 170000          170000       ;ODT START ADDRESS
(1) 000142 000300          300          ;PRIORITY
(1)          017436          .=POINT      ;RESTORE POINTER
(1) 017436 104401 017444          $CLKVEC:    TYPE,CLKMES
(1) 017442 000000          HALT
(1) 017444 005015 045514 042526    CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
(1) 017452 020103 047111 042524
(1) 017460 051122 050125 020124
(1) 017466 020055 044504 041523
(1) 017474 047117 042516 052103
(1) 017502 046040 041524 000040
8822          000001          .END

```

| | | | |
|---------|--------|--------|-------|
| AAA | 003200 | 8148# | |
| ABASE = | 000000 | 8148 | |
| ACDW1 = | 000000 | 8148 | |
| ACDW2 = | 000000 | 8148 | |
| ACPUOP= | 000000 | 8148 | |
| ACTREG | 001166 | 8148#* | 8729 |
| ADDW0 = | 000000 | 8148 | |
| ADDW1 = | 000000 | 8148 | |
| ADDW10= | 000000 | 8148 | |
| ADDW11= | 000000 | 8148 | |
| ADDW12= | 000000 | 8148 | |
| ADDW13= | 000000 | 8148 | |
| ADDW14= | 000000 | 8148 | |
| ADDW15= | 000000 | 8148 | |
| ADDW2 = | 000000 | 8148 | |
| ADDW3 = | 000000 | 8148 | |
| ADDW4 = | 000000 | 8148 | |
| ADDW5 = | 000000 | 8148 | |
| ADDW6 = | 000000 | 8148 | |
| ADDW7 = | 000000 | 8148 | |
| ADDW8 = | 000000 | 8148 | |
| ADDW9 = | 000000 | 8148 | |
| ADEVCT= | 000000 | 8148 | |
| ADEVN = | 000000 | 8148 | |
| ADRCNT= | 013711 | 8729#* | |
| AENV = | 000000 | 8148 | |
| AENVN = | 000000 | 8148 | |
| AFATAL= | 000000 | 8148 | |
| AMADR1= | 000000 | 8148 | |
| AMADR2= | 000000 | 8148 | |
| AMADR3= | 000000 | 8148 | |
| AMADR4= | 000000 | 8148 | |
| AMAMS1= | 000000 | 8148 | |
| AMAMS2= | 000000 | 8148 | |
| AMAMS3= | 000000 | 8148 | |
| AMAMS4= | 000000 | 8148 | |
| AMSGAD= | 000000 | 8148 | |
| AMSGLG= | 000000 | 8148 | |
| AMSGTY= | 000000 | 8148 | |
| AMTYP1= | 000000 | 8148 | |
| AMTYP2= | 000000 | 8148 | |
| AMTYP3= | 000000 | 8148 | |
| AMTYP4= | 000000 | 8148 | |
| APASS = | 000000 | 8148 | |
| APRIOR= | 000000 | 8148 | |
| APTCSU= | 000040 | 7008# | 8729 |
| APTENV= | 000001 | 7008# | 8729 |
| APTSIZ= | 000200 | 7008# | 8148 |
| APTSPO= | 000100 | 7008# | 8729 |
| ASWREG= | 000000 | 8148 | |
| ATESTN= | 000000 | 8148 | |
| AUNIT = | 000000 | 8148 | |
| AUSWR = | 000000 | 8148 | |
| AVECT1= | 000000 | 8148 | |
| AVECT2= | 000000 | 8148 | |
| BASEAD | 001154 | 8148#* | 8729* |

| | | | | |
|----------|---------|--------|-------|------|
| \$APTHD | 002136 | 8148# | | |
| \$ASTAT= | ***** U | 7008 | | |
| \$ATYC | 000024 | 7008# | | |
| \$ATY1 | 000000 | 7008# | | |
| \$ATY3 | 000006 | 7008# | 8729 | |
| \$ATY4 | 000016 | 7008# | 8729 | |
| \$AUTOB | 001434 | 8148# | | |
| \$BASE | 001602 | 8148# | | |
| \$BDADR | 001422 | 8148# | | |
| \$BDDAT | 001426 | 8148# | | |
| \$BELL | 001516 | 8148# | 8729 | |
| \$CDW1 | 001606 | 8148# | | |
| \$CDW2 | 001610 | 8148# | | |
| \$CHARC | 013370 | 8729#* | | |
| \$CKSWR= | ***** U | 8729 | | |
| \$CLKVE | 017436 | 8821# | | |
| \$CMTAG | 001400 | 8148# | | |
