DIAGNOSTICS

ADDENDUM

DIAGNOSTIC MANUAL

The attached pages are new or replacement pages effective April 01, 1981.

DIAGNOSTIC DEPARTMENT

September 15, 1981

PRODUCT ENHANCEMENTS

FIXTST15	OP-1/15 Fix Data Switch Test (5300-1103-XX).
MINI96	96 TPI Mini-diskette Subsystem Test (5000-11118-2).
RAMSW15	OP-1/15 RAM/ROM Switch Test (5300-1109-X).
SERPR15	OP-1/15 Serial Printer Adapter (5300-1119-X).
R3750	Attribute Overlay for OP-1/RW (508-3750-004).
KBDTSTP5	OP-1/15 Phillips Keyboard Test.
KBDTSTKT	OP-1/15 Keytronics Keyboard Test.
KBDP386	Tests Keyboards with PCO 386.
W3430	Attribute Overlay for WETTST (508-3430-002).

CORRECTED PRODUCT DEFICIENCIES

ASCTST	Revised to run with PCO 199. (CRAB #70)
BSC2	Timing changed to run on OP-1/15.
DSKSTP3	Commands added to exercise Bad Track Latch Reset and Bad Track Format. (CRAB #66)
IOTST4	Revised to check 4 high order interrupt bits on OP-1/R when PCO 338 is implemented. (CRAB #60)
PDCIFL	Revised to test 96 TPI Tandon mini drives.
RTCTST4	Revised to check 4 high order interrupt bits on OP-1/R when PCO 338 is implemented. (CRAB #60)
WRDTST	Clear Screen command fixed. (CRAB #71)

OUTSTANDING COMPLAINTS

PAR #	
394	Add test to check DTR with PCO 350.
396	Intermittant failures in AIOTST4, tests 3B,3C when run on OP-1/R with Sync Adapter.
397	Need Character Generator Fonts for visual observation in WETTST.
398	Need test to force and check for parity error.
401	Add select address test to AIOTST4.

OUTSTANDING COMPLAINTS

CRAB #	PROGRAM NAME	BUG
51	IOTSTM	Does not check DTR line
61	WETTST/RTEST	Does not test PCO 352
62	AIOTST4	Fails when used with sync adapter
63	VIDTST4/WETTST	Need character generator listings to compare to test
65	DSKTST	Need test to force parity error

PRODUCT DEFICIENCIES

None.

ADDENDUM

DIAGNOSTIC MANUAL

SECTION I

The attached pages are new or updated pages effective April 01, 1981

DIAGNOSTICS

SECTION 1

09/15/81

TABLE OF CONTENTS

DIAGNOSTIC MANUAL

(SECTION I)

September 15, 1981

Supplement Filing Instructions

Section I	Remove	Insert (total pages)	Description
Table of Contents	April 1981	September 1981 (2)	Table of Contents
21	21-6	21-6 (1)	KBDTST
22	None	22-10; 22-11 (2)	WETTST
29A	None	Cover Sheet; 29A-1 to 29A-6 (7)	MINI96
39	39-1	39-1 (1)	PDCIFL
42	None	Cover Sheet; 42-1 to 42-3 (4)	RAMSW15
43	None	Cover Sheet; 43-1 to 43-4; Figure 1; Figure 2 (7)	SERPR15

Note: These instructions should be inserted at the beginning of the DIAGNOSTIC Manual.

SECTION	PROGRAM NAME	TITLE	DATE
29	MINTST	Mini-Diskette Subsystem Test	01/15/80
29A	MINI96	Mini-Diskette Subsystem Test (96 TPI)	09/15/81
37	MPDCTST	Micro-Programmable Diskette Controller Test	07/15/80
39	PDCIFL	Sub-Assembly Programmable Diskette Controller Test	09/15/81
40	DSPSW	OP-1/15 Display Switch Test	02/24/81
41	LFI15	OP-1/15 Line Frequency Interrupt Test	03/23/81
42	RAMSW15	OP-1/15 RAM/ROM Switch Test	09/15/81
43	SERPR15	OP-1/15 Serial Printer Adaptor Test	09/15/81

ADDENDUM

DIAGNOSTIC MANUAL

SECTION II

The attached pages are new or updated pages effective <u>April 01, 1981</u>

DIAGNOSTICS 4K

SECTION II

09/15/81

TABLE OF CONTENTS

DIAGNOSTIC 4K MANUAL

(SECTION II)

September 15, 1981

Supplement Filing Instructions

Section II	Remove	Insert (total pages)	Description
Table of Contents	April 1981	September 1981 (2)	Table of Contents
2	2-1	2-1 (1)	RTCTST4
3	3-1; 3-4	3-1; 3-4 (2)	FIXTST4/FIXTST15
6	6-1 to 6-3	6-1 to 6-3	IOTST4

Note: These instructions should be inserted at the beginning of the DIAGNOSTIC 4K Manual.

BOOTSTRAPS

SECTION III

10/31/80

TABLE OF CONTENTS

APPENDICES

04/01/81

SECTION	PROGRAM NAME	TITLE	DATE
A	Appendix A	Run Modification Feature	06/15/77
B	Appendix B	4K Format Test Module Run Options/ Loading the 4K Diagnostic Programs	09/01/78
C	Appendix C	RUN, RUNSS, RUNXP	10/31/80
D	Appendix D	SEND	10/31/80
E	Appendix E	SENDS	10/31/80
F	Appendix F	Limited Distribution	10/31/80
G	Appendix G	Archived Diagnostics	07/15/80
H	Appendix H	(A)synchronous or (S)ynchronous	10/31/80
	Addendum		

DIATST

DIABLO PRINTER TEST

DIATST - DIABLO PRINTER TEST PROGRAM

Applicable Assemblies

5000-1127-X	Byte String Controller Board	(X=1-4)
5000-1136-1	Character Printer Board	
5000-1170-1	Word Mover Controller Board	

General Description

The purpose of the DIATST program is to determine if the Diablo Printer, the Character Printer board and the print portion of the Byte String Controller or Word Mover Controller are working properly and, if not, to give an indication of which print functions are incorrect. The program requires assistance from the operator to perform certain actions and analyze the PRINTOUT (See Figure 1) at the end of the last test.

16K of memory is required to run DIATST.

This manual applies only to the 8080 version of DIATST.

Loading Procedure

DIATST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If DIATST loads properly, it will identify itself and wait for operator action.

Operator Action

At the end of the wait period, DIATST needs to know if there is an Asynchronous or Synchronous adapter. Respond by typing an (A) for Asynchronous or (S) for Synchronous adapter.

DIATST will then ask for the serial number of the printer being tested. Enter the serial number and type a carriage return or to bypass it completely, just type a carriage return.

All tests including Test 01 operate automatically. Test 01 displays a message on the screen requesting the operator to perform a certain task. The operator indicates compliance by typing the SPACE bar after performing the requested task. When the space code is sensed DIATST continues to the next part of Test 01 or to the next test.

When the entire test has been completed, the prompt **"Type Space to Repeat DIATST"** will be displayed. If a space is typed on the keyboard, DIATST will restart.

Errors

All program detectable errors are indicated by an appropriate error message on the screen and the simultaneous activation of the bell. The bell will ring only to notify the operator of an error. The error message on the screen attempts to give a description of the nature of the problem. In most cases, this message should be adequate in diagnosing the problem. Otherwise, refer to the detailed description of each test to determine the purpose and results for the displayed error message.

After an error message is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test and depressing PROG will restart the program.

Even if DIATST proceeds from Test 00 to Test 16 without a displayed error, there could still be a printed error, since DIATST has no way to examine the characters printed or to ascertain if paper movement commands are functioning properly. After the final test the operator must compare the printout produced by the tests to the correct PRINTOUT in Figure 1.

Test Description

All test operations are described in this section. The program will halt and the specified error message will be displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing for on the top of each page. Below the description, all possible error messages are listed, with an explanation of the cause of the message. However, in addition to the specified messages, other messages may be displayed. These messages are general to all the tests and are listed below:

PLEASE MAKE PRINTER READY

This condition could be caused by improper data cable hook-up, lack of ribbon in printer, lack of power to printer, lack of printer controller board in OP-1, lack of paper (optional), cover open (optional) or printer carriage motion impeded by the left or right hard stops or by a foreign object.

A restore command may be issued from the keyboard by typing a shifted prog. This will make the printer ready if the not ready condition was caused by the carriage exceeding the left or right margins.

After righting the cause of the not ready condition, type space to continue.

TEST 00 Test that the print portion of the Byte String Controller does not respond to incorrect select address, does respond to correct select address and INIT de-selects a selected Byte String Controller.

CONTROLLER SELECTED WITH WRONG ADDRESS (XX)

IFL to one of the addresses 034, 0B0, 094 or 0A4 gave a result other than OFF (open bus). XX is the incorrect select address.

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (B4)

IFL to the correct Byte String Controller addresss(OB4) gave a result of OFF (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Byte String Controler, IFL did not get a result of OFF (open bus).

TEST 01 Tests that NOT READY bit from Printer can be read by software.

NOT READY FLAG LOOKS READY

NOT READY bit low as if printer is powered up, instead of being powered down and/or disconnected from the OP-1.

NOT READY FLAG LOOKS NOT READY

NOT READY bit high as if printer is not connected, or in a fault condition.

TEST 02 Tests whether Controller NOT BUSY bit (IFL bit 7) is set when DVCL and INIT commands are issued.

Expected IFL Status: 080

Multiple ERROR messages as below are possible.

IFL BIT 7 NG NOT BUSY FLAG

NOT BUSY FLAG low when it should be high.

IFL BIT 1 NG NOT READY

NOT READY FLAG high when it should be low.

IFL BIT 0 NG PRINTER BUSY

PRINTER BUSY FLAG high when it should be low.

TEST 03 Test that COM2 and COM3 do not cause the printer to activate.

A simple move is executed by issuing a COM2 (Byte String Controller) and a COM3 (Byte String and Word Mover Controller). If either of these caused the printer busy line to go busy the test will fail.

COM2 CAUSES PRINTER BUSY TO GO ACTIVE.

COM3 CAUSES PRINTER BUSY TO GO ACTIVE.

TEST 04 Tests if NOT BUSY bit (IFL bit 7) is set by a DVCL and INIT cleared for the duration of a print command (COM1) and is set upon completion of printing; also, test if PRINTER BUSY bit (IFL bit 0) is reset before a print command and is set during execution of the print command. In addition, test if locations PCAH (0823) and PCAL (0822) are initialized to the contents of PSAH (0821) and PSAL (0820) by a print command and eventually increment till they equal PTAH (0825) and PTAL (0824) upon print completion.

NOT BUSY IS NOT SET BY DVCL AND INIT

After a DVCL and INIT, IFL bit 7 was low when it should be high.

PRINTER BUSY IS SET BEFORE PRINT COMMAND ISSUED

Before a print command, IFL bit 0 washigh when it should be low.

CONTROLLER BUSY FLAG DOES NOT LOOK BUSY DURING A PRINT

After a print command, IFL bit 7 was high when it should be low.

PRINTER BUSY FLAG DOES NOT LOOK BUSY DURING A PRINT

After a print command, IFL bit 0 was low when it should be high.

CURRENT ADDRESS NOT INITIALIZED PROPERLY

After a print command, PCAH and PCAL were not loaded to the address in PSAH and PSAL.

CONTROLLER NOT BUSY FLAG STAYS BUSY TOO LONG

After about one second, IFL bit 7 is still low when it should be high.

CURRENT ADDRESS DID NOT STOP AT CORRECT ADDRESS

After the print command is finished, PCAH and PCAL are not equal to PTAH and PTAL.

<u>TEST 05</u> Tests that locations PCAH and PCAL increment until they point to a character matching the character in PTC if PTAH bit 7 is low, but increment until they equal PTAH and PTAL if PTAH bit 7 is high.

PRINT DID NOT STOP AT TERMINATING CHARACTER

During a print command with PTAH bit 7 low, PCAH and PCAL incorrectly incremented past a character that matches the character in PTC.

PRINT DID NOT STOP AT TERMINATING ADDRESS

During a print command with PTAH bit 7 high, PCAH and PCAL did not increment up to PTAH and PTAL.

<u>TEST 06</u> Tests for check mechanism fault from extended carriage movement in either direction.

NOT READY FLAG LOOKS READY

After issuing either of the print commands predetermined to cause a fault, the NOT READY bit did not go high.

<u>TEST 07</u> Tests if the RESTORE command functions properly.

RESTORE COMMAND DID NOT RETURN CARRIAGE COMPLETELY

After issuing a restore command, an attempt is made to print a line requiring the full platen width (13.20 inches). However, printing was aborted and the NOT READY bit went high, presumably because the carriage hit against the right hard stop.

<u>TEST 08</u> Tests if during print functions, the PRINTER BUSY signal holds off data transfer from the printer controller.

PRINTER BUSY SIGNAL DOES NOT HOLD OFF DATA TRANSFER

During a print command, PCAH and PCAL continuously incremented, with no hold off pause, until they reached PTAH and PTAL.

TEST 09 Tests all horizontal carriage movement bits.

BIT X IS NOT FUNCTIONING PROPERLY

A horizontal carriage movement bit was not moving its appropriate distance: 2 raised to the power of bit X (X represents the bit number from 0 to 9).

This test is performed by moving the carriage a total of 13.2 inches (792 increments). The carriage is moved the maximum number of times its appropriate distance is a divisor of 792 increments. The remainder of the 13.2 inches is comprised of one movement using the bits already verified. After this the NOT READY bit must go from low to high when the carriage is extended one sixth of an inch to the right, since it should hit the right hard stop causing a fault.

TEST 10 Visual test of vertical paper movement bits.

No possible error messages, refer to Figure 1 for verification.

TEST 11 Tests if the fast move bit causes carriage movement and printing to speed up by more than 25 percent.

FAST MOVE IS TOO SLOW

The speed of carriage movement in conjunction with printing was not increased sufficiently by setting the fast move bit.

TEST 12 Visual test of forward and backward carriage movement, and the printability of all capital letters on print wheel.

The message "THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS" is printed twice.

No possible error messages, refer to Figure 1 for verification.

TEST 13

Test the half space capability of increasing every carriage movement by 1/120th of an inch.

HALF SPACING IS NOT FUNCTIONING PROPERLY

A print line of many small movements is calculated to cause a total carriage move of 11.33 inches. The half spacing capability should increase the total movement of the print line to 13.2 inches. After this the NOT READY bit must go from low to high when the carriage is extended one sixth of an inch to the right, since it should hit the right hard stop causing a fault.

TEST 14

Visual test of ribbon dropping capability.

On single color ribbons the second line should appear lighter. On dual color ribbons the two lines should differ in color.

No possible error messages, refer to Figure 1 for verification.

TEST 15

Visual test of print wheel alignment.

The vertical lines should just touch and be accurately aligned with a skew of no more than 0.005 inch.

No possible error messages, refer to Figure 1 for verification.

TEST 16

Visual test that all 96 characters on the print wheel are printable.

All ASCII characters between 020 and 07F Hexadecimal are printed.

No possible error messages, refer to Figure 1 for verification.

```
T

  TEST NUMBER 05
TEST TERMINATING CHARACTER (
TEST TERMINATING ADDRESS

  TEST NUMBER 06
########################### LEFT ################### TEST CHECK MECHANISM FAULTS ################### RIGHT ########################

  TEST NUMBER 07
****************************************************** TEST RESTORE COMMAND ******************************************************

  TEST NUMBER 08
TEST OF BUSY SIGNAL

  TEST NUMBER 09
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (
1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2
  3                3               3               3               3               3               3               3               3               3
   4                                  4                               4                               4                               4
      5                                                                     5                                                               5
            6                                                                                                                                        6
                         7
                                                   8
                                                                                                      9


  TEST NUMBER 10
       02
         3
           4

               5


                       6





                                       7










                                                                      8




















                                                                                                                                     9


  TEST NUMBER 11
TEST FAST MOVE BIT
TEST FAST MOVE BIT

  TEST NUMBER 12
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS

  TEST NUMBER 13
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ TEST HALF SPACE CAPABILITY @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

  TEST NUMBER 14
TEST RIBBON UP * * * * *


  TEST NUMBER 15
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

  TEST NUMBER 16
¢!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~¬
¢!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~¬

DIATST COMPLETE, FOR DIABLO PRINTER SERIAL NUMBER 01234-ABCD
```

```
T

TEST NUMBER 04
TEST TERMINATING CHARACTER (
TEST TERMINATING ADDRESS

TEST NUMBER 05
########################### LEFT ################### TEST CHECK MECHANISM FAULTS ################### RIGHT ###########################

TEST NUMBER 06
***************************************************** TEST RESTORE COMMAND *****************************************************

TEST NUMBER 07
TEST OF BUSY SIGNAL

TEST NUMBER 08
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2       2
  3               3               3               3               3               3               3               3               3               3
    4                                 4                               4                               4                               4
      5                                                                     5                                                               5
            6                                                                                                                                             6
                          7
                                                    8
                                                                                                      9


TEST NUMBER 09
      02
        3
          4
              5

                      6



                                      7



                                                                      8



                                                                                                                                      9

TEST NUMBER 10
TEST FAST MOVE BIT
TEST FAST MOVE BIT

TEST NUMBER 11
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS

TEST NUMBER 12
############################################## TEST HALF SPACE CAPABILITY ##############################################

TEST NUMBER 13
TEST RIBBON UP * * * * *
TEST RIBBON DROP * * *

TEST NUMBER 14
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

TEST NUMBER 15
¢!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}¯¬
¢!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}¯¬

DIATST COMPLETE, FOR DIABLO PRINTER SERIAL NUMBER 01234-ABCD
ALL TESTS PASSED WITH NO PROGRAM DETECTABLE ERRORS
```

RAMCOM

MULTI-ALGORITHM MEMORY TEST

RAMCOM - MULTI-ALGORITHM MEMORY TEST

Program	Applicable Machine Assemblies
RAMCOM	8080 DOS, OP-1/R 8K Chips and OP-1/R 16K Chips
RAMH80	8080 HDOS
RAMH80	8080 4K Memory Board
RAMR8	OP-1/R 8K Chips
RAMR16	OP-1/R 16K Chips
RAM08	8008 DOS

Applicable Assemblies

5000-1107-X	8K Memory Board	(X=1,2)
5000-1139-X-1	RRP Memory Board	(X=1-7)
5000-1139-8-2	RRP Memory Board	
5000-1140-X	RRPM Board	(X=1-8)
5000-1142-X-Y	32K RAM Board	(X=1-8, Y=1,2)
5000-1155-X-Y-Z	Universal Memory Board	(X=1-5, Y=0-4, Z=0-4)

General Description

The purpose of the RAMCOM program (and subsets) is to determine if the Read/Write Random Access Memory (RAM) is working properly and, if not, to give an indication of which addresses and bits are incorrect. The program requires no operator interaction unless it is desired to change the test parameters which, upon loading, default to test all existing RAM with no stop on error detection.

RAMCOM is a multi-algorithm RAM test that is also self relocatable. The algorithms used are a "marching ones and zeros" and "RAM leakage with neighboring bits refreshed". To facilitate testing all portions of RAM equally, RAMCOM changes it's location after every loop.

2K of memory (addresses 0800 to 0FFF) is required to run RAMCOM. This 2K block is known to exist as RAM in all hardware configurations since there must be RAM at addresses 0800 to 0868 for the Display and Input/Output Microprocessors, and RAM must exist in 1K (minimum) blocks at 1K address boundaries.

This manual applies to both 8008 and 8080 versions of RAMCOM.

Loading Procedure

RAMCOM can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If RAMCOM loads properly, it will identify itself and wait five seconds before testing all existing RAM. This delay is necessary to allow all hardware vectors to stick.

Operator Action

The total 64K range of memory is divided into 64 blocks, each of 1K length. For operator convenience, these blocks are displayed as 16 groups of 4 on the second line of the display screen, as shown in Figure 1. Each group of 4, representing a block of length 4K, is delimited by a reversed hexadecimal digit representing the most significant digit of the beginning address of the 4K block. Within a 4K block marked with the reversed digit X, the 4 individual 1K blocks correspond to addresses X000 to X3FF, X400 to X7FF, X800 to XBFF, and XC00 to XFFF. To serve as a reminder to the operator, the digits 0, 4, 8, and C are displayed in the last 4K block (marked with a reversed F) but only if this last block does not exist as RAM. Portions of memory to be tested are indicated by an asterisk (*) in the appropriate position(s) on the second line of the display screen while areas of non-existant RAM or of existing ROM or PROM memory (which cannot be tested by RAMCOM) are denoted by the absence of asterisks. The 2K block in which the program resides is marked by reversed P's.

Now, when RAMCOM is first loaded, a cursory RAM existance check is performed and asterisks are automatially filled in for every 1K block which is determined to consist of RAM. After this, testing commences.

At any time while RAMCOM is testing, the operator may change the portion of memory to be tested and continue either with or without a stop on error detection. However, a pause might be discerned after the operator types "SHIFTED HOME" and before RAMCOM beeps to acknowledge that the operator has control.

The following is a list of valid test parameter commands:

Home (with SHIFT key) will give operator control and allow the operator to change test parameters.

Cursor Right (key) - Stop testing and move the cursor, consisting of a continuously changing character on the second line of the display screen, one position to the right. This is in preparation for changing the character at the cursor from a space to an asterisk or vice versa.

Cursor Left (key) - Stop testing and move the cursor one position to the left.

Asterisk (* and SHIFT keys) - Stop testing, write an asterisk at the cursor position, and move the cursor one position to the right. The asterisk signifies that the 1K block so marked will be tested when testing resumes.

Space (SPACE bar) - Stop testing, clear the character at the cursor position, and move the cursor one position to the right. The absence of an asterisk signifies that the 1K block so marked will be skipped when testing resumes.

Restart (PROG key) - Return to the beginning of RAMCOM, re-search and mark all existing RAM with asterisk, clear the loop count, error count, and actual, expected, and over-write error values, and commence testing with no stop on error detection and relocation enabled.

Go (G key without SHIFT key) - Continue testing the 1K blocks that are currently marked with an asterisk and do not stop testing on error detection. If any asterisks were changed from their previous positions, clear the loop count, error count, and actual, expected, and over-write error values. The Go mode is useful if the opertor wishes to determine the overall integrity of the RAM memory.

Stop (S key without SHIFT key) - Continue testing the 1K blocks that are currently marked with an asterisk but halt testing and beep if an error is detected. If any asterisks were changed from their previous positions, clear the loop count, error count, and actual, expected, and over-write error values. The stop mode is useful if the operator wished to analyze the various address and data values existing when an error is detected.

No Relocating (N key without SHIFT key) - Continue testing without relocating the program after each loop. Typing the N key will put an "N" on the screen between Errors and Address. Can only be reset by typing the PROG key.

Errors

All errors are indicated by RAMCOM incrementing the error count and updating the actual, expected, and over-write values displayed on the top line of the display screen, as shown in Figure 1. Also if the Stop mode is in effect, each error will beep and halt testing, thereby allowing the operator to examine the updated error information.

Errors are of two types: Data errors and Address errors. Data errors are caused by failure of a RAM chip to "hold" the data written into it by a previous write pass. This could be the result of a faulty or marginal chip, incorrect insertion of a chip in it's socket, or grounded, floating, or interconnected data lines. A faulty or incorrectly inserted chip or a bad data line will generate very many errors (over 0400 hex for each 1K block) per loop. A marginal chip will usually have only a few errors recorded for each loop. All data errors are characterized by having the actual and expected values differ by at most only a few bits, i.e. ACT=DE, EXP=FF, OVRWRT=00.

Address errors are caused by the writing of correct data at an incorrect location. This subsequently shows up when the data is read. Address errors could be the result of grounded, floating, or interconnected addressing lines. Address errors tend to generate very many errors (over 0400 for each 1K block) per loop and are characterized by having the actual and overwrite values differ by at most a few bits, i.e. ACT=F7, EXP=00, OVRWRT=FF. Also, since addressing lines are common for a whole memory board, a bad Addressing line will usually cause errors for all RAM on the board.

In addition to the individual error values updated each time an error is detected, "stuck" data bits are displayed on lines 3 and 4 of the display screen. There are 16 stuck bits displays, one for each 4K block in the 64K memory. The horizontal position of any asterisk(*) above each stuck bits display has nothing whatever to do with the interpretation of the information displayed in the bits; asterisks merely specify which 1K block(s) are participating in the test and hence the address range represented by the stuck bits.

Each stuck bits display shows the accumulated errors for all eight data bits of the 4K block which is displayed above it. The individual bits are easily identified; bits 7, 6, 5, and 4 are displayed from left to right on display line 3, while bits 3, 2, 1, and 0 are dispalyed immediately below them on display line 4. Each bit is displayed as one of the following characters:

CHARACTER	MEANING
- (hyphen)	The bit was always correct.
0	The bit was incorrectly low when it should have been high.
1	The bit was incorrectly high when it should have been low.
X	The bit was both incorrectly low and high at different times.

An error at a known bit and in a known address range can be mapped to a particular chip using the appropriate Figure 2, 3, 4, or 5. Assume that the error shown in Figure 1 is detected, i.e. bit 2 in address range 1000 to 17FF is stuck high. Then, if an RRP Memory Board is being used, Figure 3 shows that the bad chip is third up in the second leftmost block of 8 chips on the board in slot 4, holding the board with the connector "fingers" on the left. If, however, a 32K RAM Board is being used, Figure 4 indicates that the bad chip is located on the top half of the board, the third chip in from the left hand side, and in the second row of chips on the board in slot 4.

Test Description

When RAMCOM is first loaded, or when the PROG key is depressed by the operator, a cursory RAM existance check is performed in order to fill in the initial asterisks. This cursory check uses the following algorithm:

1. One location in the 1K block to be checked is read.
2. The inverse of the original contents is written to the location.
3. The location is read again and the new contents is compared with the original contents. If the new contents and the original contents differ, even if only by one bit out of the eight bits, the 1K block is assumed to exist as RAM and an asterisk is written in the appropriate location on the display screen. If the new and original contents are identical, the assumption is made that the 1K block is non-existant RAM or existing ROM or PROM and no asterisk is written.

4. Steps 1 through 3 are repeated for all 1K blocks which could exist as RAM. Thus, if an 8K maximum length memory board such as an 1107, 1139, or 1140 is plugged into slot 4, steps 1 through 3 are performed only for the first 16 1K blocks. If a 32K maximum length board such as an 1142, or 1155 is plugged into slot 4, steps 1 through 3 are performed for all 64 1K blocks.

After the cursory RAM existance check, testing commmences. A multi-algorithm testing sequence is used which proceeds as follows:

1. Load 00 into all locations in all 1K blocks under test.
2. Read from the first location in the first 1K block and compare the actual value with the expected value. Update any errors to the display screen.

3. Over-write the inverse of the expected value to the current location.

4. Repeat steps 2 and 3 for all locations in all 1K blocks under test, with the current location going forward (from low addresses to high addresses).

5. Read from the last location in the last 1K block and compare the actual with the expected value (which is now the inverse of the value in step 1).

6. Over-write the inverse of the expected value to the current location.

7. Repeat steps 5 and 6 for all locations in all 1K blocks under test, with the current location going backward (from high addresses to low addresses).

8. Repeat steps 1 through 7 seven times with the hex values 01, 03, 07, 0F, 1F, 3F, 7F.

9. Load 0FF hex into all locations in all 1K blocks under test.

10. Overwrite the inverse of the original value to the current location.

11. Refresh the contents of the 8 surrounding bits on each 4K chip.

12. Check the value of the current location on each 4K chip to verify no leakage occurred into or out of the bit under test on each 4K chip.

13. Repeat steps 10, 11, and 12 for all groups of 4K chips under test.

14. Repeat steps 10 through 13 for all 4K locations in a 4K chip.

15. Repeat steps 9 through 14 with the hex value 00.

16. a). Overwrite the activity flag (flashing asterisk) with an R signifying relocating in process.

    b). Starting with the program location, search the RAM under test for two contiguous 1K blocks that remain error free up to this point (wrapping around to the beginning if necessary).

    c). After a new destination for the program is located, re-write the program to this location.

    d). Overwrite the R with the activity flag once again.

    e). Increment the loop counter on the display and continue testing from this new area.

17. Load 0FF hex into all locations in all 1K blocks under test.

18. Read from the first location in the first 1K block and compare the actual value with the expected value. Update any errors to the display screen.

19. Over-write the inverse of the expected value to the current location.

20. Repeat steps 18 and 19 for all locations in all 1K blocks under test, with the current location going forward (from low addresses to high addresses).

21. Read from the last location in the last 1K blocks under test, with the current location going forward (from low addresses to high addresses).

22. Over-write the inverse of the expected value to the current location.

23. Randomly select and de-select the 1K blocks under test.

24. Repeat steps 17 through 28 seven times with the hex values 01, 03, 07, 0F, 1F, 3F, 7F, FF, FE, FC, F8, F0, E0, C0, 80, 00, 01, ... (Note that after 16 values, the values repeat).

25. Repeat steps 9 through 25 until operator intervention.

```
RAMCOM   1.2  ERRS=0001     FWD ADDR=17A0 ACT=04 EXP=00 OVRWRT=FF  *  LOOP=----
0**PP1****2    3    4    5    6    7    8    9    A    B    C    D    E    F
=---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 ---- -1-- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
```

RAMCOM DISPLAY SCREEN

Figure 1

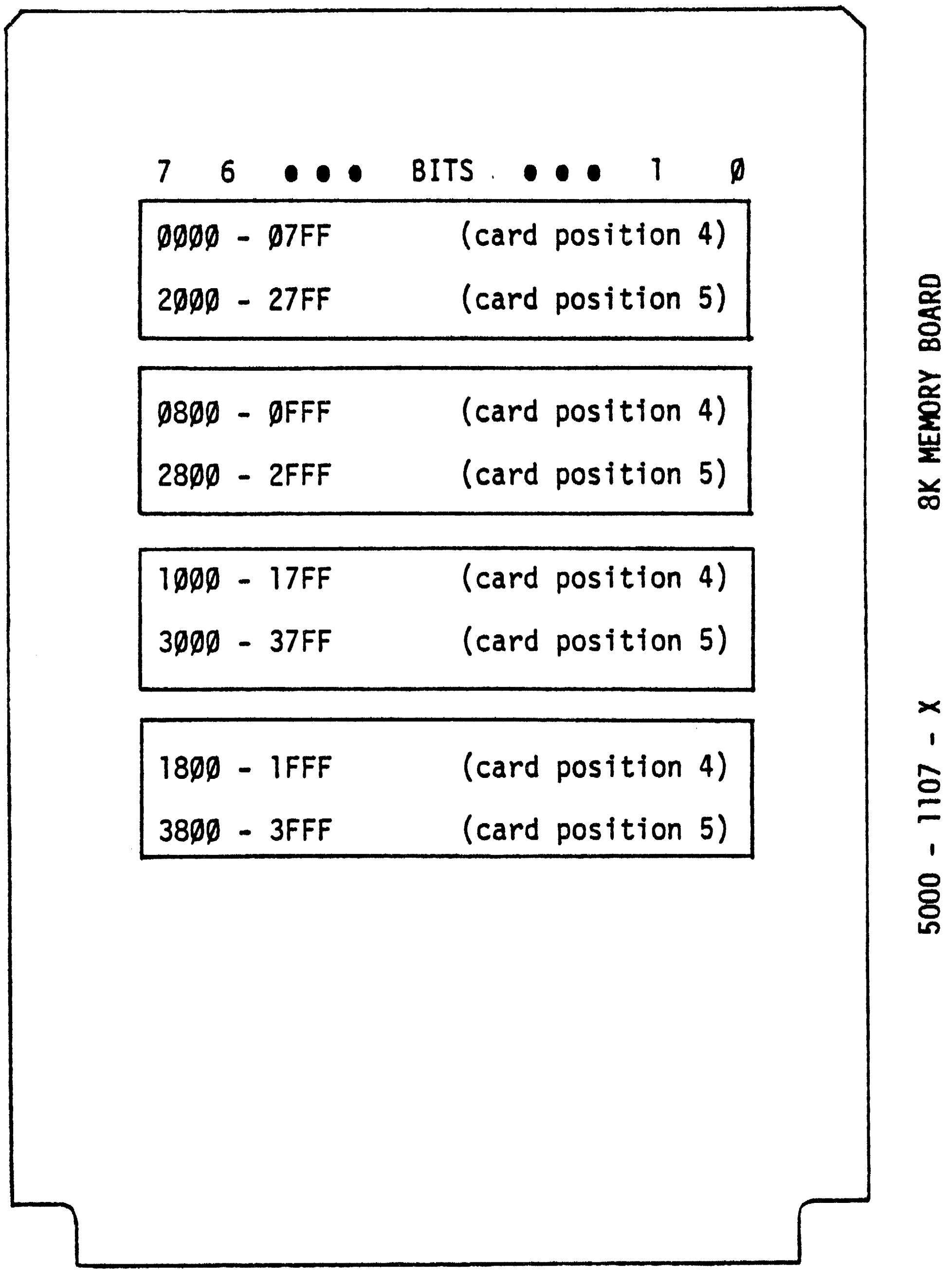

7 6 ● ● ● BITS ● ● ● 1 Ø
ØØØØ - Ø7FF (card position 4)
2ØØØ - 27FF (card position 5)
Ø8ØØ - ØFFF (card position 4)
28ØØ - 2FFF (card position 5)
1ØØØ - 17FF (card position 4)
3ØØØ - 37FF (card position 5)
18ØØ - 1FFF (card position 4)
38ØØ - 3FFF (card position 5)
8K MEMORY BOARD
5000 - 1107 - X

FIGURE 2

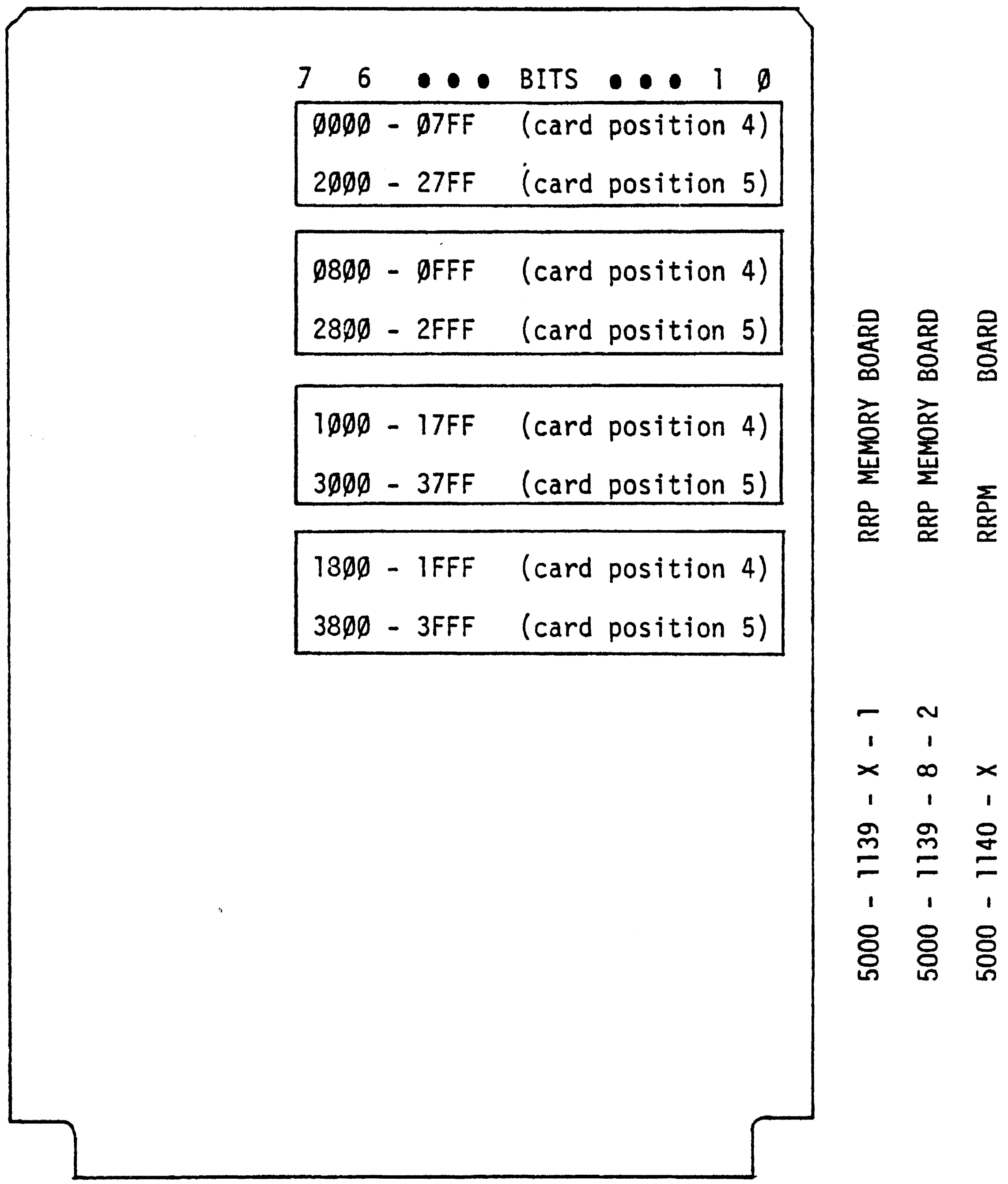
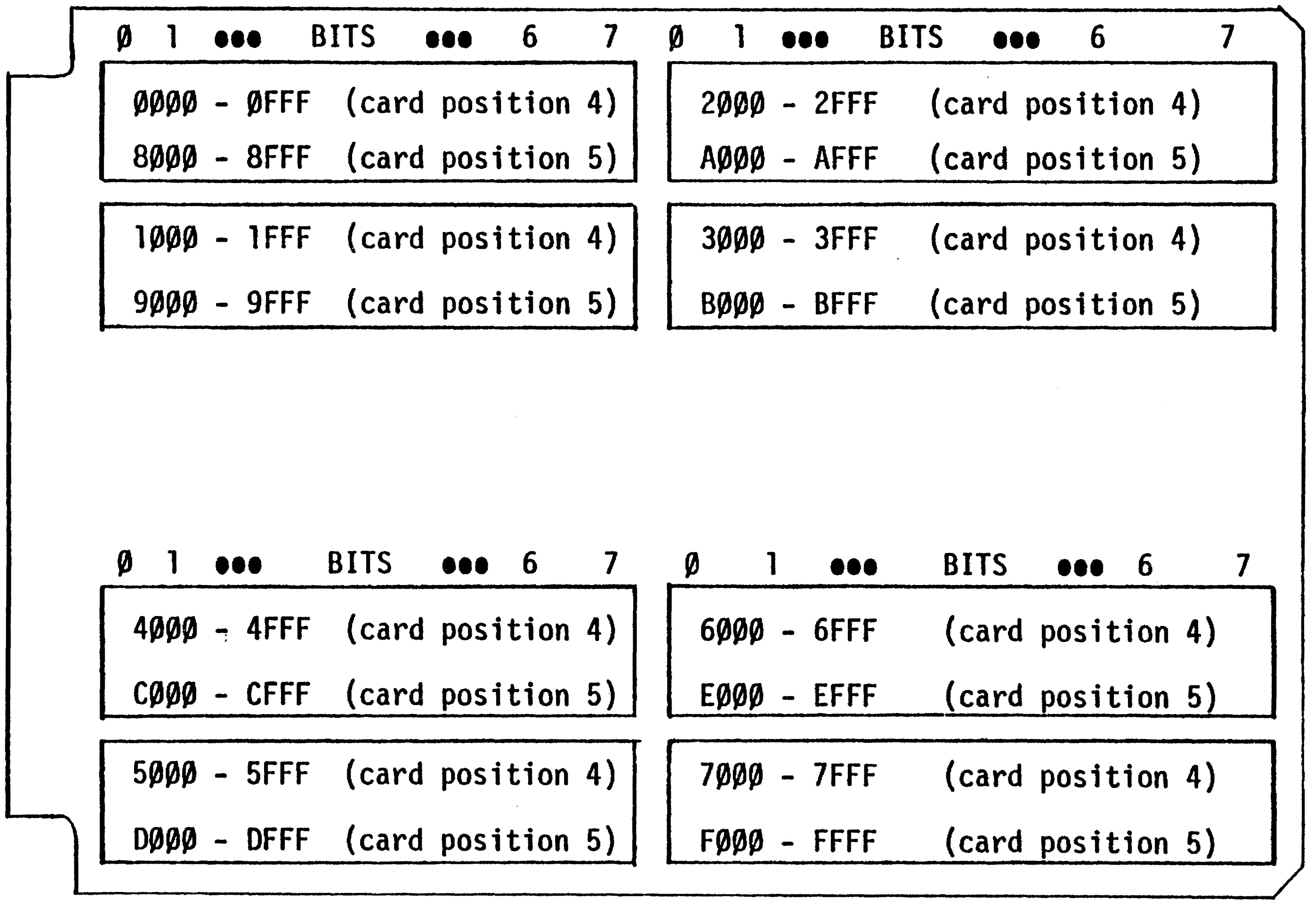

7 6 ● ● ● BITS ● ● ● 1 Ø
ØØØØ - Ø7FF (card position 4)
2ØØØ - 27FF (card position 5)
Ø8ØØ - ØFFF (card position 4)
28ØØ - 2FFF (card position 5)
1ØØØ - 17FF (card position 4)
3ØØØ - 37FF (card position 5)
18ØØ - 1FFF (card position 4)
38ØØ - 3FFF (card position 5)
5000 - 1139 - X - 1 RRP MEMORY BOARD
5000 - 1139 - 8 - 2 RRP MEMORY BOARD
5000 - 1140 - X RRPM BOARD

FIGURE 3

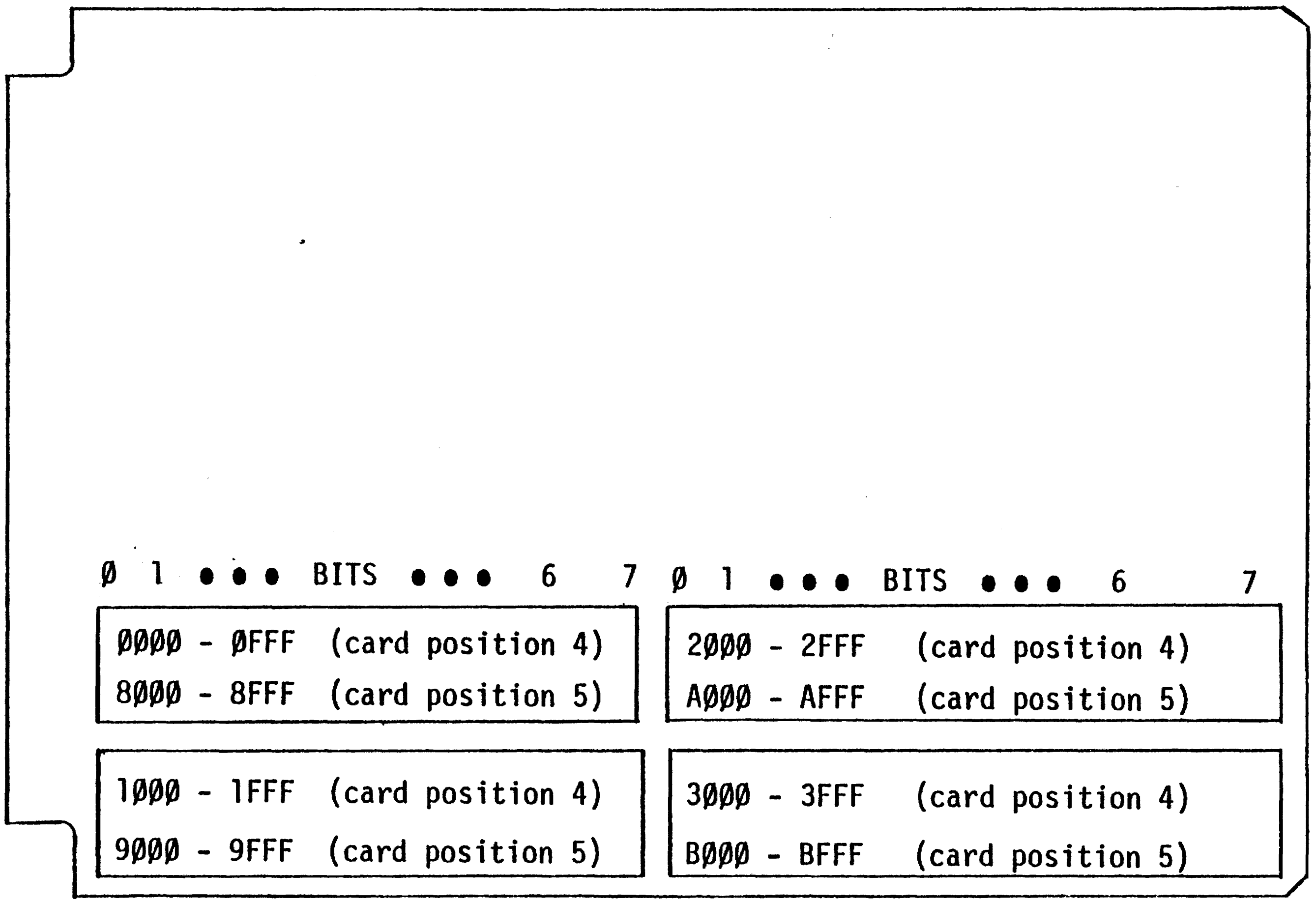
FIGURE 4



Ø 1 ●●● BITS ●●● 6 7
ØØØØ - ØFFF (card position 4)
8ØØØ - 8FFF (card position 5)
1ØØØ - 1FFF (card position 4)
9ØØØ - 9FFF (card position 5)
Ø 1 ●●● BITS ●●● 6 7
2ØØØ - 2FFF (card position 4)
AØØØ - AFFF (card position 5)
3ØØØ - 3FFF (card position 4)
BØØØ - BFFF (card position 5)
Ø 1 ●●● BITS ●●● 6 7
4ØØØ - 4FFF (card position 4)
CØØØ - CFFF (card position 5)
5ØØØ - 5FFF (card position 4)
DØØØ - DFFF (card position 5)
Ø 1 ●●● BITS ●●● 6 7
6ØØØ - 6FFF (card position 4)
EØØØ - EFFF (card position 5)
7ØØØ - 7FFF (card position 4)
FØØØ - FFFF (card position 5)

5000 - 1142 - X - Y 32K RAM BOARD

FIGURE 5


Ø 1 ● ● ● BITS ● ● ● 6 7
Ø 1 ● ● ● BITS ● ● ● 6 7
ØØØØ - ØFFF (card position 4)
8ØØØ - 8FFF (card position 5)
2ØØØ - 2FFF (card position 4)
AØØØ - AFFF (card position 5)
1ØØØ - 1FFF (card position 4)
9ØØØ - 9FFF (card position 5)
3ØØØ - 3FFF (card position 4)
BØØØ - BFFF (card position 5)

5000 - 1155 - X - Y - Z UNIVERSAL MEMORY BOARD

PARRAM

MULTI-ALGORITHM MEMORY TEST WITH PARITY CHECK

PARRAM - MULTI-ALGORITHM MEMORY TEST WITH PARITY CHECK

Applicable Assemblies

5000-11114	CPU-M Card

General Description

The purpose of the PARRAM program is to determine if the Read/Write Random Access Memory (RAM) and the Memory Parity Error Detection is functioning correctly, and if not, to give an indication of which addresses and bits are incorrect. The program requires limited operator interaction while the program is running.

PARRAM is a multi-algorithm RAM test that is self-relocatable or can be operator forced relocatable. The algorithms used in this program are "marching ones and zeros" and "RAM leakage, with neighboring bits refreshed". This program will relocate after each loop to make sure that all portions of memory are tested. Also after each loop, the characters will reverse so that they are not burned permanently on the screen.

16K of memory (addresses 00 to 3FFFH) is required to run PARRAM. This 16K block is known to exist as RAM in all hardware configurations since there must be RAM at addresses 0800H to 0868H for the Display and Input/Output microprocessors, and RAM must exist in 16K (minimum) blocks.

Loading Procedure

PARRAM can be loaded into memory using any convenient available method. When PARRAM loads the screen will go blank for five seconds to allow all hardware vectors to stick.

Operator Action

Once five seconds have elapsed, three command lines will appear (see Figure 1), and an audible beep will be generated. At this time the program is waiting for a "Y", "N" or shifted "HOME", meaning, YES, test memory parity error detection, NO, do not test memory parity error detection, or, shifted "HOME", give the operator control of the test parameters. Also if 'NO' is typed and the error message 'Parity Malfunction' appears, IFL bit 5 is stuck low.

The total range of memory is divided into 4 blocks, each of 16K length. These blocks are displayed as 4 groups on the second and third lines of the display screen, as shown in Figure 1. Each group representing a block of 16K is delimited by a reversed hexadecimal digit representing the most significant digit of the beginning address of that 16K block. The portions of RAM to be tested are indicated by an (*) on the second line, corresponding to the most significant digit of the beginning address of that 16K block. Areas of non-existent RAM or of existent ROM or PROM (which cannot be tested by PARRAM) are denoted by the absence of an (*). The beginning address of the program is displayed at "Program Location" on the second line of the display screen as shown in Figure 1.

Before the user indicates whether or not to check the Memory Parity Error Detection a cursory RAM check is performed, and asterisks are automatically filled in for every 16K block which consists of RAM. After the Memory Parity Error Detection question is answered, testing commences.