| \$CM1 = | 000006 | 8148# | | |
| \$CM2 = | 000014 | 8148# | | |
| \$CM3 = | 000006 | 8148# | | |
| \$CM4 = | 000006 | 8148# | | |
| \$CPUOP | 001554 | 8148# | | |
| \$CRLF | 001523 | 8148# | 8729 | |
| \$DDW0 | 001612 | 8148# | | |
| \$DDW1 | 001614 | 8148# | | |
| \$DDW10 | 001636 | 8148# | | |
| \$DDW11 | 001640 | 8148# | | |
| \$DDW12 | 001642 | 8148# | | |
| \$DDW13 | 001644 | 8148# | | |
| \$DDW14 | 001646 | 8148# | | |
| \$DDW15 | 001650 | 8148# | | |
| \$DDW2 | 001616 | 8148# | | |
| \$DDW3 | 001620 | 8148# | | |
| \$DDW4 | 001622 | 8148# | | |
| \$DDW5 | 001624 | 8148# | | |
| \$DDW6 | 001626 | 8148# | | |
| \$DDW7 | 001630 | 8148# | | |
| \$DDW8 | 001632 | 8148# | | |
| \$DDW9 | 001634 | 8148# | | |
| \$DEVCT | 001536 | 8148# | | |
| \$EVM | 001604 | 8148# | | |
| \$E = | 000002 | 6836# | | |
| \$ENDAD | 012714 | 8148 | 8729# | |
| \$ENV | 001546 | 7008 | 8148# | 8729 |
| \$ENVM | 001547 | 7008 | 8148# | 8729 |
| \$ERFLG | 001403 | 8148#* | 8729* | |
| \$ERMAX | 001415 | 8148#* | 8729* | |
| \$ERROR | 014230 | 8148 | 8729# | |
| \$ERRPC | 001416 | 8148#* | 8729* | |
| \$ERRTB | 001652 | 8148# | 8729 | |
| \$ERRTY | 014450 | 8729# | | |
| \$ERTTL | 001412 | 8148#* | 8729* | |
| \$ESCAP | 001514 | 8148#* | 8729* | |
| \$ETABL | 001546 | 8148# | | |
| \$ETEND | 001652 | 8148# | | |
| \$FATAL | 001530 | 7008* | 8148# | |

35

CNDUU-A MACY11 30(1046) 14-DEC-82 10:02 PAGE 63-10
CNDUUA.M11 29-OCT-82 13:27 CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0148

| | | | |
|----------|---------|--------|-------|
| .CONVR | 014004 | 8729# | |
| .EOP | 012374 | 8729# | |
| .INSTE | 013502 | 8729# | |
| .INSTR | 013374 | 8729# | |
| .INST1 | 013414 | 8729# | |
| .MSG | 013416 | 8729#* | |
| .PARAM | 013512 | 8729# | |
| .PFAIL | 015032 | 8148 | 8729# |
| .RES05 | 013752 | 8729# | |
| .SAV05 | 013712 | 8729# | |
| .SCOPE | 016340 | 8729# | |
| .SCOPI | 013070 | 8729# | |
| .SETFL | 014162 | 8729# | |
| .START | 002152 | 8148# | 8729 |
| .\$ASTA= | ***** U | 7008 | |
| .\$X = | 002136 | 8148# | |

| | | | |
|---------|-------|-------|------|
| .\$DB20 | 4812# | | |
| .\$DIV | 4587# | | |
| .\$EOP | 2214# | 6492# | |
| .\$ERR0 | 2700# | 6493# | 8729 |
| .\$ERR1 | 2896# | 6493# | 8729 |
| .\$MULT | 4523# | | |
| .\$POWE | 4229# | 6493# | |
| .\$RAND | 4307# | | |
| .\$RDDE | 3891# | | |
| .\$RDOC | 3797# | | |
| .\$READ | 3395# | | |
| .\$R2AZ | 4958# | | |
| .\$SAVE | 3969# | | |
| .\$SB2D | 4771# | | |
| .\$SB20 | 4874# | | |
| .\$SCOP | 2454# | 6493# | 8729 |
| .\$SIZE | 4361# | | |
| .\$SUPR | 4913# | | |
| .\$TRAP | 4073# | 6493# | 8729 |
| .\$TYPB | 3287# | | |
| .\$TYPD | 3209# | | |
| .\$TYPE | 2985# | 6492# | 8729 |
| .\$TYPO | 3112# | 6493# | 8729 |
| .\$40CA | 972# | | |

. ABS. 017510 000

ERRORS DETECTED: 0

CNDUUA,CNDUUA/CRF/NL:TOC=(CNMAC2.SML,CNDUU1.M11,CNDUU2.M11,CNDUUA.M11
 RUN-TIME: 17 18 1 SECONDS
 RUN-TIME RATIO: 80/37=2.1
 CORE USED: 43K (86 PAGES)