At any time while PARRAM is testing, the operator may change the portion of memory to be tested and continue either with or without a stop on error detection. However, a pause might be discerned after the operator types "SHIFTED HOME" and before PARRAM beeps to acknowledge that the operator has control.

The following is a list of valid test parameter commands:

Home (with SHIFT key) - Will give operator control and allow the operator to change test parameters.

Cursor Right (C3 key) - Stop testing and move the cursor on the second line of the display screen, one position to the right. This is in preparation for changing the character at the cursor from a space to an asterisk or vice versa.

Cursor Left (C1 key) - Stop testing and move the cursor one position to the left.

Asterisk (unshifted) - Stop testing, write an asterisk at the cursor position, and move the cursor one position to the right. The asterisk signifies that the 16K block so marked will be tested when testing resumes.

Space (SPACE bar) - Stop testing, clear the character at the cursor position, and move the cursor one position to the right. The absence of an asterisk signifies that the 16K block so marked will be skipped when testing resumes.

Restart (PROG key) - Return to the beginning of PARRAM, wait for a "Y" or "N", meaning whether or not to test Memory Parity Error Detection; re-search and mark all existing RAM with an asterisk, clear the loop count, error count, actual and expected values, and commence testing with no stop on error detection and relocation enabled.

Relocate (R key, unshifted) - Immediately forces relocation to occur to the next chip currently marked with an asterisk, and which does not have any errors reported in it. An "R" will replace the asterisk in the activity flag during this process, and an asterisk will replace the R after the operation is complete.

Go on error (G key, unshifted) - Continue testing the 16K blocks that are currently marked with an asterisk and do not stop testing on error detection. The Go mode is useful if the operator wishes to determine the overall integrity of the RAM.

Stop (S key, unshifted) - Continue testing the 16K blocks that are currently marked with an asterisk but halt testing and beep if an error is detected. The stop mode is useful if the operator wished to analyze the various address and data values existing when an error is detected. An "S" will appear on the first line of the display screen (see Figure 1) to indicate Stop on error mode. This mode can be reset by typing "G" for "Go on error", or the PROG key to restart the program.

No Relocating (N key, unshifted) - Continue testing without relocating the program after each loop. Typing the N key will put an "N" on the first line of the display screen (see Figure 1). This can only be reset by typing the PROG key.

Errors

All errors except one are indicated by PARRAM by incrementing the error count and updating the actual and expected values displayed on the first line of the display screen, as shown in Figure 1. Also if the Stop mode is in effect, each error will beep and halt testing, thereby allowing the operator to examine the updated error information on the third line of the display screen. To continue testing where the error occurred the key marked "C" has to be depressed.

The exception to this type of error is if the Memory Parity Error Detection hardware does not work correctly. If the operator decides to test the Parity Hardware and there is a malfunction an error message will be displayed on the third line of the display screen. At this time the operator has to re-evaluate the integrity of the hardware, and can try again by depressing one of the three keys described in the Operator Action section.

Data errors are caused by failure of a RAM chip to "hold" the data written into it by a previous write pass. This could be the result of a faulty or marginal chip, incorrect insertion of a chip in it's socket, or grounded, floating, or interconnected data lines. A faulty or incorrectly inserted chip or bad data line will generate very many errors (over 0400 hex for each 16K block) per loop. A marginal chip will usually have only a few errors recorded for each loop. All data errors are characterized by having the actual and expected values differ by at most only a few bits, i.e. ACT=DE, EXP=FF.

In addition to the individual error values updated each time an error is detected, "stuck" data bits are displayed on line 3 of the display screen. There are 4 stuck bits displays, one for each 16K block in the 64K memory.

Each stuck bits display shows the accumulated errors for the parity bit and all eight data bits of the 16K block that is characterized by a reversed hexadecimal digit representing the most significant digit of the beginning address of that 16K block, which is displayed to the left of each group. The individual bits are easily identified; P for parity, bits 7,6,5,4,3,2,1 and 0 are displayed from left to right on the second display line as shown in Figure 1. Each bit is displayed as one of the following characters:

CHARACTER	MEANING
- (hyphen)	The bit was always correct.
0	The bit was incorrectly low when it should have been high.
1	The bit was incorrectly high when it should have been low.
X	The bit was incorrectly low and high at different times.

An error at a known bit and in a known address can be mapped to a particular chip using Figure 2. Assume that the error shown in Figure 1 is detected, i.e., P bit at address 3FFFH is stuck low.

When looking at Figure 2

Bank 0 = addresses 00000-3FFFH
Bank 1 = addresses A000H-7FFFH
Bank 2 = addresses 8000H-BFFFH
Bank 3 = addresses C000H-FFFFH

on a 64K machine. The address shown in Figure 1 (3FFFH) is located in Bank 1, and the bit that caused the error was bit P.

Test Description

When PARRAM is first loaded, or when the PROG key is depressed by the operator, a cursory RAM existance check is performed in order to fill in the initial asterisks. The cursory check uses the following algorithm:

1. One location in the 16K block to be checked is read.

2. The inverse of the original contents is written to the location.

3. The location is read again and the new contents are compared with the original contents. If the new contents and the original contents differ, even if only by one bit out of the eight bits, the 16K block is assumed to exist as RAM and an asterisk is written in the appropriate location on the display screen. If the new and original contents are identical, the assumption is made that the 16K block is non-existant RAM or existing ROM or PROM and no asterisk is written.

4. Steps 1 through 3 are repeated for all 16K blocks which could exist as RAM.

After the cursory RAM existance check is complete, the Memory Parity Error Detection hardware is checked only if the operator decided to do so. This hardware check is done using the following algorithm:

1. Parity is changed from odd (normal state) to even parity.

2. A value is written into a memory location.

3. Parity is changed back to the normal state (from even to odd).

4. The value is read back from that same location, causing an expected error. If no error occurred then an error message is put on the display screen and PARRAM jumps back to the initial parity question to try again.

After the Memory Parity Error Detection question is answered, testing commences. A multi-algorithm testing sequence is used which proceeds as follows:

1. Load 00 into all locations in all 16K blocks under test.

2. Read from the first location in the first 16K block and compare the actual value with the expected value. Update any errors to the display screen.

3. Over-write the inverse of the expected value to the current location.
4. Repeat steps 2 and 3 for all locations in all 16K blocks under test, with the current location going forward (from low addresses to high addresses).
5. Read from the last location in the last 16K block and compare the actual with the expected value (which is now the inverse of the value in step 1).
6. Over-write the inverse of the expected value to the current location.
7. Repeat steps 5 and 6 for all locations in all 16K blocks under test, with the current location going backward (from high addresses to low addresses).
8. Repeat steps 1 through 7 seven times with the hex values 01, 03, 07, 0F, 1F, 3F, 7F.
9. Load 0FF hex into all locations in all 16K blocks under test.
10. Overwrite the inverse of the original value to the current location.
11. Refresh the contents of the 8 surrounding bits on each chip.
12. Check the value of the current location on each chip to verify no leakage occurred into or out of the bit under test on each chip.
13. Repeat steps 10, 11 and 12 for all groups of 16K chips under test.
14. Repeat steps 10 through 13 for all 16K locations in each 16K chip.
15. Repeat steps 9 through 14 with the value 00H.
16. a) Overwrite the activity flag (flashing asterisk) with an R signifying relocating in process.

    b) Starting with the program location, search the RAM under test for the next 16K block that remained error free up to this point (wrapping around to the beginning if necessary).

    c) After a new destination for the program is located, re-write the program to this location.

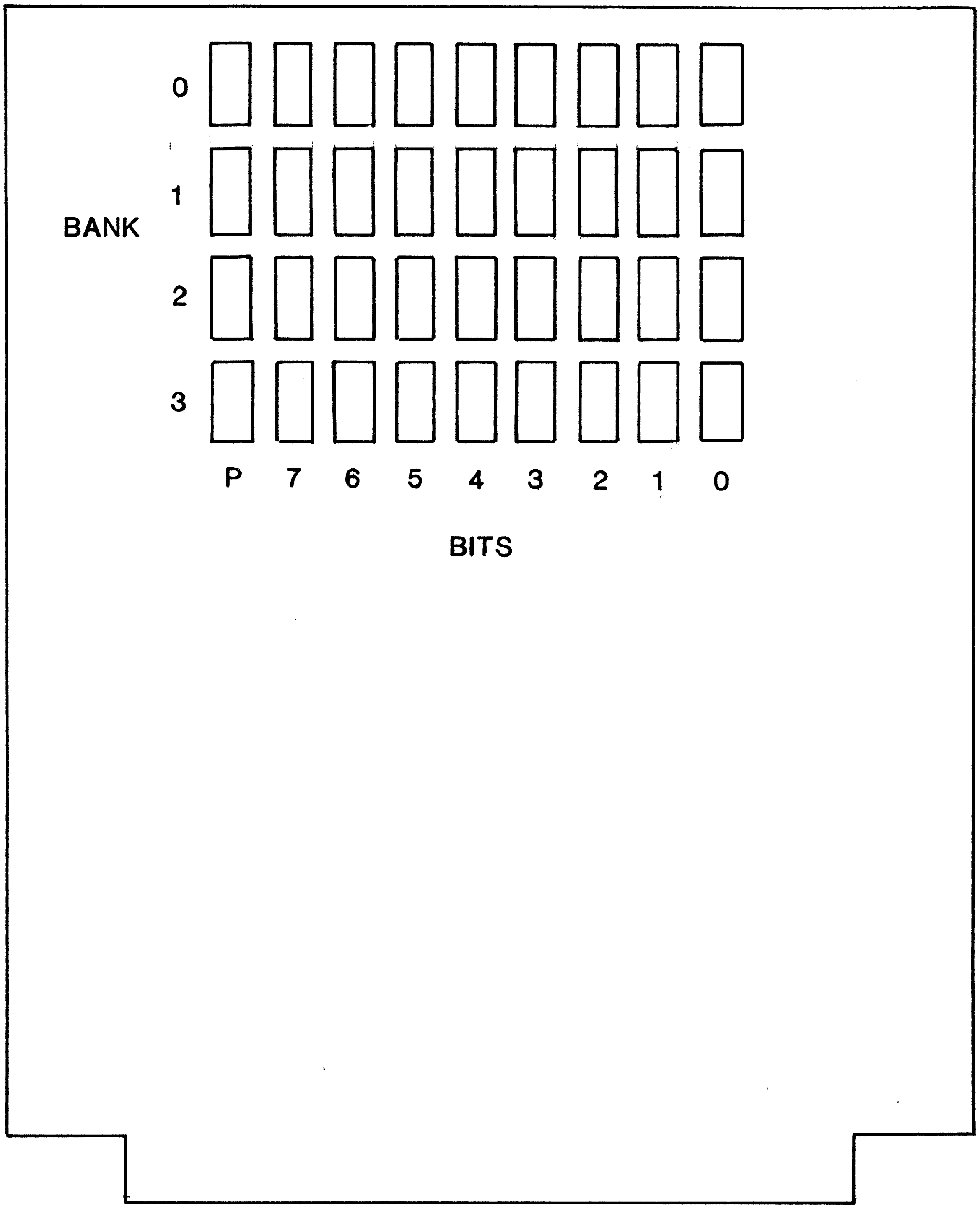
    d) Overwrite the R with the activity flag once again.

    e) Increment the loop counter on the display and continue testing from this new area.
17. Repeat steps 1 through 16 until operator intervention.

```
PARRAM    1.0     ERRS=0001            ADDR=3FFF   ACT=3F   EXP=3F      N S *
LOOP=----
*P76543210*P76543210*P76543210*P76543210          TEST PARITY - Y      PGM
LOC=1028
00--------4---------8---------C---------
```

PARRAM SCREEN DISPLAY

- Figure 1 -



CPU-M CARD

FIGURE 2

PRNTST

CENTRONICS PRINTER TEST

PRNTST - CENTRONICS/OKIDATA PRINTER TEST

Applicable Assemblies

5000-1101-1	Printer Controller Board	
5000-1127-X	Byte String Controller Board	(X=1-4)
5000-1185-1	Printer Controller Board *	

General Description

The purpose of the PRNTST program is to determine if the Centronics or Okidata printer and the Printer Controller are working properly and, if not, to give an indication of which print functions are incorrect. The program requires assistance from the operator to perform certain actions and analyze the PRINTOUT (see Figures 1-4) at the end of the last test.

8K of memory is required to run PRNTST.

This manual applies to both 8008 and 8080 versions of PRNTST.

Loading Procedure

PRNTST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If PRNTST loads properly, it will identify itself and immediately start Test 00.

Operator Action

The operator must answer three questions after PRNTST initially loads by depressing the proper key. These questions are the type of Printer (Okidata or Centronics), column width (80 or 132) and whether an 1185-1 board.

The first three tests (Tests 00, 01 and 02) ask the operator to manually control the printer. A message appears on the display screen and PRNTST waits for the operator to follow the direction. The operator indicates compliance by typing the space bar after performing the requested operation. When the space code is sensed, PRNTST continues to the next part of the test or to the next test.

After Test 02, all tests except Test 12 execute automatically. When the entire test has been completed, the prompt "TEST COMPLETED, TYPE SPACE TO REPEAT PRNTST" will be displayed. If a space is typed on the keyboard, PRNTST will be re-run.

Refer to Appendix A for specialized test run options.

* NOTE: 5000-1185-1 Board requires a COMTST8 Diagnostic Plug to execute Test 12.

Errors

All program detectable errors are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The bell will ring only to notify the operator of an error. The error message displayed on the screen attempts to give a description of the nature of the problem. In most cases, this message should be adequate to diagnose and fix the error. Otherwise, refer to the detailed description of the specific test to determine the purpose and expected results for the displayed error message.

After an error message is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, and depressing the PROG key will restart the program.

Even if PRNTST proceeds from Test 00 to Test 10 without a displayed error, there could still be a printed error since PRNTST has no way to examine the characters printed by the Printer. After the final test the operator must compare the print-out produced by the tests to the correct corresponding PRINTOUT shown in Figures 1-4. Also, during Test 08, the operator must listen for the Printer to beep as the line "BELL SHOULD BE AUDIBLE" is printed.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what is being tested. Below the description, all possible error messages are listed, with an explanation of the cause of the message. However, in addition to the specified messages, other messages may be displayed. These messages are general to all the tests and are listed below:

PLEASE MAKE PRINTER READY

(Turn on printer, plug printer into OP-1, select printer with "Select" pushbutton).

PLEASE SELECT PRINTER
(Select printer with "Select" pushbutton).

Test 00 - Test of Controller Select

Tests that the Printer Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selected a selected Printer Controller. The correct device select address is 0B4 hex. The incorrect select addresses are those single byte addresses whose high order nibbles "or-ed" with their low order nibbles do not result in 0F hex. This is derived from the fact that a 4 input AND gate performs the device selection from the address bits of the select address bits of the select address. Thus a select address of 0FF hex would attempt to select all devices.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL to the select address XX hex gave a result other than 0FF hex (open bus).

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (B4)

IFL to the correct Printer Controller address (0B4 hex) gave a result of 0FF hex (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Printer Controller, IFL did not get a result of 0FF hex (open bus).

Test 01 - Test of Printer Selected Flag

Tests that the Select bit (IFL bit 6) from the printer "Select" pushbutton can be read by the software.

ACTUAL IFL BITS = XXXX,XXXX
ERROR DETECTED AT IFL BIT 6: SELECTED

Either IFL bit 6 was low when it should have been high, or high when it should have been low.

Test 02 - Test of Printer Not Ready Flag

Tests that the Not Ready bit (IFL bit 1) from the printer hardware can e read by the software.

ACTUAL IFL BITS = XXXX,XXXX
ERROR DETECTED AT IFL BIT 6: SELECTED

Either IFL bit 6 was low when it should have been high, or high when it should have been low.

ERROR DETECTED AT IFL BIT 1: NOT READY

Either IFL bit 1 was low when it should have been high, or high when it should have been low.

Test 03 - Test of Controller Not Busy Flag and Current Address

Test that the Not Busy bit (IFL bit 7) is set by a DVCL and INIT, cleared for the duration of a print command (COM1), and is set upon completion of print. Also, test that locations PCAH (0823) and PCAL (0822) are initialized to PSAH (0821) and PSAL (0820) by a print command and eventually increment til they equal PTAH (0825) and PTAL (0824) upon print completion.

NOT BUSY FLAG IS NOT SET BY INIT AND DVCL

After an INIT and DVCL, IFL bit 7 was low when it should have been high.

COM1 DOES NOT RESET NOT BUSY FLAG

After a print command, IFL bit 7 was high when it should have been low.

CURRENT ADDRESS IS NOT INITIALIZED CORRECTLY

After a print command, PCAH and PCAL were not loaded to the address in PSAH and PSAL.

NOT BUSY FLAG DOES NOT SET AFTER A PRINT

About two seconds after the print command was given, IFL bit 7 was still low when it should have been high.

CURRENT ADDRESS DOES NOT STOP AT CORRECT ADDRESS

After the print command is finished, PCAH and PCAL are not equal to PTAH and PTAL. (Note that the print message is set up such that termination on character and address will occur at the same location.)

Test 17 Tests that no interrupt is generated when the Not Busy flag is low, the interrupt mask is set to allow an interrupt only from the Printer Adapter, and the interrupts are enabled - disabled. If this test fails, the interrupt priority for the slot that the board is in should be checked with that entered during initialization. If the interrupt priorities match then the interrupt is bad.

Tests 18-21 are visual tests.

Tests 18-1B should be run in sequence.

Test 18 Test of Form Feed. Paper should advance to top of form. "TOP OF FORM" should be printed on top line.

Test 19 Test of carriage return/line feed. Paper should advance one line "NEXT LINE DOWN" should be printed.

Test 1A Test of Carriage Return. On Okidata Printer the issuing of a Carriage Return also causes a line feed to occur. The message printed should be:

"〉〉〉〉〉〉 ERROR IF X's BEFORE THIS"

On Centronics Printers the internal buffer is printed and the print head is positioned at the beginning of that line. The message printed should be "XXXXXX ERROR IF NOT X's BEFORE THIS".

On the Printronix Printer the internal line buffer is not printed and the internal buffer pointer is positioned to the start of the buffer. The message: "////// ERROR IF X's BEFORE THIS".

Test 1B Test of Vertical Tab. The paper should advance to the sixth line and the message printed should be: "6 LINES BELOW TOP OF FORM". The vertical tab on the Printronix requires the use of the VFO therefore, three carriage return/line feeds are issued and the Vertical Tab is not tested.

Test 1C Tests that all characters are printable.

SECTION 1 6 slewed lines of upper case characters ASCII codes 20-5F hex.

SECTION 2 6 slewed lines of lower case characters ASCII codes 60-7E hex, repeated.

If the printer does not support lower case characters, Section 2 will be printed in upper case.

Test 04 - Test of Controller Terminating Character and Address

Tests that locations PCAH and PCAL increment until they point to a character matching the character in PTC if PTAH bit 7 is low, but increment until they equal PTAH and PTAL if PTAH bit 7 is high.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER

After a print command with a terminating character before the terminating address, PCAH and PCAL are not equal to the address of the terminating character.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING ADDRESS

After a print command which had a character equal to PTC, but also had PTAH bit 7 high, PCAH and PCAL are not equal to PTAH and PTAL.

Test 05 - Test of Controller Hold Off after Print Command

Tests that during a time consuming printer function such as Carriage Return (with data in the Printer internal line buffer), the printer does not acknowledge receipt of the next character, thereby holding off data transfer from the printer controller.

PRINTER BUSY SIGNAL DOES NOT HOLD OFF CONTROLLER

During a print command with a Carriage Return and Line Feed in the middle, PCAH and PCAL continuously incremented, with no hold off pause, until they reached PTAH and PTAL.

PRINTER BUSY SIGNAL HOLDS OFF CONTROLLER AT INCORRECT ADDRESS

During a print command with a Carriage Return and Line Feed in the middle, PCAH and PCAL paused somewhere between PSAH, PSAL and PTAH, PTAL but not at the correct place (the address of the Carriage Return.)

Test 06 - Test of INIT and DVCL and COM1

Tests that INIT and DVCL clear the Printer internal line buffer. Also tests that INIT, DVCL, and COM1 abort any previously unfinished print command.

INIT DOES NOT CLEAR PRINTER BUFFER

A message left in the Printer internal line buffer was not annihilated by an INIT.

DVCL DOES NOT CLEAR PRINTER BUFFER

A message left in the Printer internal line buffer was not annihilated by a DVCL.

DE-SELECT COMMAND DOES NOT DE-SELECT PRINTER

Sending a De-select control (ASCII control DC3) in the middle of a print lien did not reset IFL bit 6.

COM1 DOES NOT RESET NOT BUSY FLAG

Attempt to make Not Busy Flag (IFL bit 7) stick low by seding a De-select control (ASCII control DC3) in the middle of a print line was unsuccessful. IFL bit 7 was high.

INIT DOES NOT ABORT A PRINT

INIT did not make a stuck low IFL bit 7 go high.

DVCL DOES NOT ABORT A PRINT

DVCL did not make a stuck low IFL bit 7 go high.

COM1 DOES NOT ABORT A PRINT

A new print command issued during a previously unfinished command did not re-initialize PCAH and PCAL to PSAH and PSAL.

AUTO-SELECT COMMAND DOES NOT SELECT PRINTER

Sending, to a De-selected Printer, a Select control (ASCII control DC1) as the first character in a print message did not make IFL bit 6 go high.

Test 07 - Test of Controller Interrupt

Tests that with master interrupt enabled and printer interrupt mask bit set, the Printer Controller Interrupts with the Not Busy bit (IFL bit 7) set, and does not interrupt with the Not Busy bit reset.

PRINTER CONTROLLER DOES NOT INTERRUPT WITH NOT BUSY FLAG SET

No interrupt is generated with the printer interrupt mask bit set, master interrupt enabled, and IFL bit 7 high.

DE-SELECT COMMAND DOES NOT DE-SELECT PRINTER

An attempt to make the printer busy by sending a De-select control (ASCII control DC3) in the middle of a print line did not reset IFL bit 6.

COM1 DOES NOT RESET NOT BUSY FLAG

Attempt to make Not Busy flag (IFL bit 7) stick low by sending a De-select control (ASCII control DC3) in the middle of a print line was unsuccessful. IFL bit 7 was high.

PRINTER CONTROLLER INTERRUPTS WITH NOT BUSY FLAG RESET

An interrupt is generated with the printer interrupt mask bit set, master interrupt enabled, and IFL bit 7 low.

COM1 DOES NOT ABORT A PRINT

A new print command issued during a previously unfinished command did not re-initialize PCAH and PCAL to PSAH and PSAL.

AUTO-SELECT COMMAND DOES NOT SELECT PRINTER

Sending, to a De-selected Printer, a Select control (ASCII control DC1) as the first character in a print message did not make IFL bit 6 go high.

Test 08 - Test of Printer with only Visual Error Detection

Visual and Audible test of all printer control codes as listed below:

Form Feed (ASCII FF) - move to top of next sheet of paper.

Carriage Return (ASCII CR) - print all characters in internal line buffer.

Line Feed (ASCII LF) - move paper up one line.

Vertical Tab (ASCII VT) - for the Centronics, move to the next group of eight lines on the paper. For the Okidata, move forward the number of lines specified (5).

Expanded (ASCII SO) - print the following line in double width.

Delete (ASCII DEL) - clear internal line buffer.

Bell (ASCII BEL) - ring bell.

Also, for the Centronics Printer, one line is written with overstruck characters to confirm that the Centronics Printer does not automatically line feed upon receiving a CR. For both Printers, another line is written in lower case ASCII characters to confirm that lower case characters are being printed as their upper case equivalents on the Centronics Printer, but in lower case on the Okidata Printer.

No error messages are possible.

Test 09 - Test of Printer with only Visual Error Detection

Visual test of printing all 64 printable upper case ASCII characters. The characters are printed twice.

No error messages are possible.

Test 10 - Test of Printer with only Visual Error Detection.

Visual test of printing the foxer message. The following message is printed twice:

THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.

No error messages are possible.

Test 11 - Test of Printer with only Visual Error Detection

Visual test that all columns are printable.

No error messages are possible.

Test 12 - Testing SYD and L2

This test is only executed if the Printer Controller is an 1185-1. It is first verified that the SYD and L2 signals can be raised and lowered by appropriate COM3 commands. The operator is then asked to insert a diagnostic plug. The plug takes the SYD and L2 outputs and reroutes them into the HWA and BUSY inputs. This allows the program to verify that SYD and L2 signals can be successfully outputted off the controller into the external environment.

SYD STUCK HIGH

A COM3 command with bit 1 low did not lower IFL bit 4.

SYD STUCK LOW

A COM3 command with bit 1 high did not raise IFL bit 4.

L2 STUCK HIGH

A COM3 command with bit 0 low did not lower IFL bit 5.

L2 STUCK HIGH

A COM3 command with bit 0 high did not raise IFL bit 5.

SYD OFFBOARD OUTPUT STUCK HIGH

Raising SYD did not lower HWA.

SYD OFFBOARD OUTPUT STUCK LOW

Lowering SYD did not raise HWA.

L2 OFFBOARD OUTPUT STUCK HIGH

Raising L2 did not lower BUSY.

L2 OFFBOARD OUTPUT STUCK LOW

Lowering L2 did not raise BUSY.

TEST 13 - Testing COM2 with only Visual Error Detection

This test is only executed if the controller is an 1185-1. The test is identical to Test 10, except that a COM2 rather than a COM1 is used to start a print operation.

```
TEST NOT BUSY FLAG AND CURRENT ADDRESS
TEST TERMINATING CHARACTER
TEST TERMINATING ADDRESS
TERMINATING CHARACTER WAS CORRECTLY IGNORED
TEST BUSY SIGNAL
FROM PRINTER


TEST AUTO-SELECT
TEST AUTO-SELECT
```

```
TOP OF PAGE
NEXT LINE DOWN




8 LINES DOWN (VERTICAL TAB)
E X P A N D E D
these were sent as lower case

BELL SHOULD BE AUDIBLE
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I12
1       10        20        30        40        50        60        70        80        90       100       110       120       130↑↑
```

FIGURE 1

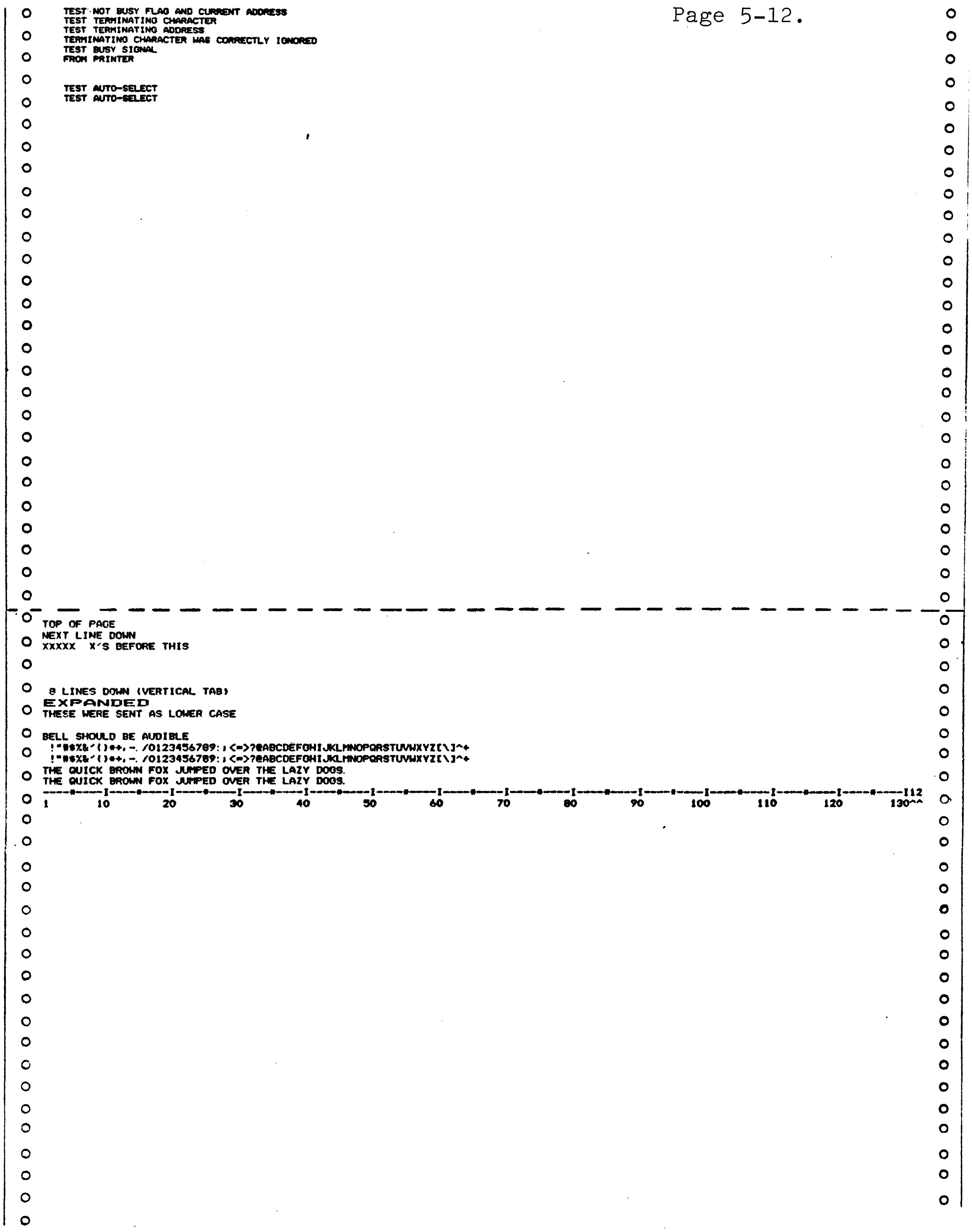
132 COLUMN OKIDATA

```
TEST NOT BUSY FLAG AND CURRENT ADDRESS
TEST TERMINATING CHARACTER
TEST TERMINATING ADDRESS
TERMINATING CHARACTER WAS CORRECTLY IGNORED
TEST BUSY SIGNAL
FROM PRINTER


TEST AUTO-SELECT
TEST AUTO-SELECT
```

```
TOP OF PAGE
NEXT LINE DOWN






8 LINES DOWN (VERTICAL TAB)
EXPANDED
these were sent as lower case

BELL SHOULD BE AUDIBLE
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I
1        10        20        30        40        50        60        70        80
```

FIGURE 2

80 COLUMN OKIDATA



```
TEST NOT BUSY FLAG AND CURRENT ADDRESS
TEST TERMINATING CHARACTER
TEST TERMINATING ADDRESS
TERMINATING CHARACTER WAS CORRECTLY IGNORED
TEST BUSY SIGNAL
FROM PRINTER


TEST AUTO-SELECT
TEST AUTO-SELECT
```

```
TOP OF PAGE
NEXT LINE DOWN
XXXXX  X'S BEFORE THIS


 8 LINES DOWN (VERTICAL TAB)
EXPANDED
THESE WERE SENT AS LOWER CASE

BELL SHOULD BE AUDIBLE
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^+
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^+
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I12
1       10        20        30        40        50        60        70        80        90        100       110       120       130^^
```

FIGURE 3

132 COLUMN CENTRONICS

```
TEST NOT BUSY FLAG AND CURRENT ADDRESS
TEST TERMINATING CHARACTER
TEST TERMINATING ADDRESS
TERMINATING CHARACTER WAS CORRECTLY IGNORED
TEST BUSY SIGNAL
FROM PRINTER


TEST AUTO-SELECT
TEST AUTO-SELECT
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - -
TOP OF PAGE
NEXT LINE DOWN
XXXXX  X'S BEFORE THIS


 8 LINES DOWN (VERTICAL TAB)
EXPANDED
THESE WERE SENT AS LOWER CASE

BELL SHOULD BE AUDIBLE
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^+
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^+
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS.
----*----I----*----I----*----I----*----I----*----I----*----I----*----I----*----I
1        10        20        30        40        50        60        70        80
```

FIGURE 4

80 COLUMN CENTRONICS

PRNEX

CENTRONICS / OKIDATA / PRINTRONIX PRINTER EXERCISER

PRNEX - PRINTER EXERCISE PROGRAM

Applicable Assemblies

5000-1101-1	Printer Controller
5000-1127-X	Byte String Controller
5000-1170	Word Mover Controller

General Description

PRNEX is an operator usable printer exercise program that allows for visual inspection of all printable characters on a Centronics, Printronix or Okidata Printer.

Loading Procedure

PRNEX can be loaded into memory using any conveniently available loading procedure. When loaded properly, it will identify itself and await operator input. At this point, the printer should be plugged in, turned on and selected.

Operator Action

The following message is displayed on the screen when PRNEX has been loaded:

"TYPE 1 FOR 80 COLS OR 2 FOR 132 COLS"

If "1" is typed, the buffer size for all printed lines will be 80 characters. If "2" is typed, the buffer size will be 132 characters.

The next message to appear on the screen will be:

"INPUT: SPACE - ALL CHARS, RETURN - OPEN BUFFER, PROG - RESTART"

If the SPACE bar is typed, the printer will commence printing successive lines of all printable ASCII characters (20H-7EH) and will continue until the PROG key is depressed. The first character on the first line will be 20H (ASCII SPACE). The first character on the next line will be 21H (ASCII !), the next, 22H, etc., producing a skewed column effect.

If a printable ASCII character other than SPACE is typed, the printer will commence printing successive lines of that character and will continue until the PROG key is depressed.

Typing the RETURN key allows the operator to then type ASCII characters, including control characters; to the buffer visible on the display screen. The following control characters are recognized:

Bell (07H) - control 'G'
Vertical Tab (0BH) - control 'K'
Form Feed (0CH) - control 'L'
Expanded Mode (0EH) - control 'N'
Delete (7FH) - shifted 'DEL'

Printable alphabetic characters typed to the buffer will be upper case when the F0 key is lighted.

When the RETURN key is typed again, the printer will commence printing successive lines consisting of the characters in the screen buffer. The printing operation will continue until the PROG key is depressed.

DSTTST

DISKETTE CONTROLLER TEST

DSTTST - DISKETTE CONTROLLER TEST

Applicable Assemblies

5000-1135-1	Diskette Controller Board

General Description

The purpose of the DSTTST program is to determine if the Diskette Drives and Diskette Controller are working properly and, if not, to give an indication of which functions are incorrect. The program requires assistance from the operator to perform certain actions in the IFL test section.

16K (or more) of memory is required to run DSTTST.

This manual applies to the 8080 versions of DSTTST only.

Loading Procedure

DSTTST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If DSTTST loads properly, it will identify itself and pause before requesting test parameters.

Operator Action

On line 3 of the display screen the message "Type (S)ynchronous; (A)synchronous" will appear. At this time type an (S) if there is a Synchronous adapter or an (A) if there is an Asynchronous adapter. After answering the previous question, the following messages appear on the display screen and permit the operator to change the test parameters or to leave the defaults as they are:

*SKIP IFL TESTS (Y OR N)? =Y
This option skips the lengthy operator interactive "IFL tests" (Tests 0-9) and proceeds immediately to the "run time test" (Test 10) which continuously exercises the diskette. This section must be run at least once through.

*STOP AT AN ERROR (Y OR N)? =N
Tests 8, 9, and 10 will not stop at an error unless the operator answers Yes. When DSTTST does stop at an error, the operator is given control.

*NUMBER OF SECTORS (1-16)? =16
The number of the highest sector at which an operation will start in Test 10. The legal values are dependent on buffer size and are listed in Figure 1.

*DRIVE TO BE TESTED (0-3)? =1
All tests are performed using this drive.

*TRACKS TO BE TESTED (MIN-0, MAX-76)? =0,2,4,32,76
The tracks at which operations will be performed in Test 10. Single tracks must be separated by a comma or a space. If a hyphen separates two numbers, all tracks between and including the two numbers are selected.

*BUFFER SIZE (256 BYTE BLOCKS)? =16
This specifies a buffer length of 256 bytes times the inputted number to be used for operations in test 10. A range of 1-17 is permitted.

*OPERATIONS (A-ALL R-READ W-WRITE)? =A
Operations of read only, write only, or both write and read can be specified for test 10.

RETRY ATTEMPTS BEFORE ERROR? =0
Any number between 0 and 255 will be accepted. During the run time test (test 10) DSTTST will re-execute an operation in which an error is detected this many times before the error is reported.

The RETURN key should be typed after all of the changes have been made.

All tests except 2, 3, and 7 execute automatically and without operator intervention. Tests 2, 3, and 7 ask the operator to manually turn the diskette drive on and off and to protect and unprotect the diskette.

Errors

All errors are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The error message displayed on the screen attempts to give a description of the nature of the problem. In some cases, this should be adequate to diagnose and fix the error. Otherwise, refer to the detailed description of the specific test to determine the purpose and expected results for the displayed error message. For the IFL status tests, the message in Figure 2 is applicable.

After an error message is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, and depressing the PROG key will restart the program.

Because of the inverted pyramid test strategy used in tests 0-9 of DSTTST, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

Test Description

All test operations are described in this section. The program will halt and the specified error message will be displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing for on the top of the page. Below the description, all possible error messages are listed, with an explanation of the cause of the message.

BUFFER SIZE (256 BYTE BLOCKS)	NUMBER OF SECTORS
1	1-16
2	1, 3, 5, 7, 9, 11, 13, 15
3	1, 4, 7, 10, 13
4	1, 5, 9, 13
5	1, 6, 11
6	1, 7•
7	1, 8
8	1, 9
9	1, 9
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1

LEGAL NUMBER OF SECTORS
FOR VARIOUS BUFFER SIZES
Figure 1

Test 00

Tests that the Diskette Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selects a selected Diskette Controller.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL to the select address XX hex gave a result other than OFF hex (open bus).

CONTROLLER NOT SELECTED WITH CORRECT ADDRESS (5A)

IFL to the correct Diskette Controller address (05A hex) gave a result of OFF hex (open bus).

INIT DOES NOT DE-SELECT CONTROLLER

After an INIT with the Diskette Controller selected, IFL did not get a result of OFF hex (open bus).

Test 01

Tests that the Not Busy flag (IFL bit 7) is set before a Read, reset during a Read, and set again after a Read. Also tests that both DVCL and INIT individually abort a read operation.

NOT BUSY WAS HIGH DURING READ COMMAND

While a read is being executed the Not Busy flag (IFL bit 7) was set.

NOT BUSY WAS LOW AFTER 1 SECOND OF INACTIVITY

One second after a read was executed the controller looked busy.

DVCL DOES NOT ABORT READ COMMAND

During a read DVCL was executed but the controller remained busy.

INIT DOES NOT ABORT READ COMMAND

During a read an INIT was executed but the controller remained busy.

Test 02

Tests the Not Ready flag (IFL bit 1)
Asks the operator to remove the diskette and disconnect power to the drive. Controller tries to increment the head.

NOT READY BIT LOOKS READY

The Controller looks ready with the power disconnected and the diskette removed.

Asks the operator to apply A.C. power to the drive. Controller tries to increment the head.

NOT READY BIT LOOKS READY

The Controller looks ready with the diskette removed from the drive.

For IFL errors see Figure 2.

Test 01

Tests that the Not Busy flag (IFL bit 7) is set before a Read, reset during a Read, and set again after a Read. Also tests that both DVCL and INIT individually abort a read operation.

NOT BUSY WAS HIGH DURING READ COMMAND

While a read is being executed the Not Busy flag (IFL bit 7) was set.

NOT BUSY WAS LOW AFTER 1 SECOND OF INACTIVITY

One second after a read was executed the controller looked busy.

DVCL DOES NOT ABORT READ COMMAND

During a read DVCL was executed but the controller remained busy.

INIT DOES NOT ABORT READ COMMAND

During a read an INIT was executed but the controller remained busy.

Test 02

Tests the Not Ready flag (IFL bit 1)
Asks the operator to remove the diskette and disconnect power to the drive. Controller tries to increment the head.

NOT READY BIT LOOKS READY

The Controller looks ready with the power disconnected and the diskette removed.

Asks the operator to apply A.C. power to the drive. Controller tries to increment the head.

NOT READY BIT LOOKS READY

The Controller looks ready with the diskette removed from the drive.

For IFL errors see Figure 2.

Test 03

Tests the Write Protect flag (IFL bit 0).

Asks the operator to make the diskette write protected. Controller tries to increment the head to check the bit.

For errors see Figure 2.

•

Test 04

Test of the Track Zero flag (IFL bit 3).

Controller increments the head three times to move it off of track 0 and to read the flags.

TRACK ZERO BIT STUCK HIGH

The Track Zero flag is set when the head is not on track 0.

Controller decrements the head once and then checks for the Track Zero flag. If the head is not over track 0 then the procedure is repeated up to 78 times.
For errors see Figure 2.

Test 05

Test of the Read Error flag (IFL bit 5).

The Controller moves the head to Track Zero and reads the flags.

READ ERROR BIT IS STUCK HIGH

The Read Error flag (IFL bit 5) was set when a transfer did not take place.

DSTTST assumes that track 0 sector 0 is not formatted as a 256 byte block. It tries to cause a read error by reading track 0 sector 0 as a 256 byte block. If a read error does not occur, then the buffer size is decreased by one byte and another read command is executed. This procedure is repeated up to 200 times.

NOTE: If read errors don't occur try doing the read with a diskette that contains DOS (which is recorded as a 2K byte block starting at track 0 sector 0).

Test 06

Test of the Activity Timeout flag (IFL bit 6).

DSTTST checks the Activity Timeout flag (IFL bit 6) to see that it's reset.

THE TIMEOUT BIT IS STUCK LOW

The Activity Timeout flag (IFL bit 6) was set without an operation being executed.

DSTTST tries to cause the Activity Timeout flag to set by reading a nonexistant sector. The Activity Timeout flag is supposed to set after 600 msec. The tolerance that is being tested is 540-720 msec.

THE TIMEOUT BIT NEVER WENT HIGH

THE TIMEOUT IS TOO LONG LONG

The Activity Timeout flag set, but it took more than 720 msec.

THE TIMEOUT IS TOO SHORT

The Activity Timeout flag set in less than 540 msec.

Test 07

Tests that the status bits are correct with the controller inactive.

DSTTST asks the operator to make the diskette write enabled.

For errors see Figure 2.

Test 08

Tests that 256 and 2048 byte Write operations will transfer data into numerically adjacent sectors and inter-sector gaps.

DSTTST writes either 256 or 2048 bytes in a single Write. The number of bytes to be written is taken from a table. That number of bytes is written starting at sector 0. On the first unwritten sector a different pattern is written to destroy any bytes which might have overflowed onto this sector from the first Write. Finally the original pattern is checked. This procedure is repeated for Writes of 1 sector and 8 sectors.

This test operates on track 0.

For errors see Figure 3.

Test 09

Tests that 256 and 2048 byte Write operations will transfer data onto numerically adjacent sectors and inter-sector gaps.

DSTTST writes either 256 or 2048 bytes in a single Write. The number of bytes to be written is taken from a table. That number of bytes is written starting at sector 0. On the first unwritten sector a different pattern is written to destroy any bytes which might have overflowed onto this sector from the first Write. Finally the original pattern is checked. This procedure is repeated for Writes of 1 sector and 8 sectors.

This test operates on track 76.

For errors see Figure 3.

Test 10

WRITE will perform a Write followed by a Check on each operator requested sector of each selected track. The pattern written is unique for each sector, track, and loop iteration. All error conditions are tested for both Write and Check. If errors are detected, the bell will ring and the error count will be incremented. After completely testing all selected tracks, the loop iteration counter is incremented and the tests repeat.

READ will perform a Read on each operator requested sector of each selected track. All error conditions are tested after each Read is completed. The read data will then be examined (character by character) and a Match error message will be displayed if a bad character is found.

ALL will initiate Write of all selected tracks followed by Read as described above.

All testing of selected tracks is done in a random manner. All writing is done in a unique fashion for each iteration. If a test fails, the total error counter is incremented, the R (Read), W (Write), or M (Match) error counter is incremented, the bell will ring, and the test will stop if the stop after error mode was previously selected. To continue from this stop, depress the SPACE key. If further testing is necessary, the following information is available. See Figure 3.

```
TYPE SPACE TO CONTINUE DSTTST

ACTUAL IFL BITS = 1010,0100

ERROR DETECTED AT IFL BIT 7:  NOT BUSY

ERROR DETECTED AT IFL BIT 6:  TIMEOUT BIT

ERROR DETECTED AT IFL BIT 5:  READ ERROR

ERROR DETECTED AT IFL BIT 3:  TRACK ZERO

ERROR DETECTED AT IFL BIT 2:  FILE INOPERATIVE

ERROR DETECTED AT IFL BIT 1:  NOT READY

ERROR DETECTED AT IFL BIT 0:  WRITE PROTECT
```

NOTE: Any or all messages may be on the screen depending on the error found.

Each message is visible if its corresponding Diskette Controller IFL bit was opposite to what was expected.

IFL ERROR MESSAGES
Figure 2.

RUN TIME TEST ERRORS

If the stop after error mode was previously selected and an error occurs the following messages will be displayed:

TRACK abc SECTOR de LAST REFERENCED WITH A(N) f ERROR FLAGS = gh

TYPE R=READ F=FWRT D=DWRT C=CHK E=ERASE SP=CNT I=RESET W=REWRT

The ERROR (f) will be either R for Read, W for Write, or M for Match

The FLAGS (gh) is the value of the IFL status byte.

The following data entries are allowed.

SP=CNT = Continue testing the rest of the tracks
R=READ = Read the track and sector specified by abc/de
I=RESET = Move to track 0 and then back to the current track
F=FWRT = Format the output buffer (on screen) and write it to the diskette at the track and sector specified by abc/de
D=DWRT = Data Entry into buffer via the keyboard. When a return is typed, write the data to the diskette at the track and sector specified by abc/de
C=CHK = Performs a verify and beeps if error detected
W=REWRT = Writes buffer to the diskette, then performs a verify

(shift) cursor up - scroll screen up (not on display screen)
(shift) cursor down - scroll screen down (not on display screen)
(shift) cursor left - decrement head (not on display screen)
(shift) cursor right - increment head (not on display screen)

NOTE: If the PROG key is typed at any time during execution, the program will restart as if just loaded.

RUN TIME TEST ERRORS
Figure 3.

BSC2

BINARY SYNCHRONOUS II CONTROLLER TEST

BSC2 - BINARY SYNCHRONOUS II CONTROLLER TEST

Applicable Assemblies

5000 - 1192	Binary Synchronous II Controller

Required Test Assemblies

COMTST1 Diagnostic Cable

General Description

The purpose of the BSC2 program is to determine if the Binary Synchronous II Controller is working properly and, if not, to give an indication of which functions are incorrect. Prior to running BSC2, the Asynchronous I/O Adapter must be tested by IOTST and be known good. A COMTST1 Diagnostic Cable is needed to run BSC2. The program requires no operator interaction unless an error is detected.

16K (or more) of memory is required to run BSC2.

This manual applies to the 8080 versions of BSC2 only.

Loading Procedure

BSC2 can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If BSC2 loads properly, it will identify itself and wait for operator action.

Operator Action

At the end of the wait period, BSC2 wants to know if there is an Asynchronous or Synchronous adapter. To communicate with BSC2 type in (A) for Asynchronous or (S) for Synchronous. The next message to appear is the following:

"PLEASE ENTER THE SELECT ADDRESS: C3"

The user may change the select address by entering a two character hex address. The select address will default to C3 by typing a carriage return.

All tests operate automatically and without operator intervention. When the entire test has been completed, the prompt "TYPE SPACE TO REPEAT BSC2" will be displayed. If a space is typed on the keyboard, BSC2 will restart.

Errors

All errors are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The error message displayed on the screen attempts to give a description of the nature of the problem. In some cases, this should be adequate to diagnose and fix the error. Otherwise, refer to the detailed description of the specific test to determine the purpose and expected results for the displayed error message.

After an error message is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, and depressing the PROG key will restart the program.

Because of the inverted pyramid test strategy used in BSC2, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in the subsequent subtest since the function is assumed to be working in all subtests after the one in which it is tested.

COMTST1 Diagnostic Cable

The COMTST1 Diagnostic Cable is necessary for testing the Binary Synchronous II Controller. The cable must be inserted into the Asynchronous I/O Adapter connector and slot D1 (as shown in Figure 1) on the rear of the OP-1 prior to program execution. The schematic of the COMTST1 Diagnostic Cable is shown in Figure 2.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing for on the top of the page. Below the description all possible error messages are listed, with an explanation of the cause of the message.

TEST 00 - CONTROLLER SELECT TEST

Tests that the Binary Synchronous II Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selects a selected Binary Synchronous II Controller. The correct device select address is OC3 hex. The incorrect addresses are those single byte addresses whose high order nibbles "or-ed" with their low order nibbles do not result in OF hex. This is derived from the fact that a 4 input AND gate performs the device selection from the address bits of the select address. Thus a select address of 0FF hex would attempt to select all devices.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL to the select address XX hex gave a result other than 0FF hex (open bus).

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (C3)

IFL to the correct Binary Synchronous II Controller address (OC3 hex) gave a result of 0FF hex (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Binary Synchronous II Controller, IFL did not get a result of 0FF (open bus).

TEST 01 - TEST STATIC FLAGS DATA TERMINAL READY AND DATA SET READY.

Tests that Data Terminal Ready can be set and reset by OFL bit 4, and if set can be individually reset by an INIT and DVCL. Also tests that setting and resetting Data Set Ready can be detected by IFL bit 1. Also tests that with no transmit in progress (Request to Send reset), Transmitted Data is a consistent "1" (-12V.). The tests are conducted using known good flags on the Asynchronous I/O Adapter, and yield conclusive results.

REQUEST TO SEND STUCK HIGH

An INIT and DVCL could not reset Request To Send.

REQUEST TO SEND RESET DOES NOT SET TRANSMITTED DATA TO A STOP BIT

After an INIT and DVCL have reset Request To Send, Transmitted Data is not a "1" (-12V.).

DATA TERMINAL READY STUCK HIGH

An INIT, DVCL, and OFL (with bit 4 reset) could not reset Data Terminal Ready.

DATA SET READY STUCK HIGH

With input to Data Set Ready a known "0" (-12V.). the Data Set Ready flag (IFL bit 0) is set.

DATA TERMINAL READY STUCK LOW

An OFL (with bit 4 set) could not set Data Terminal Ready.

DATA SET READY STUCK LOW

With input to Data Set Ready a known "1" (+12V.) the Data Set Ready flag (IFL bit 0) is reset.

DVCL DOES NOT RESET DATA TERMINAL READY

After a DVCL with Data Terminal Ready set, it is still set.

CLOCKING TRANSMIT WITHOUT A COMI DOES NOT SET TRANSMITTED DATA TO STOP BIT

With Request to Send reset, Transmitted Data is not a "1" (-12V.) while clocking the transmitter eight times.

TEST 02 - TEST TRANSMIT CAPABILITIES AND INTERRUPT

Tests that a Transmit (COMI) resets the Not Busy flag (IFL bit 7), sets Request To Send, waits for Clear To Send and then initializes the Main Channel Current address and increments it until it reaches the Terminating address. Also tests that INIT, DVCL, and reaching the Terminating address individually set the Not Busy flag (IFL bit 7) and reset Request To Send. Also tests that the controller interrupts with IFL bit 7 set.

CURRENT ADDRESS REACHED TERMINATING ADDRESS PREMATURELY

During a Transmit (COMI) of only ASCII SYN Characters the Main Channel Current address reached its terminating address without the expected number of clock pulses.

TRANSMIT DOES NOT SET REQUEST TO SEND

During a Transmit (COMI) Request To Send was "0" (-12V.).

TRANSMIT DOES NOT RESET NOT BUSY

During a Transmit (COMI) the Not Busy flag (IFL bit 7) was set.

TRANSMIT DOES NOT SET REQUEST TO SEND OR RESET NOT BUSY

During a Transmit (COMI) Request To Send was "0" (-12V.) and the Not Busy flag (IFL bit 7) was set.

INIT DOES NOT SET NOT BUSY

During a Transmit (COMI) an INIT did not cause the Not Busy flag (IFL bit 7) to set.

INIT DOES NOT RESET REQUEST TO SEND

During a Transmit (COMI) an INIT did not cause Request To Send to become a "0" (-12V.).

INIT DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

During a Transmit (COMI) an INIT did not cause the Not Busy flag (IFL bit 7) to set and Request To Send to become a "0" (-12V.).

DVCL DOES NOT SET NOT BUSY

During a Transmit (COMI) a DVCL did not cause the Not Busy flag (IFL bit 7) to set.

DVCL DOES NOT RESET REQUEST TO SEND

During a Transmit (COMI) a DVCL did not cause Request To Send to become a "0" (-12V.).

DVCL DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

During a Transmit (COMI) a DVCL did not cause the Not Busy flag (IFL bit 7) to set and Request To Send to become a "0" (-12V.).

CLEAR TO SEND LOW DOES NOT HOLD OFF TRANSMIT

Transmission of data from the Main Channel takes place without Clear To Send a "1" (+12V.).

CURRENT ADDRESS NOT INITIALIZED CORRECTLY

Issuing a Transmit (COMI) does not cause the Main Channel Current address to be initialized to the Starting address.

REACHING TERMINATING ADDRESS DOES NOT SET NOT BUSY

The end of a transmit did not cause the Not Busy flag (IFL bit 7) to set.

REACHING TERMINATING ADDRESS DOES NOT RESET REQUEST TO SEND

The end of a transmit did not cause Request To Send to become a "0" (-12V.).

REACHING TERMINATING ADDRESS DOES NOT RESET REQUEST TO SEND

The end of a transmit did not cause Request to Send to become a "0" (-12V.).

REACHING TERMINATING ADDRESS DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

The end of a transmit did not cause the Not Busy flag (IFL bit 7) to set and Request To Send to become a "0" (-12V.).

CURRENT ADDRESS DOES NOT STOP AT TERMINATING ADDRESS

During a Transmit (COMI) the Not Busy flag (IFL bit 7) set but the Main Channel Current address incremented past the Terminating address.

CONTROLLER DOES NOT INTERRUPT WITH NOT BUSY SET

With the Not Busy flag (IFL bit 7) set an Interrupt is not generated.

CONTROLLER INTERRUPTS WITH NOT BUSY RESET

With the Not Busy flag (IFL bit 7) reset an Interrupt is generated.

TEST 03 - TEST RING DETECTED

Tests the Ring Detect mode, specifically that transitions on Ring Indicator are properly indicated by the Ring Detected flag (IFL bit 0). Also tests the INIT, DVCL, and, under appropriate conditions, COMI reset the Ring Detected flag. Also tests that resetting the Ring Detect mode does not affect the Ring Detected flag.

RING DETECTED STUCK HIGH

An INIT, DVCL, and COMI could not reset the Ring Detected flag (IFL bit 0).

RING DETECTED STUCK LOW

With Ring Indicator a "1" (+12V.) in the Ring Detect mode the Ring Detected flag (IFL bit 0) remains reset.

RESETTING RING INDICATOR RESETS RING DETECTED

After setting the Ring Detected flag (IFL bit 0) by entering Ring Detect mode and making Ring Indicator a "1" (+12V.), making Ring Indicator a "0" (-12V.) causes the Ring Detected flag (IFL bit 0) to reset.

INIT DOES NOT RESET RING DETECTED WITH RING INDICATOR SET

A INIT could not reset the Ring Detected flag (IFL bit 0) while in Ring Detect mode.

DVCL DOES NOT RESET RING DETECTED WITH RING INDICATOR SET

A DVCL could not reset the Ring Detected flag (IFL bit 0) while in Ring Detect mode.

COM1 RESETS RING DETECTED.

A COM1 erroneously reset the Ring Detected flag (IFL bit 0).

RESETTING RING INTERRUPT MODE DOES NOT RESET RING DETECTED.

Resetting the Ring Interrupt mode by issuing a COM3 does not cause the Ring Detected flag (IFL bit 0) to be reset.

RING DETECT STUCK LOW WHEN NOT RING INTERRUPT MODE

With Ring Indicator a "1" (+12V.) while not in the Ring Interrupt mode the Ring Detected flag (IFL bit 0) remains reset.

TEST 04 - ACTIVITY TIMEOUT TEST

Tests that the Activity Timeout flag (IFL bit 3) can be set in 3 seconds (+ or - 10%) by issuing a Receive (COM1) and not receiving a SYN character or by issuing a TRANSMIT (COM1) and not transmitting a data character. Also tests that an INIT, DVCL, and COM1 individually resets the Activity Timeout flag.

ACTIVITY TIMEOUT STUCK HIGH

AN INIT, DVCL, and COM1 could not reset the Activity Timeout flag (IFL bit 3).

ACTIVITY TIMEOUT STUCK LOW

Issuing a Receive (COM1) without clocking the receiver does not set the Activity Timeout flag (IFL bit 3) in a five second time period.

ACTIVITY TIMEOUT SETS BEFORE 2.4 SECONDS

Issuing a Receive (COM1) without clocking the receiver does set the Activity timeout flag (IFL bit 3) but requires less than the minimum allowed time of 2.4 seconds.

ACTIVITY TIMEOUT SETS AFTER 3.6 SECONDS

Issuing a Receive (COM1) without clocking the receiver does set the Activity Timeout flag (IFL bit 3) bit requires more than the maximum allowed time of 3.6 seconds.

INIT DOES NOT RESET ACTIVITY TIMEOUT

After setting the Activity Timeout flag (IFL bit 3) an INIT does not reset it.

DVCL DOES NOT RESET ACTIVITY TIMEOUT

After setting the Activity Timeout flag (IFL bit 3) a DVCL does not reset it.

COM1 DOES NOT RESET ACTIVITY TIMEOUT

After setting the Activity Timeout flag (IFL bit 3) a COM1 does not reset it.

AFTER A COM1 ACTIVITY TIMEOUT DOES NOT SET WITHIN BOUNDS

After setting the Activity Timeout flag (IFL bit 3) a COM1 resets it but does not again set it after 3 seconds (+ or - 10%).

ACTIVITY TIMEOUT DOES NOT SET DURING TRANSMIT WITHOUT SENDING A DATA CHARACTER

Issuing a Transmit (COM1) without clocking the transmitter does not set the Activity timeout flag (IFL bit 3) in a five second time period.

TEST 05 - TIMER TIMEOUT TEST

Tests that the Timer Timeout flag (IFL bit 4) can be set in 1 second (+ or - 10%) by issuing a Start Timer (COM3). Also tests that an INIT, DVCL, Abort Timer, Reset Timer, and Start Timer individually reset the Timer Timeout flag.

TIMER TIMEOUT STUCK HIGH

An INIT, DVCL, Abort Timer, and Reset Timer could not reset the Timer Timeout flag (IFL bit 4).

TIMER TIMEOUT STUCK LOW

A Start Timer (COM3) does not set the Timer Timeout flag (IFL bit 4) in a five second time period.

TIMER TIMEOUT SETS BEFORE 0.8 SECONDS

A Start Timer (COM3) does set the Timer Timeout flag (IFL bit 4) but requires less than the minimum allowed time of 0.8 seconds.

TIMER TIMEOUT SETS AFTER 1.2 SECONDS

A Start Timer (COM3) does set the Timer Timeout flag (IFL bit 4) but requires more than the maximum allowed time of 1.2 seconds.

INIT DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) an INIT does not reset it.

DVCL DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) a DVCL does not reset it.

ABORT TIMER DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) an Abort Timer (COM3) does not reset it.

RESET TIMER DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) a Reset Timer (COM3) does not reset it.

START TIMER DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) a Start Timer (COM3) does not reset it.

CONTROLLER DOES NOT INTERRUPT WITH TIMER TIMEOUT SET

With the Timer Timeout flag (IFL bit 4) set an Interrupt is not generated.

CONTROLLER INTERRUPTS WITH TIMER TIMEOUT RESET

With the Timer Timeout flag (IFL bit 4) reset an Interrupt is generated.

TEST 06 - TEST TRANSMITTING LEAST SIGNIFICANT 7 DATA BITS AND PARITY.

Tests transmission of the parity bit and the least significant 7 data bits excluding control characters (00-01F hex). Also tests that bit 6 of the OFL (LRC) is not stuck low.

NO SYN CHARACTER SENSED

The first 2 characters in the Main Channel transmit buffer are the ASCII SYN character (016 hex). These characters are used to align the bits being received. If none of these characters are sensed, it implies Transmission is not taking place.

TIMER TIMEOUT SETS AFTER 1.1 SECONDS

A Start Timer (COM3) does set the Timer Timeout flag (IFL bit 4) but requires more than the maximum allowed time of 1.1 seconds.

INIT DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) an INIT does not reset it.

DVCL DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) a DVCL does not reset it.

ABORT TIMER DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) an Abort Timer (COM3) does not reset it.

RESET TIMER DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) a Reset Timer (COM3) does not reset it.

START TIMER DOES NOT RESET TIMER TIMEOUT

After setting the Timer Timeout flag (IFL bit 4) a Start Timer (COM3) does not reset it.

CONTROLLER DOES NOT INTERRUPT WITH TIMER TIMEOUT SET

With the Timer Timeout flag (IFL bit 4) set an Interrupt is not generated.

CONTROLLER INTERRUPTS WITH TIMER TIMEOUT RESET

With the Timer Timeout flag (IFL bit 4) reset an Interrupt is generated.

TEST 06 - TEST TRANSMITTING LEAST SIGNIFICANT 7 DATA BITS AND PARITY.

Tests transmission of the parity bit and the least significant 7 data bits excluding control characters (00-01F hex). Also tests that bit 6 of the OFL (LRC) is not stuck low.

NO SYN CHARACTER SENSED

The first 2 characters in the Main Channel transmit buffer are the ASCII SYN character (016 hex). These characters are used to align the bits being received. If none of these characters are sensed, it implies Transmission is not taking place.

CHARACTER WITH HEX VALUE XX WAS SENSED AS A CONTROL CHARACTER

The Secondary Channel contains only 0FF hex characters so there should not be any Active Control Characters. During transmission if the Active Control Character flag (IFL bit 6) goes high the character pointed to by the Current address of the Secondary Channel is the XX in the message.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits transmitted. The leftmost bit reflects the status of the parity bit and the next 7 from left to right are bits 6 through 0 respectively.

A = 0 if that bit is stuck low;
 1 if that bit is stuck high;
 X if that bit is not stuck but was incorrect at one point;
 - if that bit worked properly.

TEST 07 - TEST TRANSMITTING ALL 8 DATA BITS.

Tests transmission of all 8 data bits. Also tests that bit 6 of the OFL (CRC) is not stuck high.

NO SYN CHARACTER SENSED

The first 2 characters in the Main Channel transmit buffer are the ASCII SYN character (016 hex). These characters are used to align the bits being received. If none of thse characters are sensed, it implies Transmission is not taking place.

CHARACTER WITH HEX VALUE XX WAS SENSED AS A CONTROL CHARACTER

The Secondary Channel contains only 0FF hex characters so there should not be any Active Control Characters. During transmission if the Active Control Character flag (IFL bit 6) goes high the character pointed to by the Current address of the Secondary Channel is the XX in the message.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits transmitted. The data bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
 1 if that bit is stuck high;
 X if that bit is not stuck but was incorrect at one point;
 - if that bit worked properly.

TEST 08 - TEST SECONDARY CHANNEL CURRENT ADDRESS AND THE ACTIVE CONTROL CHARACTER FLAG

Tests that during transmission from the Main Channel, detection of a control character initializes the Secondary Channel Current address and increments it until it reaches a matching control character in the secondary buffer. Also tests that the Secondary Channel does not totally prohibit CPU memory access during scanning. Also tests that the Active Control Character flag is set when the Secondary Channel locates a character matching the one which initiated the scan. Also tests that after locating a character and setting the Active Control Character flag it can be reset by INIT, DVCL, COM1, and COM2.

SECONDARY CHANNEL DOES NOT SCAN FOR CONTROL CHARACTERS

Transmitting a string of control characters failed to cause the Secondary Channel Current address to initialize and increment through the secondary buffer.

SECONDARY CHANNEL SCANNING AND CPU EXECUTION IMPROPERLY INTERLEAVED

The Secondary Channel is set up with its buffer being over 50 bytes long and the control character being scanned for at the end of the buffer. By continuously examining the Secondary Channel Current address it is found that it did scan the entire secondary buffer, but the CPU could never "catch" it in the middle of the buffer.

SECONDARY CHANNEL CURRENT ADDRESS NOT INITIALIZED CORRECTLY

The Secondary Channel Current address is set up to be 200 bytes before the Starting address. After beginning a Transmission of control characters, the Current address did not initialize to the Starting address before scanning.

SECONDARY CHANNEL CURRENT ADDRESS POINTING TO CHARACTER WITH HEX VALUE XX

The Active Control Character flag (IFL bit 6) set but not on the first control character transmitted, the hex value of the character found is displayed (XX).

ACTIVE CONTROL CHARACTER DOES NOT SET

After sensing a control character on the Main Channel the secondary buffer is scanned and the correct character is located but the Active Control Character flag (IFL bit 6) did not set.

INIT DOES NOT RESET ACTIVE CONTROL CHARACTERS

After setting the Active Control Character flag (IFL bit 6) an INIT does not reset it.

DVCL DOES NOT RESET ACTIVE CONTROL CHARACTER

After setting the Active Control Character flag (IFL bit 6) a DVCL does not reset it.

COM2 DOES NOT RESET ACTIVE CONTROL CHARACTER

After setting the Active Control Character flag (IFL bit 6) a Transmit (COM2) does not reset it.

COM2 RESETS ACTIVE CONTROL CHARACTER BUT KEEPS IT RESET

With the Active Control Character flag (IFL bit 6) set issuing a Transmit (COM2) resets it, but does not again set it during the transmission of a known control character.

COM2 DOES NOT LOCATE NEXT CONTROL CHARACTER

A Transmit (COM2) causes the Active Control Character flag (IFL bit 6) to set however, the character pointed to by the Secondary Channel was not the next control character in the Transmit buffer.

COM1 DOES NOT RESET ACTIVE CONTROL CHARACTER

After setting the Active Control Character flag (IFL bit 6) a Transmit (COM1) does not reset it.

COM1 RESETS ACTIVE CONTROL CHARACTER BUT KEEPS IT RESET

With the Active Control Character flag (IFL bit 6) set issuing a Transmit (COM1) resets it, but does not again set it during the re-transmission of the original buffer.

COM1 DOES NOT LOCATE FIRST CONTROL CHARACTER

A Transmit (COM1) causes the Active Control Character flag (IFL bit 6) to set, however the character pointed to by the Secondary Channel was not the first control character in the Transmit buffer.

CONTROLLER DOES NOT INTERRUPT WITH ACTIVE CONTROL CHARACTER SET

With the Active Control Character flag (IFL bit 6) set an Interrupt is not generated.

CONTROLLER INTERRUPTS WITH ACTIVE CONTROL CHARACTER RESET

With the Active Control Character flag (IFL bit 6) reset an Interrupt is generated.

TEST 09 - TEST IF USRT TRANSMITS SYN CHARACTERS

Tests if during a Transmission from the Main Channel, the USRT transmits the SYN characters as a default if it is unable to read characters from the Main Channel buffer.

USRT DOES NOT TRANSMIT SYNC IDLE CHARACTERS

A Transmit (COM1) is issued to begin transmission of a buffer that contains an active control character. After the Active Control Character flag (IFL bit 6) sets, the bits being transmitted from the USRT are not SYN characters.

TEST 10 - TEST ASCII AND EBCDIC MODES

Tests if the ASCII/EBCDIC mode selection (OFL bit 7) is working.

ACTIVE CONTROL CHARACTER DOES NOT SET DURING TRANSMIT OF CONTROL CHARACTER

During a Transmit (COM1) of a buffer known to contain control characters, the Not Busy flag (IFL bit 7) sets prior to the Active Control Character flag (IFL bit 6).

ASCII MODE ERROR

During a Transmit (COM1) in the ASCII mode the Active Control Character flag (IFL bit 6) sets for an EBCDIC control character.

EBCDIC MODE ERROR

During a Transmit (COM1) in the EBCDIC mode the Active Control Character flag (IFL bit 6) sets for an ASCII control character.

TEST 11 - PROTOCOL COMMAND TEST

Tests all Protocol commands during a transmit. This test utilizes the same Main and Secondary Channel buffers throughout the entire test. The hex contents of the Main buffer in hex is SYN, 0FE, 0FF, 0FF, 0FF, 0FF, 00, 0FF, 0FF, 0FF, 0FF, 01, 0FF, 0FF, 0FF, 0FF while the Secondary buffer contains 02, 01, 00, 0FF, 0FE.

COM1 DOES NOT LOCATE FIRST CONTROL CHARACTER

After a Transmit (COM1) the Active Control Character flag (IFL bit 6) did not set even though the buffer contains known control characters.

BIT 2 OF PROTOCOL COMMAND STUCK HIGH

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 2 (restart Main Channel) reset incorrectly caused the transmission to restart at the beginning of the buffer. The Active Control Character flag (IFL bit 6) was set by the first active control character, 00 hex.

BIT 3 OF PROTOCOL COMMAND STUCK HIGH

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 3 (treat next character as control character) reset incorrectly caused by Active Control Character flag (IFL bit 6) to be set by a non-control character, 0FF hex.

ACTIVE CONTROL CHARACTERS SET WITH INCORRECT CHARACTER

During a Transmit (COM1), issuing a Protocol Command (COM2) caused the Active Control Character flag (IFL bit 6) to set on a character other than the expected one.

UNEXPECTED PATTERN IN CHARACTERS TRANSMITTED

During a Transmit (COM1), issuing a Protocol Command (COM2) caused the characters to be transmitted incorrectly.

BIT 7 OF PROTOCOL COMMAND STUCK LOW

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 7 (accept character in holding register into data stream) set did not transmit the character in the holding register to the receiver.

BIT 7 OF PROTOCOL COMMAND STUCK HIGH

During a Transmit (COM1), issuing a COM2 with bit 7 (accept character in holding register into the data stream) reset transmitted the character in the holding register to the receiver.

BIT 4 OF PROTOCOL COMMAND DOES NOT SET NOT BUSY

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 4 (termination sequence) set failed to set the Not Busy flag (IFL bit 7).

BITS 4 AND 5 OF PROTOCOL COMMAND INCORRECT

Attempt to use bits 4 and 5 to accumulate and transmit an LRC character was unsuccessful. During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 5 (start BCC accumulation) set followed by a second Protocol Command with bit 4 (termination sequence) set failed to transmit the LRC.

BIT 3 OF PROTOCOL COMMAND STUCK LOW

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 3 (treat next character as control character) set did not cause the Active Control Character flag (IFL bit 6) to set on the next character transmitted.

BIT 2 of PROTOCOL COMMAND STUCK LOW

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 2 (restart Main Channel) set did not cause the transmission to restart at the beginning of the buffer.

BIT 6 OF PROTOCOL COMMAND IS MALFUNCTIONING

During a Transmit (COM1), issuing a Protocol Command (COM2) with bit 6 (accept character in holding register into BCC accumulation) set or reset has no effect on the BCC accumulation.

TEST 12 - TEST LRC

Tests LRC accumulator during transmission of 1 character at a time. Each character from 01-7F hex will be tested to yield the correct LRC. Also the LRC of a string of characters will be tested.

-AAA,AAAA STUCK BITS FOR LRC OF ALL CHARACTERS

This stuck message is displayed if there is an error in any of the LRC bits transmitted. The bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
  1 if that bit is stuck high;
  X if that bit is not stuck but was incorrect at one point;
  - if that bit worked properly.

LRC INCORRECT FOR A STRING OF CHARACTERS DURING TRANSMIT

The transmitted LRC of a string of known characters does not agree with the known LRC.

TEST 13 - TEST CRC

Tests accumulation of CRC by transmitting 16 different known strings of characters whose CRC has already been determined to test all bits high and low.

ACCUMULATED CRC INCORRECT

The transmitted CRC of a string of known characters does not agree with the known CRC.

TEST 14 - TEST LOSS OF CARRIER

Tests if loss of carrier has an effect on reception. Reception should take place only if Carrier Detect is a "1" (+12V.).

RECEIVE DOES NOT RESET NOT BUSY

During a Receive (COM1) the Not Busy flag (IFL bit 7) was set.

LOSS OF CARRIER DOES NOT INHIBIT RECEPTION

During a Receive (COM1) with Carrier Detect a known "0" (-12V.) characters are still received by the Main Channel.

RECEPTION DOES NOT OCCUR WITH CARRIER

During a Receive (COM1) with Carrier Detect a known "1" (+12V.) Characters are not received by the Main Channel.

TEST 15 - TEST RECEIVING SYN CHARACTERS

Tests the number of SYNC IDLE characters necessary to initiate reception. This test utilizes a buffer to be transmitted to the Binary Synchronous Controller that is a sequential increase of SYNC IDLE characters separated by the "inverse" of the SYNC IDLE character (X, S, X, S, S, X, S, S, S, X, S, S, S, S, X, S, S, S, S, S where S=SYNC IDLE X="inverse" of SYNC IDLE). All characters from OO-FF hex are tested as SYNC IDLE characters.

X SYN CHARACTERS ARE NEEDED TO RECEIVE SYN CHARACTER YY

Two SYNC IDLE characters are supposed to begin reception. However, if X is not equal to two it is inserted and YY is the particular SYNC IDLE character being tested.

THE SECOND SYN CHARACTER (YY) DOES NOT ENTER BUFFER

Using the buffer described it was found that two SYNC IDLE characters referenced by YY were necessary to begin reception but the second one did not enter the buffer.

UNEXPECTED PATTERN IN CHARACTERS RECEIVED FOR SYN CHARACTER (YY)

There is an unknown character in the receive buffer which prevents analysis for SYNC IDLE character YY.

TEST 16 - TEST RECEIVING 8 DATA BITS

Tests the receiver for correct character reception of all characters from 00-0FF hex with no parity.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits received. The data bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at one point;
    - if that bit worked properly.

TEST 17 - TEST RECEIVING 7 DATA BITS WITH ODD PARITY

Tests the receiver for correct character reception of all characters from 00-7F hex with parity odd.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits received. The leftmost bit reflects the status of the parity bit and the next 7 from left to right are bits 6 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

TEST 18 - TEST RECEIVE ERROR

Tests that the Receive Error flag (IFL bit 5) can detect Parity and LRC errors. Also tests that an INIT and DVCL can individually reset the Receive Error flag.

RECEIVE ERROR STUCK LOW

Transmitting a known bad parity character to the receiver while in the receive mode does not set the Receive Error flag (IFL bit 5).

INIT DOES NOT RESET RECEIVE ERROR

Issuing an INIT with the Receive Error flag (IFL bit 5) set cannot reset it.

DVCL DOES NOT RESET RECEIVE ERROR

Issuing a DVCL with the Receive Error flag (IFL bit 5) set cannot reset it.

INCORRECT LRC DOES NOT SET RECEIVE ERROR

Transmitting a known bad LRC to the receiver while in the receive mode does not set the Receive Error flag (IFL bit 5).

TEST 19 - TEST OVERRUN FLAG DURING RECEIVE

Test that if the next character received after a Protocol bit 3 command is not a character in the Secondary Channel that the overrun flag (IFL bit 2) will set. In addition tests that an INIT and DVCL can independently reset it.

OVERRUN STUCK HIGH

Issuing an INIT and a DVCL failed to reset the OVERRUN flag (IFL bit 2).

OVERRUN FLAG NOT SET BY ISSUING PROTOCOL BIT # (COM2)

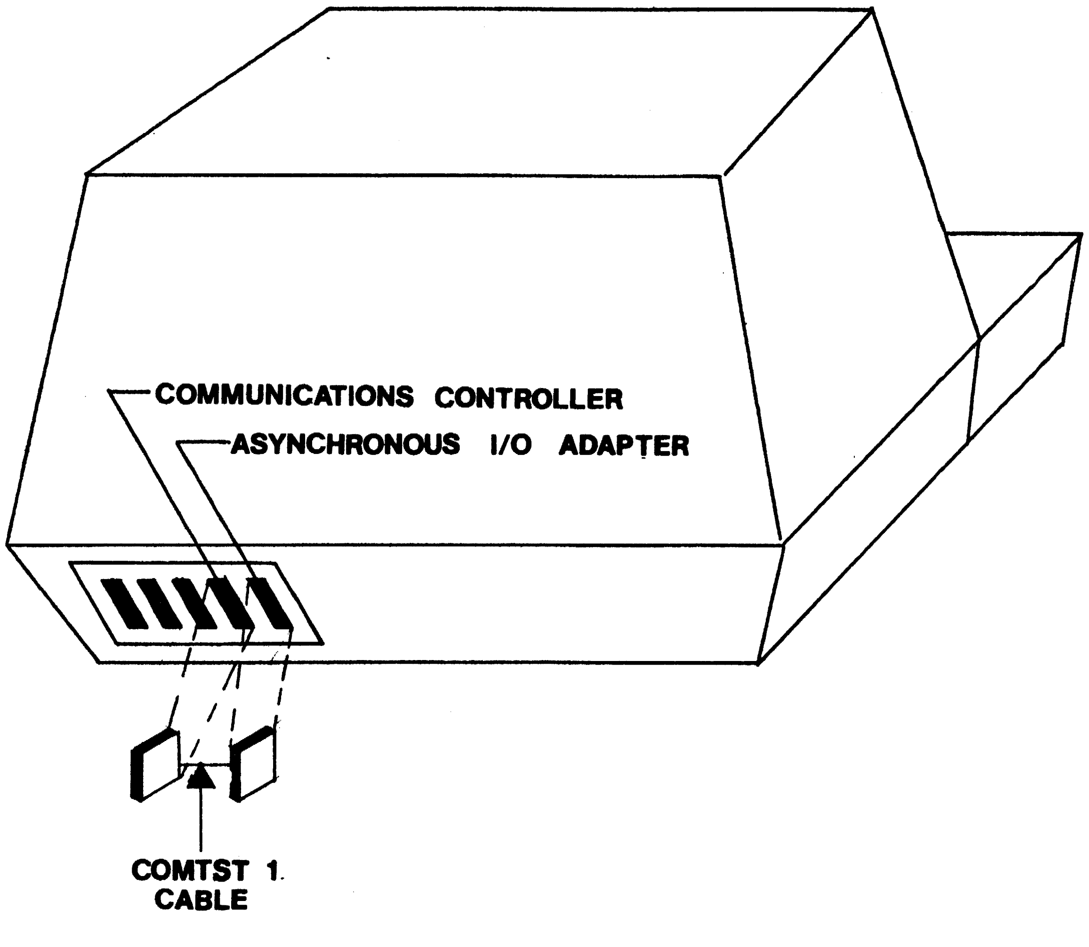
During a Receive (COM1) after detecting an Active Control Character, issuing a Protocol command with bit 3 failed to set the Overrun flag on the next non Active Control character which was received.

INIT DOES NOT RESET OVERRUN

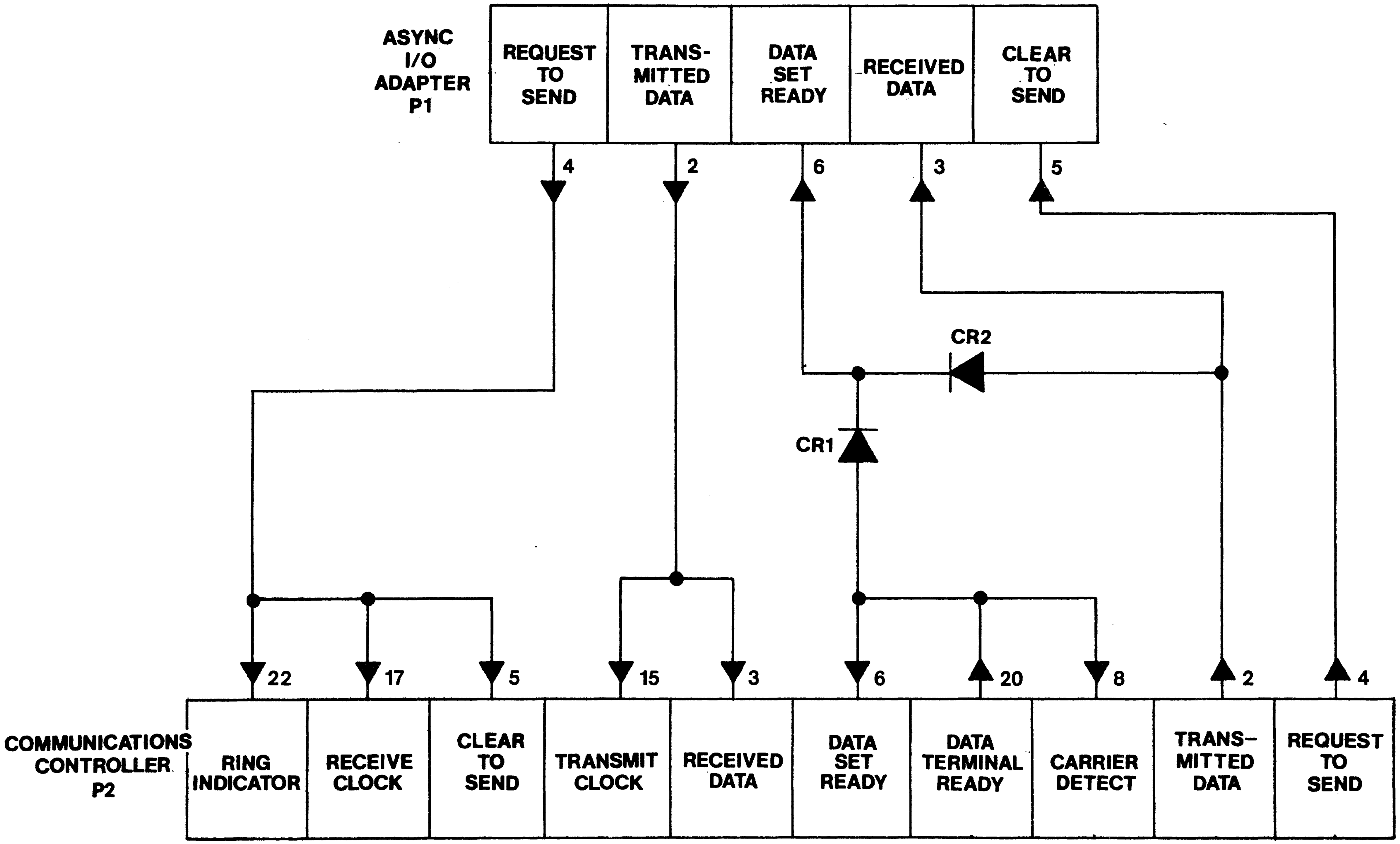
Issuing an INIT failed to reset the Overrun flag (IFL bit 2).

DVCL DOES NOT RESET OVERRUN

Issuing a DVCL failed to reset the Overrun flag (IFL bit 2).



COMTST1 DIAGNOSTIC CABLE INSERTION GUIDE
FIGURE 1



COMTST1 DIAGNOSTIC CABLE
FIGURE 2

SYN2

SYNCHRONOUS II CONTROLLER TEST

SYN2 - SYNCHRONOUS II CONTROLLER TEST

Applicable Assemblies

5000-1193-1	Synchronous II Communications Controller

Required Test Assemblies

COMTST1 Diagnostic Cable
COMTST12 Diagnostic Cable

General Description

The purpose of the SYN2 program is to determine if the Synchronous II Controller is working properly and, if not, to give an indication of which functions are incorrect. Prior to running SYN2, the Asynchronous I/O Adapter must be tested by IOTST and be known good. A COMTST1 and the COMTST12 Diagnostic Cables are needed to run SYN2. The program requires no operator interaction unless an error is detected.

12K (or more) of memory is required to run SYN2.

This manual applies to the 8080 version of SYN2 only.

Loading Procedure

SYN2 can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If SYN2 loads properly, it will identify itself and pause for a moment to enter run-mode information. It is desirable to insert both plugs of the COMTST1 Diagnostic Cable into their respective connectors on the OP-1, as shown in Figure 1, before SYN2 is loaded. If this is not possible, such as will be the case when SYN2 is loaded through one of these connectors, then when SYN2 starts it will immediately detect an error and halt. At this time, remove the loading cable and insert the COMTST1 Diagnostic Cable. Then type the PROG key to clear the error count and restart SYN2.

Operator Action

After the run-mode information pause, the following message will appear:

"PLEASE ENTER SELECT ADDRESS: C3"

The operator may change the select address by entering a two character hexadecimal number. The select address will default to C3 by typing a carriage return.

The second message that will appear is:

"IS STRAP A-B IMPLEMENTED? TYPE Y or N."

Respond to this question by typing a (Y) for yes or a (N) for no. SYN2 will immediately start testing after this question is answered.

All tests except Test 20 operate automatically and without operator intervention. When the entire test has been completed, the prompt "TYPE SPACE TO REPEAT SYN2" will be displayed. If a space is typed on the keyboard, SYN2 will restart.

Refer to Appendix A for specialized test run options.

Errors

All errors are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The error message displayed on the screen attempts to give a description of the nature of the problem. In some cases, this should be adequate to diagnose and fix the error. Otherwise, refer to the detailed description of the specific test to determine the purpose and expected results for the displayed error message.

After an error message is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, and depressing the PROG key will restart the program.

Because of the inverted pyramid test strategy used in SYN2, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

COMTST1 Diagnostic Cable

The COMTST1 Diagnostic Cable is necessary for testing the Synchronous II Controller. The cable must be inserted into the Asynchronous I/O Adapter connector and slot D1 (as shown in Figure 1) on the rear panel of the OP-1 prior to program execution. The schematic of the COMTST1 Diagnostic Cable is shown in Figure 2.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing for on the top of the page. Below the description all possible error messages are listed, with an explanation of the cause of the message.

Test 00 - Controller Select Test.

Tests that the Synchronous Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selects a selected Synchronous Controller. The correct device select address is OC3 hex. The incorrect select addresses are those single byte addresses which have 3 or less bits high. This is derived from the fact that a 4 input AND gate performs the device selection from the address bits of the select address. Thus a select address of OFF hex would attempt to select all devices.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL or INP to the select address XX hex gave a result other than 0FF hex (open bus).

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (C3)

IFL to the correct Synchronous Controller address (OC3 hex) gave a result of 0FF hex (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Synchronous Controller, IFL did not get a result of 0FF hex (open bus).

Test 01 - Test static flags Data Terminal Ready and Data Set Ready.

Tests that Data Terminal Ready can be set and reset by OFL bit 4. also tests that setting and resetting Data Set Ready can be detected by IFL bit 1. It also tests that with no transmit in progress (Clear To Send reset), Transmitted Data transmits SYN characters. The tests are conducted using known good flags on the Asynchronous I/O Adapter, and yield conclusive results.

REQUEST TO SEND STUCK HIGH

An INIT and DVCL could not reset Request To Send.

TRANSMITTED DATA STUCK HIGH

In order to use the known good Data Set Ready on the Asynchronous I/O Adapter to test Data Terminal Ready, Transmitted Data must be a "1" (-12V.) because of diode CR2 in the COMTST1 Diagnostic Cable. An attempt is made to bring Transmitted Data to a "1" by transmitting with the USRT while clocking the transmitter with Clear To Send reset. This should transmit the ASCII SYN character (16 hex) which has some bits "0" and others "1". Thus, clocking the Transmit Clock with Clear To Send reset could not make Transmitted Data a "1" (-12V.).

DATA TERMINAL READY STUCK HIGH

An OFL (with bit 4 reset) could not reset Data Terminal Ready.

DATA SET READY STUCK HIGH

With input to Data Set Ready a known "0" (-12V.) the Data Set Ready flag (IFL bit 0) is set.

DATA TERMINAL READY STUCK LOW

An OFL (with bit 4 set) could not set Data Terminal Ready.

DATA SET READY STUCK LOW

With input to Data Set Ready a known "1" (+12V.) the Data Set Ready flag (IFL bit 6) is reset.

Test 02 - Test transmit capabilities and interrupt.

Tests that a Transmit (COM3) resets the Not Busy flag (IFL bit 7), sets Request To Send, waits for Clear To Send and then initializes the Synchronous Buffer Current address and increments it until it reaches the Terminating address. Also tests that INIT, DVCL, reaching Terminating address, and reaching Terminating character individually set the Not Busy flag (IFL bit 7) and reset Request To Send. Also tests that the controller interrupts with IFL bit 7 set.

CURRENT ADDRESS REACHED TERMINATING ADDRESS PREMATURELY

During a Transmit (COM3) of only ASCII SYN Characters the Synchronous Buffer Current address reached its terminating address without the expected number of clock pulses.

TRANSMIT DOES NOT SET REQUEST TO SEND

During a Transmit (COM3) Request To Send was "0" (-12V.).

TRANSMIT DOES NOT RESET NOT BUSY

During a Transmit (COM3) the Not Busy flag (IFL bit 7) was set.

TRANSMIT DOES NOT SET REQUEST TO SEND OR RESET NOT BUSY

During a Transmit (COM3) Request To Send was "0" (-12V.) and the Not Busy flag (IFL bit 7) was set.

INIT DOES NOT SET NOT BUSY

During a Transmit (COM3) an INIT did not cause the Not Busy flag (IFL bit 7) to set.

INIT DOES NOT RESET REQUEST TO SEND

During a Transmit (COM3) an INIT did not cause Request To Send to become a "0" (-12V.).

INIT DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

During a Transmit (COM3) an INIT did not cause the Not Busy flag (IFL bit 7) to set and Request To Send to become a "0" (-12V.).

DVCL DOES NOT SET NOT BUSY

During a Transmit (COM3) a DVCL did not cause the Not Busy flag (IFL bit 7) to set.

DVCL DOES NOT RESET REQUEST TO SEND

During a Transmit (COM3) a DVCL did not cause Request To Send to become a "0" (-12V.).

DVCL DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

During a Transmit (COM3) a DVCL did not cause the Not Busy flag (IFL bit 7) to set, and the Request To Send did not become a "0" (-12V.).

CLEAR TO SEND LOW DOES NOT HOLD OFF TRANSMIT

Transmission of data from the Synchronous Buffer takes place without Clear To Send being a "1" (+12V.).

CURRENT ADDRESS NOT INITIALIZED CORRECTLY

Issuing a Transmit (COM3) does not cause the Synchronous Buffer Current address to be initialized to the Starting address.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER OR ADDRESS

During a Transmit (COM3) the Synchronous Buffer Current address incremented past the terminating character and address.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING ADDRESS

During a Transmit (COM3) the Synchronous Buffer Current address incremented past the Terminating address.

REACHING TERMINATING ADDRESS DOES NOT SET NOT BUSY

The end of a transmit did not cause the Not Busy flag (IFL bit 7) to set.

REACHING TERMINATING ADDRESS DOES NOT RESET REQUEST TO SEND

The end of a transmit did not cause Request To Send to become a "0" (-12V.).

REACHING TERMINATING ADDRESS DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

The end of a transmit did not cause the Not Busy flag (IFL bit 7) to set and Request To Send to become a "0" (-12V.).

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER

During a Transmit (COM3) the Synchronous Buffer Current address incremented past the Terminating character.

REACHING TERMINATING CHARACTER DOES NOT SET NOT BUSY

The end of a transmit did not cause the Not Busy flag (IFL bit 7) to set.

REACHING TERMINATING CHARACTER DOES NOT RESET REQUEST TO SEND

The end of a transmit did not cause Request To Send to become a "0" (-12V.).

REACHING TERMINATING CHARACTER DOES NOT SET NOT BUSY OR RESET REQUEST TO SEND

The end of a transmit did not cause the Not Busy flag (IFL bit 7) to set and Request To Send to become a "0" (-12V.).

INTERRUPT PRIORITY LEVEL NO. 6 NOT ISSUED WITH NOT BUSY SET

With the Not Busy flag (IFL bit 7) set Interrupt 6 is not generated.

INTERRUPT PRIORITY LEVEL NO. 6 ISSUED WITH NOT BUSY RESET

With the Not Busy flag (IFL bit 7) reset Interrupt 6 is generated.

Test 03 - Test Ring Detected.

Tests the Ring Detected flag, specifically that transitions on Ring Indicator are properly indicated by the Ring Detected flag (IFL bit 0).

RING DETECTED STUCK HIGH

After an INIT and DVCL with Ring Indicator a "0" (-12V.) the Ring Detected flag (IFL bit 0) remains set.

RING DETECTED STUCK LOW

With Ring Indicator a "1" (+12V.) the Ring Detected flag (IFL bit 0) remains reset.

RESETTING RING INDICATOR DOES NOT RESET RING DETECTED

After setting the Ring Detected flag (IFL bit 0) by making Ring Indicator a "1" (+12V.), making Ring Indicator a "0" (-12V.) does not cause the Ring Detected flag (IFL bit 0) to reset.

Test 04 - Test Terminating Characters.

Test that all 256 (00-0FF hex) characters can be used as terminating characters. The test is performed by setting up the Synchronous Buffer with the terminating character which is preceded and followed by the inverse of the terminating character. A Transmit (COM3) is issued and upon termination of transmission the Current address is examined to verify termination was caused by the terminating character.

CHARACTER (XX) DOES NOT TERMINATE TRANSMISSION

The Current address of the Synchronous Buffer incremented past the terminating character (XX) during a Transmit (COM3).

Test 05 - Test transmitting least significant 7 data bits and parity.

Tests transmission of the parity bit and the least significant 7 data bits excluding control characters (00-01F hex).

NO SYN CHARACTER SENSED

The first 2 characters in the Synchronous transmit buffer are the ASCII SYN character (016 hex). These characters are used to align the bits being received. If none of these characters are sensed, it implies Transmission is not taking place.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits transmitted. The leftmost bit reflects the status of the parity bit and the next 7 from left to right are bits 6 through 0 respectively.

A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at one point;
    - if that bit worked properly.

Test 06 - Test transmitting all 8 data bits.

Tests transmission of all 8 data bits.

NO SYN CHARACTER SENSED

The first 2 characters in the Synchronous transmit buffer are the ASCII SYN character (016 hex). These characters are used to align the bits being received. If none of thse characters are sensed, it implies Transmission is not taking place.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits transmitted. The data bits from left to right are bits 7 through 0 respectively.

0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 07 - Test if USRT transmits Marks.

Tests that during a Transmission from the Synchronous Buffer, the USRT transmits Marks (0FF hex) as a default if it is unable to read characters from the Synchronous Buffer because Clear To Send is kept low.

USRT DOES NOT TRANSMIT MARKS

A Transmit (COM3) is issued to begin transmission of a buffer, but Clear To Send is kept low. This should force the USRT to transmit Marks (0FF hex) as a default because the USRT is not permitted to access the characters from the Synchronous Buffer.

Test 08 - Test Loss of Carrier.

Tests if loss of carrier has an effect on reception. Reception should take place only if Carrier Detect is a "1" (+12V.).

RECEIVE DOES NOT RESET NOT BUSY

During a Receive (COM2) the Not Busy flag (IFL bit 7) was set.

NO RECEPTION WITH CARRIER DETECT STRAPPED HIGH

On boards with strap A-B implemented Reception should have taken place even though Carrier was not externally raised.

LOSS OF CARRIER DOES NOT INHIBIT RECEPTION

During a Receive (COM2) with Carrier Detect a known "0" (-12V.), characters are still received by the Synchronous Buffer.

RECEPTION DOES NOT OCCUR WITH CARRIER

During a Receive (COM2) with Carrier Detect a known "1" (+12V.), characters are not received by the Synchronous Buffer.

Test 09 - Test receiving SYN characters.

Tests the number of consecutive SYN characters necessary to initiate reception. Also tests that received SYN characters do not enter the receive buffer. This test utilizes multiple transmit messages each with an increasing number of SYN characters followed by ten pad characters (0FF hex.). All characters from 00-FE hex are tested as SYN characters.

X SYN CHARACTERS ARE NEEDED TO RECEIVE SYN CHARACTER YY

Two or three SYN characters are supposed to begin reception. However, if X is not equal to two or three, it is inserted and YY is the particular SYNC IDLE character being tested.

SYN CHARACTER (YY) ENTERS BUFFER

Using the transmit messages described it was found that two or three SYN characters referenced by YY were necessary to begin reception but one of these SYN characters entered the receive buffer.

UNEXPECTED PATTERN IN CHARACTERS RECEIVED FOR SYN CHARACTER (YY)

There is an unknown character in the receive buffer which prevents analysis for SYNC IDLE character YY.

SYN CHARACTER YY DOES NOT BEGIN RECEPTION

Reception is tried using from 0 to 5 SYN characters to begin the Synchronous II Controller receiving. If reception does not take place with 5 SYN characters, it is assumed that SYN character YY will not initiate reception.

Test 10 - Test receiving 8 data bits.

Tests the receiver for correct character reception of all characters from 00-0FF hex with no parity.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits received. The data bits from left to right are bits 7 through 0 respectively.

```
A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at
      one point;
    - if that bit worked properly.
```

Test 11 - Test Character Error.

Tests that the Character Error flag (IFL bit 5) can detect Parity errors. Also tests that an INIT and DVCL can individually reset the Character Error flag.

CHARACTER ERROR STUCK HIGH

An INIT and DVCL could not reset the Character Error Flag (IFL bit 5).

CHARACTER ERROR STUCK LOW

Transmitting a known bad parity character to the receiver while in the receive mode does not set the Character Error flag (IFL bit 5).

INIT DOES NOT RESET CHARACTER ERROR

Issuing an INIT with the Character Error flag (IFL bit 5) set cannot reset it.

DVCL DOES NOT RESET CHARACTER ERROR

Issuing a DVCL with the Character Error flag (IFL bit 5) set cannot reset it.

Test 10 - Test receiving 8 data bits.

Tests the receiver for correct character reception of all characters from 00-0FF hex with no parity.

XXH WAS RECEIVED AS AN ATTENTION CHARACTER

This message will appear if the transmitted character was interpreted as an attention character.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits received. The data bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 11 - Test Character Error.

Tests that the Character Error flag (IFL bit 5) can detect Parity errors. Also tests that an INIT and DVCL can individually reset the Character Error flag.

CHARACTER ERROR STUCK HIGH

An INIT and DVCL could not reset the Character Error Flag (IFL bit 5).

CHARACTER ERROR STUCK LOW

Transmitting a known bad parity character to the receiver while in the receive mode does not set the Character Error flag (IFL bit 5).

INIT DOES NOT RESET CHARACTER ERROR

Issuing an INIT with the Character Error flag (IFL bit 5) set cannot reset it.

DVCL DOES NOT RESET CHARACTER ERROR

Issuing a DVCL with the Character Error flag (IFL bit 5) set cannot reset it.

Test 12 - Test receiving 7 data bits with odd parity.

Tests the receiver for correct character reception of all characters from 00-7F hex with parity odd.

PARITY ERROR DURING RECEIVE

During a Receive (COM2) of characters with known good parity, the Character Error flag (IFL bit 5) set.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits received. The leftmost bit reflects the status of the parity bit and the next 7 from left to right are bits 6 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 13 - Test Attention Receive.

Tests separately that for all Attention Characters in the ranges 00-01F hex and 080-09F hex, the Receive-Attention (COM1) does not cause the Synchronous II Controller to receive when non-attention characters are transmitted to it.

ATTENTION CHARACTER YY NOT NEEDED FOR RECEPTION

After a Receive-Attention (COM1), reception took place even though there was no corresponding Attention character transmitted to the Synchronous II Controller.

Test 14 - Test Receiving Attention Characters.

Tests that reception is initiated in the Receive-Attention (COM1) separately for all Attention characters in the ranges 00-01F hex and 080-09F hex (not including the ASCII SYN characters, 016 and 096 hex). Also tests that reception begins with the character immediately following the Attention character by examining the first character in the Synchronous Buffer.

RECEPTION BEGINS X CHARACTERS AFTER ATTENTION CHARACTER YY

The first character in the Synchronous Buffer should be the character that immediately followed the Attention character in the transmitted string. If the first received character was indeed part of the transmitted string but not the character immediately following the attention character its position in reference to the Attention character in the transmitted string is determined (X) along with the Attention character causing the fault.

ATTENTION CHARACTER YY ENTERS BUFFER

Reception should begin with the character immediately following the Attention character (YY), not with the Attention character itself.

UNEXPECTED PATTERN IN CHARACTERS RECEIVED FOR ATTENTION CHARACTER YY

For Attention character YY the buffer contained a character that could not be processed. Either the character was received incorrectly or reception did not begin.

RECEPTION BEGINS WITHOUT ATTENTION CHARACTER FOLLOWING SYNC

Reception began when a pad character was between the last SYNC character and the Attention Character.

Test 15 - Test receiving 7 data bits with odd parity in Attention Mode.

Tests the receiver for correct character reception of all characters from 00-7F hex with parity odd. Also tests that bit 7 of the characters transferred into memory is 0.

PARITY ERROR DURING RECEIVE

During a Receive-Attention (COM1) of characters with known good parity the Character Error flag (IFL bit 5) set.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits received. The data bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 16 - Test OCC Generation.

By transmitting to the Synchronous II Controller a known string of characters, one can compute any discrepancy in the OCC generation. This test changes only one character of the known string each of 256 times through a loop and computes the expected OCC, which upon reception it compares against the actual OCC. It tests all 256 values of OCC (00-FF hex). Neither the SYN character nor the Attention character should be computed in the OCC; however if the hardware is incorrectly computing them in the OCC, this error has been nullified by transmitting an even number of each.

ATTENTION RECEIVE DOES NOT GENERATE OCC

If the OCC found in the buffer at the end of each and every reception is equal to the BCC transmitted to the Synchronous II Controller it means that reaching the terminating character is not causing the OCC to be substituted for the BCC in the Synchronous buffer.

AAAA,AAAA STUCK BITS FOR OCC GENERATION

This stuck message is displayed if there is an error in any of the OCC bits received. The bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 17 - Test effect of Attention and SYN characters on OCC generation.

Tests that OCC generation begins with the character immediately after the Attention character. Also tests that SYN characters and Attention characters do not get computed in the OCC.

OCC GENERATION INCORRECT

A known string of characters transmitted to the Synchronous II Controller did not generate the expected OCC.

SYN CHARACTER IS INCLUDED IN OCC GENERATION

An odd number of SYN characters at the beginning of the transmitted string generated a different OCC then when an even number of SYN characters were transmitted to the Synchronous II Controller.

ATTENTION CHARACTER IS INCLUDED IN OCC GENERATION

An odd number of Attention characters at the beginning of the transmitted string generated a different OCC then when an even number of Attention characters were transmitted to the Synchronous II Controller.

Test 18 - Test effect of a second Attention character on reception and OCC generation.

Tests that receiving a second Attention character will cause the Synchronous II Controller to restart reception with the character immediately following the second Attention character and will restart the OCC generation with the same character.

A SECOND ATTENTION CHARACTER DOES NOT RESTART RECEPTION

The first character found in the Synchronous Buffer was not the character immediately following the second Attention character.

A SECOND ATTENTION CHARACTER DOES NOT RESTART OCC GENERATION

The OCC found in the Synchronous Buffer was not the same as the OCC that should have been computed from the character after the second Attention character up to and including the BCC character.

Test 19 - Test that after the BCC is received the line appears open (0FF hex gets pushed onto the line).

TRAILING PAD AFTER BCC IS NOT 0FF HEX

Expected character did not match character received.

REQUEST TO SEND DROPS PRIOR TO BIT 7 OF PAD CHARACTER

Not enough pad bits were sensed before request to send signal went low.

Test 20 - Test that when incoming data is all high, the Synchronous II Controller looks for SYN characters.

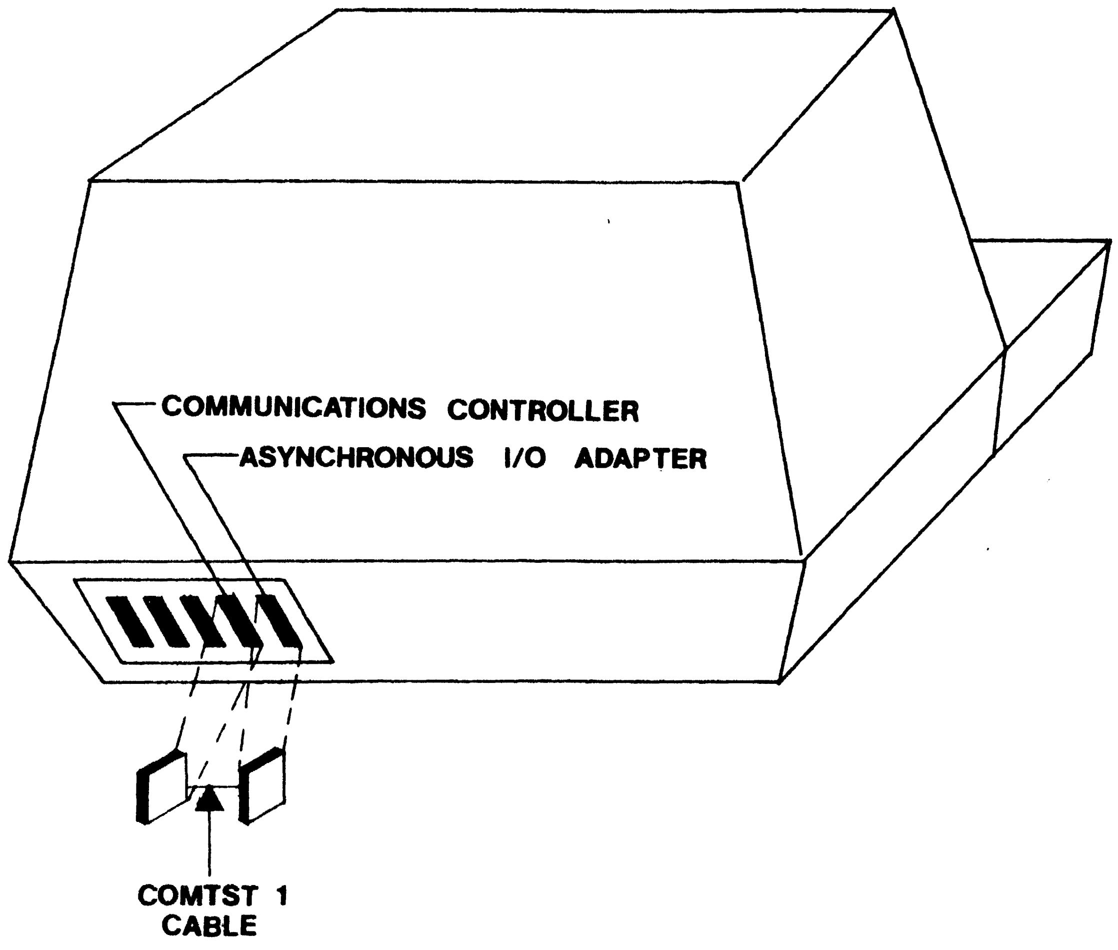
SYNC AQUISITION MODE MALFUNCTIONING

After receiving 32 bits of consecutive 1's synchronization was not lost.

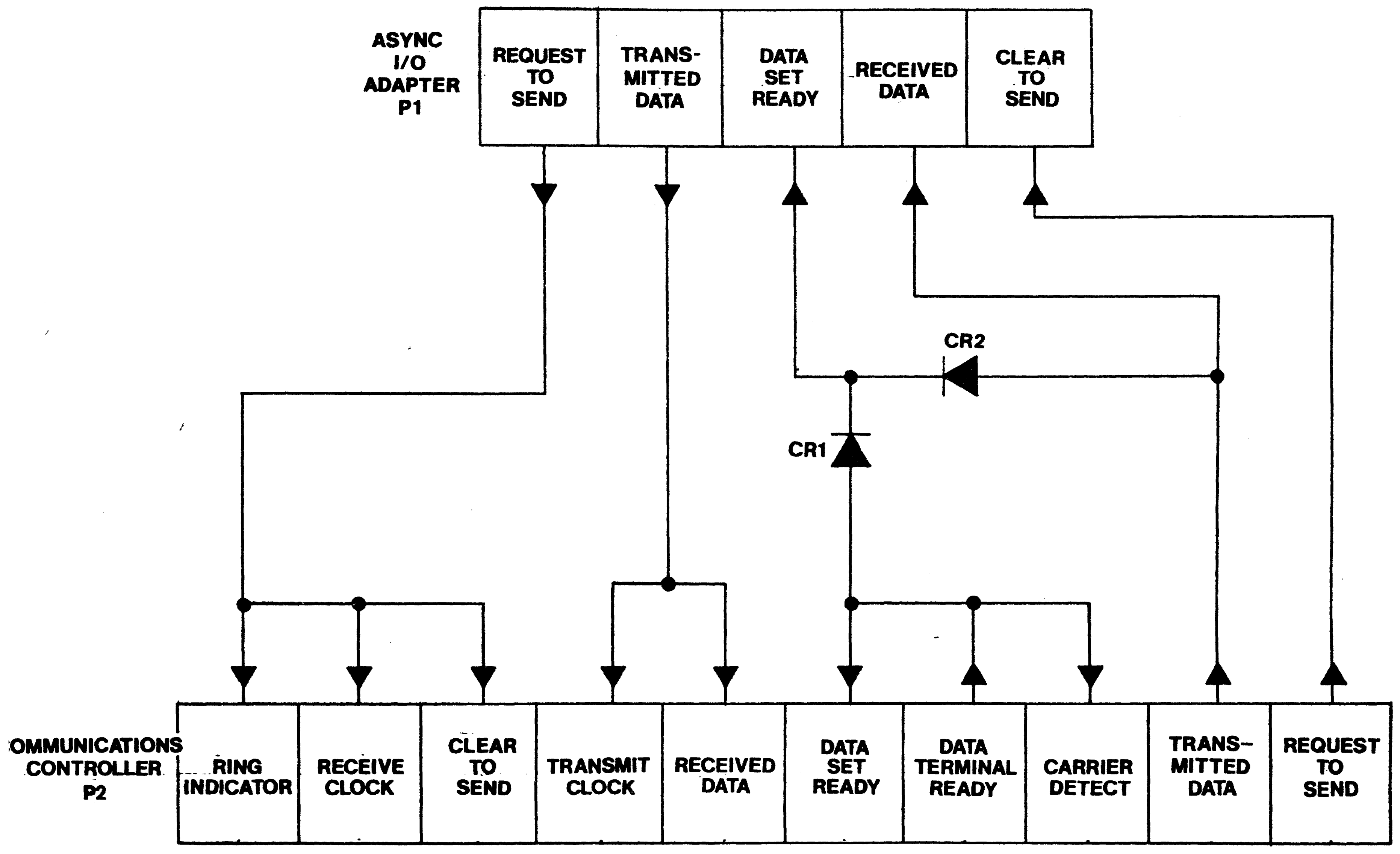
Test 21 - Test that when a byte of data is received synchronization is not lost.

SYNC AQUISITION MODE MALFUNCTIONS

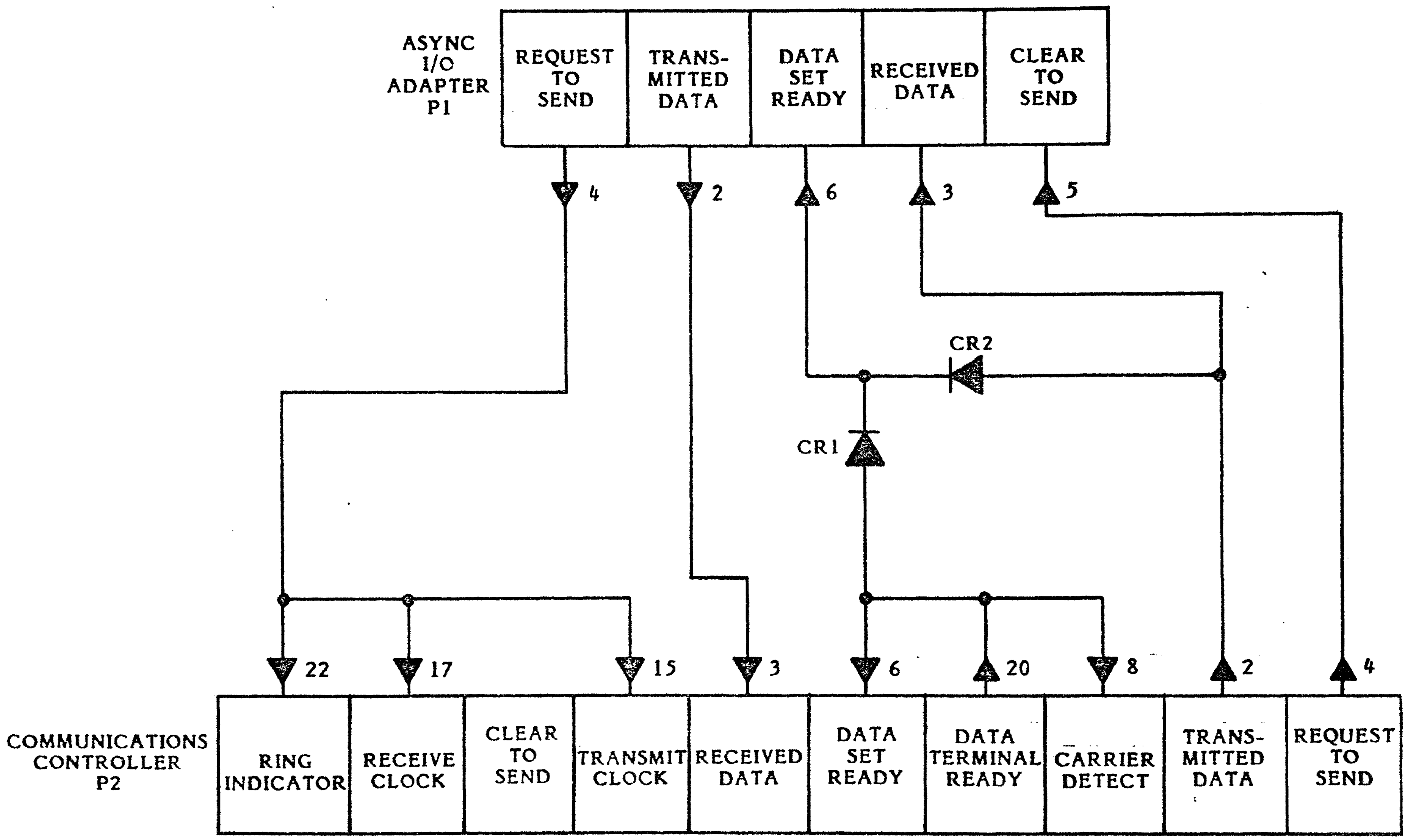
Currently in SYNC aquisition mode, when synchronization was not supposed to be lost.



COMTST1 DIAGNOSTIC CABLE INSERTION GUIDE
FIGURE 1



COMTST1 DIAGNOSTIC CABLE
FIGURE 2



COMTST12 DIAGNOSTIC CABLE

Figure 3

WRDTST/WRD210

WORD MOVER CONTROLLER TEST

WRDTST/WRD210 - WORD MOVER CONTROLLER TEST

Applicable Assemblies		Program Name
5000-1170	Word Mover Controller	(WRDTST)
5000-1170 PCO 210	Word Mover Controller	(WRD210)

General Description

The purpose of the WRDTST (and/or WRD210) program is to determine if the Word Mover Controller is working properly and, if not, to give an indication of which functions are incorrect.

16K (or more) of memory is required to run WRDTST.

This manual applies only to the 8080 version of WRDTST.

Loading Procedure

WRDTST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If WRDTST loads properly, it will immediately identify itself and pause for optional test run instructions (refer to Appendix A).

Operator Action

After pausing to accept optional test run instructions (if any), the message, "Type (S)ynchronous, (A)synchronous" will appear. At this time type (S) for Synchronous or (A) for Asynchronous adapter. After this, the message "ENTER TROUBLE-SHOOT MODE? (Y)es, (N)o" will appear on line 3 of the display screen. Key in a (Y) for yes or an (N) for no. Trouble-shooting mode is described in the Appendix following the test descriptions. This message will be erased and replaced by, "ARE 90-9F WORD DELIMITERS". This message refers to the hardware jumper options on the board. The operator should type a (Y) for yes or an (N) for no. At this time WRDTST will begin test execution.

Errors

All errors in Tests 00-09 are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The bell will ring only to notify the operator of an error. The error message displayed on the screen attempts to give a description of the nature of the problem. In some cases, this should be adequate to diagnose and fix the error. Otherwise refer to the detailed description of the specific test to determine the purpose and expected results for the displayed error message.

Errors - Continued

After an error message is displayed the operator has four ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, depressing the PROG key will restart the program and typing SHIFT PROG will return control to the operating system.

Because of the inverted pyramid test strategy used in WRDTST, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing. Below the description, all possible error messages are listed, with an explanation of the cause of the message.

Errors - Continued

After an error message is displayed the operator has four ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, depressing the PROG key will restart the program. A shifted PROG will not return control to the operating system.

Because of the inverted pyramid test strategy used in WRDTST, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing. Below the description, all possible error messages are listed, with an explanation of the cause of the message.

TEST 00 - Controller Select Test

Tests that the Word Mover Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selects a selected Word Mover Controller. The correct device select address is OB4 (hex). The incorrect select addresses are those single byte addresses which have 3 or less bits high. This is derived from the fact that a 4 input AND gate performs the device selection from the address bits of the select address. Thus a select address of OFF hex would attempt to select all devices.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL to the select address XX hex have a result other than OFF hex (open bus).

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (B4)

IFL to the correct Word Mover Controller address (OB4 hex) gave a result of OFF hex (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Word Mover Controller, IFL did not get a result of OFF hex (open bus).

TEST 01 - Testing Not Busy Flag (IFL Bit 7)

Tests that the Printer Busy Flag (IFL Bit 0) is not stuck low. That the Not Busy flag (IFL Bit 7) is set before a move, reset during a move, and set again after a move. That both DVCL and INIT individually abort a move and set the Not Busy Flag.

PRINTER BUSY FLAG STUCK LOW (IFL BIT 0)

After the Word Mover has been selected, a DVCL and an INIT instruction are executed. The Word Mover is reselected and the Printer Busy flag (IFL bit 0) is found to be low. The printer cable should be disconnected at this time.

MOVE WAS NOT EXECUTED

This message is displayed only if the Word Mover remains inactive for two consecutive moves. The source (line 5) is composed of FF's on the first move. Inactivity is established in the following manner. The Word Mover is issued commands to move line 5 on the screen to line 15. The Not Busy flag (IFL bit 7) is examined and found to be high, indicating no present board activity. SOCL (location 0822) is immediately examined and found to be equal to SOSL (location 0820), indicating no part of the source has been transferred to the Word Mover. The program repacks the source with 00's and again tries to move it to line 15. If there is still no activity the error message is displayed.

MOVE TERMINATED BEFORE NOT BUSY (IFL BIT 7) COULD BE EXAMINED

This message is displayed only if 2 consecutive moves are prematurely terminated before the Not Busy flag (IFL bit 7) can be detected low. The source (line 5) is composed of FF's on the first move. Premature termination is established in the following manner. The Word Mover is issued commands to move line 5 on the screen to line 15. The Not Busy flag (IFL bit 7) is immediately examined and found to be high, indicating no present board activity. The current vectors are examined and found to be static. SOCL (location 0822) is found to be not equal to SOSL (location 0820). This indicates that the move was begun, but terminated prematurely. The program repacks the source with 00's and repeats the move. If the move again prematurely terminates, the error message is displayed.

NOT BUSY (IFL BIT 7) LOW, BUT MOVE WAS NOT EXECUTED

The Word Mover is issued commands to move line 5 to line 15. The Not Busy flag (IFL bit 7) is immediately examined and found to be low. The program re-examines the Not Busy flag (IFL bit 7) after waiting long enough for the move to terminate and finds it low. The current vectors are examined and found to be static, indicating no move is in progress. SOCL (location 0822) is found to be equal to SOSL (location 0820), indicating no part of the source has been transferred to the Word Mover.

MOVE EXCEEDED MAXIMUM TIME ALLOWANCE

The Word Mover is issued commands to move line 5 to line 15 on the screen. The Not Busy flag (IFL bit 7) is imediately examined and found to be low. The program re-examines the Not Busy flag (IFL bit 7) after waiting long enough for the move to terminate and finds it low. The current vectors are examined and found to be changing, indicating that the move has not yet terminated. DVCL and INIT instructions are issued to try to abort the move and the error message is displayed.

NOT BUSY (IFL BIT 7) LOW AFTER MOVE TERMINATED

The Word Mover is issued commands to move line 5 to line 15 on the screen. The Not Busy flag (IFL bit 7) is immediately examined and found to be low. The program re-examines the Not Busy flag (IFL bit 7) after waiting long enough for the move to terminate and finds it still low. The current vectors (locations 0822, 0823, 0862 and 0863) are examined and found to be static, indicating that no move is in progress; if they are not equal, the error message is displayed.

NOT BUSY (IFL BIT 7) HIGH BEFORE MOVE TERMINATED

The Word Mover is issued commands to move line 5 to line 15 on the screen. When the Not Busy flag (IFL bit 7) goes high, the current vectors (locations 0822, 0823, 0862 and 0863) are examined. If the vectors are found to be changing, indicating that a move is still in progress, the error message is displayed.

DVCL DOES NOT TERMINATE MOVE

The Word Mover is issued commands to move line 5 to line 15 on the screen. The Not Busy flag (IFL bit 7) is immediately examined and confirmed to be low. A DVCL instruction is executed. The Not Busy flag (IFL bit 7) is re-examined and found to still be low. The current vectors (locations 0822, 0823, 0862 and 0863) are examined; if they are changing, the error message is displayed.

DVCL DOES NOT SET NOT BUSY (IFL BIT 7)

The Word Mover is issued commands to move line 5 to line 15 on the screen. The Not Busy flag (IFL bit 7) is immediately examined and confirmed to be low. A DVCL instruction is executed. The Not Busy flag (IFL bit 7) is re-examined and found to still be low. The current vectors (locations 0822, 0823, 0862 and 0863) are examined; if they are static, the error message is displayed.

INIT DOES NOT TERMINATE MOVE

The Word Mover is issued commands to move line 5 to line 15 on the screen. The Not Busy flag (IFL bit 7) is immediately examined and confirmed to be low. An INIT instruction is executed. The Word Mover is selected and the Not Busy flag (IFL bit 7) is re-examined and found to still be low. The current vectors (locations 0822, 0823, 0862 and 0863) are examined; if they are changing, the error message is displayed.

INIT DOES NOT SET NOT BUSY (IFL BIT 7)

The Word Mover is issued commands to move line 5 to line 15 on the screen. The Not Busy flag (IFL bit 7) is immediately examined and confirmed to be low. An INIT instruction is executed. The Word Mover is selected. The Not Busy flag (IFL bit 7) is re-examined and found to still be low. The current vectors (locations 0822, 0823, 0862 and 0863) are examined; if they are static, the error message is displayed.

TEST 02 - Testing Interrupt

Tests that the CPU is interrupted when the Not Busy flag (IFL bit 7) is set and is not interrupted when the Not Busy flag (IFL bit 7) is reset.

NOT BUSY FLAG STUCK LOW

After DVCL and INIT instructions are executed, the Word Mover is selected and the Not Busy flag (IFL bit 7) is examined. If the Not Busy flag (IFL bit 7) is low, the error message is displayed.

NO INTERRUPT DETECTED WITH NOT BUSY (IFL BIT 7) HIGH

An INIT instruction is executed. The Word Mover is selected and the Not Busy flag (IFL bit 7) is verified to be high. The interrupts are enabled. If the CPU does not detect an interrupt, the error message is displayed.

INTERRUPT DETECTED WITH NOT BUSY (IFL BIT 7) LOW

The Word Mover is issued commands to move line 5 to line 15 on the screen. During the move, the interrupts are enabled. If the CPU detects an interrupt during the move, the error message is displayed.

TEST 03 - Testing Current Vectors

Tests that the current vectors; SOCL, SOCH (locations 0822 and 0823), DECL and DECH (locations 0862 and 0863), are initialized by a move command, and that the contents of the destination current vectors are equal to the contents of the destination terminating vectors; SOTL, SOTH (locations 0824 and 0825), DETL and DETH (locations 0864 and 0865), when the move is terminated.

DESTINATION CURRENT VECTORS NOT EQUAL TO TERMINATING VECTORS

The program sets the current vectors; SOCL, SOCH (locations 0822 and 0823), DECL and DECH (locations 0862 and 0863), to be equal to the starting vectors. The Word Mover is issued commands to move line 5 to line 15 on the screen. When this move is terminated, the destination current and terminating vectors (locations 0864 and 0865) should be equal; if they are not, the error message is displayed.

SOURCE CURRENT VECTORS NOT EQUAL TO TERMINATING VECTORS PLUS 256

The program sets the current vectors; SOCL, SOCH (locations 0822 and 0823), DECL and DECH (locations 0862 and 0863), to be equal to the starting vectors. The Word Mover is issued commands to move line 5 to line 15 on the screen. When this move is terminated the source current vectors should be 256 address locations greater than the source terminating vectors (locations 0824 and 0825); if they are not, the error message is displayed.

SOURCE CURRENT VECTORS DID NOT INITIALIZE

The source current vectors (locations 0822 and 0823) are set 1 page before the starting vectors (locations 0820 and 0821). Commands for a move from line 5 to line 15 on the screen are executed. When the move terminates, the source current vectors are examined. If the source current vectors are not equal to the source terminating vectors (locations 0824 and 0825) plus 256 address locations, the error message is displayed.

DESTINATION CURRENT VECTORS DID NOT INITIALIZE

The destination current vectors (locations 0862 and 0863) are set 1 page before the starting vectors (locations 0860 and 0861). Command for a move from line 5 to line 15 on the screen are executed. When the move terminates the destination current vectors are examined. If the destination current vectors are not equal to the destination terminating vectors (locations 0864 and 0865), the error message is displayed.

MOVE EXCEEDED MAXIMUM TIME ALLOWANCE

The program times all moves in this test. Any move exceeding a period specified by the program will be aborted with DVCL and INIT instructions.

TEST 04 - Test That Every Character Can Be Moved

The source is displayed on the upper right of the screen. It consists of 17 characters. The first 16 characters are sequentially increasing, the last character is a terminating character equal to the complement of the first character. The source is moved to an empty line directly below it. The program examines the characters moved and displays their hex equivalents on the same line on the left side of the screen. If the character is moved incorrectly the hex equivalent is reversed on the screen. This process is repeated 16 times in order to display the hex equivalents for all 256 characters.

INCORRECT MOVE EXECUTED

After a move, all characters in the source and destination buffers are compared. If they are not equal, the error message is displayed.

TEST 05 - Tests That Margins Are Ignored By Move Without Wrap

A source 78 characters long is created on line 5 on the screen. The left and right margins are set at 00 and FF initially. Commands to move the source to line 15 are executed. If the move was executed properly, new margins are set and the move is repeated. The margins tested are shown below:

LEFT	RIGHT
00	FF
01	7F
03	3F
07	1F
0F	0F
1F	07
3F	03
7F	01
FF	00

The source, except for the terminating character, is shifted once to the left after each move. The margins are marked by "↓" on the line above the destination.

INCORRECT MOVE EXECUTED

After a move, all characters in the source and destination buffers are compared. If they are not equal, the error message is displayed.

TEST 06 - Tests That COM3 Command Byte Is Ignored By Move Command Without Wrap

A source 78 characters long is created on line 5 of the screen. The COM3 command byte (character locator) is 00 initially. If the command to move was executed properly, the move will be repeated with another COM3 command byte. The command byte will take on the following values: 00, 01, 03, 07, 0F, 01F, 03F, 07F, and 0FF hex. The source, except for the terminating character, is shifted once to the left after every move.

INCORRECT MOVE EXECUTED

After a move all characters in the source and destination buffers are compared. If they are not equal, the error message is displayed.

TEST 07 - Testing Terminating Commands

Tests that all possible terminating conditions will terminate a move. The source is created on line 5. The expected termination is marked in the source by a character having a reversed field. Commands are executed to move the source to line 15. After the move, the current vectors are examined to determine if the move terminated properly. The termination condition being tested is always displayed on line 9. An explanation of how the termination condition is tested and its error messages follow.

A). Testing move terminated on destination address

The termination point is marked by a reverse field "T". After the move, line 15 should include all source characters up to and including "T".

DESTINATION CURRENT NOT EQUAL TO TERMINATING ADDRESS

After the move, the contents of the destination current vectors were found to be not equal to the contents of the destination terminating vectors, indicating that the move was not terminated on the correct address.

B). Testing move terminated on source address

The termination point is marked by a reverse field "T". After the move line 15 should contain all source characters up to and including the "T".

CURRENT DESTINATION ADDRESS INCORRECT

After the move, the contents of the current destination vectors were found to be not equal to the contents of the starting destination vectors plus 014 hex.

CURRENT SOURCE ADDRESS INCORRECT

After the move, the contents of the current source vectors were found to be not equal to the contents of the terminating source vectors.

C). Testing move terminated on destination character

The termination character is the only reversed character in the middle of line 5. After the move line 15 should contain all source characters up to and including the reversed character. The following termination characters are tested, 00, 01, 02, 04, 08, 010, 020, 040, 080 hex.

CURRENT DESTINATION ADDRESS INCORRECT

After the move, the contents of the current destination vectors were found to be not equal to the contents of the starting destination vectors plus 028 hex.

CURRENT SOURCE ADDRESS INCORRECT

After the move, the contents of the current source vectors were found to be not equal to the contents of the starting source vectors plus 040 hex.

D). Testing move terminated on source character

The termination character is the only reversed character in the middle of line 5. After the move, line 15 should contain all source characters up to and including the reversed character. The following termination characters are tested, 0FF, 07F, 0BF, 0DF, 0EF, 0F7, 0FB, 0FD, and 0FE hex.

CURRENT DESTINATION ADDR[illegible]SS INCORRECT

After the move, the contents of the current destination vectors were found to be not equal to the contents of the starting destination vectors plus 028 hex.

CURRENT SOURCE ADDRESS INCORRECT

After the move, the contents of the current source vectors were found to be not equal to the contents of the starting source vectors plus 028 hex.

E). Testing move terminated on destination control character

The termination character is the only reversed character in the middle of line 5. After the move, line 15 should contain all source characters up to and including the reversed (control) character. Control characters 101 thru 01F hex are tested.

CURRENT DESTINATION ADDRESS INCORRECT

After the move, the contents of the current destination vectors were found to be not equal to the contents of the starting destination vectors plus 028 hex.

CURRENT SOURCE ADDRESS INCORRECT

After the move, the contents of the current source vectors were found to be not equal to the contents of the starting source vectors plus 040 hex.

F). Testing move terminated on source control character

The termination character is the only reversed character in the middle of line 5. After the move, line 15 should contain all source characters up to and including the reversed (control) character. Control characters 010 thru 01F hex are tested.

CURRENT DESTINATION ADDRESS INCORRECT

After the move, the contents of the current destination vectors were found to be not equal to the contents of the starting destination vectors plus 028 hex.

CURRENT SOURCE ADDRESS INCORRECT

After the move, the contents of the current source vectors were found to be not equal to the contents of the starting destination vectors plus 028 hex.

G). Tests that destination terminating character is ignored when DETM bit 7 is high

The termination character is the 2nd reversed character at the end of line 5. The 1st reversed character in the middle of line 5 is a destination terminating character. However, since DETM bit 7 is high, the move should not terminate on this character, but should terminate on the 2nd reversed character (source terminating character). The table below shows the values of the terminating characters on each move:

DESTINATION CHARACTER	SOURCE CHARACTER
11	FF
12	11
13	12
.	.
.	.
.	.
20	1F
40	20
80	40
01	80
02	01
04	02
08	04
10	08

MOVE TERMINATED ON DESTINATION CHARACTER

After the move, the contents of the current destination vectors were found to be equal to the contents of the starting destination vectors plus 028 hex. This is the address of the destination termination character.

MOVE NOT TERMINATED ON SOURCE TERMINATING CHARACTER

After the move, the contents of the current destination vectors were found to be equal to the contents of the starting destination vectors plus 04E hex. This is the address of the source terminating character.

H). Tests that source terminating character is ignored when SOTH BIT 7 is high

The termination character is the 2nd reversed character at the end of line 5. The 1st reversed character in the middle of line 5 is a source terminating character. However, since SOTH bit 7 is high, the move should not terminate on this character, but should terminate on the 2nd reversed character (destination terminating character). The table below shows the values of the terminating characters on each move.

SOURCE CHARACTER	DESTINATION CHARACTER
10	0F
11	10
12	11
.	.
.	.
.	.
1F	1E
20	10
40	20
80	40
01	80
02	01
04	02
08	04

MOVE TERMINATED ON SOURCE CHARACTER

After the move, the contents of the current destination vectors were found to be equal to the contents of the starting destination vectors plus 028 hex. This is the address of the source termination character.

MOVE DID NOT TERMINATE ON DESTINATION CHARACTER

After the move, the contents of the current destination vectors were found to be unequal to the contents of the starting destination vectors plus 04E hex. This is the address of the destination terminating character.

I). Testing move terminated on source character when SOTH bit 7 is high.

The termination character is the only reversed character in the middle of line 5. Since COM2 bit 3 is set, the move will terminate on a source control character even though SOTH bit 7 is high. The table below shows the values of the terminating characters on each move:

SOURCE CHARACTER	DESTINATION CHARACTER
10	FF
11	FE
12	FD
.	.
.	.
.	.
1F	F0

CURRENT DESTINATION ADDRESS INCORRECT

After the move, the contents of the current destination vectors were found to be not equal to the contents of the starting destination vectors plus 28 hex. This is the address of the source terminating control character.

J). Testing Move Terminated on Destination Character when DETH Bit 7 is High

SOURCE CHARACTER	DESTINATION CHARACTER
80	10
40	11
20	12
10	13
08	14
04	15
02	16
01	17
80	18
.	.
.	.
.	.
01	1F

CURRENT DESTINATION ADDRESS INCORRECT

After the move, the contents of the current destination vectors were not equal to the contents of the starting vecotrs plus 028 hex. This is the address of the destination terminating control character.

TEST 08 - Test That Page Width Is Ignored By Move WithOut Wrap

The source (lines 5 thru 9) is moved to the destination (lines 15 thru 19). The move is executed once for every legitimate page width (COM2 bits 5 and 6).

INCORRECT MOVE EXECUTED

After the move, the source is matched character for character with the destination. If any mismatch is found, the error message is displayed.

TEST 09 - Test Terminating Flags

Tests that the Source Termination flag (IFL bit 5) is not stuck high, is set by a source terminated move, is not set by a destination terminated move, and is reset by DVCL and INIT instructions. Test that the Destination Termination flag (IFL bit 4) is not stuck high, is set by a destination terminated move, is not set by a source terminated move, and is reset by DVCL and INIT instructions. Test that both flags can be set simultaneously. All moves in this test are terminated on an address not a character. The terminating address is always set to be one greater than the starting address.

SOURCE TERMINATION FLAG STUCK HIGH

DVCL and INIT instructions are executed. The Word Mover is selected and the Source Termination flag (IFL bit 5) is examined. If the Source Termination flag (IFL bit 5) is found set, the error message is displayed.

DESTINATION TERMINATION FLAG STUCK HIGH

DVCL and INIT instructions are executed. The Word Mover is selected and the Destination Termination flag (IFL bit 4) is examined. If the Destination Termination flag (IFL bit 4) is found set, the error message is displayed.

SOURCE TERMINATION FLAG STUCK LOW

The source vectors are set to terminate a move after transferring one character from line 5 to line 15. The destination vectors do not terminate the move. After the move, the Source Termination flag (IFL bit 5) is examined. If the Source Termination flag (IFL bit 5) is low, the error message is displayed.

MOVE TERMINATED BY SOURCE SET DESTINATION TERMINATION FLAG

The source vectors are set to terminate a move after transferring one character from line 5 to line 15. The destination vectors do not terminate the move. The Destination Termination flag (IFL bit 4) is examined after the move. If the Destination Termination flag (IFL bit 4) is set, the error message is displayed.

DVCL DOES NOT RESET SOURCE TERMINATION FLAG

The source vectors are set to terminate a move after transferring one character from line 5 to line 15. The destination vectors do not terminate the move. The Source Termination flag (IFL bit 5) is examined and confirmed to be set. A DVCL instruction is executed and the Source Termination flag (IFL bit 5) is re-examined. If the Source Termination flag (IFL bit 5) is found set, the error message is displayed.

INIT DOES NOT RESET SOURCE TERMINATION FLAG

The source vectors are set to terminate a move after transferring one character from line 5 to line 15. The destination vectors do not terminate the move. The Source Termination flag (IFL bit 5) is examined and confirmed to be set. An INIT instruction is executed. The Word Mover is reselected and the Source Termination flag (IFL bit 5) is examined. If the Source Termination flag (IFL bit 5) is found set, the error message is displayed.

DESTINATION TERMINATION FLAG STUCK LOW

The destination vectors are set to terminate a move after transferring one character from line 5 to line 15. The source vectors do not terminate the move. After the move, the Destination Termination flag (IFL bit 4) is examined. If the Destination Termination flag (IFL bit 4) is low, the error message is displayed.

MOVE TERMINATED BY DESTINATION SET SOURCE TERMINATION FLAG

The destination vectors are set to terminate a move after transferring one character from line 5 to line 15. The source vectors do not terminate the move. The Source Termination flag (IFL bit 5) is examined after the move. If the Source Termination flag (IFL bit 5) is set, the error message is displayed.

DVCL DOES NOT RESET DESTINATION TERMINATION FLAG

The destination vectors are set to terminate a move after transferring one character from line 5 to line 15. The source vectors do not terminate the move. The Destination Termination flag (IFL bit 4) is examined and confirmed to be set. A DVCL instruction is executed and the Destination Termination flag (IFL bit 4) is re-examined. If the Destination Termination flag is found set, the error message is displayed.

INIT DOES NOT RESET DESTINATION TERMINATION FLAG

The destination vectors are set to terminate a move after transferring one character from line 5 to line 15. The source vectors do not terminate the move. An INIT instruction is executed. The Word Mover is reselected and the Destination Termination flag (IFL bit 4) is examined. If the Destination Termination flag is found set, the error message is displayed.

TERMINATING FLAGS ARE NOT SET SIMULTANEOUSLY

Both source and destination vectors are set to terminate a move after transferring one character from line 5 to line 15. After the move, both Source and Destination flags (IFL bits 4 and 5) are examined. If either bit is found low, the error message is displayed.

TEST 10 - Testing Margins

Initially, the left and right margins are set at 027 and 028 hex and the line length is set to 80. COM2 bit 4 is set so that no nulls will be written between the right and left margins. After this move has been executed, the program decreases the left margin by one, increases the right margin by one, and repeats the move. This process is continued until the left margin equals zero. At this point COM2 bit 4 is reset so that the area between margins is filled with nulls and the line length is set to 80. The program increases and decreases the left and right margins respectively by one between moves until the original values (027 and 028 hex) are reached. The program then sets the line length to 132 and repeats the opening and shrinking processes described above. Finally the line length is set to 160 and the margins are again opened and shrunk. There are 234 moves with wrap executed and checked character by character in this test. For an explanation of the error message format, see the appendix at the end of this manual.

TEST 11 - Testing COM3

The left and right margins are set at 03A and 04F hex. Initially the COM3 command byte (character locator) is 04F hex. After every move, the COM3 command byte is decremented by one until a value of zero is attained. There are 80 moves with wrap executed and checked character by character in this test. For an explanation of the error message format, see the appendix at the end of this manual.

TEST 12 - Test Word Greater Than 64 Bytes

Initially, the left and right margins are set at 00 and 01. COM2 bit 4 is set so that no nulls are written between margins. After each move, the right margin is incremented by one until a value of 027 hex is reached. At this point, COM2 bit 4 is reset so that the area between margins is filled with nulls and the line length is set to 160. The right margin is now decremented by one after each move until a value of 01 is reached. The source for this test contains two unusual words:

i. The first word in the source text is, "N@@@@U@@@L@@L@S" (@ represents a null). The Word Mover should delete any nulls (00 hex) before transferring a character to the destination. Therefore, this word should appear in the destination as, "NULLS".

ii. The second word in the source text is greater than 64 bytes. Words of this length are wrapped differently than the shorter words contained in the remainder of the source.

There are 78 moves with wrap executed and checked character by character in this test. For an explanation of the error message format, see the appendix at the end of this manual.

TEST 13 - Testing Control Delimiters

Initially the left and right margins are set at 00 and 01. COM2 bit 4 is set so that no nulls are written between margins. COM2 bit 1 is set so that control delimiters are recognized. After each move, the right margin is incremented by one until a value of 03C hex is reached. At this point COM2 bit 4 is reset so that the area between margins is filled with nulls. The COM3 command byte, destination starting address, and the left margin are incremented by one after every move until the left margin is equal to 03B hex. The source for this test has 3 unusual features:

i) The first seventeen words in the source are separated from each other by 1 control delimiter (010-01F hex). These words are used to verify that control delimiters are recognized when COM2 bit 1 is set.

ii) The 17th word in source is, "N@@@@U@@@L@@L@S" (@ represents null). The Word Mover should delete any nulls (00 hex) before transferring a character to the destination. Therefore, this word should appear in the destination as, "NULLS".

iii) The 18th word in the source is greater than 64 bytes. Words of this length are wrapped differently than the shorter words contained in the remainder of the source.

There are 119 moves with wrap executed and checked character by character in this test. For an explanation of the error message format, see the appendix at the end of this manual.

TEST 14 - Testing Buffer Integrity

A) A move is destination terminated after 1 character has been transferred to the destination. The destination starting vectors are made equal to the terminating vectors. The terminating vectors are increased to extend the destination buffer by 4 lines (4 x 80)-1. The COM3 command byte is incremented by one and another move is executed.

B) A move is source terminated after 1 word has been transferred to the destination. The source starting vectors are made equal to the terminating vectors. The terminating vectors are increased so that the destination buffer (4 lines) can be completely filled.

TEST 15 - Testing Move Terminated on Destination Address with Word Wrap

Each complete wrap in this test is composed of a series of 1 byte moves. After a character has been moved to the destination. The starting destination address is made equal to the current address and the terminating address is made equal to the current address plus one. The move is then restarted with a COM3 command. The command byte associated with the COM3 is always equal to the position in the line where the next data byte should appear.

The margin commands are changed after each complete wrap. The right margin is incremented from 1 to 79, the left margin is incremented from 0 to 78.

The purpose of this test is to determine if all ECO's are properly implemented. If all previous tests pass, the problem is very probably an improper ECO.

TEST 16 - Testing Move Terminated on Source Character with Word Wrap

Each complete wrap in this test is composed of a series of 1 byte moves. After a character has been moved to the destination, the source starting address is made equal to the current address. The data byte in the memory location equal to the current address plus one is placed in the source terminating character vector. The move is then restarted with a COM3 command. The command byte associated with the COM3 is always equal to the position in the line where the next data byte should appear.

The right margin command is decremented after each complete wrap from 79 to 1.

APPENDIX

ERROR CHECKING PROCEDURES AND MESSAGES IN TEST 10 THRU 14

The program will examine the commands and vectors for the WORD MOVER. As the hardware begins execution of the move, the program duplicates the same move with software and places its output in a buffer beneath the title, "***SOFTWARE WRAP***". The hardware places its output in a destination buffer beneath the title, "*** HARDWARE WRAP ***". Both buffers have a length of 5 lines (5 x 80 bytes). The program then begins comparing the buffers character by character.

If a mismatch occurs, the erroneous hardware character and its corresponding software character will 'spin' on the screen. The spinning software character will be placed in the message, "EXPECTED CHARACTER = X". The erroneous hardware character will be placed in the message, "ACTUAL CHARACTER = X". Instructions for continuing will also be displayed on the screen. These instructions are explained below.

If no mismatch occurs the next move will be executed.

INSTRUCTION KEYS

SHIFTED "R" KEY	Repeat the wrap and halt for further instructions. The screen is cleared before the wrap is repeated.
"R" KEY	Repeat the entire test.
"S" KEY	Skip to the next test.
"SPACE" KEY	Continue the current test.
"PROG" KEY	Restart the program.

TEST 17 - Testing 90-9F as Terminating Characters in Addition to IOM Commands

A string of characters is placed in the source buffer. In the middle of the buffer, characters 90-9F are inserted one at a time and a move is executed. The current locations should contain the address of the control character.

TEST 18 - Testing FIFOS and Shift Registers (WRDTST Only)

This test checks the long term data retention of the FIFOS and shift registers.

The source and destination buffers are setup for 320D bytes each (64 bytes for the FIFO and 256 bytes for the shift register). A move with wrap is executed (LEFT MAR.=0, RIGHT MAR.=50H). The entire source buffer is read in and held for five seconds. Then, a COM3 is issued to clear the FIFO and shift registers. The data is then checked.

This test is executed twice. First with the data pattern 55H, AAH, 55H, etc. Then with the pattern AAH, 55H, AAH, etc.

If an error is found the following message will appear:

ADDRESS AAAAH IS DEFECTIVE. EXP=BBH ACT=CCH where

AAAAH = The relative address of the error. If AAAAH = 0000H to 003FH the FIFO is defective. If AAAAH = 0040H to 013FH the shift register is defective.

BBH = The expected value either 55H or AAH.

CCH = The actual value read from the buffer.

Test 19 - Testing Words Less Than 64 Bytes Long

A word between 2 and 64 characters long is placed at the end of the line. The right margin is set to the last character of the word. A move with wrap is performed. The word should be moved to the second line of the destination buffer.

APPENDIX

ERROR CHECKING PROCEDURES AND MESSAGES IN TEST 10 THRU 14

The program will examine the commands and vectors for the WORD MOVER. As the hardware begins execution of the move, the program duplicates the same move with software and places its output in a buffer beneath the title, "***SOFTWARE WRAP***". The hardware places its output in a destination buffer beneath the title, "*** HARDWARE WRAP ***". Both buffers have a length of 5 lines (5 x 80 bytes). The program then begins comparing the buffers character by character.

If a mismatch occurs, the erroneous hardware character and its corresponding software character will 'spin' on the screen. The spinning software character will be placed in the message, "EXPECTED CHARACTER = X". The erroneous hardware character will be placed in the message, "ACTUAL CHARACTER = X". Instructions for continuing will also be displayed on the screen. These instructions are explained below.

If no mismatch occurs the next move will be executed.

INSTRUCTION KEYS

SHIFTED "R" KEY	Repeat the wrap and halt for further instructions. The screen is cleared before the wrap is repeated.
"R" KEY	Repeat the entire test.
"S" KEY	Skip to the next test.
"SPACE" KEY	Continue the current test.
"PROG" KEY	Restart the program.

TROUBLE SHOOTING MODE

This mode allows the operator to set up and execute move commands. A source buffer is displayed at the top of the screen. The vectors, commands and flags are displayed beneath the source buffer. The margins are marked on the screen with down arrows (081 hex). The destination starting and terminating addresses are marked with brackets (03E and 03C hex). A list of the command keys is given below:

FUNCTION KEYS

<u>"X" KEY</u>	Execute the commands displayed in the middle of the screen. DVCL and CLICK instructions are issued immediately before each move. The CLICK is issued to allow the technician to trigger an oscilloscope at the beginning of the move command. All vectors and flags will be updated after each move. Holding the key down will continuously repeat the move.
<u>"C" KEY</u>	Clear the destination buffer.
<u>"I" KEY</u>	Reset the vectors, commands and source buffer to their initial values.
<u>"PROG" KEY</u>	Escape the Trouble Shooting Mode and restart the program.
<u>" " KEY</u>	Scroll screen up.
<u>" " KEY</u>	Scroll screen down.

To change vectors and commands, the operator must enter another mode by typing the "HOME" key. Upon typing the "HOME" key, the cursor address and contents are displayed at the top of the screen. This message can also be used as a mode indicator. None of the function keys above, except "PROG" are active in this mode. To change a command, the operator must move the cursor using the arrow keys to a position under the data to be changed. New data typed is displayed on the screen. The source buffer and vectors may be changed in the same manner. If the operator positions the cursor in the destination buffer and types the "F4" key, the text starting location and commands will automatically change to correspond to the cursor position. That is, the destination starting vector will be set one before the cursor address and the left margin and COM3 command bytes will be set to the value of the cursor position on the line. If the operator attempts to make the left margin greater than the right margin the error message "INVALID MARGIN" will appear on line 2. If the operator positions the cursor outside the destination buffer and types the "F4" key, the error message, "INVALID DESTINATION STARTING ADDRESS," will appear on line 2.

After all changes have been made on the screen, the operator should type the ENTER key (C9). This will cause the program to accept all valid changes made on the screen. Any command or vector not given a hex value will not be accepted. Instead, a beep will be issued and the cursor will be positioned below the illegal character. If all changes are valid, the program will leave this mode and wait for the operator to type one of the previous function keys listed above.

ASCTST

ASYNCHRONOUS CONTROLLER TEST

ASCTST - ASYNCHRONOUS CONTROLLER TEST

Applicable Assemblies

5000-1134-X	Asynchronous Communications Controller	(X = 1 to 10)

Required Test Assemblies

IOTST80 Diagnostic Plug	(Dash 1 thru 10)
COMTST3 Diagnostic Plug	(Dash 1 and 6)
COMTST4 Diagnostic Plug	(Dash 2 and 7)
COMTST5 Diagnostic Plug	(Dash 4 and 9)
COMTST6 Diagnostic Plug	(Dash 4 and 9)
COMTST7 Diagnostic Plug	(Dash 3 and 8)

General Description

The purpose of the ASCTST program is to determine if the Asynchronous Controller is working properly and, if not, to give an indication of which functions are incorrect. Prior to running ASCTST, the Asynchronous I/O Adapter must be tested by IOTST and be known good. From one to three Diagnostic Plugs are needed to run ASCTST.

16K (or more) of memory is required to run ASCTST.

This manual applies to both 8008 and 8080 versions of ASCTST.

Loading Procedure

ASCTST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If ASCTST loads properly, it will identify itself and request the Dash number of the board under test and the select address.

Operator Action

After the run-mode information pause, the following message will appear: "PLEASE ENTER THE SELECT ADDRESS: 69". The user may change the select address by typing a two character hex address. The select address will default to select address 69 by typing a carriage return.

The next message to appear is: "PLEASE ENTER DASH # (ENTER 0 FOR DASH 10)" will be displayed. At this time the operator must type the Dash number of the board under test.

Following this action, there will be a message displayed requesting the particular plug or plugs to be inserted into their respective connectors on the OP-1, as shown in Figure 1. After following the instructions, typing a space will start ASCTST.

For Dash 4 and 9 the message "PLEASE REMOVE THE COMTST5 PLUG AND INSERT THE COMTST6 PLUG" will be displayed during Test 26. After following the instructions typing a space will cause ASCTST to continue.

All tests except Test 26 (Dash 4 and 9 only) operate automatically and without operator intervention. When the entire test has been completed, the prompt "TYPE SPACE TO REPEAT ASCTST" will be displayed. If a space is typed on the keyboard, ASCTST will restart.

Refer to Appendix A for specialized test run options.

Errors

All errors are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The bell will ring only to notify the operator of an error. The error message displayed on the screen attempts to give a description of the nature of the problem. In some cases, this should be adequate to diagnose and fix the error. Otherwise, refer to the detailed description of the specific test to determine the purpose and expected results for the displayed error message.

After an error message is displayed, the operator has four ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, depressing the PROG key will restart the program, and typing PROG (with the SHIFT key) will return control to the operating system.

Because of the inverted pyramid test strategy used in ASCTST, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

Diagnostic Plugs

The Diagnostic Plugs are necessary for testing the Asynchronous Communications Controller. The IOTST80 Diagnostic Plug must be inserted into the Asynchronous I/O Adapter connector and the COMTSTX (X = 3 to 7) Diagnostic Plug into the connector D1 (as shown in Figure 1) on the rear panel of the OP-1 prior to program execution. The schematics of these plugs are shown in Figures 2 thru 7.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing. Below the description all possible error messages are listed, with an explanation of the cause of the message.

Test 00 - Controller Select Test

Tests that the Asynchronous Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selects a selected Asynchronous Controller. The correct device select address is 069 hex. The incorrect select addresses are those single byte addresses which have 3 or less bits high. This is derived from the fact that a 4 input AND gate performs the device selection from the address bits of the select address. Thus a select address of 0FF hex would attempt to select all devices.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL or INP to the select address XX hex gave a result other than 0FF hex (open bus).

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (0C3)

IFL to the correct Asynchronous Controller address (069 hex) gave a result of 0FF hex (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Asynchronous Controller, IFL did not get a result of 0FF hex (open bus).

Test 01 - Test Static Flags

NOTE: For Dash 1 and 6 only.

Tests that the static input flags and output flags can be properly set and reset. If the Ring Detector flag (IFL bit 0) appears to be unresponsive to the Break flag (OFL bit 3), the Transmitter is used to attempt to "unstick" the Ring Detector flag (IFL bit 0).

INIT AND DVCL DOES NOT SET TRANSMITTED DATA TO A STOP BIT

After issuing an INIT and DVCL, Transmitted Data was set to a start bit (+12V.).

RING DETECTOR, BREAK, OR TRANSMITTED DATA STUCK HIGH

The Ring Detector flag (IFL bit 0) appeared to be unresponsive to the Break flag (OFL bit 3). The Transmitter failed to "unstick" the Ring Detector flag by transmitting an alternating string of 1's and 0's.

BREAK STUCK LOW

Issuing an OFL with the Break flag (OFL bit 3) set does not set the known good Ring Detector flag (IFL bit 0).

DVCL RESETS BREAK

Issuing a DVCL with the Break flag (OFL bit 3) set, reset the known good Ring Detector flag (IFL bit 0).

INIT DOES NOT RESET BREAK

Issuing an INIT with the Break flag (OFL bit 3) set, does not reset the known good Ring Detector flag (IFL bit 0).

DATA TERMINAL READY OR DATA SET READY STUCK HIGH

The Data Set Ready flag (IFL bit 1) could not be reset by issuing an INIT and an OFL with the Data Terminal Ready flag (OFL bit 1) reset.

DATA TERMINAL READY OR DATA SET READY STUCK LOW

The Data Set Ready flag (IFL bit 1) could not be set by issuing an OFL with the Data Terminal Ready flag (OFL bit 1) set.

DVCL RESETS DATA TERMINAL READY

Issuing a DVCL with the Data Terminal Ready flag (OFL bit 1) set, reset the known good Data Set Ready flag (IFL bit 1).

INIT DOES NOT RESET DATA TERMINAL READY

Issuing an INIT with the Data Terminal Ready flag (OFL bit 1) set, does not reset the known good Data Set Ready flag (IFL bit 1).

SUPERVISORY TRANSMIT DATA OR CARRIER DETECTOR STUCK HIGH

The Carrier Detector flag (IFL bit 2) could not be reset by issuing an INIT and an OFL with the Supervisory Transmit Data flag (OFL bit 2) reset.

SUPERVISORY TRANSMIT DATA OR CARRIER DETECTOR STUCK LOW

The Carrier Detector flag (IFL bit 2) could not be set by issuing an OFL with the Supervisory Transmit flag (OFL bit 2) set.

DVCL RESETS SUPERVISORY TRANSMIT DATA

Issuing a DVCL with the Supervisory Transmit Data flag (OFL bit 2) set, reset the known good Carrier Detector flag (IFL bit 2).

INIT DOES NOT RESET SUPERVISORY TRANSMIT DATA

Issuing an INIT with the Supervisory Transmit Data flag (OFL bit 2) set, does not reset the known good Carrier Detector flag (IFL bit 2).

REQUEST TO SEND OR SUPERVISORY RECEIVED DATA STUCK HIGH

The Supervisory Received Data flag (IFL bit 3) could not be reset by issuing an INIT and an OFL with the Request To Send flag (OFL bit 0) reset.

REQUEST TO SEND OR SUPERVISORY RECEIVED DATA STUCK LOW

The Supervisory Received Data flag (IFL bit 3) could not be set by issuing an OFL with the Request To Send flag (OFL bit 0) set.

DVCL RESETS REQUEST TO SEND

Issuing a DVCL with the Request To Send flag (OFL bit 0) set, reset the known good Supervisory Received Data flag (IFL bit 3).

INIT DOES NOT RESET REQUEST TO SEND

Issuing an INIT with the Request To Send flag (OFL bit 0) set, does not reset the known good Supervisory Received Data flag (IFL bit 3).

Test 02 - Test Transmit capabilities and Interrupt

Tests that a Transmit (COM1) resets the Main Channel Not Busy flag (IFL bit 6), sets Request To Send, waits for Clear To Send, and then initializes the Asynchronous Buffer Current address and increments it until it reaches the Terminating address. Also tests that INIT, DVCL, reaching Terminating address, and reaching Terminating character individually set the Main Channel Not Busy flag and reset Request To Send. Also tests that the controller interrupts with the Main Channel Not Busy flag set.

MAIN NOT BUSY STUCK LOW

Issuing an INIT and a DVCL could not set the Main Channel Not Busy flag (IFL bit 6).

TRANSMIT DOES NOT SET REQUEST TO SEND

During a Transmit (COM1) Request To Send was "0" (-12V.).

TRANSMIT DOES NOT RESET MAIN NOT BUSY

During a Transmit (COM1) the Main Channel Not Busy flag (IFL bit 6) was set.

TRANSMIT DOES NOT SET REQUEST TO SEND OR RESET MAIN NOT BUSY

During a Transmit (COM1) Request To Send was "0" (-12V.) and the Main Channel Not Busy flag (IFL bit 6) was set.

INIT DOES NOT SET MAIN NOT BUSY

During a Transmit (COM1) an INIT did not cause the Main Channel Not Busy flag (IFL bit 6) to set.

INIT DOES NOT RESET REQUEST TO SEND

During a Transmit (COM1) an INIT did not cause Request To Send to become a "0" (-12V.).

INIT DOES NOT SET MAIN NOT BUSY OR RESET REQUEST TO SEND

During a Transmit (COM1) an INIT did not cause the Main Channel Not Busy flag (IFL bit 6) to set and Request To Send to become a "0" (-12V.).

DVCL DOES NOT SET MAIN NOT BUSY

During a Transmit (COM1) a DVCL did not cause the Main Channel Not Busy flag (IFL bit 6) to set.

DVCL DOES NOT RESET REQUEST TO SEND

During a Transmit (COM1) a DVCL did not cause Request To Send to become a "0" (-12V.).

DVCL DOES NOT SET MAIN NOT BUSY OR RESET REQUEST TO SEND

During a Transmit (COM1) a DVCL did not cause the Main Channel Not Busy flag (IFL bit 6) to set and Request To Send to become a "0" (-12V.).

CLEAR TO SEND LOW DOES NOT HOLD OFF TRANSMIT

NOTE: For Dash 1 and 6 only

Transmission of data from the Asynchronous Buffer takes place without Clear To Send a "1" (+12V.).

CURRENT ADDRESS NOT INITIALIZED CORRECTLY

Issuing a Transmit (COM1) does not cause the Asynchronous Buffer Current address to be initialized to the Starting address.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER OR ADDRESS

During a Transmit (COM1) the Asynchronous Buffer Current address incremented past the Terminating character and address.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING ADDRESS

During a Transmit (COM1) the Asynchronous Buffer Current address incremented past the Terminating address.

REACHING TERMINATING ADDRESS DOES NOT SET MAIN NOT BUSY

The end of a transmit did not cause the Main Channel Not Busy flag (IFL bit 6) to set.

REACHING TERMINATING ADDRESS DOES NOT RESET REQUEST TO SEND

The end of a transmit did not cause Request To Send to become a "0" (-12V.).

REACHING TERMINATING ADDRESS DOES NOT SET MAIN NOT BUSY OR RESET REQUEST TO SEND

The end of a Transmit (COM1) did not cause the Main Channel Not Busy flag (IFL bit 6) to set and Request To Send to become a "0" (-12V.).

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER

During a Transmit (COM1) the Asynchronous Buffer Current address incremented past the Terminating character.

REACHING TERMINATING CHARACTER DOES NOT SET MAIN NOT BUSY

The end of a transmit did not cause the Main Channel Not Busy flag (IFL bit 6) to set.

REACHING TERMINATING CHARACTER DOES NOT RESET REQUEST TO SEND

The end of a transmit did not cause Request To Send to become a "0" (-12V.).

REACHING TERMINATING CHARACTER DOES NOT SET MAIN NOT BUSY OR RESET REQUEST TO SEND

The end of a transmit did not cause the Main Channel Not Busy flag (IFL bit 6) to set and Request To Send to become a "0" (-12V.).

INTERRUPT PRIORITY LEVEL NO. 6 NOT ISSUED WITH MAIN NOT BUSY SET

With the Main Channel Not Busy flag (IFL bit 6) set Interrupt 6 is not generated.

INTERRUPT PRIORITY LEVEL NO. 6 ISSUED WITH MAIN NOT BUSY RESET

With the Main Channel Not Busy flag (IFL bit 6) reset, Interrupt 6 is generated.

Test 03 - Test Terminating Characters

Tests that all 256 (00-0FF hex) characters can be used as terminating characters. The test is performed by setting up the Asynchronous Buffer with the terminating character preceded and followed by the inverse of the terminating character. A Transmit (COM1) is issued and upon termination of transmission the Current address is examined to verify that termination was caused by the terminating character.

CHARACTER (XX) DOES NOT TERMINATE TRANSMISSION

The Current address of the Asynchronous Buffer incremented past the Terminating character (XX) during a Transmit (COM1).

<u>Test 04 - Test Baud rate with maximum tolerance</u>

Tests transmission using the Main Channel Not Busy flag by timing how long the hardware stays busy during transmission of a known number of characters. At all speeds, communication parameters are set for 1 stop bit, 8 data bits, no parity bit, and thus 10 bits per character (including the start bit) are expected. Since character length has not yet been tested, however, timing ranges can accomodate as few as 9 bits, normally the least transmittable per character, or as many as 12 bits, normally the most transmittable per character. All speeds supported by the Asynchronous controller are tested.

MAIN NOT BUSY DOES NOT SET WITH BAUD RATE X

During testing of Baud X the Main Channel Not Busy flag (IFL bit 6) failed to set in a reasonable time period.

BAUD RATE SET FOR X BUT IS INCORRECTLY Y

During testing of Baud X it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance expected for Baud Y.

BAUD RATE SET FOR X BUT IS INCORRECTLY FASTER THAN 38400

During testing of Baud X it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance exceeding Baud 38400.

BAUD RATE SET FOR X BUT IS INCORRECTLY SLOWER THAN 110

During testing of Baud X it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance below Baud 110.

BAUD RATE SET FOR X BUT IS INCORRECTLY BETWEEN Y AND Z

During testing of Baud X it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance between Baud Y and Z.

<u>Test 05 - Test transmitting 7 data bits</u>

Tests transmission of the least significant 7 data bits (00-07F hex).

-AAA,AAAA STUCK BITS FOR 7 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits transmitted. The rightmost 7 bits from right to left are data bits 6 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

<u>Test 06 - Test OUT command</u>

Tests that the number of stop bits, data bits, and parity odd or even can be correctly set in all combinations. This scheme tries each combination in turn and determines if the bit stream transmitted is correct. If not, an error is recorded for that combination. Then an analysis is performed to see if there is a preponderance of errors for any particular state of one of the OUT command bits.

MULTIPLE OUT COMMAND ERRORS

Testing the stop bits, data bits, and parity bits yielded errors in over 50% of the transmissions. At least 3 of the 4 most significant bits in the OUT command are bad.

OUT COMMAND ERROR X

A significant number of errors were generated by changing the OUT command before each transmission. These errors corresponded to the OUT command with bit or bits X in their respective state, where X is equal to one to four of the following: 1 STOP BIT, 2 STOP BITS, 7 DATA BITS, 8 DATA BITS, PARITY, NO PARITY, ODD, or EVEN.

Test 07 - Test transmitting 8 data bits

Tests transmission of all 8 data bits.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits transmitted. The data bits from left to right are data bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 08 - Test Baud rate with minimum tolerance

Tests transmission using the Main Channel Not Busy flag by timing how long the hardware stays busy during transmission of a known number of characters. At all speeds, communication parameters are set for 1 stop bit, 8 data bits, no parity bit, and thus 10 bits per character (including the start bit) are expected. Since the OUT command has been tested and is presumably good, timing ranges are used which can accomodate only 10 bits. All speeds supported by the Asynchronous controller are tested.

MAIN NOT BUSY DOES NOT SET WITH BAUD RATE X

During testing of Baud X, the Main Channel Not Busy flag (IFL bit 6) failed to set in a reasonable time period.

BAUD RATE SET FOR X BUT IS INCORRECTLY Y

During testing of Baud X, it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance related to Baud Y.

BAUD RATE SET FOR X BUT IS INCORRECTLY FASTER THAN 38400

During testing of Baud X it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance exceeding Baud 38400.

BAUD RATE SET FOR X BUT IS INCORRECTLY SLOWER THAN 110

During testing of Baud X, it was found that the Main Channel Not Busy flag (IFL bit 6) incorrectly set during a time variance below Baud 110.

BUAD RATE SET FOR X BUT IS INCORRECTLY BETWEEN Y AND Z

During testing of Baud X it was found that the Main Channel Not busy flag (IFL bit 6) incorrectly set during a time variance between Baud Y and Z.

Test 09 - Test transmit with redundancy

NOTE: For Dash 1 thru 5 only.

Tests that all 256 (00-0FF hex) possible OCC values can be generated. Also tests that the Current address stops two address locations past the Terminating character to allow for the OCC transmission.

TERMINATING CHARACTER (XX) DOES NOT GENERATE OCC

After the Main Channel Not Busy flag (IFL bit 6) set, the Main Channel Current address was examined and not found to be two address locations past the Terminating character.

AAAA,AAAA STUCK BITS FOR OCC GENERATION

This stuck message is displayed if there is an error in any of the bits OCC transmitted. The bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at one point;
    - if that bit worked properly.

Test 10 - Test Main Channel Receive capabilities

Tests that a Receive (COM1) resets the Main Channel Not Busy flag (IFL bit 6). Also tests that a Receive increments the Current address up to the Terminating character or address and then sets the Main Channel Not Busy flag.

RECEIVE DOES NOT RESET MAIN NOT BUSY

During a Receive (COM1) the Main Channel Not Busy flag (IFL bit 6) was set.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER OR ADDRESS

During a Receive (COM1) the Main Channel Current address incremented past the Terminating character and address.

Test 09 - Test transmit with redundancy

NOTE: For Dash 1 thru 5 only.

Tests that all 256 (00-0FF hex) possible OCC values can be generated. Also tests that the Current address stops two address locations past the Terminating character to allow for the OCC transmission.

TERMINATING CHARACTER (XX) DOES NOT GENERATE OCC

After the Main Channel Not Busy flag (IFL bit 6) set, the Main Channel Current address was examined and not found to be two address locations past the Terminating character.

AAAA,AAAA STUCK BITS FOR OCC GENERATION

This stuck message is displayed if there is an error in any of the bits OCC transmitted. The bits from left to right are bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 10 - Test Main Channel Receive capabilities

Tests that a Receive (COM1) resets the Main Channel Not Busy flag (IFL bit 6). Also tests that a Receive increments the Current address up to the Terminating character or address and then sets the Main Channel Not Busy flag.

CURRENT ADDRESS NOT INITIALIZED CORRECTLY

During a receive (COM1) the main channel current address was not set to the Main Channel Starting Address +1.

RECEIVE DOES NOT RESET MAIN NOT BUSY

During a Receive (COM1) the Main Channel Not Busy flag (IFL bit 6) was set.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER OR ADDRESS

During a Receive (COM1) the Main Channel Current address incremented past the Terminating character and address.

Test 11 - Test Character Error

Tests that the Character Error flag (IFL bit 5) can detect parity errors. Also tests that a COM1, INIT, and DVCL can individually reset the Character Error flag.

CHARACTER ERROR STUCK HIGH

An INIT and DVCL could not reset the Character Error flag (IFL bit 5).

CHARACTER ERROR STUCK LOW

Transmitting a known bad parity character to the receiver while in the receive mode does not set the Character Error flag (IFL bit 5).

COM1 DOES NOT RESET CHARACTER ERROR

Issuing a new Receive (COM1) with the Character Error flag (IFL bit 5) set cannot reset it.

INIT DOES NOT RESET CHARACTER ERROR

Issuing an INIT with the Character Error flag (IFL bit 5) set cannot reset it.

DVCL DOES NOT RESET CHARACTER ERROR

Issuing a DVCL with the Character Error flag (IFL bit 5) set cannot reset it.

Test 12 - Test Main Channel receiving 7 data bits with odd parity

Tests the receiver for correct character reception of all characters from 00-07F hex with parity odd.

PARITY ERROR DURING RECEIVE

During a Receive (COM1) of characters with known good parity, the Character Error flag (IFL bit 5) set.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits received. The leftmost bit reflects the status of the parity bit and the next 7 from left to right are data bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 13 - Test Main Channel receiving 8 data bits with no parity

Tests the receiver for correct character reception of all characters from 00-0FF hex with no parity.

PARITY ERROR DURING RECEIVE

During a Receive (COM1) of characters with no parity, the Character Error flag (IFL bit 5) set.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits received. The data bits from left to right are data bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
1 if that bit is stuck high;
X if that bit is not stuck but was incorrect at one point;
- if that bit worked properly.

Test 14 - Test Receive Data with Redundancy

NOTE: For Dash 1 thru 5 only.

Tests that the Main Channel receives a character after the terminating character, and includes this character in the OCC generation. Also tests all 256 (00-0FF hex) possible OCC values are generated correctly.

PARITY ERROR DURING RECEIVE

During a Receive (COM1) of characters with known good parity, the Character Error flag (IFL bit 5) set.

RECEIVE DATA WITH REDUNDANCY DOES NOT GENERATE OCC

The character in the Main Channel receive buffer was not the expected OCC, but in actuality was the transmitted BCC character following the terminating character.

AAAA,AAAA STUCK BITS FOR OCC GENERATION

This stuck message is displayed if there is an error in any of the OCC bits generated. The bits from left to right are bits 7 through 0 respectively.

```
A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at one
      point;
    - if that bit worked properly.
```

Test 15 - Test Main Channel Halt

Tests that issuing a Halt (COM1) on the Main Channel while receiving will stop reception on the Main Channel, set the Main Not Busy flag (IFL bit 6) and Interrupt 6. Also tests that the Main Channel Halt will not affect the Secondary Channel.

HALT SETS SECONDARY NOT BUSY AND DOES NOT SET MAIN NOT BUSY

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM1) to the Main Channel incorrectly stopped reception on the Secondary and not the Main Channel.

HALT DOES NOT SET MAIN NOT BUSY AND SECONDARY NOT BUSY

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM1) to the Main Channel could not stop reception on the Main Channel.

HALT SETS MAIN NOT BUSY AND SECONDARY NOT BUSY

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM1) to the Main Channel incorrectly stopped reception on both channels.

HALT DOES NOT SET INTERRUPT PRIORITY LEVEL NO. 6

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM1) failed to set the interrupt.

HALT SETS INTERRUPT PRIORITY LEVEL NO. 7

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM1) incorrectly set the wrong interrupt.

Test 16 - Test Secondary Channel Halt

Tests that issuing a Halt (COM2) on the Secondary Channel while receiving will stop reception on the Secondary Channel, set the Secondary Not Busy flag (IFL bit 7), and Interrupt 7. Also tests that the Secondary Channel Halt will not affect the Main Channel.

HALT SETS MAIN NOT BUSY AND DOES NOT SET SECONDARY NOT BUSY

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM2) to the Secondary Channel incorrectly stopped reception on the Main but not the Secondary Channel.

HALT DOES NOT SET MAIN NOT BUSY AND SECONDARY NOT BUSY

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM2) to the Secondary channel could not stop reception on the Secondary Channel.

HALT SETS MAIN NOT BUSY AND SECONDARY NOT BUSY

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM2) to the Secondary Channel incorrectly stopped reception on both channels.

HALT DOES NOT SET INTERRUPT PRIORITY LEVEL NO. 7

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM2) failed to set the interrupt.

HALT SETS INTERRUPT PRIORITY LEVEL NO. 6

During a Receive (COM1) on the Main Channel and a Receive (COM2) on the Secondary Channel, issuing a Halt (COM2) incorrectly set the wrong interrupt.

Test 17 - Test Secondary Channel Receive capabilities

Tests that a Receive (COM2) resets the Secondary Channel Not Busy flag (IFL bit 7). Also tests that a Receive increments the Current address up to the Terminating character or address and then sets the Secondary Channel Not Busy flag.

RECEIVE DOES NOT RESET MAIN NOT BUSY

During a Receive (COM2) the Secondary Channel Not Busy flag (IFL bit 7) was set.

CURRENT ADDRESS DOES NOT STOP AT TERMINATING CHARACTER OR ADDRESS

During a Receive (COM2) the Secondary Channel Current address incremented past the Terminating character and address.

Test 18 - Test Secondary Channel receiving 7 data bits with odd parity

Tests the Secondary Channel for correct character reception of all characters from 00-07F hex with parity odd.

PARITY ERROR DURING RECEIVE

During a Receive (COM1) of characters with known good parity, the Character Error flag (IFL bit 5) set.

AAAA,AAAA STUCK BITS FOR 7 DATA BITS ODD PARITY

This stuck message is displayed if there is an error in any of the bits received. The leftmost bit reflects the status of the parity bit and the next 7 from left to right are bits 6 through 0 respectively.

A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at one point;
    - if that bit worked properly.

Test 19 - Test Secondary Channel receiving 8 data bits with no parity

Tests the Secondary Channel for correct character reception of all characters from 00-0FF hex with no parity.

PARITY ERROR DURING RECEIVE

During a Receive (COM1) of characters with no parity, the Character Error flag (IFL bit 5) set.

AAAA,AAAA STUCK BITS FOR 8 DATA BITS NO PARITY

This stuck message is displayed if there is an error in any of the bits received. The bits from left to right are data bits 7 through 0 respectively.

A = 0 if that bit is stuck low;
    1 if that bit is stuck high;
    X if that bit is not stuck but was incorrect at one point;
    - if that bit worked properly.

Test 20 - Test Attention Secondary Channel

Tests that all 256 Attention characters (00-0FF hex) terminate reception by the Secondary Channel. Also tests that the Secondary Channel receives characters overstoring them in the first location of the receive buffer.

ATTENTION CHARACTER X DOES NOT TERMINATE RECEPTION

Transmitting Attention character X (0D0-0FF hex) to the Secondary Channel failed to set the Secondary Not Busy flag (IFL bit 7).

ATTENTION CHARACTER X DOES NOT ENTER SECONDARY BUFFER

Attention character X (00-0FF hex) does set the Secondary Not Busy flag (IFL bit 7) but upon examining the first location of the Secondary Channel buffer the Attention character was not found.

Test 21 - Test Attention Receive Data

Tests that no characters enter the Main Channel receive buffer prior to the Secondary Channel receiving an Attention character, and that the first character to enter the Main Channel receive buffer is the character immediately following the Attention character. Also tests that reception of an Attention character sets Interrupt Priority level number 7. Also tests that a second Attention character restarts reception by the Main Channel.

CHARACTER X ENTERED BUFFER WHEN NO ATTENTION CHARACTER WAS SENT

With the Attention character set up to be 00 hex, transmitting a string of characters from 030 to 03F hex caused character X to enter the Main Channel receive buffer.

ATTENTION CHARACTER X ENTERS BUFFER

Reception by the Main Channel began with Attention character X (00-0FF hex) rather than the character immediately following it.

CHARACTERS RECEIVED BEFORE ATTENTION CHARACTER X

Attention character X (00-0FF hex) was not necessary to begin reception by the Main Channel, nor did Attention character X restart the receiver.

UNEXPECTED PATTERN IN CHARACTERS RECEIVED FOR ATTENTION CHARACTER X

For Attention character X the receive buffer contained a character that could not be processed. Either the character was received incorrectly or reception did not begin.

RECEPTION BEGINS X CHARACTERS AFTER ATTENTION CHARACTER Y

The first character in the Main Channel buffer should be the character that immediately followed the Attention character in the transmitted string. If the first received character was indeed part of the transmitted string but not the character immediately following the Attention character, its position in reference to the Attention character in the transmitted string is determined (X) along with the Attention character causing the fault.

INTERRUPT PRIORITY LEVEL NO. 7 NOT ISSUED BY RECEIVING ATTENTION CHARACTER X

Reception began properly on the Main Channel but Interrupt 7 was not issued by the reception of Attention character X on the Secondary Channel.

A SECOND ATTENTION CHARACTER DOES NOT RESTART RECEPTION

Upon completion of transmitting a string of characters containing two Attention characters, the first character in the Main Channel buffer was not the character immediately following the second Attention character.

Test 22 - Test Receive Data with Attention Recognition

Test that the Main Channel does begin reception prior to an Attention character being received by the Secondary Channel. Also, test that an Attention character restarts reception.

RECEPTION DOES NOT START WITHOUT AN ATTENTION CHARACTER

Transmitting a known string of characters not containing an Attention character did not cause the Main Channel to begin reception.

AN ATTENTION CHARACTER DOES NOT RESTART RECEPTION

After transmitting a known string of characters containing an Attention character, the first character in the Main Channel buffer was not the character immediately following the Attention character.

Test 23 - Test Receive Data with Attention Recognition and Redundancy

NOTE: For Dash 1 thru 5 only.

Test that the OCC is being generated beginning with the first character received, and that receiving an Attention character will restart the OCC generation.

ERROR IN OCC GENERATION

Transmitting a known string of characters without an Attention character did not generate the expected OCC value in the receive buffer.

ATTENTION CHARACTER DOES NOT RESTART OCC GENERATION

Transmitting a known string of characters containing an Attention character did not cause the OCC generation to be restarted with the character following the Attention character.

Test 24 - Test Attention Receive Data with Redundancy

NOTE: For Dash 1 thru 5 only.

Test that the OCC is being generated beginning with the first character received after the Attention character, and that receiving a second Attention character will restart the OCC generation.

ERROR IN OCC GENERATION

Transmitting a known string of characters with an Attention character did not generate the expected OCC value in the receive buffer.

A SECOND ATTENTION CHARACTER DOES NOT RESTART OCC GENERATION

Transmitting a known string of characters containing two Attention characters did not cause the OCC generation to be restarted with the character immediately following the second Attention character.

Test 25 - Test Auto Answer

NOTE: For Dash 1 and 6 only.

Tests that when a ring is detected or carrier is removed, Interrupt 7 will be generated along with either the Ring Detector flag (IFL bit 0) for ring indication or the Loss of Carrier flag (IFL bit 4) for carrier loss indication.

INTERRUPT GENERATED WITH CARRIER DETECTOR SET AND RING DETECTOR RESET

With carrier present and no ring indication, Interrupt 7 was generated, after issuing an Auto Answer (COM2).

RING INDICATOR DOES NOT GENERATE INTERRUPT PRIORITY LEVEL NO. 7

With carrier and ring indication present, Interrupt 7 was not generated after issuing an Auto Answer (COM2).

RESETTING CARRIER DETECTOR DOES NOT SET LOSS OF CARRIER

With carrier present and no ring indication, issuing an Auto Answer (COM2) then dropping carrier did not set the Loss of Carrier flag (IFL bit 4).

LOSS OF CARRIER DOES NOT GENERATE INTERRUPT PRIORITY LEVEL NO. 7

With carrier present and no ring indication, issuing an Auto Answer (COM2) then dropping carrier did not generate Interrupt 7.

TEST 26 - TEST DATA TERMINAL READY

NOTE: For Dash 4 and 9 only.

Tests that the Data Terminal Ready flag (OFL bit 1) can be properly set and reset.

DATA TERMINAL READY OR RING DETECTOR STUCK HIGH

The known good Ring Detector flag (IFL bit 0) could not be reset by issuing an INIT and an OFL with the Data Terminal Ready flag (OFL bit 1) reset.

DATA TERMINAL READY OR RING DETECTOR STUCK HIGH

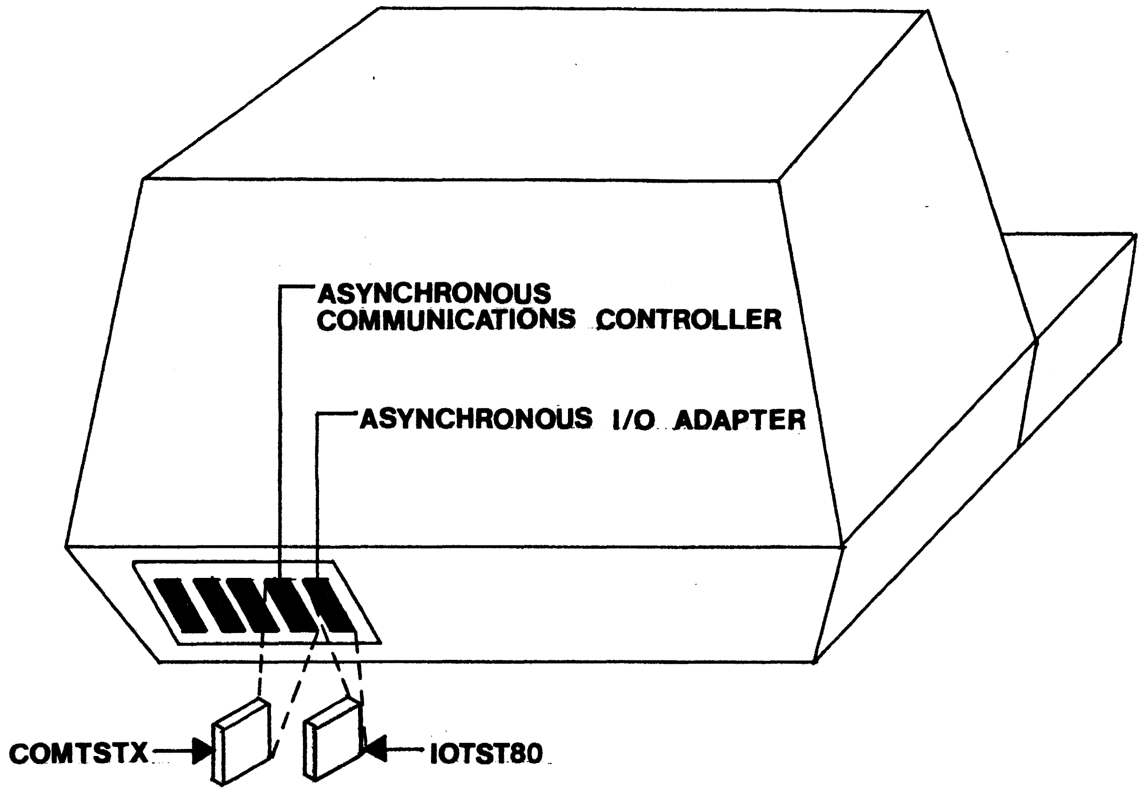
The known good Ring Detector flag (IFL bit 0) could not be set by issuing an OFL with the Data Terminal Ready flag (OFL bit 1) set.

DVCL RESETS DATA TERMINAL READY

Issuing a DVCL with the Data Terminal Ready flag (OFL bit 1) set, reset the known good Ring Detector flag (IFL bit 0).

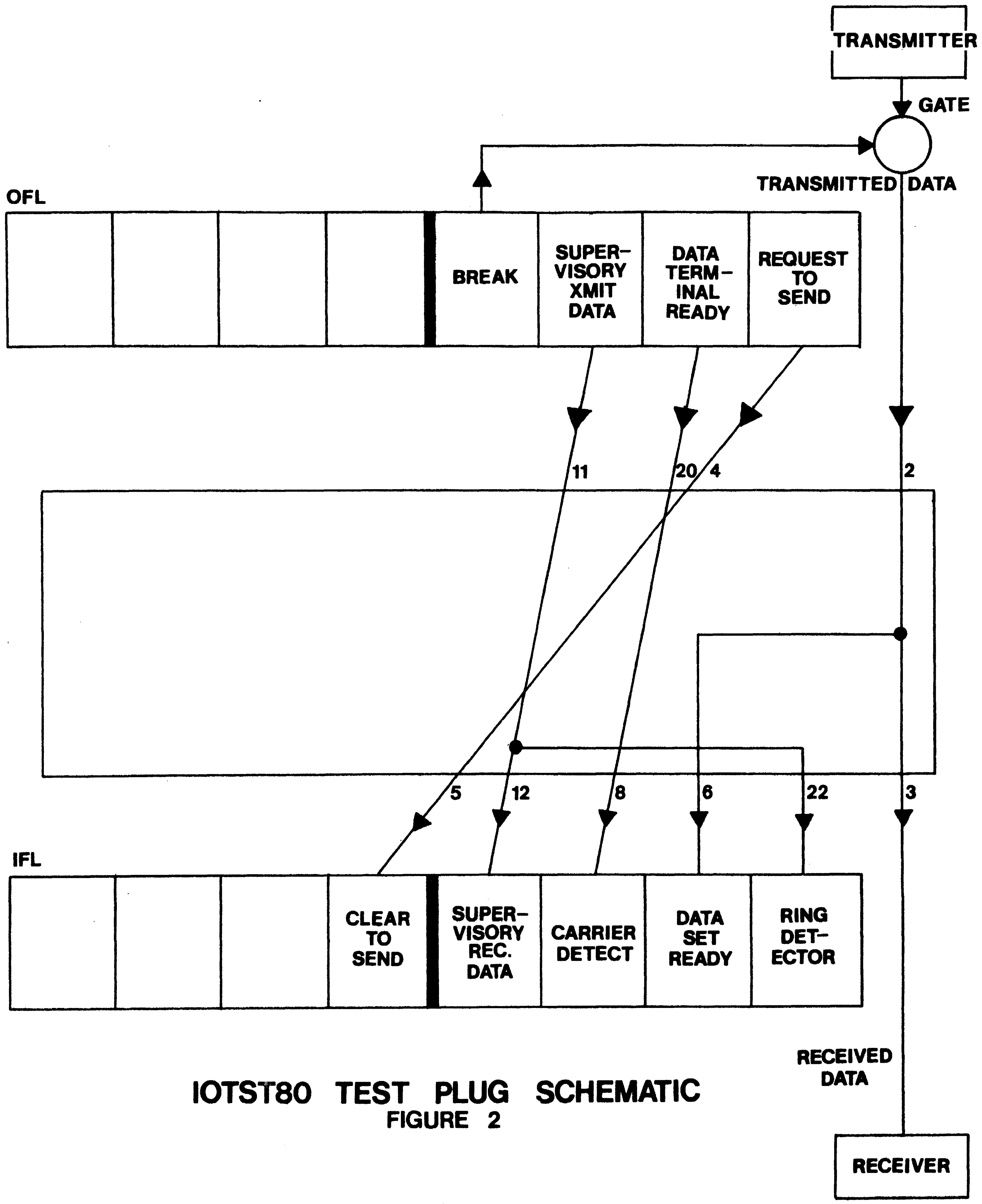
INIT DOES NOT RESET DATA TERMINAL READY

Issuing an INIT with the Data Terminal Ready flag (IFL bit 1) set, does not reset the known good Ring Detector flag (IFL bit 0).

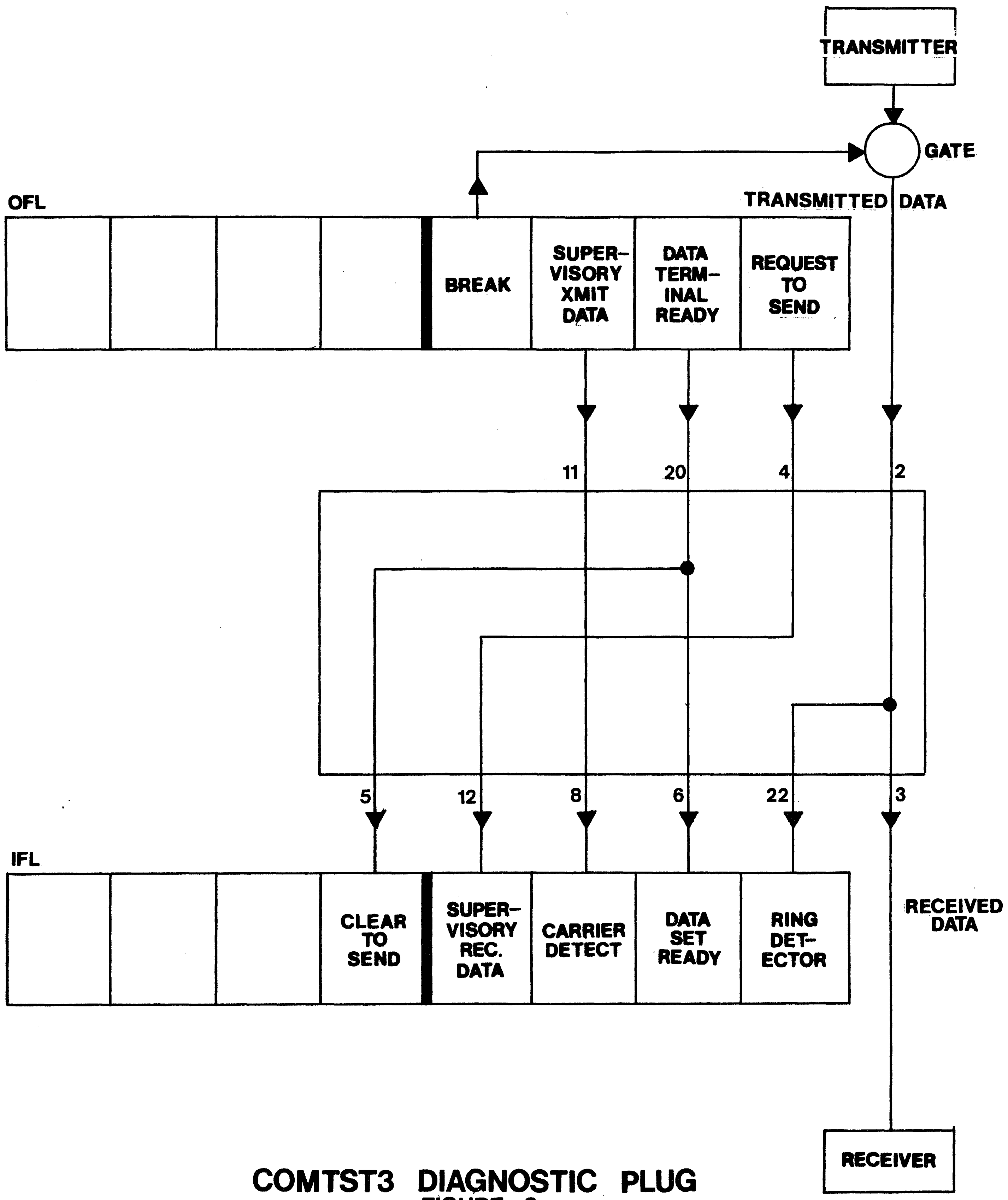


COMTSTX DIAGNOSTIC PLUG INSERTION GUIDE

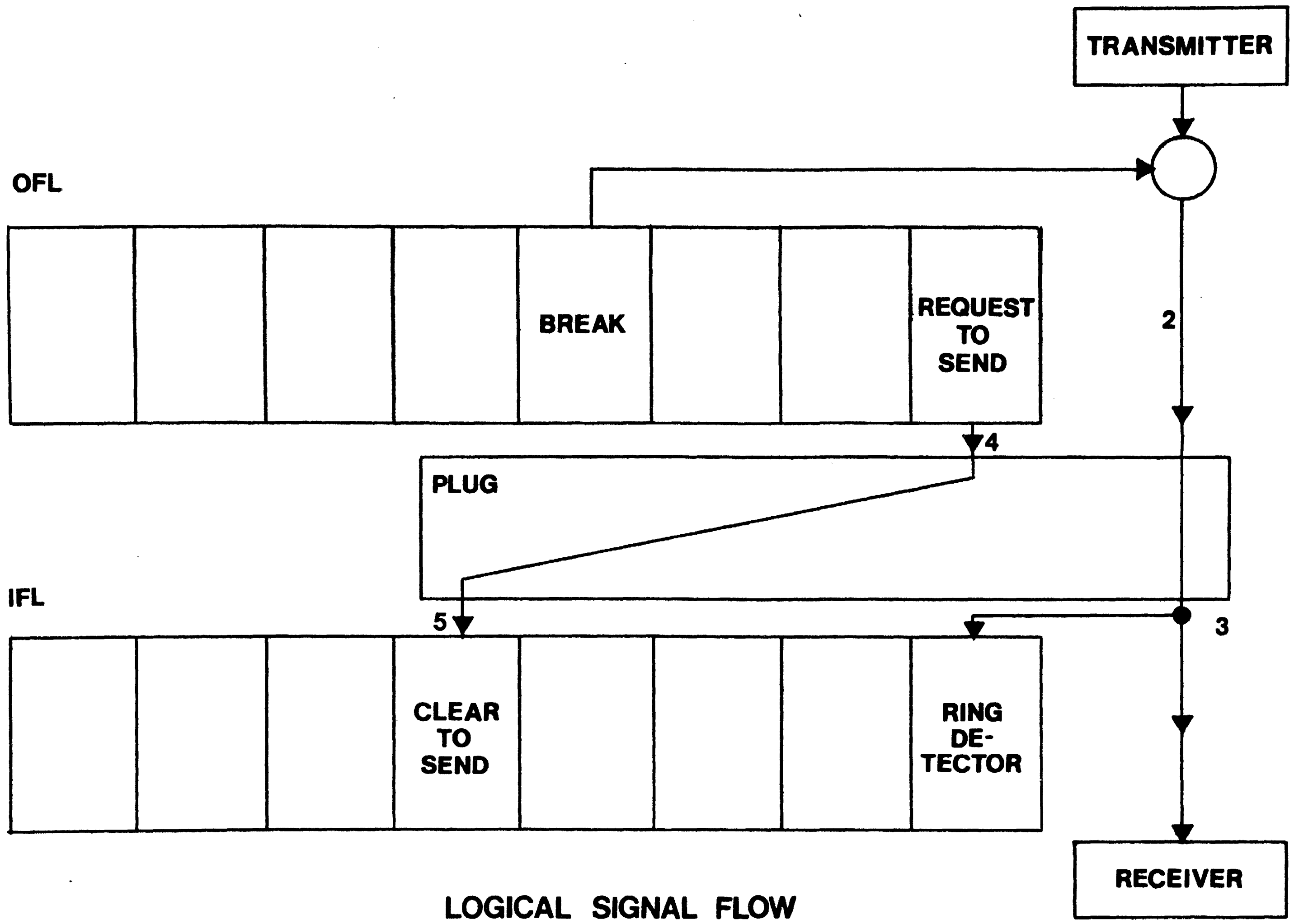
FIGURE 1



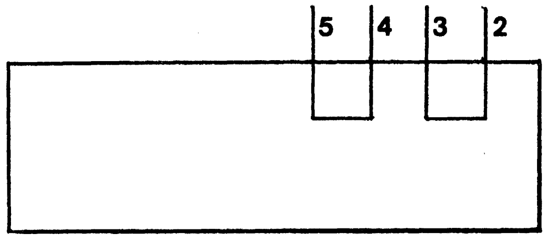
IOTST80 TEST PLUG SCHEMATIC
FIGURE 2



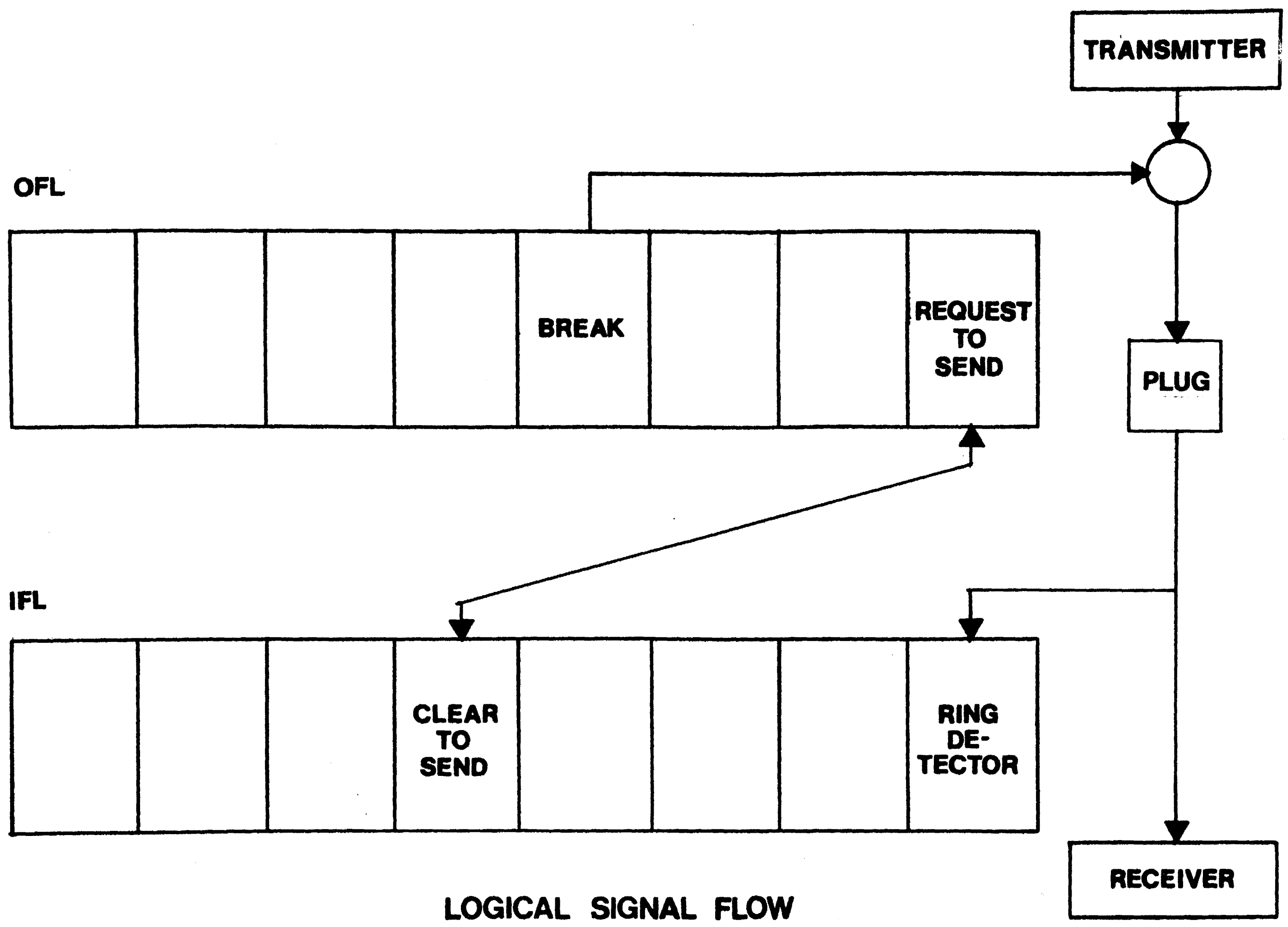
COMTST3 DIAGNOSTIC PLUG
FIGURE 3



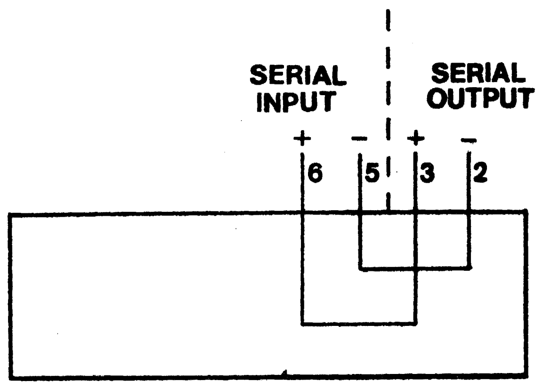
LOGICAL SIGNAL FLOW



COMTST4 DIAGNOSTIC PLUG
FIGURE 4

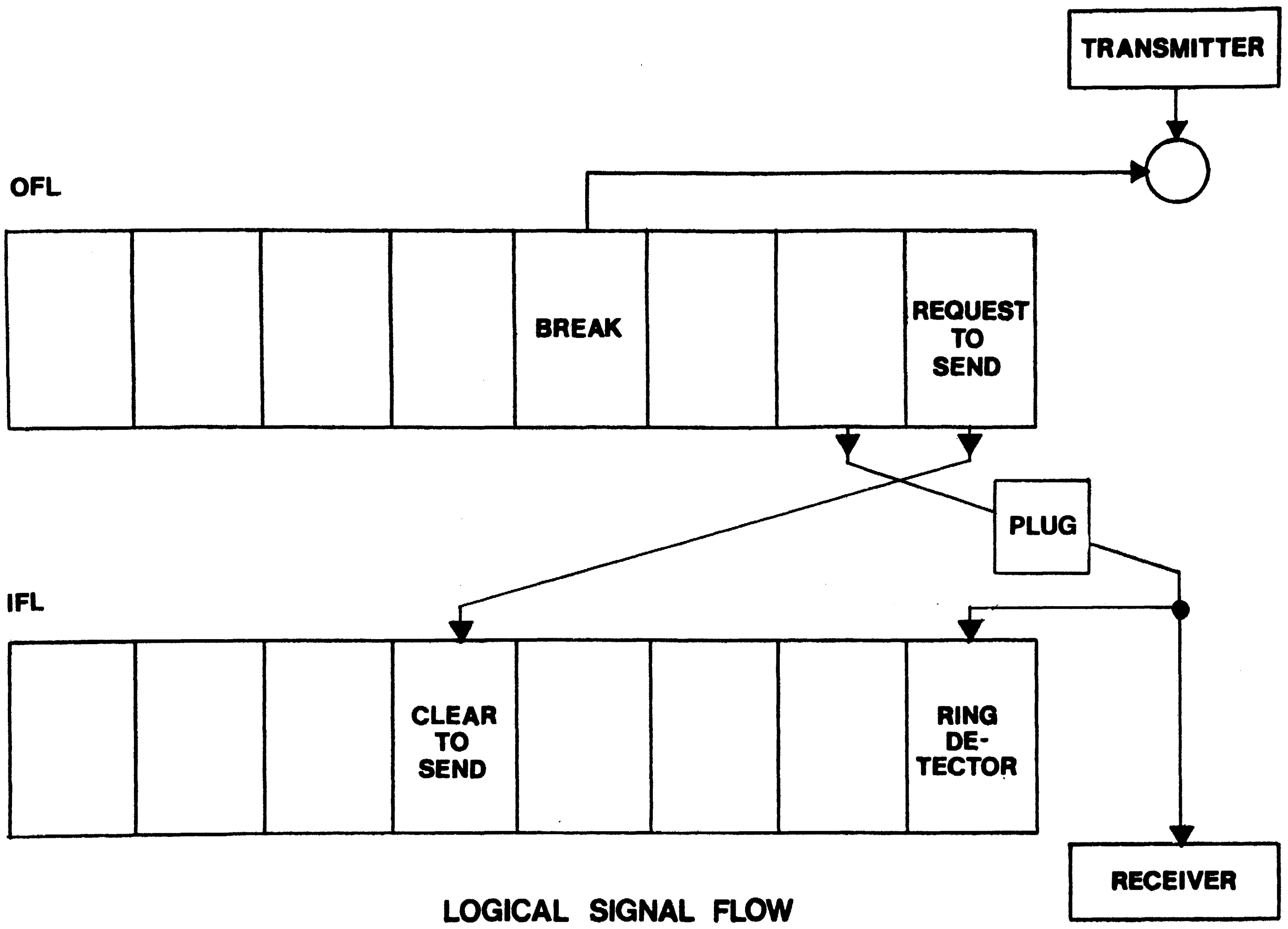


LOGICAL SIGNAL FLOW

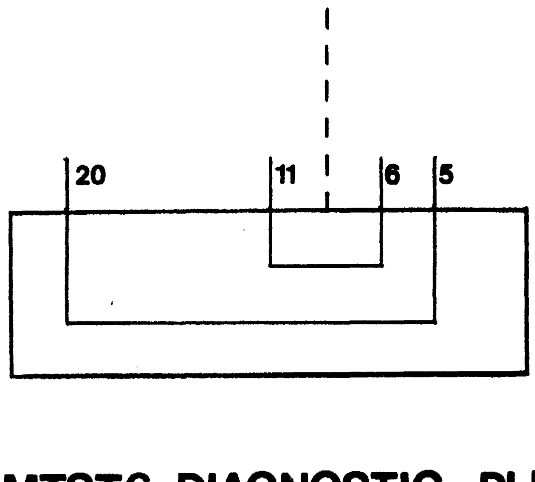


COMTST5 DIAGNOSTIC PLUG

FIGURE 5

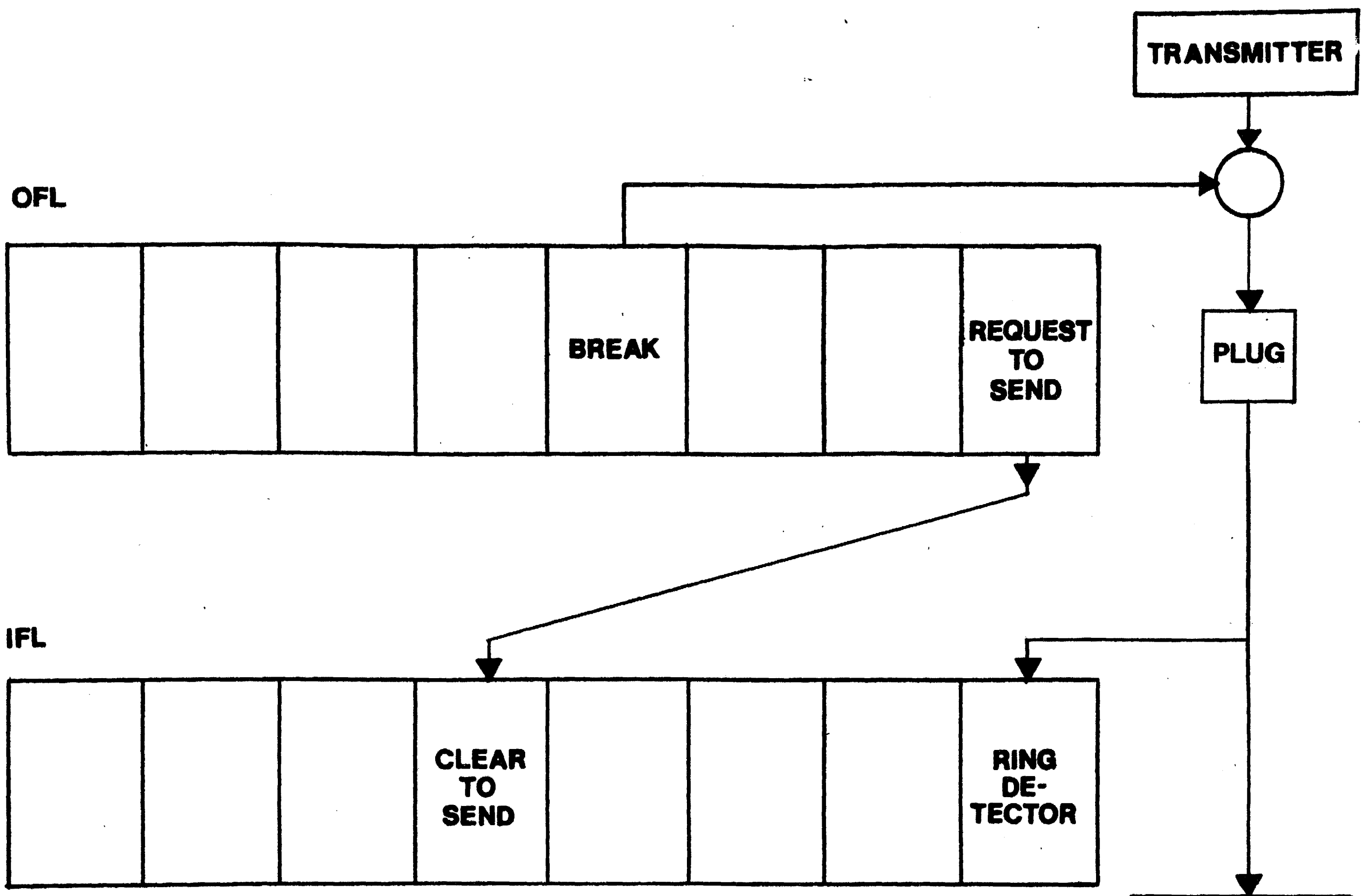


LOGICAL SIGNAL FLOW

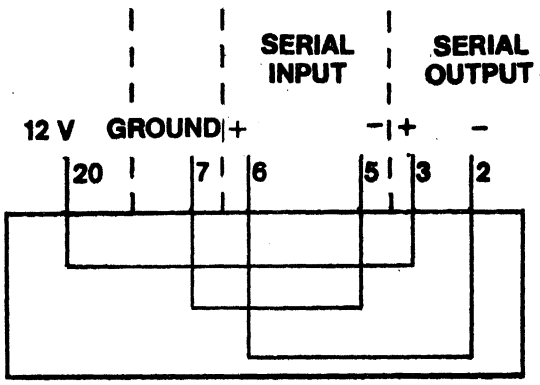


COMTST6 DIAGNOSTIC PLUG

FIGURE 6



LOGICAL SIGNAL FLOW



COMTST7 DIAGNOSTIC PLUG

FIGURE 7

IOMTST / WIOMTST

INPUT / OUTPUT MICROPROCESSOR TEST

IOMTST/WIOMTST - I/O MICROPROCESSOR TEST PROGRAM

Applicable Assemblies		Applicable Test
5000-1103-X	I/O Microprocessor	Both
5000-1127-X	Byte String Controller	IOMTST
5000-1170-X	Word Mover Controller	WIOMTST

General Description

The purpose of the IOMTST program is to determine if the I/O Microprocessor is working properly and, if not, to give an indication of which functions are defective. The program requires the use of from 1 to 4 known good Byte String Controllers to exercise the IOM. These controllers may be installed in device slots 7 through 10 in any configuration in order that all IOM vectoring may be tested. IOMTST works on an inverted pyramid testing structure, i.e., the more basic IOM functions are tested first, with each succeeding test being based on the assumption that the previous tests have been run successfully.

WIOMTST is identical to IOMTST described in the previous section, but with the following modifications:

1. The Word Mover Controller rather than the Byte String Controller is used to exercise the IOM.

2. For all data moves, the command issued is a COM3 (move without wraparound).

3. Because of internal hardware buffering, all tests of terminating conditions (Source and Destination) use the Destination Current Address to monitor IOM progress.

16K of RAM is required to run IOMTST and WIOMTST.

Loading Procedure

IOMTST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If IOMTST loads properly, it will identify itself and pause for optional test run instructions (refer to Appendix A).

Operator Action

After pausing to accept optional test run instructions (if any) the following message is displayed:

BYTE STRING CONTROLLERS (TYPE Y or N):

The operator enters Y (yes) or N (no) in a displayed table indicating which device controller slots contain Byte String Controllers for the current test run. In addition to the Y and N keys, the only other active key for entering data is the down arrow , which advances the cursor in the data table. When all the data has been entered, the operator is instructed to type the RETURN key, at which time the program will immediately begin executing Test 00.

IOMTST is a completely self-contained program. All tests are self-running. When the entire program has been completed, the prompt **"TYPE SPACE TO REPEAT IOMTST"** will be displayed on the screen.

Errors

All program detectable errors are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The bell will ring only to notify the operator of an error. The error message attempts to give a description of the nature of the problem. After an error message is displayed, the operator has four ways with which to proceed. Typing the SPACE bar continues the program to the next test, typing R will repeat the current test, typing the PROG key will restart the program and typing the PROG key shifted will exit to the operating system.

Test Description

All test operations are described in this section. The program will halt and the specified error message is displayed if expected results are not obtained.

Test 00 - TEST THAT ALL CONTROLLERS GENERATE INTERRUPTS

The existing Byte String Controllers are selected and cleared of activity and then checked to see that all generate interrupts. A data move (COM2, unconditional move) is then executed and the interrupts are enabled, testing that the controllers do not interrupt the CPU while busy.

INTERRUPT NOT GENERATED FOR CONTROLLER SLOT XX

An interrupt was not generated by the controller in the indicated slot number(s) when all controllers were not busy.

CONTROLLER INTERRUPTS WHEN NOT BUSY RESET SLOT XX

The controller in the indicated slot number(s) interrupted the CPU during a data move.

Test 01 - TEST OF SOURCE CURRENT ADDRESS

All hardware vectors are set up such that the Source Current address is 256 bytes before the Source Starting address. A data move (COM2, unconditional move) is executed for each existing controller, testing that the Source Current address is initialized at the Source Starting address when the move command is issued.

SOURCE CURRENT ADDRESS NOT INITIALIZED AT SOURCE STARTING ADDRESS, SLOT XX

The Source Current address for the indicated slot, examined immediately after the move command was issued, was found to be incorrectly initialized with respect to the Source Starting address.

Test 02 - TEST OF DESTINATION CURRENT ADDRESS

All hardware vectors are set up such that the Destination Current address is 256 bytes before the Destination Starting address. A data move (COM2, unconditional move) is executed for each existing controller, testing that the Destination Current address is initialized at the Destination Starting address when the move command is issued.

DESTINATION CURRENT ADDRESS NOT INITIALIZED AT DESTINATION STARTING ADDRESS, SLOT XX

The Destination Current address for the indicated slot, examined immediately after the move command was issued, was found to be incorrectly initialized with respect to the Destination Starting address.

Tests 03-06 - TEST OF COMBINED SOURCE TERMINATING CONDITIONS

The hardware vectors are set up such that a unique Source Terminating character occurs in the source buffer and coincides with the Source Terminating address. Bit 7 of the Source Terminating address (high) is reset to allow for termination on character or address. The destination buffer is made larger than the source buffer and Bit 7 of the Destination Terminating address (high) is set so that the move is not terminated by the destination conditions. A data move (COM2, unconditional move) is executed and the Source Current address is examined to see that it does, in fact, terminate correctly, i.e., that it terminates on the expected terminating address. Since the expected terminating character occurs at this same location, the test demonstrates, if run successfully, that a move can be terminated on at least one of the two terminating conditions (character or address). A separate test is run for each device controller slot. If no controller exists in a slot, the test is skipped.

SOURCE CURRENT ADDRESS DID NOT TERMINATE ON TERMINATING CHARACTER OR ADDRESS

The Source Current address was found to increment past the expected Source Terminating address.

MOVE WAS TERMINATED BEFORE TERMINATING CHARACTER AND ADDRESS

The Source Current address was found to have stopped incrementing at a location before the expected Source Terminating address.

<u>Tests 07-10 - TEST OF COMBINED DESTINATION TERMINATING CONDITIONS</u>

The hardware vectors are set up such that a unique Destination Terminating character occurs in the source buffer and coincides with the Destination Terminating address. Bit 7 of the Destination Terminating address (high) is reset to allow for termination on character or address. The source buffer is made larger than the destination buffer and Bit 7 of the Source Terminating address (high) is set so that the move is not terminated by the source conditions. A data move (COM2, unconditional move) is executed and the Destination Current address is examined to see that it does, in fact, terminate correctly, i.e., that it termiates on the expected terminating address. Since the expected terminating character occurs at this same location, the test demonstrates, if run successfully, that a move can be terminated on at least one of the two terminating conditions (character or address). In addition, the characters moved to the destination buffer are checked for integrity, i.e., that they are identical to the original characters in the source buffer. A separate test is run for each device controller slot. If no controller exists in a slot, the test is skipped.

DESTINATION CURRENT ADDRESS DID NOT TERMINATE ON TERMINATING CHARACTER OR ADDRESS

The Destination Current address was found to increment past the expected Destination Terminating address.

MOVE WAS TERMINATED BEFORE TERMINATING CHARACTER AND ADDRESS

The Destination Current address was found to have stopped incrementing at a location before the expected Destination Terminating address.

FALSE CHARACTER INTEGRITY IN DESTINATION BUFFER

A character was found in the destination buffer which did not match the actual character to be moved from the source buffer. (If a character matchs correctly, it is blanked out. The erroneous character therefore will be the first character visible in the destination buffer.)

<u>Tests 11-14 - TEST OF ALL SOURCE TERMINATING CHARACTERS</u>

The source buffer is set up such that the Source Terminating character appears before the Source Terminating address. Bit 7 of the Source Terminating address is made low to allow for termination on character or address. Bit 7 of the Destination Terminating address is set to allow for termination on address only. A separate data move (COM2, unconditional move) is executed with the Source Terminating character vector set for each of 256 possible ASCII characters. The Source Current address is examined to determine whether the move was, in fact, terminated on the proper character. (For each data move, the expected terminating character appears on the screen directly to the right of the word "SOURCE".) A separate test is run for each device controller slot. The slots not under test have a destination buffer of 5 bytes in length only. If no controller exists in a slot, the test is skipped.

MOVE TERMINATED BEFORE TERMINATING CHARACTER, ACTUAL: XX, EXPECTED: XX

The Source Current address stopped incrementing before the address of the expected Source Terminating character.

MOVE DID NOT TERMINATE ON TERMINATING CHARACTER, EXPECTED: XX

The Source Current address was found to increment past the address of the expected Source Terminating character.

Tests 15-18 - TEST OF ALL DESTINATION TERMINATING CHARACTERS

The source buffer is set up such that the Destination Terminating character will appear before the Destination Terminating address. Bit 7 of the Destination Terminating address is made low to allow for termination on character or address. Bit 7 of the Source Terminating address is set to allow for termination on address only. A separate data move (COM2, unconditional move) is executed with the Destination Terminating character vector set for each of 256 possible ASCII characters. The Destination Current address is examined to determine whether the move was in fact, terminated on the proper character. (For each data move, the expected terminating character appears on the screen directly to the right of the word "SOURCE".) A separate test is run for each device controller slot. The slots not under test have a destination buffer of 5 bytes in length only. If no controller exists in a slot, the test is skipped.

MOVE TERMINATED BEFORE TERMINATING CHARACTER, ACTUAL: XX, EXPECTED: XX

The Destination Current address stopped incrementing before the address of the expected Destination Terminating character.

MOVE DID NOT TERMINATE ON TERMINATING CHARACTER, EXPECTED: XX

The Destination Current address was found to increment past the address of the expected Destination Terminating chracter.

Tests 19-22 - TEST OF BIT 7, TERMINATING ADDRESS (HIGH)

The hardware vectors are set such that the Source Terminating character appears within the source buffer, and Bit 7 of the Source Terminating address (high) is set. A data move is executed and the Source Current address is examined to check that it ignores the Source Terminating character, i.e., that it increments past the address of the terminating character. The vectors are then reset so this procedure can check Bit 7 of the Destination Terminating address. A separate test is run for each device controller slot. If no controller exists in a slot, the test is skipped.

TERMINATING CHARACTER WAS NOT IGNORED WITH BIT 7 SET, SOURCE (or DESTINATION)

The Current address was found to terminate on the address of the terminating character when the terminating character should have been ignored.

Tests 23-26 - TEST OF ALL SOURCE TERMINATING ADDRESS BITS

Bit 7 of the Source Terminating address (high) is set to allow termination on address only, and then 15 separate data moves are executed, each testing a different address bit. The Source Current address is examined to check that it does, in fact, terminate on the proper terminating address. The first 8 least significant address bits are configured as follows:

```
1XXXXXXX        00000001
1XXXXXXX        00000010
1XXXXXXX        00000100   etc.
```

The next 7 bits are:

```
10000001        XXXXXXXX
10000010        XXXXXXXX
10000100        XXXXXXXX   etc.
```

A separate test is run for each device controller slot. The slots not under test have a destination buffer of 5 bytes in length only. If no controller exists in a slot, the test is skipped.

MOVE DID NOT TERMINATE ON TERMINATING ADDRESS, EXPECTED: XXXX

The Source Current address was found to increment past the expected Source Terminating address.

MOVE WAS TERMINATED BEFORE TERMINATING ADDRESS, EXPECTED: XXXX, ACTUAL: XXXX

The Source Current address was found to terminate before the expected Source Terminating address.

Tests 27-30 - TEST OF DESTINATION TERMINATING ADDRESS

Bit 7 of the Destination Terminating address is set to allow termination on address only, and a single data move (COM2, unconditional move) is executed, set for termination on an arbitrary address. Only one address need be tested because the destination addresses are checked by the same comparator as the source addresses, which were tested in Tests 23-26. The Destination Current address is examined to check that it does, in fact, terminate on the proper address. A separate test is run for each device controller slot. The slots not under test have a destination buffer of 5 bytes in length only. If no controller exists in a slot, the test is skipped.

MOVE DID NOT TERMINATE ON TERMINATING ADDRESS, EXPECTED: XXXX

The Destination Current address was found to increment past the expected Destination Terminating address.

MOVE WAS TERMINATED BEFORE TERMINATING ADDRESS, EXPECTED: XXXX, ACTUAL: XXXX

The Destination Current address was found to stop incrementing before the expected Destination Terminating address.

DSKTST

DISK SYSTEM TEST

DSKTST - DISK SYSTEM TEST

Applicable Assemblies

5000-1119	Disk Controller in slot 8 of each terminal 20K of RAM per terminal 8080 terminal(s) Any Ontel File Controller and Disk Drive configuration

General Description

The purpose of the DSKTST program is to determine if all equipment in a Disk System (Disk Controllers, File Controller and Drives) are working properly and, if not, to give an indication of which functions are incorrect.

20K (or more) of memory is required to run DSKTST.

This manual applies only to the 8080 version of DSKTST.

Loading Procedure

DSKTST can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If DSKTST loads properly, it will immediately identify itself and pause for optional test run instructions (refer to Appendix A).

Operator Action

After pausing to accept optional test run instructions, DSKTST asks if there is a Synchronous or Asynchronous adapter. At this time type (A) for Asynchronous or (S) for Synchronous. After this DSKTST will display a menu of test parameters and their default values. DSKTST then waits for the operator to make any parameter changes or enter the parameters currently displayed. The operator makes a change by moving the cursor to the appropriate parameter field and typing in the desired parameter. The cursor will respond to any of the arrow keys (the 4 nearest neighbors surrounding the Home key). After all parameter changes, if any, have been made, the operator types, "C" (hex 063) to enter the parameters. The diagnostic tests will begin execution if all entered parameters are legitimate. If an illegal parameter is entered, the cursor will be placed over the parameter in question and DSKTST will wait for the operator to re-enter the parameter before beginning the tests. If "PROG" is depressed during the execution of any test the program will restart. That is, DSKTST will pause to accept optional test run instructions and then display the parameter menu. However, the menu will show the parameter values previously entered, not the original default values.

Parameter Description

"*SURFACE (U-UPPER, L-LOWER, B-BOTH)" - Specifies which surfaces are to be exercised. If "B" is entered, the upper surface is exercised first.

"*AUTO STOP (Y or N)" - When entered as Y, DSKTST returns control to the operating system after completing 50 loops in Test 10.

"*VERIFY FORMAT (Y or N)" - A utility to check that all entered tracks have been formatted properly. The format is checked by reading the entire track and:

a. Comparing the first byte of every sector with the format code byte for the track.

b. Comparing the first 20 bytes following the code byte and the last 20 bytes of each sector with 00 hex.

The first byte not conforming to conditions a and b is marked with a spinning character. If an error is detected the program will unconditionally stop and wait for an operator instruction.

This utility and the Format utility are completely independent. They can be run together or alone. Upon their completion the operator must type SPACE to start the exerciser portion of Test 10.

"*STOP ON ERROR (Y or N)" - refers only to the disk exerciser portion of Test 10, not the Format and Verify utilities. "Y" means stop unconditionally on any error and wait for instructions. "N" means stop only on non-recoverable system errors (any command returning with an IFL not equal to 080 hex eight successive times). An IFL error will cause a command retry up to eight succesive times before halting the program. Counters for all errors are displayed on the screen. An asterisk (*) beneath a counter marks the most recent errors. If "Y" was entered the current IFL status is always displayed. IF "N" was entered either 080 hex or the most recent IFL error is displayed.

"*SECTOR SIZE 1-256, 2-512 BYTE BLOCK)" - specifies the number of bytes per sector.

"*DISK (0-3)" - specifies which disks are to be exercised. On a two platter drive, such as the Diablo, 0 corresponds to the removable disk and 3 corresponds to the fixed disk.

"*DRIVES TO BE TESTED (0-3)" - specifies which drives are to be exercised. Drives are exercised in ascending numerical order. Entering a drive more than once will not increase the number of times it is exercised. DSKTST will eliminate any redundancies in the drive field.

"*NUMBER OF SECTORS (1-24)" - all sectors from 0 to n-1 will be tested.

"*ASSIGN UNIT NUMBER (0-3)" - this terminal identification byte will be recorded on every sector tested.

"*FORMAT TRACKS (Y or N)" - a utility enabling the operator with a virgin or destroyed surface to run the diagnostic. Immediately preceeding Test 10, Initialize and Check track commands are executed on every entered track. If the verify format parameter was entered as "Y", both utilities are run concurrently. That is, following the Check track command, a Read command is executed and the formatted track is verified. Any command causing an IFL error unconditionally stops the program. The stop on error parameter has no effect here, the operator must always issue an instruction to restart the utility.

"*SELECT ADDRESS OF DISK CONTROLLER" - specifies the hardware select address of the Disk Controller to be tested. Any hex number may be entered, however, no more than one controller should be selected or DSKTST will give erroneous results.

"*SLOT NUMBER" - specifies the physical slot location of the Disk Controller to be tested.

"*TRACKS TO BE TESTED (MIN-0, MAX-407)" - specifies the tracks to be exercised. A space, comma or multiples of are valid delimiters between single tracks. A single hyphen may be used to indicate a group of tracks. The hyphen may not be used in conjunction with spaces or commas.

Errors

All errors in Tests 00-09 are indicated by an appropriate error message on the display screen and the simultaneous activation of the bell. The bell will ring only to notify the operator of an error. The error message displayed on the screen attempts to give a description of the nature of the problem. In some cases, this should be adequate to diagnose and fix the error. Otherwise, refer to the detailed description of the specific test to determine the purpose and expected results for the displlayed error message.

After an error message is displayed the operator has four ways to proceed. Typing the SPACE bar will continue testing on the next test, typing the R key (without the SHIFT key) will repeat the current test, depressing the PROG key will restart the program and typing SHIFT PROG will return control to the operating system.

Because of the inverted pyramid test stragegy used in DSKTST, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

Test Description

All test operations are described in this section. The program will halt after displaying an error message if expected results are not obtained.

On the following pages, each test is listed with a brief description of what it is testing. Below the description, all possible error messages are listed, with an explanation of the cause of the message.

TEST 00 - CONTROLLER SELECT TEST

Tests that the Disk Controller does not respond to an incorrect select address, does respond to the correct address, and INIT de-selects a selected Disk Controller. The correct device select address is 087 (hex). The incorrect select addresses are those single byte addresses which have 3 or less bits high. This is derived from the fact that a 4 input AND gate performs a device selection from the address bits of the select address. Thus a select address of 0FF hex would attempt to select all devices.

CONTROLLER SELECTED WITH INCORRECT ADDRESS (XX)

IFL to the select address XX hex gave a result other than 0FF hex (open bus).

CONTROLLER IS NOT SELECTED WITH CORRECT ADDRESS (087)

IFL to the correct Disk Controller address (087 hex) gave a result of 0FF hex (open bus).

INIT DOES NOT DESELECT CONTROLLER

After an INIT to the selected Disk Controller, IFL did not get a result of 0FF hex (open bus).

TEST 01 - TESTING THE NOT READY FLAG

Tests that the Not Ready flag is not stuck high or low. That DVCL and INIT reset the Not Ready flag.

NOT READY FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Not Ready flag was examined and found to be low.

NOT READY FLAG STUCK LOW

This test assumes that less than 4 drives are connected to the File Controller. Restore commands are executed on all drives. If none of the Restore commands set the Not Ready flag, the error message is displayed.

INIT DOES NOT CLEAR NOT READY FLAG

This test assumes that less than 4 drives are connected to the File Controller. Following the execution of a Restore command on a drive not connected to the File Controller, the Not Ready flag is verified to be set. An INIT command is executed and the Disk Controller is re-selected. If the Not Ready flag is still found to be high, the error message is displayed.

DVCL DOES NOT CLEAR NOT READY FLAG

This test assumes that less than 4 drives are connected to the File Controller. Following the execution of a Restore command on a drive not connected to the file Controller the Not Ready flag is verified to be set. A DVCL command is executed. If the Not Ready flag is still found to be high, the error message is displayed.

<u>TEST 02 - TESTING THE NOT BUSY FLAG (IFL BIT 7)</u>

Tests that all Disk I/O commands reset the Not Busy flag, that the Not Busy flag sets after the command has been completed, and that DVCL or INIT will immediately abort the execution of an command and set the Not Busy flag.

NOT BUSY FLAG STUCK LOW

After INIT and DVCL instructions were executed, the Not Busy flag was examined and found to be low.

NOT BUSY REMAINED HIGH DURING A RESTORE OPERATION

Commands to begin a Restore were executed. While the Restore was in progress, the Not Busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER A RESTORE OPERATION

Commands to begin a Restore were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING A RESTORE OPERATION

Commands to begin a Restore were executed. While the Restore was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the Restore. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING A RESTORE OPERATION

Commands to begin a Restore were executed. While the Restore was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the Restore. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

NOT BUSY REMAINED HIGH DURING A CHECK TRACK OPERATION

Commands to begin a Check TRACK were executed. While the Check TRACK was in progress, the Not Busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER A CHECK TRACK OPERATION

Commands to begin a Check TRACK were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING A CHECK TRACK OPERATION

Commands to begin a Check TRACK were executed. While the Check TRACK was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the Check TRACK. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING A CHECK TRACK OPERATION

Commands to begin a Check TRACK were executed. While the Check TRACK was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the Check TRACK. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

NOT BUSY REMAINED HIGH DURING AN UPDATE READ OPERATION

Commands to begin an UPDATE READ were executed. While the UPDATE READ was in progress, the Not Busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER AN UPDATE READ OPERATION

Commands to begin an UPDATE READ were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING AN UPDATE READ OPERATION

Commands to begin an UPDATE READ were executed. While the UPDATE READ was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the UPDATE READ. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING AN UPDATE READ OPERATION

Commands to begin an UPDATE READ were executed. While the UPDATE READ was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the UPDATE READ. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

NOT BUSY REMAINED HIGH DURING AN UPDATE WRITE OPERATION

commands to begin an UPDATE WRITE were executed. While the UPDATE WRITE was in progress, the Not busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER AN UPDATE WRITE OPERATION

Commands to begin an UPDATE WRITE were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING AN UPDATE WRITE OPERATION

Commands to begin an UPDATE WRITE were executed. While the UPDATE WRITE was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the UPDATE WRITE. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING AN UPDATE WRITE OPERATION

Command to begin an UPDATE WRITE were executed. While the UPDATE WRITE was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the UPDATE WRITE. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

NOT BUSY REMAINED HIGH DURING AN UPDATE CHECK OPERATION

Commands to begin an UPDATE CHECK were executed. While the UPDATE CHECK was in progress, the Not Busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER AN UPDATE CHECK OPERATION

Commands to begin an UPDATE CHECK were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING AN UPDATE CHECK OPERATION

Commands to begin an UPDATE CHECK were executed. While the UPDATE CHECK was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the UPDATE CHECK. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING AN UPDATE CHECK OPERATION

Commands to begin an UPDATE CHECK were executed. While the UPDATE CHECK was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the UPDATE CHECK. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

NOT BUSY REMAINED HIGH DURING A READ OPERATION

Commands to begin a READ were executed. While the READ was in progress, the Not Busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER A READ OPERATION

Commands to begin a READ were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING A READ OPERATION

Commands to begin a READ were executed. While the READ was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the READ. After attempting to abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING A READ OPERATION

Command to begin a READ were executed. While the READ was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the READ. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

NOT BUSY REMAINED HIGH DURING A WRITE OPERATION

Commands to begin a WRITE were executed. While the WRITE was in progress, the Not Busy flag was examined and found to be high.

NOT BUSY REMAINED LOW AFTER A WRITE OPERATION

Commands to begin a WRITE were executed. The program continually examined the Not Busy flag for a period of two seconds during which the flag was always low.

DVCL DOES NOT IMMEDIATELY SET NOT BUSY DURING A WRITE OPERATION

Commands to begin a WRITE were executed. While the WRITE was in progress, the Not Busy flag was examined and verified to be low. A DVCL was executed to abort the WRITE . After attempting the abort, the program examined the Not Busy flag and found it to still be low.

INIT DOES NOT IMMEDIATELY SET NOT BUSY DURING A WRITE OPERATION

Commands to begin a WRITE were executed. While the WRITE was in progress the Not Busy flag was examined and verified to be low. An INIT was executed to abort the WRITE. After attempting the abort, the program examined the Not Busy flag and found it to still be low.

<u>TEST</u> <u>03</u> - <u>TESTING THE WRITE PROTECT FLAG</u>

Tests that the Write Protect flag (IFL bit 0) is not stuck high or low. That no Disk I/O commands sets the Write Protect flag when the drive is unprotected. That a Write commands sets the Write Protect flag when the drive is protected.

WRITE PROTECT FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Write Protect flag was examined and found to be high.

WRITE PROTECT FLAG STUCK LOW

The execution of a Write command with the drive write protected failed to set the Write Protect flag.

DVCL DOES NOT CLEAR WRITE PROTECT FLAG

After the Write Protect flag was verified to be high, the program executed a DVCL. The Write Protect flag was re-examined and found to still be high.

INIT DOES NOT CLEAR WRTE PROTECT FLAG

After the Write Protect flag was verified to be high, the program executed an INIT and re-selected the Disk Controller. The Write Protect flag was re-examined and found to still be high.

WRITE PROTECT FLAG HIGH AFTER READ OPERATION

After executing a Read command, the Write Protect flag was found to be high.

WRITE PROTECT FLAG HIGH AFTER UPDATE READ OPERATION

After executing an Update Read operation, the Write Protect flag was found to be high.

WRITE PROTECT FLAG HIGH AFTER UPDATE CHECK OPERATION

After executing an Update Check operation, the Write Protect flag was found to be high.

WRITE PROTECT FLAG HIGH AFTER CHECK OPERATION

After executing a Check operation, the Write Protect flag was found to be high.

WRITE PROTECT FLAG HIGH AFTER RESTORE OPERATION

After executing a Restore operation, the Write Protect flag was found to be high.

WRITE PROTECT FLAG LOW AFTER FORMAT OPERATION

The drive is write protected and the Write Protect flag is verified to be low. After an Initialize Track (format) command is executed, the Write Protect flag is found to be low.

WRITE PROTECT HAS NOT BEEN TURNED OFF

The operator instruction, "PLEASE TURN OFF WRITE PROTECT. TYPE SPACE TO CONTINUE ", is displayed on the screen. After the space is depressed the program executes a Write command. The Write Protect flag is then examined and found to be high.

<u>TEST</u> <u>04</u> - <u>TESTING</u> <u>THE</u> <u>FORMAT</u> <u>ALLOW</u> <u>SWITCH</u>

Tests that an Initialize Track command sets the Write Protect flag (IFL bit 0) when the Format Allow switch is turned off. Tests that an Initialize Track command does not set the Write Protect flag when the Format Allow switch is turned on.

FORMAT ALLOW SWITCH FAILED TO TURN OFF

The operator instruction, "PLEASE TURN OFF FORMAT ALLOW SWITCH. TYPE SPACE TO CONTINUE", is displayed on the screen. After space is depressed, the program executes an Initialize Track command. The Write Protect flag is then examined and found to be low.

FORMAT ALLOW SWITCH FAILED TO TURN ON

The operator instruction, "PLEASE TURN ON FORMAT ALLOW SWITCH. TYPE SPACE TO CONTINUE", is displayed on the screen. After the space is depressed, the program executes an Initialize Track command. The Write Protect flag is then examined and found to be high.

<u>TEST</u> <u>05</u> - <u>TESTING</u> <u>READ</u> <u>ERROR</u> <u>FLAG</u>

Tests that the Read Error flag (IFL bit 5) is not stuck high or low. That INIT and DVCL reset the Read Error flag. That Update Check, Update Read, Check and Read operations on a formatted sector set the Read Error flag. That all other operations reset the Read Error flag.

READ ERROR FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Read Error flag was examined and found to be high.

READ ERROR FLAG STUCK LOW

After reading a sector of a formatted track the Read Error flag failed to go high.

READ ERROR FLAG HIGH AFTER DVCL

The Read Error flag was forced high by reading a sector of a formatted track, following the Read, a DVCL was executed which failed to clear the Read Error flag.

READ ERROR FLAG HIGH AFTER INIT

The Read Error flag was forced high by reading a sector of a formatted track. Following the Read, an INIT was executed which failed to clear the Read Error flag.

UPDATE CHECK FAILED TO SET READ ERROR FLAG

After executing an Update Check on a sector of a formatted track, the Read Error flag was found to be low.

UPDATE WRITE FAILED TO CLEAR READ ERROR FLAG

The program reads a sector of a formatted track and verifies that the Read Error flag is high. After writing back the sector with an Update Write the Read Error flag was examined and found to still be high.

RESTORE FAILED TO CLEAR READ ERROR FLAG

The program reads a sector of a formatted track and verifies that the Read Error flag is high. After performing a Restore, the Read Error flag was examined and found to still be high.

CHECK FAILED TO SET READ ERROR FLAG

After executing a Check on a sector of a formatted track, the Read Error flag was found to be low.

FORMAT FAILED TO CLEAR READ ERROR FLAG

The program reads a sector of a formatted track and verifies that the Read Error flag is high. After performing an Initialize Track (format), the Read Error flag was examined and found to still be high.

WRITE FAILED TO CLEAR READ ERROR FLAG

The program reads a sector of a formatted track and verifies that the Read Error flag is high. After writing back the sector, the Read Error flag was examined and found to still be high.

READING A GOOD SECTOR SET THE READ ERROR FLAG

A Read is performed ten times on a known good sector. If the Read Error flag is high after any of the Read operations, the error message is displayed.

TEST 06 - TESTING ADDRESS AND SECTOR ERROR FLAGS

Tests that the Address and Sector Error flags (IFL bits 4 and 6 respectively) are not stuck high, are set by a Check operation performed on a track greater than 407, and are reset by DVCL and INIT instructions.

ADDRESS ERROR FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Address Error flag was examined and found to be high.

SECTOR ERROR FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Sector Error flag was examined and found to be high.

ADDRESS ERROR AND SECTOR ERROR FLAGS STUCK HIGH

After INIT and DVCL instructions were executed, the Sector Error flag and the Address Error flag were examined and found to be high.

ADDRESS ERROR AND SECTOR ERROR FLAGS STUCK LOW

Both the Address Error and Sector Error flags remained low after commands to perform a Check operation on a track greater than 407 were executed.

ADDRESS ERROR FLAG STUCK LOW

The Address Error flag remained low after commands to perform a Check operation on a track greater than 407 were executed.

SECTOR ERROR FLAG STUCK LOW

The Sector Error flag remained low after commands to perform a Check operation on a track greater than 407 were executed.

ADDRESS ERROR AND SECTOR ERROR FLAGS HIGH AFTER DVCL

The Address Error and Sector Error flags were forced high by a Check on an imaginary track. Following a Check, a DVCL was executed which failed to clear either flag.

SECTOR ERROR FLAG HIGH AFTER DVCL

The Sector Error flag was forced high by a Check on an imaginary track. Following the Check, a DVCL was executed which failed to clear the Sector Error flag.

ADDRESS ERROR FLAG HIGH AFTER DVCL

The Address Error flag was forced High by a Check on an imaginary track. Following the Check, a DVCL was executed which failed to clear the Address Error flag.

ADDRESS ERROR AND SECTOR ERROR FLAGS HIGH AFTER INIT

The Address Error and Sector Error flags were forced high by a Check on an imaginary track. Following the Check, a INIT was executed which failed to clear either flag.

SECTOR ERROR FLAG HIGH AFTER INIT

The Sector Error flag was forced high by a Check on an imaginary track. Following the Check, a INIT was executed which failed to clear the Sector Error flag.

ADDRESS ERROR FLAG HIGH AFTER INIT

The Address Error flag was forced High by a Check on an imaginary track. Following the Check, a INIT was executed which failed to clear the Address Error flag.

ADDRESS ERROR AND SECTOR ERROR FLAGS HIGH AFTER RESTORE

The Address Error and Sector Error flags were forced high by a Check on an imaginary track. Following the Check, a Restore was executed which failed to clear either flag.

SECTOR ERROR FLAG HIGH AFTER RESTORE

The Sector Error flag was forced high by a Check on an imaginary track. Following the Check, a Restore was executed which failed to clear the Sector Error flag.

ADDRESS ERROR FLAG HIGH AFTER RESTORE

The Address Error flag was forced High by a Check on an imaginary track. Following the check, a Restore was executed which failed to clear the Address Error flag.

TEST 07 - TESTING UPDATE TIME OUT FLAG

Tests that the Update Time Out flag (IFL Bit 3) is not stuck high, that the timer has a period of 2 seconds ± 10%, that DVCL clears the Update Time Out flag and unlocks the File Controller.

UPDATE TIME OUT FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Update Time Out flag was examined and found to be high.

UPDATE TIME OUT FLAG STUCK LOW

An Update Read operation was executed. No commands were sent to the File Controller for a period of 5.4 seconds. Following this period, commands for another Update Read were executed. After the second Update Read, the Update Time Out flag was examined and found to be low.

DVCL DOES NOT CLEAR UPDATE TIME OUT FLAG

After the Update Time Out flag was set, a DVCL was executed. Following the DVCL, the Update Time Out flag was examined and found to be high.

DVCL DOES NOT CLEAR UPDATE LOCK OUT

After the File Controller was locked out, that is any operation is aborted and sets the Update Time Out flag, a DVCL was executed to unlock the File Controller. Following the DVCL an Update Read was executed and found to set the Update Time Out flag indicating that the File Controller is still locked out.

UPDATE TIME OUT TOO SHORT

After a successful Update Read operation has been executed, no commands were sent to the File Controller for a period of 1.8 seconds. Following this period, a second Update Read was executed and found to set the Update Time Out flag.

UPDATE TIME OUT TOO LONG

After a successful Update Read operation has been executed, no commands were sent to the File Controller for a period of 2.2 seconds. Following this period, a second Update Read was executed and found not to set the Update Time Out flag.

TEST 08 - TESTING ACTIVITY TIME OUT FLAG (IFL BIT 2)

Tests that the Activity Time Out flag is not stuck high, and that the timer has a period of 16 seconds $\pm$ 10%.

ACTIVITY TIME OUT FLAG STUCK HIGH

After INIT and DVCL instructions were executed, the Activity Time Out flag was examined and found to be high.

ACTIVITY TIME OUT FLAG STUCK LOW

Commands to start a Read operation are executed. While the Read is in progress commands to execute all possible operations simultaneously are executed (lower 4 bits in DDRV are high). This should start the activity timer. The program waits up to 20 seconds for the timer to time out and set the Activity Time Out flag. If after 20 seconds the flag is not set, the error message is displayed.

ACTIVITY TIME OUT TOO SHORT

The time has a period less than 14.3 seconds.

ACTIVITY TIME OUT TOO LONG

The timer has a period greater than 17.9 seconds.

TEST 09 - TESTING INTERRUPT

Tests that the Disk Controller interrupts the CPU when Not Busy is high, does not interrupt when Not Busy is low, and does not interrupt when Not Busy is high but the interrupt mask is zeroed.

NOT INTERRUPTED WITH NOT BUSY HIGH

After Not Busy is verified to be high, the interrupts are enabled with SMSK set to 020 Hex. If an interrupt is not detected, the error message is displayed.

INTERRUPTED WITH NOT BUSY LOW

Read commands are executed to lower the Not Busy flag. While the Read is in progress, the interrupts are enabled. If an interrupt is detected, the error message is displayed.

INTERRUPTED WITH MASK SET TO ZERO

After Not Busy is verified to be high and SMSK has been set to 0 Hex, the interrupts are enabled. If an interrupt is detected, the error message is displayed.

TEST 10 - TWO UTILITIES AND AN EXERCISER DIAGNOSTIC WHICH EXHAUSTIVELY TEST THE ENTIRE DISK SYSTEM

Control Keys

Once Test 10 identifies itself a number of program control keys become continuously active (interrupt driven). Since the screen and cursor commands are interrupt driven, they may be serviced at any time without interfering with the test in progress.

"PROG" (0FF hex) - Restart the program. That is, first pause for optional test run instructions, then display the parameter menu. The parameter fields will contain the previously entered values.

"SHIFTED PROG" (0FF hex) - exit to the operating system.

"RIGHT ARROW" (083 hex) - move the cursor to the right.

"LEFT ARROW" (081 hex) - move the cursor to the left.

"DOWN ARROW" (080 hex) - move the cursor down.

"UP ARROW" (085 hex) - move the cursor up.

"SHIFTED DOWN ARROW" (090 hex) - scroll the screen down.

"SHIFTED UP ARROW" (095 hex) - scroll the screen up.

"HOME" (082 hex) - set the cursor and screen to their default values.

Utilities

IF any of the utilities are called the following headings will appear on line 2:

```
TRACK NO.#    SURFACE    DISK    DRIVE    OPERATION    IFL=
```

The utilities will fill in the pertinent information in reversed fields to the right of each heading. If an error occurs, the operator will know precisely what command caused it and where in the drive system the error is situated.

The utilities observe the following service priorities:

1. Tracks are serviced succesively from low number to high number.

2. A track on the lower surface of a disk is serviced before the same track on the upper surface of the disk.

3. A track on the removable disk is serviced before the same track on the fixed disk.

4. Drives are serviced in the order they were entered (assuming no redundancies). However, all called for service to a drive must be completed before service to another can begin.

If any errors occur while the utilities are being run the bell will ring and the message, "TYPE; F-FORMAT, C-CHECK, R-RESTORE, V-VERIFY, T-TRY AGAIN, SPACE-CONTINUE, will appear on line 5.

"F-FORMAT" - The operation field is updated to FORMAT. An Initialize Track command is executed on the track specified by the fields to the right of the headings. After the command is completed the IFL field is updated. If an error is detected, the bell will ring again. DSKTST waits for further instructions.

"C-CHECK FORMAT" - The operation field is updated to Check. A Check Track command is executed on the track specified by the Fields to the right of the headings. After the command is completed, the IFL field is updated. If an error is detected, the bell will ring again. DSKTST waits for further instructions.

"R-RESTORE" - The operation field is updated to Restore. A Restore command is executed. After the command is completed the IFL field is updated. If a error is detected the bell will ring again. DSKTST waits for further instructions. It is hoped that this command will free a drive that has become locked up due to a misinterpeted track address.

"V-VERIFY" - The operation field is updated to Read. The entire track specified by the fields to the right of the headings is transferred to a buffer starting on line 9. After the Read command has been completed the IFL field will be updated. If an IFL error is detected, the bell will ring again and DSKTST will wait for further instuctions. If the Read is successful the buffer will be verified according to previously outlined criteria. If a verification error is found, the character in error will be displayed above the buffer in the center of line 8. The same character in the buffer will be marked by a spinning symbol. DSKTST will ring the bell and wait for further instructions. If no verification error is found the buffer is erased and DSKTST waits for further instructions.

"T-TRY AGAIN" - The operation under program control which failed is repeated. If the operation fails again DSKTST will ring the bell and wait for further instructions. If the operation is successful the program will continue. A manual instruction which fails is not repeated by this command.

"SPACE - CONTINUE" - Control is returned to the program which continues as if no errors have been detected.

When the utilities have completely serviced the system, the message "FORMAT COMPLETED, TYPE SPACE TO CONTINUE" is displayed on line 3. If space is depressed DSKTST will begin running an exerciser diagnostic.

Exerciser

A continuous test of the reading and writing abilities of the system. The File Controller's update lock out feature is also tested if more than one terminal is running the exerciser.

Testing is done a track at a time. That is, a buffer is written to a track, read back to a terminal, and matched with every byte in the original buffer. The manner in which the buffer is transferred and the contents of the buffer continually change.

The following rules are obeyed by the exerciser:

1. Tracks are selected pseudo-randomly. The selected track is tested on all entered surfaces and disks of the drive currently being accessed before another track is selected.

2. The selected track is tested on an upper surface before being tested on the lower.

3. The selected track is tested on the removable disk before being tested on the fixed.

4. All entered tracks are tested on one drive before another drive is accessed. Drives are accessed in the order they were entered (assuming no redundancies).

5. The first operation performed on the selected track on a given surface, disk, and drive, is an Update Write. The last operation is a Read. This means the File Controller is free only during a match operation. Thus one terminal can not interfere with another. For example, one terminal may run the Format utility, while the others are running the exercser. The Verify utility may also be run concurrently with the exerciser.

Track Selection - DSKTST builds a table containing all entered track addresses. When a new track is needed a random address is generated. This address is compared with all entries in the table. If a match is detected, the entry is removed from the table and becomes the selected track. If no match is detected, meaning the address generated was not entered or has already been tested, DSKTST alternately defaults to either the highest or lowest address contained in the table. This address is removed from the table and becomes the selected track.

Sector Identification - Every sector is composed of 2 fields. The first 41 bytes of a sector contains the sector identification message, "T R K - X X X S E C - X X X S U R F - X D I S K - X D R - X U N I T - X". Unit refers to the terminal presently writing the message. The remainder of the sector contains variable data.

Test Procedure - The loop count by definition starts at 001 and increments when all entered tracks have been tested on all entered surfaces disks, and drives. When a track is being tested up to 4 read and write commands may be executed. The buffers being transferred by these commands are marked on the screen by 3 " " symbols (03E Hex) and 3 " " symbols (03C). In addition, 153 "@" symbols (00 hex) separate buffers. The loop count controls the number of commands needed to test a track, the buffer length of each command, and the data contained in the variable field of each sector.

The number of sectors to be tested is divided by the loop count. If the remainder is zero, 2 I/O commands will be used to test a track, an Update Write and a Read. The buffer length of these commands will be equal to the number of sectors being tested. If the remainder is not zero, 4 I/O commands will be used to test a track, 2 Update Writes, an Update Read and a Read. The first Update Write command will have a buffer length equal to the remainder. The second Update Write command will have a buffer length equal to the number of sectors being tested minus the remainder. The Update Read and Read commands have similar lengths.

Example:

Number of sectors = 3

Loop count = 1	Loop Count = 2
Update Write on sector 0	Update Write on sectors 0 and 1
Update Write on sectors 1 and 2	Update Write on sector 2
Update Read of sector 0	Update Read of sectors 0 and 1
Read of sectors 1 and 2	Read of sector 2

Loop Count = 3	Loop Count = 4
Update Write on sectors 0, 1 and 2	Update Write on sector 0
Read of sectors 0, 1 and 2	Update Write on sectors 1 and 2
	Update Read of sector 0
	Read of sectors 1 and 2

The first data byte in the variable field of each sector is equal to low order byte of the loop count in binary. the rest of the sector is filled with successively incremented data.

Error Messages

The exerciser detects two kinds of errors. IFL errors and Match errors. On all errors, the last I/O command executed before the error was detected is clearly labeled on lines 2 and 3. Sub-totals of all errors are displayed on lines 6, 7 and 8. An IFL error is defined as any operation which returns with a status not equal to 080 hex. Upon detection the erroneous IFL status is displayed on line 2 and all error counts are updated. A DVCL is executed to release any lock the terminal may have on the File Controller. The exerciser displays the message, "TYPE SPACE - CONTINUE, R-RETRY", on line 4 and waits for instructions from the operator.

R-RETRY - Meaning repeat the operation which failed and wait for instructions from the operator. The operator retains manual control on a retry. A DVCL is always issued after a retry to unlock the File Controller. Thus, retrying an operation on one terminal should not interfere with operations performed on another terminal. The bell will ring and the error counts will be updated each time an IFL error is detected on a repeated operation.

SPACE - CONTINUE - Meaning continue with the next operation as if no error had been detected. The operator loses manual control when space is depressed.

Because any lock on the File Controllers is cleared with a DVCL, a terminal under manual control cannot interfere with other terminals under program control. However, the reverse is not true. Other terminals are free to access the disk system. For example, if the operator retries a Read operation, data written by another terminal may be read instead of the original buffer.

When the exerciser has completed all I/O commands on a given track, the data read back is matched with the data known to be written on that track. The first mismatch is marked with a spinning symbol on the screen. The expected data byte and the actual byte read back are displayed at the start of line 1. To display more of the data buffer, all LEOLS and FLEOLS are replaced with spaces when a match error is detected. The message, "TYPE SPACE-CONTINUE, F-FORMAT, SHIFTED R-UPD RESTORE, R-READ, W-WRITE, C-CHECK, S-SELECT and DVCL", is displayed on lines 4 and 5.

SPACE-CONTINUE - Begin testing the next track. Manual control of the program is lost.

F-FORMAT - The operation field is changed to FORMAT. An Initialize Track command is executed on the track currently being tested. After the command is completed, the IFL status is displayed on line 2. If an IFL error is detected the bell will ring again. The exeriser waits for further instructions. If a track has been destroyed by the testing procedure it may be repaired with this command. An IFL error will result if a Format command is followed by a Read.

SHIFTED R - UPD RESTORE - The operation field is changed to RESTOR. A Restore command (COM2 with command byte 080 Hex) is executed on the drive currently being tested. After the command is completed, the IFL status is displayed on line 2. If an IFL error is detected the bell will ring again. The exerciser waits for further commands.

R-READ - The operation field is changed to READ. The number of sectors being tested are read into the data buffer with either one or two I/O commands, depending on the loop count. Any IFL error will ring the bell. The exerciser waits for further instructions.

W-WRITE - The operation field is changed to WRITE. The data buffer currently displayed is written to the disk with either one or two Write commands, depending on the loop count. Any IFL error will ring the bell. The exerciser waits for further instructions.

C-CHECK - The operation field is changed to CHECK. The track being tested is Checked with either one or two commands, depending on the loop count. Any IFL error will ring the bell. The exerciser waits for further instructions.

S-SELECT & DVCL - The operation field is changed to SELECT. The Disk Controller is selected and a DVCL is executed. Any IFL error will ring the bell. The exerciser waits for futher instructions.

DISKEX

HARD DISK EXERCISER

DISKEX - HARD DISK EXERCISER

GENERAL DESCRIPTION

The purpose of the DISKEX program is to exercise a working hard disk system over long periods of time to determine the type and frequency of I/O errors typically encountered by the operating system. The design of the exerciser stresses adaptability to different machine configurations and a minimal use of resources to test the entire system. To achieve these goals, DISKEX requires an operator to enter data into extensive groups of fields. The field groups are repetitive so the operator should experience little difficulty when entering data.

To provide some understanding of the capabilities and deficiencies of DISKEX, a short list of similarities and differences between DISKEX and the previously released exerciser, DSKTST, is provided below.

DIFFERENCES

1. Only 24 sector/track systems are supported. This means every sector must contain 256 bytes.

2. Only single sector operations are supported. Multi-sectored reading and writing is not tested.

3. The operator may select which sectors are to be tested. Unlike DSKTST, which tested all sectors by following a predefined algorithm, DISKEX randomly tests only the sectors chosen by the operator.

4. DISKEX has a delay option to prevent a high priority terminal from locking out terminals of lower priority.

5. DISKEX can test up to 4 disk controllers in one terminal.

6. At the completion of every I/O operation DISKEX checks that the current controller has issued an interrupt.

7. DISKEX has no provision for stopping an error or returning to the operating system automatically. The exerciser will continually run until told otherwise by the operator.

8. The IFL status of all controllers being tested is continually updated.

SIMILARITIES

1. The file controller's lock-out feature is used to prevent any controller or terminal from interfering with another.

2. The sequence of commands in both programs are the same. Update Write (lock), Update Check (continue lock), Read (unlock).

3. Operator commands are interrupt driven to ensure rapid response.

4. Invalid commands are indicated by a beep from the keyboard.

STARTING UP

DISKEX is divided into two modules. One is for Data Entry, the other contains the Exerciser. After the program is loaded, the screen will remain blank for a moment as DISKEX performs its initialization chores (find end of memory, clearing sections of memory, etc.). When initialization is finished, the program will identify itself by name and version number and display the first group of data entry fields.

DATA ENTRY

Data Entry forms an image of the system configuration and storage media areas to be tested for the exerciser module. Before describing the meaning of the fields the operator must edit. A brief summary of the editing commands is given below:

UP ARROW (085 Hex) - Move the cursor to the 1st byte of the previous field.

DOWN ARROW (080 Hex) - Move the cursor to the 1st byte of the next field.

RIGHT ARROW (083 Hex) - Move the cursor to the next byte of the current field. If executed when the cursor is at the end of the field, a down arrow operation will be performed.

LEFT ARROW (081 Hex) - Move the cursor to the previous byte of the current field. If carried out when the cursor is at the start of a field, an up arrow operation will be effected.

TAB (09 Hex) - Identical to down arrow.

HOME (082 Hex) - Move the cursor to the 1st byte of the 1st unprotected field.

SHIFTED RIGHT ARROW (093 Hex) - Move the cursor to the end of the current field.

SHIFTED LEFT ARROW (091 Hex) - Move the cursor to the 1st byte of the current field.

INSERT (087 Hex) - Insert at cursor position.

DELETE (097 Hex) - Delete at cursor position.

CLEAR (099 Hex) - Erase the current field.

REPLACE (089 Hex) - Replace the contents of the current field with either the default values or the most recently stored values.

RETURN (80D Hex) - Store the data currently displayed in the fields and bring up the next group of fields.

PROG (0EF Hex) - Abort any editing on the current group of fields and bring up the 1st group.

SHIFTED PROG (0FF Hex) - Return to HDOS.

<u>DATA ENTRY</u> - Continued

It should be noted that if the operator gives an editing command to move the cursor out of the current field or store the contents of the current field, the field must contain only valid data. Invalid data will abort the command and beep the Beeper.

In most cases, the cursor will be placed on the field byte in error and a simple explanatory message will be displayed to the right of the field.

FIELD GROUP - ONE

SELECT ADDRESS OF CONTROLLERS - broken field composed of four 2 byte entries. The operator should type in the hex select addresses of the disk controllers to be tested in each entry. Only hex keystrokes will be accepted. The entries in the field will be transferred to a protected field in all succeeding field groups. Error Messages:

WHAT? - Empty field or invalid character.

DOUBLE - The field contains 2 identical select addreses.

4BITS - The cursor is positioned on an entry which does not have exactly 4 bits high.

INITIALIZE PACKS (Format, Check format, Both, Neither) - This field can be used to check or repair the storage areas used by all controllers.

ADDRESS CHECK (Write only, Read only, Neither) - If an entry other than 'N' is made in this field, an alternate exerciser will be called. The alternate exerciser writes to all media areas, reads back what it wrote, and matches. Since locks on the file controller are not established, no use of the delay option is possible. It is assumed that only one terminal will be accessing the file controller. If all media areas are tested, the alternate exerciser can detect any shorts or opens on the address lines.

FIELD GROUPS - TWO THROUGH FIVE

CURRENT CONTROLLER - 2 byte continuous protected field containing one of the select address entries from field group one. This field is a visual aide for the operator.

SLOT (7-10) - 2 byte continuous field containing the slot number of the current controller. The information supplied by this field determines which interrupt the controller will issue at the end of an I/O operation.

VECTOR - 2 byte continuous field containing the memory address of the current controller's starting vector. The exerciser refreshes the vectors before commencing any I/O operation.

FILE CONTROLLER (1-4) - 1 byte field presently unused.

DELAY (0-9) - 1 byte field indicating the amount of time the disk controller idles before establishing a lock on the file controller. The delay period can be used to prevent lock out of low priority terminals. The needed period length is a function of the total system configuration, which is unknown to the exerciser program and therefore must be set by the operator. No delay is required if less than 3 terminals are exercising a file controller.

DRIVES (1-4) - Broken field made up of 4 one byte elements containing the addresses of the drives to be tested. Double entries are not permitted.

FIELD GROUPS - TWO THROUGH FIVE - Continued

PLATTERS (1-4) - Broken field composed of 4 one byte entries containing addresses of the platters to be tested. On DIABLO and PERTEC drives, 1 corresponds to the removeable pack and 4 corresponds to the bottom fixed platter.

SURFACES (Upper, Lower, Both) - 1 byte field specifying which surfaces are to be tested.

TRACKS (0-404) - 35 byte continuous field containing the addresses of the tracks to be tested. The 'thru' operator is represented by a dash. The thru operator must be preceded and followed by a numerical operand.

SECTORS (1-24) - 35 byte continuous field specifying the sectors to be tested. The 'thru' operator is represented by a dash.

When the opertor finishes entering data into the last group of fields, DISKEX leaves the data entry module and passes into the exerciser module. The exerciser displays a command block for each controller being tested before the execution of any I/O command. The command block gives a complete description of the command about to be executed, as well as the current status of the controller. Bookeeping data such as loop and error counts are also presented.

As DISKEX enters the exercise module, a number of interrupt driven commands become active. A description of these commands and their keystrokes is given below:

INTERRUPT DRIVEN COMMANDS

PROG (0EF) - Restart the program.

SHIFTED PROG (OFF) - Exit to HDOS.

SHIFTED UP (095) - Scroll the screen up.

SHIFTED DOWN ARROW (090) - Scroll the screen down.

UP ARROW (085) - Move the cursor up.

DOWN ARROW (080) - Move the cursor down.

RIGHT ARROW (083) - Move the cursor right.

LEFT ARROW (081) - Move the cursor left.

SHIFTED HOME (092) - Move the screen home and place the cursor in row 0, column 0.

HOME (082) - Move the screen home.

L (06C) - Move the screen to the start of the error log.

FIELD GROUPS - TWO THROUGH FIVE - Continued

CONTROL DELETE (01F) - Erase the error log.

SPACE (020) - Stops and restarts the exerciser.

ERRORS

When DISKEX detects either a status or match error, the keyboard is beeped and the appropriate error count is incremented. A brief description of the error is created in the error log and an attempt at recovery is made. This attempt consists of up to 7 retries followed by a restore and another retry. Errors made during the recovery are not logged. If a time out status is detected during recovery, any lock the controller has on the file controller is relinquished before another retry is initiated.

KBDTST

UNIVERSAL KEYBOARD TEST

KBDTST - Universal Keyboard Test

Applicable Assemblies

5000-1132-x-y-z	keyboard
5000-11116-1	ICO keyboard

General Description

The purpose of KBDTST is to determine if the functions of a keyboard are working correctly. KBDTST requires operator assistance throughout the test.

KBDTST may be run on an 8080 or 8085 machine. 8K (or more) of memory is required to run KBDTST.

Loading Procedure

KBDTST can be loaded into memory using any convenient method. When KBDTST loads properly, it will identify itself and await run-mode input by the operator (see Appendix B).

Errors

Because of the inverted pyramid test strategy used in KBDTST, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

On the following pages, each test is listed with a brief description of what it is testing. Below the description, all possible error messages are listed, with an explanation of the cause of the message.

T00: Tests that the keyboard is not selected by an incorrect select address. If the keyboard is selected with the wrong select address this addresss will be displayed in an error message.

T01: Tests that the keyboard is selected with the correct select address. The select address for the keyboard is 0E1H.

T02: Tests that a selected keyboard is deselected by an INIT.

T03: Tests that the Keyboard Character Available flag* can be set. This test requires that the operator depress a key on the keyboard. The operator is given ten seconds in which to do this after which a keyboard malfunction is assumed.

T04: Tests that the Keyboard Character Available flag* is not stuck high by trying to reset it with all of the INIT, DVCL, and INP commands.

T05: Tests that an INIT alone resets the Keyboard Character Available flag*. This test requires that the operator depress a key on the keyboard. The operator is given ten seconds in which to do this after which results are invalid.

T06: Tests that a DVCL alone resets that Keyboard Character Available flag*. This test requires that the operator depress a key on the keyboard. The operator is given ten seconds in which to do this after which results are invalid.

T07: Tests that an INP alone resets the Keyboard Character Available flag*. This test requires that the operator depress a key on the keyboard. The operator is given ten seconds in which to do this after which results are invalid.

T08: Tests that an interrupt is generated by the keyboard when the SMSK is set to allow keyboard interrupt and the Keyboard Character Available flag* is set. This test requires that the operator depress a key on the keyboard. The test will not proceed until this is done.

T09: Tests that no interrupt is generated by the keyboard when the SMSK is set to allow keyboard interrupt and the Keyboard Character Available flag* is reset.

T0A: Tests that the SMSK holds off keyboard interrupts when the SMSK is set to allow no interrupts and the Keyboard Character Available flag* is set. This test requires that the operator depress a key on the keyboard. The test will not proceed until this is done.

T0B: Tests that the keyboard goes into character code repeat mode within the documented time (.5 to 1.2 seconds). This test requires that the operator depress and hold down a key on the keyboard. The test will not proceed until a key is depressed. When the required number of character codes have been generated the operator will be asked to release the key. If the operator does not release the key or the keyboard does not leave repeat mode an error message will be displayed.

T0C: Tests that the keyboard character code repeat frequency is 15 times/second (+/-10%). This test requires that the operator depress and hold down a key on the keyboard. The test will not proceed until a key is depressed. When the required number of character codes have been generated the operator will be asked to release the key. The test will not proceed until the key is released.

T0D: Tests the beep function. Ten beeps will be generated which the operator must listen for.

T0E: Tests the click function. Twenty clicks will be generated which the operator must listen for.

T0F: Tests that the keyboard has N-key rollover. The operator will be asked to depress ten keys simultaneously. The test will not proceed until a keystroke is detected.

T10: Tests that the status lights are working properly. The operator will be asked to depress a key eight times and watch the status lights. An image of what the status lights should be will be displayed on the screen.

T11-T13: Reserved for later use.

T14: Tests that the keys of the ASCII pad generate the proper codes in the unshifted, shifted, control, and shifted control modes. An image of the keyboard will be displayed with instruction for proceeding on line two of the screen. Instructions for proceeding will appear on the bottom line of the screen if an error is detected.

T15: Tests the shift lock key. The operator will be asked to put the keyboard into shift lock mode and then depress a key. Then the operator will be asked to take the keyboard out of shift lock mode and depress the same key.

T16: Tests that the keys of the cursor pad generate the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14.

T17: Tests that the keys of the numeric pad generate the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14.

T18: (For standard and ICO type keyboards). Tests that the keys of the function pad generate the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14.

T18: (For CCI type keyboards). Tests that the keys of function pad one generate the proper codes in unshifted, shifted, control and shifted modes. Procedure is the same as in T14.

T19: (For CCI type keyboards). Tests that the keys of function pad two generate the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14.

T1A: (For CCI type keyboards). Tests that the keys of function pad three generate the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14.

T1B: (For CCI type keyboards). Tests that the keys of function pad four generate the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14.

T1C-T1E: Reserved for later use.

T1F: (For standard and ICO type keyboards). Tests that the prog key generates the proper codes in unshifted, shifted, control, and shifted control modes. Procedure is the same as in T14. Shifted Control Prog will reboot the system to avoid this depress ctrl/@ and no error will be generated.

* Keyboard Character Available flag is IFL bit 7.

HOW TO CREATE A VERSION OF KBDTST

(For Standard and ICO Type Keyboards)

Each version of KBDTST must be made by linking together the modules specific to that version. Linking the modules can be accomplished by using the following command line:

```
>LINKEDIT COMB (P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11
```

where P1 - P5 are any of the keyboard feature names:

```
      REPT or REPTF, AUD, ROLL, LITE, SHLK
where REPT is the code for the keyboard repeat feature (15 char/sec)
      REPTF is the code for the keyboard repeat feature (33 char/sec)
      AUD is the code for the Audible Alarm (Beep and Click)
      ROLL is the code for the N-key rollover feature
      LITE is the code for the Status light feature
  and SHLK is the code for the shift lock key feature
```

P6 - P10 are the code numbers for the ASCII, CURSOR, NUMERIC, FUNCTION, and PROG pads respectively. The code meanings can be found in Table 1.

P11 is a two character code that will be appended to the name of the binary file name.

If any of the feature names are not required for KBDTST they must be substituted by a backslash (\).

Example:

```
>LINKEDIT COMB (REPT AUD SHLK \ \ 00 00 02 01 00
```

The order of the feature names is not important.

The modules needed to create KBDTST must be on the device assigned to the UTIL1 channel. The complete binary file (KBDTSTXX MBIN80) will be placed on the device assigned to the BINOUT channel.

FILE	DESCRIPTION
KBDASC00	Standard ASCII pad (with control or shift lock key)
KBDASC01	ASCII pad with 'Y' and 'Z' keys reversed (PCO 215)
KBDASC02	ASCII pad with 'DEL' and 'O' keys reversed (PCO 273)
KBDASC03	ASCII pad with 'Y' and 'Z' keys reversed and 'DEL' and 'O' keys reversed (PCO 214)
KBDASC04	Olivetti ASCII pad for Germany and Austria
KBDASC05	Olivetti ASCII pad for Holland
KBDASC06	Olivetti ASCII pad for Australia and the USA
KBDASC07	Olivetti ASCII pad for Italy
KBDASC08	Olivetti ASCII pad for France
KBDASC09	Olivetti ASCII pad for Great Britain
KBDASC10	Olivetti ASCII pad for Canada
KBDASC11	CCI ASCII pad (for USC with module KBDNNSTD)
KBDASC12	Lockheed ASCII pad
KBDCUR00	Standard cursor pad
KBDCUR01	Olivetti cursor pad
KBDCUR02	CCI cursor pad (for use with module KBDNNSTD)
KBDNUM00	Standard numeric pad (13 keys)
KBDNUM01	Numeric pad (13 keys including control key or shift lock key)
KBDNUM02	Numeric pad (16 keys with codes parallel to keys 9, 10, 11) (PCO 142)
KBDNUM03	Numeric pad (16 keys with codes parallel to keys 10, 11, 12) (PCO 107 and PCO 108)
KBDNUM04	Olivetti numeric pad
KBDNUM05	16 Key Numeric pad with control key
KBDNUM06	CCI numeric pad (for use with module KBDNNSTD)
KBDFUN00	Standard function pad (with or without lights)
KBDFUN01	Function pad (keys 9, 10, 11 deactivated) (PCO 142)
KBDFUN02	Function pad (keys 10, 11, 12 deactivated) (PCO 107 and PCO 108)
KBDFUN03	CCI function pads (for use with module KBDNNSTD)
KBDPRG00	Standard prog key

ASC is the code for ASCII pad
CUR is the code for CURSOR pad
NUM is the code for NUMERIC pad
FUN is the code for FUNCTION pad
PRG is the code for PROG pad

The two digits that follow each file name is the code for that type of key pad. These are the numbers to be used as P6 - P10 when creating KBDTST.

Table 1.

FILE	DESCRIPTION
KBDASC00	Standard ASCII pad (with control or shift lock key)
KBDASC01	ASCII pad with 'Y' and 'Z' keys reversed (PCO 215)
KBDASC02	ASCII pad with 'DEL' and 'O' keys reversed (PCO 273)
KBDASC03	ASCII pad with 'Y' and 'Z' keys reversed and 'DEL' and 'O' keys reversed (PCO 214)
KBDASC04	Olivetti ASCII pad for Germany and Austria
KBDASC05	Olivetti ASCII pad for Holland
KBDASC06	Olivetti ASCII pad for Australia and the USA
KBDASC07	Olivetti ASCII pad for Italy
KBDASC08	Olivetti ASCII pad for France
KBDASC09	Olivetti ASCII pad for Great Britain
KBDASC10	Olivetti ASCII pad for Canada
KBDASC11	CCI ASCII pad (for USC with module KBDNNSTD)
KBDASC12	Lockheed ASCII pad
KBDASC13	Olivetti ASCII pad for Finland
KBDASC14	Keytronics ASCII pad
KBDASC15	Philips ASCII pad
KBDCUR00	Standard cursor pad
KBDCUR01	Olivetti cursor pad
KBDCUR02	CCI cursor pad (for use with module KBDNNSTD)
KBDNUM00	Standard numeric pad (13 keys)
KBDNUM01	Numeric pad (13 keys including control key or shift lock key)
KBDNUM02	Numeric pad (16 keys with codes parallel to keys 9, 10, 11) (PCO 142)
KBDNUM03	Numeric pad (16 keys with codes parallel to keys 10, 11, 12) (PCO 107 and PCO 108)
KBDNUM04	Olivetti numeric pad
KBDNUM05	16 Key Numeric pad with control key
KBDNUM06	CCI numeric pad (for use with module KBDNNSTD)
KBDNUM07	OP-1/15 numeric pad
KBDFUN00	Standard function pad (with or without lights)
KBDFUN01	Function pad (keys 9, 10, 11 deactivated) (PCO 142)
KBDFUN02	Function pad (keys 10, 11, 12 deactivated) (PCO 107 and PCO 108)
KBDFUN03	CCI function pads (for use with module KBDNNSTD)
KBDPRG00	Standard prog key

ASC is the code for ASCII pad
CUR is the code for CURSOR pad
NUM is the code for NUMERIC pad
FUN is the code for FUNCTION pad
PRG is the code for PROG pad

The two digits that follow each file name is the code for that type of key pad. These are the numbers to be used as P6 - P10 when creating KBDTST.

Table 1.

WETTST

WORD & ETED DISPLAY TEST

WETTST/WETTST15 and WET8A/WET8B - WORD/ETED DISPLAY TEST

Applicable Assemblies

5000-1145	ETED Display Microprocessor
5000-1164	WORD Display Microprocessor
5100-1109	OP-1/R WORD Display Microprocessor
5300-1102	OP-1/15 Logic Board

General Description

WETTST and WET8A/WET8B were designed to test any of the three above Display Microprocessors in any of their configurations. Any attribute prom or plug where applicable may be used on the display board provided the corresponding overlay is available. WETTST WETTST15 and WET8A/WET8B requires operator action and visual verification for each subtest. WET8A and WET8B have been created by splitting WETTST into 8K segments.

More than 16K of memory is required to run WETTST.
8K (or more) or memory is required to run WET8A/WET8B.
16K (or more) of memory is required to run WETTST15 (OP-1/15).

Loading Procedure

WETTST/WETTST15/WET8A/WET8B can be loaded into memory using any conveniently available loading method. WET8A/WET8B must be downline loaded or entered using the applicable RUN program WETTST and WET8A/WET8B are currently configured to test a 5000-1164 board containing attribute prom 508-01640-002. WETTST15 is currently configured to test a 5300-1102 board containing attribute prom 508- 02930-002. Thus, to test a different configuration requires an additional procedure. This procedure is after successfully loading WETTST, WETTST15 or WET8A/WET8B load the correct overlay in the same fashion. To simplify this procedure, one may combine WETTST, WETTST15 or WET8A/WET8B with the correct overlay generating a new binary. After WETTST15 has loaded correctly, it will relocate 600H of its messages to location 0900H, so this program is capable of running in a (OP-1/15) 16K environment.

Operator Action

After first loading if WETTST is being used with an ETED or WORD display overlay the operator must answer 'Y' or 'N' to whether an alternate character set is present. No questions are asked for an OP-1/R WORD display. After the first question the operator must give the carrier plug number for an OP-1 WORD display board only. The plug number is of the form 1364-XY (X=1 thru 3, Y=A thru J).

Errors

There are no program-detectable errors. Errors are recognized by operator inspection of the display screen.

R:C-02/24/81

Test Description

All subtests are described on the following pages. Most subtests require the 'SPACE' bar to be depressed to advance to the next subtest. However, all subtests display a message describing the way to proceed.

Character Generator Test

A slewed pattern of characters is displayed filling the entire screen.

Character Generator Test: WETTST/WET8A/WETTST15

A slewed pattern of characters is displayed filling the entire screen.

Cursor Movement Test: WETTST/WET8A/WETTST15

Tests that the cursor can be scrolled horizontally across an 80 column screen and can be scrolled vertically down a 24 line screen.

1. Horizontal Scrolling - The cursor position is labeled in decimal on lines 3 and 4. The contents of the cursor vectors are displayed on line 6 in decimal. The value of CURHOR is incremented from 00 to 79 each time the space bar is depressed.

2. Vertical Scrolling - The cursor position is labeled in decimal in columns 78 and 79. The contents of the cursor vectors are displayed on line 6 in decimal. The value of CURVRT is incremented from 00 to 23 each time the space bar is depressed. With WETTST15, this test has become a two part test. It vertically scrolls the cursor through line 0 to 24 with the 25th line enabled, and then with the 25th line disabled. If the cursor appears on the 25th line while it is disabled status bit 5 of location 807H is stuck high.

3. Cursor Blanking Test - The value of CURVRT is set to 25 decimal which should move the cursor off the screen.

Character Presentation Test: WETTST/WET8A/WETTST15

The messages "TAGGED CHARACTER LINE", and "NORMAL CHARACTER LINE" are displayed. All bytes composing the first message have bit 7 high. When space is depressed DISPLY1 (location 802 hex) is loaded with a new value. The expected visual appearance of the messages (blinking, reversed, etc.) is indicated to the right of the messages for every DISPLY1 value. The following hex values are loaded into DISPLY1: 00, 02, 04, 06, 80, 40, 42, 44, 46, C0 (if applicable).

After the DISPLY1 values have been tested, the normal character set is displayed. The alternate character set and the combined character set may be displayed by typing space. If the alternated character has not been requested typing space executes the next test.

LEOL and FLEOL Test: WETTST/WET8A/WETTST15

On the left side of the screen an indication (blanked or normal) of the visual appearance of an adjacent message is given. This test is not applicable for the OP-1/R.

Disable and Enable Test: WETTST15

Tests that the 25th line can be enabled and disabled by lowering and raising bit 5 of location 807H. If a message appears on the 25th line during the disable portion of the test, this bit is stuck high.

Home Addressing Test: WETTST/WET8A/WETTST15

Every time a space is typed the contents of the home vectors are changed so that messages in numerically ascending order are displayed in the upper left portion of the screen. With WETTST15, the contents of the home vectors for the 25th line are tested simultaneously with the 803 and 804 home vectors. The messages are also displayed in ascending order in the left position of the 25th line.

Wrap Addressing Test: WETTST/WET8A/WETTST15

Every time a space is typed, the contents of the wrap vectors are changed so that messages in numerically ascending order are displayed in the lower left portion of the screen. After this the message:

"THIS MESSAGE SHOULD NOT APPEAR ON BOTTOM LINE ELSE WRAP NOT WORKING"

will be displayed on the second line of the screen. If it does appear on the bottom line then memory is wrapping on 8K boundaries.

Special Blankable Characters Test: WETTST/WET8B/WETTST15

On the left side of the screen under the heading "Blankable Character Map", the 256 character matrix is represented by an 'S' symbolizing the corresponding position of the suppressable characters in the "Character Matrix" on the right and periods (.) symbolizing the nonsuppressable characters. If either an 'S' or '.' is a suppressable character a different symbol will be required when creating an overlay. Typing the 'SPACE' bar will cause the suppressable characters and their 'S' representations to blank thus causing a similarity in the two matrices. One additional note is that to suppress characters the enhancements must be enabled and this will affect the screen.

Enhancements Operators Test: WETTST/WET8B/WETTST15

If a WORD DISPLAY board is being tested, WETTST will identify the active plug which should be inserted. After verifying that the proper plug is in place, the operator should type space to continue the test.

Three test messages, "HOME, WRAP and TAGGED" are displayed at the top of the screen. During the test these messages are surrounded by start and stop attribute characters to change their appearance. The message "HOME" is located at the end of memory. "WRAP and TAGGED" are located after the contents of the wrap vectors which is far from the end of memory. "TAGGED" is composed of bytes with bit 7 high.

In the center of the screen a matrix is displayed which indicates the expected appearance of the test messages.

At the bottom of the screen the contents of memory locations 0800 thru 0807 hex are labeled and displayed. The contents of these locations may be changed by typing the F0 thru F3 keys (shifted or unshifted) followed by 2 hex digits.

Of special note, if space is a suppressable character the attributes will not affect them.

Attribute Check Test: WETTST/WET8B/WETTST15

Tests that the attribute can be seen while in the visual enhancement mode (Location 802=01) and also tests that the attribute can start away from the start of Home and wrap position.

Screen Width Test: WETTST/WET8B/WETTST15

This test is not applicable for an ETED Display board.

This test will verify that the 80, 132 (not applicable for OP-1/RW or the OP-1/15), and 160 column screen widths are functioning properly. To verify quickly the correct screen widths, the Right arrow can be depressed which should bring the last displayable character immediately to line 2.

E394 is an OVERLAY for testing Attribute Prom # 508-00394-002 for an 5000-1145 board.

START ATTRIBUTE 1:	09
END ATTRIBUTE 1:	05
START ATTRIBUTE 2:	1B
END ATTRIBUTE 2:	1D
START ATTRIBUTE 3:	02
END ATTRIBUTE 3:	0C
SUPPRESSABLE CHARACTERS:	00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 8A, 8B, 8C, 8D, 8E, 8F, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9C, 9D, 9E, 9F

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

W730 is an OVERLAY for testing Attribute Prom # 508-00730-002 for an 5000-1164 board.

START ATTRIBUTE 1:	15, 95
END ATTRIBUTE 1:	16, 96
START ATTRIBUTE 2:	11, 91
END ATTRIBUTE 2:	12, 92
START ATTRIBUTE 3:	13, 93
END ATTRIBUTE 3:	14, 94
SUPPRESSABLE CHARACTERS:	11, 12, 13, 14, 15, 16, 20, 91, 92, 93, 94, 95, 96

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

E784 is an OVERLAY for testing Attribute Prom # 508-00784-002 for an 5000-1145 board.

START ATTRIBUTE 1:	09, 89
END ATTRIBUTE 1:	05, 85
START ATTRIBUTE 2:	0D, 1B, 8D, 9B
END ATTRIBUTE 2:	0E, 1D, 8E, 9D
START ATTRIBUTE 3:	02, 82
END ATTRIBUTE 3:	0C, 8C
SUPPRESSABLE CHARACTERS:	00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1E, 1F, 20, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 8A, 8B, 8C, 8D, 8E, 8F, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9C, 9D, 9E, 9F

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

W794 is an OVERLAY for testing Attribute Prom # 508-00794-002 for an 5000-1164 board.

START ATTRIBUTE 1:	09, 89
END ATTRIBUTE 1:	05, 85
START ATTRIBUTE 2:	0D, 1B, 8D, 9B
END ATTRIBUTE 2:	0E, 1D, 8E, 9D
START ATTRIBUTE 3:	02, 82
END ATTRIBUTE 3:	0C, 82
SUPPRESSABLE CHARACTERS:	00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 8A, 8B, 8C, 8D, 8E, 8F, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9C, 9D, 9E, 9F

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

W1140 is an OVERLAY for testing Attribute Prom # 508-01140-002 for an 5000-1164 board or its equivalent

START ATTRIBUTE 1:	OEH, 8EH
END ATTRIBUTE 1:	NA
START ATTRIBUTE 2:	18H, 98H
END ATTRIBUTE 2:	NA
START ATTRIBUTE 3:	19H, 99H
END ATTRIBUTE 3:	NA
END ALL ATTRIBUTES:	0FH, 8FH
SUPPRESSABLE CHARACTERS:	0EH, 0FH, 18H, 19H 8EH, 8FH, 98H, 99H

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

W1274 is an OVERLAY for testing Attribute Prom # 508-01274-002 for an 5000-1164 board.

START ATTRIBUTE 1:	09
END ATTRIBUTE 1:	05
START ATTRIBUTE 2:	1B
END ATTRIBUTE 2:	1D
START ATTRIBUTE 3:	02
END ATTRIBUTE 3:	0C
SUPPRESSABLE CHARACTERS:	01, 02, 04, 05, 06, 07, 08, 09, 0B, 0C, 0D, 0E, 0F, 15, 16, 17, 1B, 1C, 1D, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 8A, 8B, 8C, 8D, 8E, 8F, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9C, 9D, 9E, 9F, FF

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

W1900 is an OVERLAY for testing Attribute Prom # 508-01900-002 for an 5000-1164 board.

START ATTRIBUTE 1:	81
END ATTRIBUTE 1:	82
START ATTRIBUTE 2:	83
END ATTRIBUTE 2:	84
START ATTRIBUTE 3:	85
END ATTRIBUTE 3:	86
SUPPRESSABLE CHARACTERS:	00, 13, 14, 15, 16, 19, 1A, 1B, 7E, 80, 81, 82, 83, 84, 85, 86, 8A, 8B, 93, 94, 95, 96, 9A, 9B, FE

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

RTEST is an temporary OVERLAY for testing RTEST for an 5000-1109 board.

START ATTRIBUTE 1:	01, 03, 81, 83
END ATTRIBUTE 1:	02, 04, 82, 84
START ATTRIBUTE 2:	05, 07, 85, 87
END ATTRIBUTE 2:	06, 08, 86, 88
START ATTRIBUTE 3:	09, 89
END ATTRIBUTE 3:	0A, 8A
SUPPRESSABLE CHARACTERS:	00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 8A, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9C, 9D, 9E, 9F

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE: '.'

W3210 is an OVERLAY for testing Attribute Prom # 508-03210-002 for an 5000-1164 board or its equivalent.

START ATTRIBUTE 1:	01, 81
END ATTRIBUTE 1:	02, 82
START ATTRIBUTE 2:	N/A
END ATTRIBUTE 2:	N/A
START ATTRIBUTE 3:	N/A
END ATTRIBUTE 3:	N/A
SUPPRESSABLE CHARACTERS:	00, 01, 02, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1E, 1F, 80, 81, 82, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9E, 9F
SUPPRESSABLE CHARACTER REPRESENTATIVE:	'S'
NONSUPPRESSABLE CHARACTER REPRESENTATIVE:	'.'

W3430 is an Overlay for testing Attribute Prom #508-03430-002 for an 5000-1164 board or its equivalent.

START ATTRIBUTE 1:	01, 03, 81, 83
END ATTRIBUTE 1:	02, 04, 82, 84
START ATTRIBUTE 2:	05, 07, 85, 87
END ATTRIBUTE 2:	06, 08. 86, 88
START ATTRIBUTE 3:	09, 89
END ATTRIBUTE 3:	0A, 8A
SUPPRESSABLE CHARACTERS:	00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1E, 1F, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 8A, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 9A, 9B, 9E, 9F
SUPPRESSABLE CHARACTER REPRESENTATIVE:	'S'
NONSUPPRESSABLE CHARACTER REPRESENTATIVE:	'.'

W1640 is an OVERLAY for testing Attribute Prom # 508-01640-002 for a 5000-1164 board.

START ATTRIBUTE 1:	01H, 03H, 81H, 83H
END ATTRIBUTE 1:	02H, 04H, 82H, 84H
START ATTRIBUTE 2:	05H, 07H, 85H, 87H
END ATTRIBUTE 2:	06H, 08H, 86H, 88H
START ATTRIBUTE 3:	09H, 89H
END ATTRIBUTE 3:	0AH, 8AH
END ALL ATTRIBUTES:	NA

SUPPRESSABLE CHARACTERS: 00H-0AH, 10H-1FH, 80-8AH, 90H-9FH

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE:'.'

W3430 is an OVERLAY for testing Attribute Prom # 508-03430-002 for a 5000-1164 board.

START ATTRIBUTE 1:	01H, 03H, 81H, 83H
END ATTRIBUTE 1:	02H, 04H, 82H, 84H
START ATTRIBUTE 2:	05H, 07H, 85H, 87H
END ATTRIBUTE 2:	06H, 08H, 86H, 88H
START ATTRIBUTE 3:	09H, 89H
END ATTRIBUTE 3:	0AH, 8AH

SUPPRESSABLE CHARACTERS: 00H-0AH, 10H-1BH, 1EH, 1FH, 80H-8AH, 90H-9BH, 9EH, 9FH.

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE: '.'

R2930 is an OVERLAY for testing Attribute Prom # 508-02930-002 for an 5100-1143 OP-1/RWII

START ATTRIBUTE 1:	01H, 03H, 81H, 83H
END ATTRIBUTE 1:	02H, 04H, 82H, 84H
START ATTRIBUTE 2:	05H, 07H, 85H, 87H
END ATTRIBUTE 2:	06H, 08H, 86H, 88H
START ATTRIBUTE 3:	09H, 89H
END ATTRIBUTE 3:	0AH, 8AH
END ALL ATTRIBUTES:	NA
SUPPRESSABLE CHARACTERS:	00H, 01H, 02H, 03H, 04H, 05H, 06H, 07H, 08H, 09H, 0AH, 10H-1FH, 80H-8AH, 90H-9FH

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE: '.'

R3750 is an OVERLAY for testing Attribute Prom # 508-03750-002 for a 5100-1113 OP-1/RWII.

START ATTRIBUTE 1:	00H
END ATTRIBUTE 1:	80H
START ATTRIBUTE 2:	01H
END ATTRIBUTE 2:	81H
START ATTRIBUTE 3:	09H
END ATTRIBUTE 3:	89H
END ALL ATTRIBUTES:	NA

SUPPRESSABLE CHARACTERS:00H, 01H, 09H, 80H, 81H, 83H

SUPPRESSABLE CHARACTER REPRESENTATIVE: 'S'

NONSUPPRESSABLE CHARACTER REPRESENTATIVE: '.'.

TROUBLE

UNIVERSAL TROUBLESHOOTING TOOL

TROUBLE

TROUBLE is a universal, operator-usable diagnostic tool, valuable for both System Development and general hardware trouble-shooting. Using TROUBLE, a complex series of commands may easily be issued to any I/O device while IOM progress, IFL status, and data transfers are continuousy monitored.

Hardware commands may be single-stepped, or issued via an operator-programmed sequence of instructions.

In addition to aiding in the testing of hardware, TROUBLE features a disassembler to increase understanding of hardware logic in relation to software implementation. As an operator-designed program is being executed, the actual software commands required to perform the operation are shown on the display screen.

TROUBLE employs the twelve Function Pad keys (F0-F11) for entering program commands. An image of these keys is displayed on the screen with legends reflecting each keys's use in any of four operating modes: EDIT, BUFFR, INSTR, and RUN (see Figure 1-4). A mode is selected by depressing the F0 key repeatedly until its legend indicates the desired mode.

The EDIT mode is for entering the desired parameters. This is the mode in which the hardware vectors are initialized. One may set the Starting, Current and Terminating addresses as well as the Terminating character, in both the Main and Secondary channels. In addition, this mode is used for setting up Command byte values. For many of the Function pad keys there exists a data field above and below the legend. This is to allow the operator to execute the same command with two different Command byte values. The field above the legend corresponds to the shifted equivalent of the key. The four cursor keys are utilized to move the cursor to a desired field, In addition, the shifted up and down arrows will scroll the screen. All fields including slot number accept only hexadecimal characters.

The BUFFR mode is for setting up the Main and Secondary Channel data buffers. These buffers will accept standard key-generated ASCII characters when the F1 key legend is "ASCII". Depressing the F1 key will change its legend to "HEX", at which time any hexadecimal key code may be entered. The CLRMC and CLRSC keys are used for clearing the Main and Secondary Channel buffers respectively. The OTHER key is used for re-positioning the cursor from one buffer to the other. In addition, the four cursor keys and the shifted up and down arrows are utilized for positioning the cursor and scrolling the screen.

The INSTR mode is for constructing programs. The program entered will be executed later in the RUN mode. A program is entered by depressing the function pad keys with the legend corresponding to the desired I/O command. These commands may be entered in any order and either the shifted or unshifted fields may be utilized. As keys are entered a table will be created on the line with the cursor corresponding to the desired commands. The left and right cursor .are used for positioning to the beginning and end of this eighty byte maximum command table. If the cursor is positioned over a command byte in the table the command will be disassembled and displayed on the right of the screen. In addition to overwriting commands by positioning the cursor over the command, one may also insert and delete commands. To insert a command between any two commands, depress the key above the left arrow key (84 Hex) and then overwrite the space. To delete a command, depress the same key shifted (94 Hex).

The RUN mode is the mode in which I/O commands are actually executed. In this mode one may issue single commands by depressing the function pad key corresponding to the desired command. This mode also allows for the execution of a program previously entered in the INSTR mode by depressing "X" for execute. The RUN mode, like all other modes, allows the operator to scroll the screen by using the shifted up and down arrows.

Other keys that perform specific operations are as follows:

Control PROG - Restarts TROUBLE but does not destroy the INSTR command table nor does it destroy the contents of the Main and Secondary Channel buffers.

PROG - Restarts TROUBLE and reinitializes the INSTR table and the Main and Secondary Channel buffers.

Z - The "Z" key is active in the INSTR and RUN modes only. The "Z" is used as a command key to reset the Main and Secondary Channel Current addresses to zero. Version 0.3 on.

M - The "M" key is active in the INSTR and RUN modes only. The "M" is used as a command key to clear the Main Channel Buffer to spaces. Version 0.4 on.

S - The "S" key is active in the INSTR and RUN modes only. The "S" is used as a command key to clear the Secondary Channel Buffer to spaces. Version 0.4 on.

I - The "I" key is active in the INSTR and RUN modes. It enables a wait for two (shifted/un-shifted) additional IFL conditions similiar to the IFLWT key, with parameters entered in the EDIT mode in the key field labelled I-WT. Version 0.5 on.

Variable Output Command Keys:

These are active in the INSTR mode for programming variable output command bytes. The command keys, all located on the Numeric keypad, are:

N0 -	OFL
N12 -	OPT
N1 -	COM1
N2 -	COM2
N3 -	COM3

After one of the Command keys is typed, the Command byte value is entered by typing two hexadecimal numbers (0-F). Version 0.5 on.

```
TROUBLE  V 0.3        00          00 FF         00       00     00     00     00
   EDIT  INIT  DVCL  SEL      IFL  IFLWT  INP   OFL      OPT   COM1   COM2   COM3
                      00       00  00 FF   00    00       00     00     00     00
  SLOT = 0A
  MCSA = 1B74  SCSA = 1CB4        00 FF
  MCCA = 0000  SCCA = 0000        I-WT
  MCTA = 1CAA  SCTA = 1E8A        00 FF
  MCTC = 2A *  SCTC = 23 #
COMMANDS:


  MCSA>


                                                                            <MCTA
  SCSA>


                                                                            <SCTA
```

FIGURE -1-

```
TROUBLE  V 0.3                                                  LOCATION = 1AF5/20
  BUFFR  ASCII CLRMC CLRSC  OTHER

  SLOT = 0A
  MCSA = 1B74  SCSA = 1CB4
  MCCA = 0000  SCCA = 0000
  MCTA = 1CAA  SCTA = 1E8A
  MCTC = 2A *  SCTC = 23 #
COMMANDS:


  MCSA>


                                                                           <MCTA
  SCSA>


                                                                           <SCTA
```

FIGURE -2-

```
TROUBLE  V 0.3         00          00 FF         00       00    00    00    00
   RUN    INIT  DVCL  SEL     IFL  IFLWT  INP   OFL     OPT    COM1  COM2  COM3
                       00      00  00 FF   00    00       00    00    00    00
  SLOT = 0A
  MCSA = 1B74   SCSA = 1CB4        00 FF
  MCCA = 0000   SCCA = 0000        I-WT
  MCTA = 1CAA   SCTA = 1E8A        00 FF
  MCTC = 2A *   SCTC = 23 #
COMMANDS:


  MCSA>


                                                                            <MCTA
  SCSA>


                                                                            <SCTA
```

FIGURE -3-

```
TROUBLE  V 0.3          00          00 FF          00      00    00    00    00
  INSTR  INIT  DVCL  SEL      IFL   IFLWT   INP   OFL     OPT   COM1  COM2  COM3
                        00      00   00 FF   00    00      00    00    00    00
  SLOT = 0A
  MCSA = 1B74  SCSA = 1CB4          00 FF
  MCCA = 0000  SCCA = 0000          I-WT
  MCTA = 1CAA  SCTA = 1E8A          00 FF
  MCTC = 2A *  SCTC = 23 #
COMMANDS:


  MCSA>

                                                                          <MCTA
  SCSA>

                                                                          <SCTA
```

FIGURE -4-

SDLCTST

SDLC COMMUNICATIONS CONTROLLER TEST

SDLCTST - SDLC Communications Controller Test

Applicable Assemblies

5000-11106-01	SDLC Communications Controller
	SDLCTST Test Cable

General Description

SDLCTST is s self-contained diagnostic program for testing the SDLC Controller. The Controller to be tested should be deployed in Device slot 7 while a known good SDLC Controller with a different Select Address is placed in slot 8, 9, or 10. The SDLCTST Test Cable connects the two appropriate device slots at the rear of the terminal.

Loading

SDLCTST can be loaded by any standard method. When the program loads successfully, a data entry field is displayed on the screen. The operator should enter the appropriate information (Slot number, select address) for Controller No. 1 (Master) and No. 2 (Controller under test).

When the data has been entered properly, the RETURN key is typed, causing the program to initialize.

Operator Action

When SDLCTST initializes, the program name and version number are displayed, and the program waits for run-mode input (See Appendix C).

SDLCTST uses an inverted pyramid testing strategy, i.e., the most basic Controller functions are tested first, and all subsequent tests depend on the success of previous tests.

Description of Tests

T00 Select (Incorrect Address)

Attempt to select Controller with single-byte addresses which have 3 or less bits high. Error if IFL produces result other than 0FFH (open bus).

T01 Select (Correct Address)

Attempt to select Controller with correct address (69H). Error if IFL produces result of 0FFH (open bus).

T02 INIT De-Select

Select Controller with address 69H. Issue INIT, IFL. Error if result is not 0FFH, or if Test T01 fails.

T03 Read Controller Status

Select Controller. Issue, OFL 08H, IFL. Error if result not 00H (bit 2 ignored).

T04 Read 8273 Status

Select Controller. Clear Controller with DVCL, COM2, 01H, COM2 00H. Issue OFL 00H, IFL. Error if result not 00H.

T05 Command Busy Set by COM1

Select and clear Controller. Issue COM1 A3H (Set Port B). Read 8273 Status. Error if Bit 7 (Command Busy) not set or if Test T04 fails.

T06 Command Busy Reset

Select and Clear Controller. Issue COM1 23H (Read Port B). Error if Bit 7 (Command Busy) not reset after nominal delay.

T07 Command Result Buffer Full Set

Issue Read Port B. Read 8273 Status. Error if Bit 4 (Command Result Buffer Full) not set or if Test T06 fails.

T08 Command Result Buffer Full Reset

Set 8273 Bit 4 as in Test T07. Issue OFL 01H, IFL. Read 8273 Status. Error if Bit 4 not reset, or if Test T07, fails.

T09 Command Buffer Full Reset

Issue Set Port B. Read 8273 Status. Error if Bit 6 (Command Buffer Full) not reset, or if Test T05 fails.

T0A Command Parameter Buffer Full Reset

Issue Set Port B. Output a parameter byte by OPT. Ready 8273 Status. Error if Command Busy and Command Parameter Buffer Full both not reset, or if Test T09 fails.

T0B Port B Flags Reset

Attempt reset of RTS, DTR, and FLGDET by Reset Port B. Issue Read Port B. Error if RTS, DTR, FLGDET not reset.

T0C Port B Flags Set

Attempt to set RTS, DTR, and FLGDET by Set Port B. Issue Read Port B. Error if RTS, DTR, and FLGDET not set.

T0D Read Master Port A

Attempt to set RTS, DTR, and FLGDET. Read Master Port A. Error if CD, DSR not set, or if Test T0C fails.

T0E RTS Only Set

Reset Port B flags. Attempt to set RTS only by Set Port B. Issue Read Port B. Error if RTS only not set or if Test T0B fails.

T0F DTR Only Set

Same as Test T0E but testing DTR only.

T10 FLGDET Only Set

Same as Test T0E but testing FLGDET only.

T11 RTS Only Reset

Set Port B flags. Attempt to reset RTS only by Reset Port B. Issue Read Port B. Error if RTS only not reset, or if Test T0C fails.

T12 DTR Only Reset

Same as Test T11 but testing DTR only.

T13 FLGDET Only Reset

Same as Test T11 but testing FLGDET only.

T14 Reset RTS, DRT, FLGDET by Clear Controller

Set Port B flags. Issue DVCL, COM2 01H, COM2 00H, (Clear Controller). Issue Read Port B. Error if RTS, DTR, FLGDET not reset, or if Test T0C fails.

T15 Master DSR Only Set

Set DTR only by Set Port B. Read Master Port A. Error if DSR only not set or if Test T0F fails.

T16 Master CD Only Set

Set RTS only by Set Port B. Read Master Port A. Error if CD only not set, or Test T0E fails.

T17 Port A Flags Reset

Reset Master RTS, DTR, FLGDET by Reset Port B. Read Controller Port A. Error if CTS, DC, DSR not reset.

T18 Port A Flags Set

Set Master RTS, DTR by Set Port B. Read Controller Port A. Error if CTS, CD, DSR not set.

T19 CTS Only Set

Reset CTS, DC, DSR by Master Reset Port B. Set Master RTS. Read Controller Port A. Error if CTS only not set, or if Test T17 fails.

T1A CD Only Set

Reset CTS, CD, DSR by Master Reset Port B. Set Master RTS. Read Controller Port A. Error if CD only not set, or if Test T17 fails.

T1B DSR Only Set

Reset CTS, CD, DSR by Master Reset Port B. Set Master DTR. Read Controller Port A. Error if DSR only not set or if Test T17 fails.

T1C Controller Bits 0, 1, 6 Reset

Issue INIT. Select and clear Controller. Issue COM3 02H (Abort Timer 1), COM3 08H (Abort Timer 2) Read Controller Status. Error if IFL bits 0 (Timer 1 Time-Out), 1 (Timer 2 Time-Out) and 6 (Interrupt 6) not reset.

T1D No Interrupt When 0, 1, 6 Reset

Reset Controller Status bits 0, 1, 6 as in Test T1C. Enable Interrupt 6. Error if Interrupt occurs.

T1E Timer 1: 1 sec. +/- 20%

Enable Interrupt 6. Start Timer 1 by COM3 01. Wait for time-out interrupt to occur. If no interrupt occurs after a nominal time, the following message is displayed:

```
TIMER OVER-RUN
```

If no error, the Controller status is read. There are 2 possible error messages:

```
TIME-OUT STATUS BIT DID NOT SET
INT6 STATUS BIT DID NOT SET
```

If no error, the elapsed time is checked to see if it fell between the acceptable boundaries. The 2 error messages possible are:

```
TIME GREATER THAN +20% SPEC.
TIME LESS THAN -20% SPEC.
```

If no error, a COM3 (Abort Timer) is issued. The 2 error messages possible are:

```
COM3 (ABORT) DID NOT RESET TIME-OUT
COM3 (ABORT) DID NOT RESET INT6
```

If no error, the timer is started again, and then a COM3 (Abort) is issued immediately thereafter. The error message is:

```
TIME-OUT SET AFTER COM3 (ABORT)
```

T1F Timer 2: 10sec +/- 20%

Same as Test T1E, but testing Timer 2.

T20 Ring Detect Reset

Select and clear Controller. Reset Master DTR by Reset Port B. Re-select Controller. Read Controller Status. Error if bit 3 (Ring Detect) not reset.

T21 Ring Detect Set

Reset bit 3 (Ring Detect) as in Test T20. Set Master DTR by Set Port B. Select Controller. Set Ring Detect Mode by OFL 18H. Read Controller Status. Error if bits 3 (Ring Detect) and 7 (INT7) not set, or if Test T20 fails.

T22 <u>No Interrupt: RD Reset, RD Mode Disabled</u>

Reset Ring Detect as in Test T20. Reset Ring Detect mode by OFL 20H. Enable Interrupt 7. Error if interrupt occurs, or if Test T20 fails.

T23 <u>Interrupt: RD Set, RD Mode Enabled</u>

Set Ring Detect as in Test T21. Set Ring Detect Mode by OFL 10H. Enable Interrupt 7. Error if Interrupt does not occur, or if Test T21 fails.

T24 <u>Reset Ring Detect Mode</u>

Set Ring Detect as in Test T21. Reset Ring Detect Mode by OFL 28H. Read Controller Status. Error if Ring Detect and INT7 not reset, or if Test T21 fails.

T25 <u>Set Operating Mode</u>

Select and clear Controller. Issue COM1 91H (Set Operating Mode), OPT 04H (Buffered Mode). Read 8273 status. Error if bit 7 (Command Busy) and bit 6 (Command Buffer Full) not reset after nominal delay.

T26 <u>Command Busy Set by Transmit Frame</u>

Set Buffered Mode. Issue COM1 C8H (Transmit Frame). Read 8273 status. Error if bit 7 (Command Busy) not set, bit 6 (Command Buffer Full) not reset, or if Test T25 fails.

T27 <u>Command Busy Reset by 4 Parameters</u>

Issue Transmit Frame as in Test T26. Issue 4 required parameters by OPT. Error if Command Busy reset before last parameter issued, or if not set after last parameter issued, or if Test T26 fails.

T28 <u>TXINT, TXIRA Set</u>

Select and clear Controller. Issue Transmit Frame. Read 8273 status. Error if bits 2 (Transmitter Interrupt) and 0 (Tx Interrupt Result Available) not set after nominal delay.

T29 <u>TXINT, TXIRA Reset</u>

Issue Transmit Frame as in Test T28. Read Transmit Interrupt Result Register by OFL 02H, IFL. Read 8273 status. Error if bits 0, 2 not reset, or if Test T28 fails.

T2A Controller Transmit Interrupt

Issue Transmit Frame as in Test T28. Enable Interrupt 7. Error if no interrupt occurs, or if Test T28 fails.

T2B Controller Bits 5, 7 Set

Issue Transmit Frame and enable Interrupt 7 as in Test T2A. Read Controller status. Error if bits 5 (Transmit Interrupt) and 7 (INT7) not set, or if Test T2A fails.

T2C Read Tx Interrupt Result

Issue Transmit Frame as in Test T28. Read Transmit Interrupt Result Register. Error if value not equal to 0DH (Frame Transmit Complete), or if Test T28 fails.

T2D Main Channel Current Address

Set Main Channel Current Address (MCC) equal to FFFFH. Issue Transmit Frame with buffer size of 1 character. Read MCC. Error if MCC not equal to (Main Channel Starting Address) +1.

T2E Main Channel Termination

Issue Transmit Frame as in Test T2D, but with buffer size of 20 characters. Error if MCC not equal to MCS+20.

T2F 8273 Bits 0, 2 Set by Termination

Issue Transmit Frame as in Test T2E. Read 8273 status. Error if bits 2 (Transmitter Interrupt) and 0 (Transmit Interrupt Result Available) not set, or if Test T2E fails.

T30 Tx Interrupt Result After Termination

Issue Transmit Frame as in Test T2F. Read Transmit Interrupt Result Register. Error if value not equal to 0DH (Frame Transmit Complete), or if Test T2F fails.

T31 Controller Bits 5, 7 Set by Termination

Issue Transmit Frame as in Test T2E. Read Controller status. Error if bits 5 (Transmit Interrupt) and 7 (INT7) not set, or if Test T2E fails.

T32 Abort Transmit Frame

Select and clear Controller. Issue Transmit Frame. Issue COM1 CCH (Abort Transmit Frame). Read 8273 status. Error if bits 0 (Transmit Interrupt Result Available) and 2 (Transmitter Interrupt) not set.

T33 Abort Transmit Interrupt Result

Issue Transmit Frame, Abort Transmit Frame as in Test T32. Read Transmit Interrupt Result Register. Error if value not equal to 10H (Abort Complete), or if Test T32 fails.

T34 On-Line Transmit - 1200 Baud

Select and clear Master. Issue General Receive. Select and clear Controller, Issue Transmit Frame. When Transmit Interrupt occurs, check received data. Error if received characters do not match transmitted characters.

T35 On-line Transmit - 2400 Baud

Same as Test T34, but testing 2400 baud.

T36 On-Line Transmit - 4800 Baud

Same as Test T34, but testing 4800 baud.

T37 On-Line Transmit - 9600 Baud

Same as Test T34, but testing 9600 baud.

T38 Command Busy Set by General Receive

Select and clear Controller. Issue COM1 C0H (General Receive). Read 8273 status. Error if bit 7 (Command Busy) not set, or bit 6 (Command Buffer Full) not reset.

T39 Command Busy Reset by 2 Parameters

Issue General Receive as in Test T38. Issue 2 required parameters by OPT. Error if Command Busy reset before last parameter issued, or if set after last parameter issued, or if Test T38 fails.

T3A Controller Bits 4, 6 Set

Select and clear Controller. Issue General Receive. Select and clear Master. Issue Transmit Frame and wait for Transmit Interrupt to occur. Re-select Controller and read status. Error if bits 4 (Receiver Interrupt) or 6 (INT6) not set.

T3B 8273 Bits 1, 3 Set

Receive Frame as in Test T3A. Read 8273 status. Error if bits 3 (Receiver Interrupt), or 1 (Receiver Interrupt Result Available) not set, or if Test T3A fails.

T3C Secondary Channel Current Address

Set Secondary Channel Current Address (SCC) equal to FFFFH. Select and clear Controller. Issue General Receive. Select and clear Master. Transmit Frame with buffer size of 1 character and wait for Transmit Interrupt to occur. Read, SCC. Error if SCC not equal to (Secondary Channel Starting Address) +1.

T3D Secondary Channel Termination

Receive Frame on-line as in Test T3C, but with buffer size of 40 characters. Error if SCC not equal to (SCS+40).

T3E On-Line General Receive - 1200 Baud

Select and clear Controller. Issue General Receive. Select and Clear Master. Issue Transmit Frame. When Transmit Interrupt occurs, check received data. Error if received characters do not match transmitted characters.

T3F On-Line General Receive - 2400 Baud

Same as Test T3E, but testing 2400 baud.

T40 On-Line General Receive - 4800 Baud

Same as Test T3E, but testing 4800 baud.

T41 On-Line General Receive - 9600 Baud

Same as Test T3E, but testing 9600 baud.

T42 General Receive Interrupt Result - A1 Match

Execute On-Line General Receive at 1200 baud as in Test T3E. Read Receiver Interrupt. Result byte by OFL 03H, IFL. Error if value not equal to E0H (A1 Match), or if Test T3E fails.

T43 Read 5 Receiver Interrupt Results - 1200 Baud

Exeucte 256 character on-line Transmit Frame. When Receiver Interrupt occurs, read 5 Results and save. Read 8273 status. Error if bits 3 (Receiver Interrupt) and 1 (Receiver Interrupt Result Available) not reset. If no error, compare 5 results with expected values:

1.	A1 Match (General Receive)
2.,3.	Data bytes received (low, high)
4.	Address Field (0FFH)
5.	Control Field (0FFH)

Error if any result does not match expected value.

T44 Read 5 Receiver Interrupt Results - 2400 Baud

Same as Test T43, but testing 2400 baud.

T45 Read 5 Receiver Interrupt Results - 4800 Baud

Same as Test T43, but testing 4800 baud.

T46 Read 5 Receiver Interrupt Results - 9600 Baud

Same as Test T43, but testing 9600 baud.

T47 Result No. 2, All 256 Values

With Baud rate set for 9600, execute an on-line Transmit Frame once for each of 256 possible values of Receiver Interrupt Result No. 2 (Received Data bytes - low byte). Error if Result does not match expected value.

T48 Result No. 4, All 256 Values

Same as Test T47, but testing Result No. 4, (Address Field).

T49 Result No. 5, All 256 Values

Same as Test T47, but testing Result No. 5, (Control Field).

T4A Receiver Result: Memory Buffer Overflow

Issue a General Receive to Controller with Receiver Buffer size parameter of 10H data bytes. Attempt to Transmit Frame from Master with Frame size of 100H data bytes. Wait for Receiver Interrupt to occur and then read Receiver Result. Error if value not equal to 09H (Memory Buffer Overflow).

T4B Receiver Termination by Buffer Size

Execute on-line Transmit Frame as in Test T4A to force Memory Buffer Overflow result. Read Controller SCC Address. Error if SCC is not equal to (SCS Address) + (Receiver Buffer size parameter), or if Test T4A fails.

T4C Receiver Results: Abort Detect, Idle Detect

Initiate a 256 character, on-line Transmit Frame at 9600 baud. When transmission has begun, issue to Master Abort Transmit Frame. Wait for Receiver Interrupt. Read results. Error if Receiver not disabled after 2 results are read. If no error, the 2 saved results are compared with the expected values:

1. 04H (Abort Detected)
2. 05H (Idle Detected)

T4D **Receiver Result: Interrupt Overrun**

Execute 256 character on-line Transmit Frame. When Reciever Interrupt occurs, read first result and then delay for a nominal time. Read result 5 more times. Error if last (6th) result read not equal to 0BH (Interrupt Overrun).

T4E **Command Busy Set by Selective Receive**

Select and clear Controller. Issue COM1 C1H (Selective Receive). Read 8273 status. Error if bit 7 (Command Busy) not set or if bit 6 (Command Buffer Full) not reset.

T4F **Command Busy Reset after 4 Parameters**

Issue Selective Receive as in Test T4E. Issue 4 required parameters by OPT. Error if Command Busy reset before last parameter issued or if set after last parameter issued, or if Test T4E fails.

T50 **8273 Bits 1, 3 Set by Selective Receive**

Attempt on-line Selective Receive. Transmit Frame from Master and wait for Transmit Interrupt. Select Controller and read 8273 status. Error if bits 3 (Receiver Interrupt) and 1 (Receiver Interrupt Result Available) not set.

T51 **Controller Bits 4, 6 Set by Selective Receive**

Attempt Selective Receive as in Test T50. Read Controller status. Error if bits 4 (Receiver Interrupt) and 6 (INT6) not set, or if Test T50 fails.

T52 **Selective Receive Data - 1200 Baud**

Execute on-line Transmit Frame of 256 data bytes at 1200 baud, with Transmit Address field and Receiver A1, A2 parameters set to 0FFH. When Receiver Interrupt occurs, check data in Receive buffer. Error if data does not match transmitted data.

T53 **Selective Receive Data - 2400 Baud**

Same as Test T52, but testing 2400 baud.

T54 **Selective Receive Data - 4800 Baud**

Same as Test T52, but testing 4800 baud.

T55 **Selective Receive Data - 9600 Baud**

Same as Test T52, but testing 9600 baud.

T56 <u>Selective Receive Interrupt Results - 1200 Baud</u>

Execute on-line Transmit Frame of 256 data bytes at 1200 baud, with Transmit Address Field and Receiver A1, A2 parameters set to 0FFH. When Receiver Interrupt occurs, read and save 5 Interrupt results. Read 8273 status. Error if bits 3 (Receiver Interrupt) and 1 (Receiver Interrupt Result Available) not reset. If no error, check Receiver Results. Error if results do not match expected values.

T57 <u>Selective Receive Interrupt Results - 2400 Baud</u>

Same as Test T56, but testing 2400 baud.

T58 <u>Selective Receive Interrupt Results - 4800 Baud</u>

Same as T56, but testing 4800 baud.

T59 <u>Selective Recieve Interrupt Results - 9600 Baud</u>

Same as Test T56, but testing 9600 Baud.

T5A <u>A1 Match, All 256 Values</u>

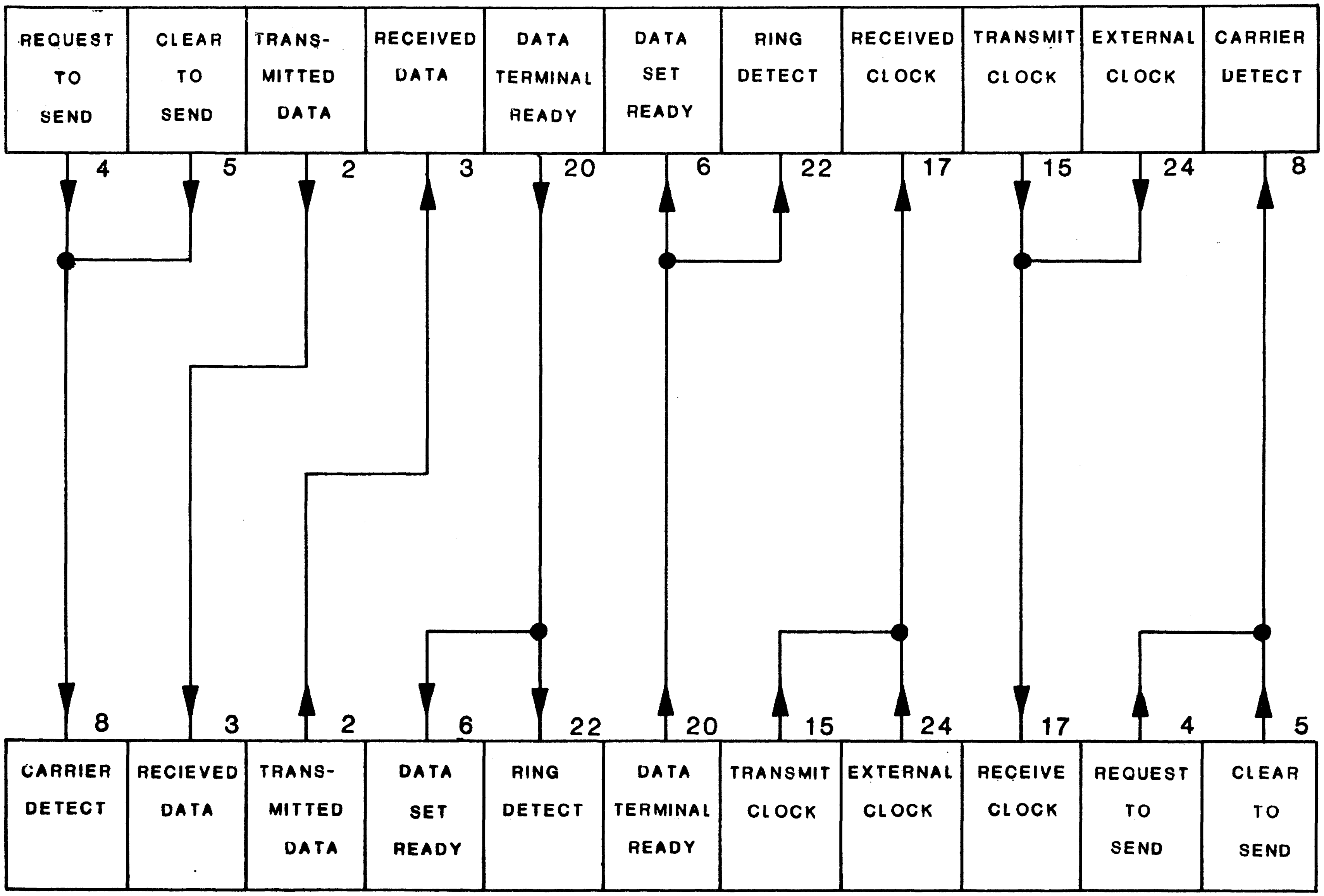
For all 256 possible hex values, execute on-line Transmit Frame of 10H data bytes at 9600 baud, with Transmit Address Field equal to A1 Receive parameter, and A1 not equal to A2. Read Receiver Interrupt results. Error if results do not match expected values.

T5B <u>A2 Match, All 256 Values</u>

Same as Test T5A, but testing A2 parameter Address match.

T5C <u>Selective Receive, No Address Match</u>

Attempt on-line Transmit Frame and Selective Receive of 1 data byte at 9600 baud. Set A1 equal to A2, and Transmit Address field not equal to either A1 or A2. Wait for Transmit Interrupt to occur. Read Receive 8273 status. Error if a Receiver Interrupt has occurred. Repeat for all 256 values of A1 and A2.



SDLC TEST CABLE

MINTST

MINI-DISKETTE SUBSYSTEM TEST

MINTST - MINI-DISKETTE SUBSYSTEM DIAGNOSTIC

Applicable Assemblies

5000-11118-1	Mini-diskette Controller
685-00100-001	Mini-diskette Drive

General Description

MINTST is a comprehensive test of the mini-diskette subsystem. The diagnostic tests one or two double density mini-diskette drives in a programmed I/O environment. It verifies the proper operation of the positioner and data transfer subassembles.

The diagnostic also incorporates: a format utility which allows the user to format and verify diskettes, a compatibility test which allows the user to verify the compatibility of two or more drives, an error log which logs test and drive statistics following an error and a manual command feature which allows the user to pass commands to the diagnostic and mini-diskette subsystem following an error.

The diagnostic assumes the CPU, the IOM and memory are functioning properly. MINTST requires at least 12K of memory to run.

Loading Procedure

MINTST can be loaded into memory using any standard loading procedure. If MINTST loads properly it will identify itself as indicated in Figure 1.

```
MINTST V??.??LP00 ERRS00 T00

FORMAT TRACKS (Y or N)..............Y
VERIFY FORMAT (Y or N)...............Y
COMPATIBILITY TESTS (Y or N).......N
SECTORS (0-31)...............................0-31
TRACKS TO BE TESTED (0-34).........0-34
DRIVES (0,1)....................................1
DEVICE NUMBER (1-4).....................3
SELECT ADDRESS............................55
```

Figure 1

Operator Action

The user can begin diagnostic testing by inserting a write enabled diskette in drive 1, typing RETURN and entering the desired run-mode information. The default parameters are the worst case parameters for drive 1 of a mini-diskette subsystem with standard device number and select address. Any or all of these parameters can be altered to suit the user's needs.

A limited number of editing commands are provided to facilitate altering the default parameters. These are:

Cursor up (C5 key) - Move the cursor to the beginning of the previous field.

Cursor down (C0 key) - Move the cursor to the beginning of the next field.

Cursor home (C2 key) - Move the cursor to the beginning of the 1st field.

Tab (Tab key) - Equivalent to the cursor down key.

Cursor right (C3 key) - Move the cursor to the next position within the current field.

Cursor left (C1 key) - Move the cursor to the previous position within the current field.

Character insert (C4 key) - Insert a space into the field at the current cursor position.

Character delete (C4 key, shifted) - Delete the character at the current cursor position.

User alternation of the eight default parameters enables testing of non-standard device assignments and the logical subdivision of the mini-diskette subsystem for fault detection and fault isolation. A detailed description of each parameter follows.

FORMAT TRACKS (Default - Yes) Format tracks 0-34 regardless of which tracks the user selected to be tested. If the user selected one or more sectors located on the left surface (sectors 16-31) both surfaces will be formatted. The diskette will only be formatted once, regardless of how many loops the diagnostic is run. After formatting the diskette the format utility changes the format parameter from Y to N. Any test except Test 4 will call the format utility if the user requested the diskette be formatted.

VERIFY FORMAT (Default - Yes) Verifies the format of tracks 0-34 regardless of which tracks the user selected to be tested. If the user selected one or more sectors located on the left surface (sectors 16-31) then both surfaces will be verified. The format will only be verified once, regardless of how many loops the diagnostic is run. After verifying the format the format utility changes the verify format parameter from Y to N. Any test except Test 4 will call the format utility and verify the format if user requested.

COMPATIBILITY TESTS (Default N) If requested the compatibility test option halts Tests 2 and 4 (regardless of the run mode option) immediately before each test attempts to recover a previously recorded data pattern. The diagnostic then prompts the user to exchange diskettes between drives to be tested for compatibility. After exchanging diskettes the user strikes the space bar and the diagnostic attempts to recover the data recorded by the alternate drive. The drives being tested for compatibility do not have to be part of a common system but the diagnostic must be run simultaneously on all systems whose drives are to be tested for compatibility.

SECTORS (Default 0-31) Specifies the sector(s) to be tested. If one or more sectors located on the left surface (sectors 16-31) is selected the diagnostic assumes the drive is double sided.

TRACKS TO BE TESTED (Default 0-34) Specifies the track(s) to be tested.

DRIVES (Default - 1) Specifies the drive(s) to be tested.

DEVICE NUMBER (Default - 3) Specifies the device number of the Mini-diskette Controller to be tested. In the OP-1 or the OP-1/70 the device numbers are slot dependent (refer to the OP-1 Reference Manual). In the OP-1/50 the device numbers are set by the header plugs. If uncertain check the header plugs to determine the device number.

SELECT ADDRESS The hardware select address of the Mini-diskette Controller to be tested.

After altering the default parameters the user must type RETURN. The diagnostic will clear the parameter menu and display the run-mode prompt. Legal responses to the run-mode prompt are, a hexadecimal test number, the R key, SPACE, RETURN or the S key. Refer to APPENDIX A of the 4K Diagnostics Manual for a detailed description of the run-mode options.

The diagnostic can be restarted at any time by typing the PROG key. Restarting the diagnostic initializes all test statistics, the error log and the test device.

Test Description

While the diagnostic is running each test displays test and drive statistics at line 5 of the display. The statistics displayed from left to right are, the diagnostic loop count, the total error count, the current operation or drive command, the drive number, the surface, the track number, the sector number, and the status byte contents. All numerical data are displayed in hexidecimal.

The sequence of tests is designed to test the least significant function first. Each successive test assumes the reliable operation of previously tested subassemblies. A detailed description of each test follows.

<u>T0</u>: Test 0 is an oscillating seek test. After restoring the heads to track 0 the test performs a series of oscillating seeks by decrementing the track offset from 34 to 0 (causing the heads to oscillate between tracks 0,34,1,33,2,32,3,31,4,30...). At the completion of this sequence of seeks the entire sequence is reversed. The test issues seeks to all tracks regardless of which tracks the user selected for testing. The test verifies that the carriage position can be offset by any legal offset, in either direction, without generating an error or striking the forward stop.

T1: Test 1 is a head positioning test. Tracks are randomly selected from the set of user selected tracks. The test issues a seek to the desired track. All the headers on the selected track, regardless of which sectors the user selected for testing, are then read. If the user selected one or more sectors from the left surface (sectors 16-31) all headers on both the left and right surfaces are read. After reading all headers on the selected track, the test checks that the actual and expected track, sector and pad fields are equal.

In addition to test and drive statistics the test displays the headers at lines 8-11. The display consists of four columns, each column contains four headers. The format of the header is track number, pad field, sector number, pad field and CRC character (the pad field is 00).

T2: Test 2 is a worst case data test. The test randomly selects tracks from the set of user selected tracks. The test then writes the worst case data pattern (hexadecimal B6 is the worst case data pattern for MFM recording technique) in each of the user selected sectors. Each sector is then checked for a valid CRC character. The sequence is repeated until the worst case data pattern has been recorded at all user selected tracks and sectors. The test recovers the worst case data pattern in the same manner it was recorded. After each read command the contents of the read and write buffer are compared to check they are equal.

If the user responded yes to the compatibility test option the test will halt and request that the user exchange diskettes between drives to be tested for compatibility prior to reading the diskette.

In addition to test and drive statistics the diagnostic displays the contents of the write buffer at lines 8-14 and the contents of the read buffer at lines 16-22 when appropriate.

T3: Test 3 is a random data test. The test randomly selects a sector from the set of user selected sectors. The test then randomly selects each of the tracks in the set of user selected tracks. A unique random data pattern is recorded in the selected sector at each of the randomly selected tracks. A read command is issued after each write command and the contents of the read and write buffers are compared to verify they are equal. This sequence is repeated until all tracks and sectors have been tested.

In addition to test and drive statistics the test displays the contents of the write buffer at lines 8-14 and the contents of the read buffer at lines 16-22 when appropriate.

T4: Test 4 is a read only test. The test randomly selects a sector from the set of user selected sectors. The test then randomly selects each of the tracks in the set of user selected tracks. The selected sector is read at each of the selected tracks. This sequence is repeated until all tracks and sectors have been read. Test 4 does not require a specific data pattern be recorded on the diskette and will read any properly formatted write enabled diskette.

A diskette cannot be read immediately after having been formatted because the format command does not generate a data mark. To prevent errors caused by the improper use of the diagnostic Test 4 does not call the format utility.

If the user responded yes to the compatibility test option the test will request the user exchange diskettes between drives to be tested for compatibility prior to starting.

In addition to test and drive statistics the test displays the contents of the read buffer at lines 16-22.

Errors

All errors are flagged by a one second audible tone. The current test and drive statistics are displayed at line 5 of the display. The contents of the status byte indicates the type of error detected for all operational errors (refer to the OP-1 or OP-1/70 Reference Manual for the status byte bit definitions). Some errors are accompanied by an error message. A detailed description of each error message follows.

HEADER ERROR: HEADER NOT FOUND FOR SELECTED SECTOR - Test 1 was unable to locate the first sector of the indicated surface and track (sector 0 or 16). The status byte is blank because this is not an operational error. All IFL flags were reset at the completion of the sequence of SEEK and READID commands but the desired header could not be found.

HEADER MATCH ERROR: ACT= EXP= - After a succession of successful READID commands Test 1 verifies that the actual and expected header data are equal. The error message is displayed with the actual and expected header data in the appropriate fields. The status byte is blank because this is not an operational error. All IFL flags were reset at the completion of the sequence of READID commands but the header data is in error.

DATA MATCH ERROR: ACT= EXP= - This message is generated by Tests 2 and 3. After reading a previously recorded data pattern the test determined that the read and write data are not equal. The read data is displayed as the actual data and the write data is displayed as the expected data. The status byte is blank because this is not an operational error. All IFL flags were reset at the completion of the read command but the read data is in error.

This error can be caused by a previous write error. Check the error log to determine whether or not the data pattern was successfully recorded at the indicated track and sector.

All errors and associated error messages are logged in the error log. The contents of the error log can be displayed by typing 'L'. The cursor up (C5 key) and the cursor down (C0 key) keys allow the user to scroll the display up or down. Typing cursor home (C2 key) returns the current test and drive statistics to the display screen. The error log is cleared whenever the diagnostic is restarted.

If the user did not elect the continous run-mode option (typing 'S' in response to the run-mode prompt selects continuous run-mode) the diagnostic will stop after an error. The user can pass commands to the failing drive, retry the operation, continue the test or restart the diagnostic. A detailed description of the commands follows.

S-RESTORE - Typing 'S' will pass a RESTORE command to the failing drive. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

F-FORMAT - Typing 'F' will pass a FORMAT command to the failing drive. The track at which the error occurred will be formatted. If the user selected one or more sectors located on the left surface (sectors 16-31) both surfaces will be formatted. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

The FORMAT command does not create a data mark and will cause Tests 2, 3 and 4 to fail is the test attempts to read a track the user just formatted. To prevent errors caused by the improper use of the diagnostic the user should not issue the FORMAT command if the diagnostic failed after a READ command.

V-VERIFY - Typing 'V' passes the VERIFY TRACK command to the failing drive. The format of the track at which the error occurred is verified. If the user selected one more sectors located on the left surface (sectors 16-31) both surfaces are verified. At the completion of the VERIFY command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

C-CHECK CRC - Typing 'C' passes a CHECK command to the failing drive. The CRC will be checked at the track and sector at which the error was detected. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

R-RETRY TEST - Typing 'R' causes the diagnostic to retry the sequence of events which caused the error. If the retry fails the diagnostic returns to the user with the current test statistics, drive statistics and any associated error messages. Unsuccessful retries are logged in the error log. If the retry is successful the diagnostic continues the test sequence.

SPACE-CONTINUE - Typing SPACE causes the diagnostic to continue the test sequence.

PROG-RESTART - Typing PROG initializes test statistics, the error log, the test device and restarts the diagnostic.

MINI96

MINI-DISKETTE SUBSYSTEM TEST

MINI96 - MINI-DISKETTE SUBSYSTEM DIAGNOSTIC

Applicable Assemblies

5000-11118-1	Mini-diskette Controller
685-00100-004	Mini-diskette Drive (96 TPI)

General Description

MINI96 is a comprehensive test of the mini-diskette subsystem. The diagnostic tests one or two quadruple density mini-diskette drives in a programmed I/O environment. It verifies the proper operation of the positioner and data transfer subassembles.

The diagnostic also incorporates: a format utility which allows the user to format and verify diskettes, a compatibility test which allows the user to verify the compatibility of two or more drives, an error log which logs test and drive statistics following an error and a manual command feature which allows the user to pass commands to the diagnostic and mini-diskette subsystem following an error.

The diagnostic assumes the CPU, the IOM and memory are functioning properly. MINI96 requires at least 12K of memory to run.

Loading Procedure

MINI96 can be loaded into memory using any standard loading procedure. If MINI96 loads properly it will identify itself as indicated in Figure 1.

```
MINI96 V??.??LP00 ERRS00 T00

FORMAT TRACKS (Y or N)..............Y
VERIFY FORMAT (Y or N)...............Y
COMPATIBILITY TESTS (Y or N).......N
SECTORS .......................................0-63
TRACKS TO BE TESTED ..............0-37
DRIVES (0,1)....................................1
DEVICE NUMBER (1-4).....................3
SELECT ADDRESS............................55
```

Figure 1

Operator Action

The user can begin diagnostic testing by inserting a write enabled diskette in drive 1, typing RETURN and entering the desired run-mode information. The default p-arameters are the worst case parameters for drive 1 of a mini-diskette subsystem with standard device number and select address. Any or all of these parameters can be altered to suit the user's needs.

A limited number of editing commands are provided to facilitate altering the default parameters. These are:

Cursor up (C5 key) - Move the cursor to the beginning of the previous field.

Cursor down (C0 key) - Move the cursor to the beginning of the next field.

Cursor home (C2 key) - Move the cursor to the beginning of the 1st field.

Tab (Tab key) - Equivalent to the cursor down key.

Cursor right (C3 key) - Move the cursor to the next position within the current field.

Cursor left (C1 key) - Move the cursor to the previous position within the current field.

Character insert (C4 key) - Insert a space into the field at the current cursor position.

Character delete (C4 key, shifted) - Delete the character at the current cursor position.

User alteration of the eight default parameters enables testing of non-standard device assignments and the logical subdivision of the mini-diskette subsystem for fault detection and fault isolation. A detailed description of each parameter follows.

FORMAT TRACKS (Default - Yes) Format tracks 0-37 regardless of which tracks the user selected to be tested. The diskette will only be formatted once, regardless of how many loops the diagnostic is run. After formatting the diskette the format utility changes the format parameter from Y to N. Any test except Test 4 will call the format utility if the user requested the diskette be formatted.

VERIFY FORMAT (Default - Yes) Verifies the format of tracks 0-37 regardless of which tracks the user selected to be tested. The format will only be verified once, regardless of how many loops the diagnostic is run. After verifying the format the format utility changes the verify format parameter from Y to N. Any test except Test 4 will call the format utility and verify the format if user requested.

COMPATIBILITY TESTS (Default N) If requested the compatibility test option halts Tests 2 and 4 (regardless of the run mode option) immediately before each test attempts to recover a previously recorded data pattern. The diagnostic then prompts the user to exchange diskettes between drives to be tested for compatibility. After exchanging diskettes the user strikes the space bar and the diagnostic attempts to recover the data recorded by the alternate drive. The drives being tested for compatibility do not have to be part of a common system but the diagnostic must be run simultaneously on all systems whose drives are to be tested for compatibility.

SECTORS (Default 0-63) Specifies the sector(s) to be tested.

TRACKS TO BE TESTED (Default 0-37) Specifies the track(s) to be tested.

DRIVES (Default - 1) Specifies the drive(s) to be tested.

DEVICE NUMBER (Default - 3) Specifies the device number of the Mini-diskette Controller to be tested. In the OP-1 or the OP-1/70 the device numbers are slot dependent (refer to the OP-1 Reference Manual). In the OP-1/50 the device numbers are set by the header plugs. If uncertain check the header plugs to determine the device number.

SELECT ADDRESS (Default - 55) The hardware select address of the Mini-diskette Controller to be tested.

After altering the default parameters the user must type RETURN. The diagnostic will clear the parameter menu and display the run-mode prompt. Legal responses to the run-mode prompt are, a hexadecimal test number, the R key, SPACE, RETURN or the S key. Refer to APPENDIX A of the 4K Diagnostics Manual for a detailed description of the run-mode options.

The diagnostic can be restarted at any time by typing the PROG key. Restarting the diagnostic initializes all test statistics, the error log and the test device.

Test Description

While the diagnostic is running each test displays test and drive statistics at line 5 of the display. The statistics displayed from left to right are, the diagnostic loop count, the total error count, the current operation or drive command, the drive number, the surface, the track number, the sector number, and the status byte contents. All numerical data are displayed in hexidecimal.

The sequence of tests is designed to test the least significant function first. Each successive test assumes the reliable operation of previously tested subassemblies. A detailed description of each test follows.

T0: Test 0 is an oscillating seek test. After restoring the heads to track 0 the test performs a series of oscillating seeks by decrementing the track offset from 37 to 0 (causing the heads to oscillate between tracks 0,37,1,36,2,35,3,34,4,33...). At the completion of this sequence of seeks the entire sequence is reversed. The test issues seeks to all tracks regardless of which tracks the user selected for testing. The test verifies that the carriage position can be offset by any legal offset, in either direction, without generating an error or striking the forward stop.

T1: Test 1 is a head positioning test. Tracks are randomly selected from the set of user selected tracks. The test issues a seek to the desired track. All the headers on the selected track, regardless of which sectors the user selected for testing, are then read. After reading all headers on the selected track, the test checks that the actual and expected track, sector and pad fields are equal.

In addition to test and drive statistics the test displays the headers at lines 8-11. The display consists of four columns, each column contains four headers. The format of the header is track number, pad field, sector number, pad field and CRC character (the pad field is 00).

T2: Test 2 is a worst case data test. The test randomly selects tracks from the set of user selected tracks. The test then writes the worst case data pattern (hexadecimal B6 is the worst case data pattern for MFM recording technique) in each of the user selected sectors. Each sector is then checked for a valid CRC character. The sequence is repeated until the worst case data pattern has been recorded at all user selected tracks and sectors. The test recovers the worst case data pattern in the same manner it was recorded. After each read command the contents of the read and write buffer are compared to check they are equal.

If the user responded yes to the compatibility test option the test will halt and request that the user exchange diskettes between drives to be tested for compatibility prior to reading the diskette.

In addition to test and drive statistics the diagnostic displays the contents of the write buffer at lines 8-14 and the contents of the read buffer at lines 16-22 when appropriate.

T3: Test 3 is a random data test. The test randomly selects a sector from the set of user selected sectors. The test then randomly selects each of the tracks in the set of user selected tracks. A unique random data pattern is recorded in the selected sector at each of the randomly selected tracks. A read command is issued after each write command and the contents of the read and write buffers are compared to verify they are equal. This sequence is repeated until all tracks and sectors have been tested.

In addition to test and drive statistics the test displays the contents of the write buffer at lines 8-14 and the contents of the read buffer at lines 16-22 when appropriate.

T4: Test 4 is a read only test. The test randomly selects a sector from the set of user selected sectors. The test then randomly selects each of the tracks in the set of user selected tracks. The selected sector is read at each of the selected tracks. This sequence is repeated until all tracks and sectors have been read. Test 4 does not require a specific data pattern be recorded on the diskette and will read any properly formatted write enabled diskette.

A diskette cannot be read immediately after having been formatted because the format command does not generate a data mark. To prevent errors caused by the improper use of the diagnostic Test 4 does not call the format utility.

If the user responded yes to the compatibility test option the test will request the user exchange diskettes between drives to be tested for compatibility prior to starting.

In addition to test and drive statistics the test displays the contents of the read buffer at lines 16-22.

Errors

All errors are flagged by a one second audible tone. The current test and drive statistics are displayed at line 5 of the display. The contents of the status byte indicates the type of error detected for all operational errors (refer to the OP-1 or OP-1/70 Reference Manual for the status byte bit definitions). Some errors are accompanied by an error message. A detailed description of each error message follows.

HEADER ERROR: HEADER NOT FOUND FOR SELECTED SECTOR - Test 1 was unable to locate the first sector of the indicated surface and track (sector 0 or 16). The status byte is blank because this is not an operational error. All IFL flags were reset at the completion of the sequence of SEEK and READID commands but the desired header could not be found.

HEADER MATCH ERROR: ACT= EXP= - After a succession of successful READID commands Test 1 verifies that the actual and expected header data are equal. The error message is displayed with the actual and expected header data in the appropriate fields. The status byte is blank because this is not an operational error. All IFL flags were reset at the completion of the sequence of READID commands but the header data is in error.

DATA MATCH ERROR: ACT= EXP= - This message is generated by Tests 2 and 3. After reading a previously recorded data pattern the test determined that the read and write data are not equal. The read data is displayed as the actual data and the write data is displayed as the expected data. The status byte is blank because this is not an operational error. All IFL flags were reset at the completion of the read command but the read data is in error.

This error can be caused by a previous write error. Check the error log to determine whether or not the data pattern was successfully recorded at the indicated track and sector.

All errors and associated error messages are logged in the error log. The contents of the error log can be displayed by typing 'L'. The cursor up (C5 key) and the cursor down (C0 key) keys allow the user to scroll the display up or down. Typing cursor home (C2 key) returns the current test and drive statistics to the display screen. The error log is cleared whenever the diagnostic is restarted.

If the user did not elect the continous run-mode option (typing 'S' in response to the run-mode prompt selects continuous run-mode) the diagnostic will stop after an error. The user can pass commands to the failing drive, retry the operation, continue the test or restart the diagnostic. A detailed description of the commands follows.

S-RESTORE - Typing 'S' will pass a RESTORE command to the failing drive. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

F-FORMAT - Typing 'F' will pass a FORMAT command to the failing drive. The track at which the error occurred will be formatted. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

The FORMAT command does not create a data mark and will cause Tests 2, 3 and 4 to fail if the test attempts to read a track the user just formatted. To prevent errors caused by the improper use of the diagnostic the user should not issue the FORMAT command if the diagnostic failed after a READ command.

V-VERIFY - Typing 'V' passes the VERIFY TRACK command to the failing drive. The format of the track at which the error occurred is verified. At the completion of the VERIFY command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

C-CHECK CRC - Typing 'C' passes a CHECK command to the failing drive. The CRC will be checked at the track and sector at which the error was detected. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

R-RETRY TEST - Typing 'R' causes the diagnostic to retry the sequence of events which caused the error. If the retry fails the diagnostic returns to the user with the current test statistics, drive statistics and any associated error messages. Unsuccessful retries are logged in the error log. If the retry is successful the diagnostic continues the test sequence.

SPACE-CONTINUE - Typing SPACE causes the diagnostic to continue the test sequence.

PROG-RESTART - Typing PROG initializes test statistics, the error log, the test device and restarts the diagnostic.

MPDCTST

MICRO-PROGRAMMABLE DISKETTE CONTROLLER TEST

MPDCTST - MICRO PROGRAMMABLE DISKETTE DIAGNOSTIC

Applicable Assemblies

5000-11110-2	MPDC Controller
5000-11110-3	MPDC Controller
5000-4033	Shugart single sided Diskette Drive
5000-4062	CDC dual sided Diskette Drive

General Description

MPDCTST is a comprehensive test of the double density, double sided diskette subsystem. The diagnostic tests one or two double density diskette drives in a programmed I/O environment. It verifies the proper operation of the positioner and data transfer subassemblies. It operates in any of Double density, Single density or IBM controller modes.

The diagnostic also incorporates: a format utility which allows the user to format and verify diskettes, a compatibility test which allows the user to verify the compatibility of two or more drives, an error log which logs test and drive statistics following an error and a manual command feature which allows the user to pass commands to the diagnostic and diskette subsystem following an error.

The diagnostic assumes the CPU, the IOM and memory are functioning properly. MPDCTST requires at least 12K of memory to run.

Loading Procedure

MPDCTST can be loaded into memory using any standard loading procedure. If MPDCTST loads properly it will identify itself as indicated in Figure 1.

```
MPDCTST V??.??LP00 ERRS00 T00

MODE (S, D, or I)............................D
FORMAT TRACKS (Y or N)..............Y
VERIFY FORMAT (Y or N)...............Y
COMPATIBILITY TESTS (Y or N).......N
SECTORS (0-63)................................0-63
TRACKS TO BE TESTED (0-76).........0-76
DRIVES (0,1,2,3)................................1
DEVICE NUMBER (1-4)......................3
SELECT ADDRESS............................59
```

Figure 1

Operator Action

The user can begin diagnostic testing by inserting a write enabled diskette in drive 1, typing RETURN and entering the desired run-mode information. The default parameters are the worst case parameters for drive 1 of a double density diskette subsystem with standard device number and select address. Any or all of these parameters can be altered to suit the user's needs.

A limited numbering of editing commands are provided to facilitate altering the default parameters. These are:

<u>Cursor up</u> (C5 key) - Move the cursor to the beginning of the previous field.

<u>Cursor down</u> (C0 key) - Move the cursor to the beginning of the next field.

<u>Cursor home</u> (C2 key) - Move the cursor to the beginning of the 1st field.

<u>Tab</u> (Tab key) - Equivalent to the cursor down key.

<u>Cursor right</u> (C3 key) - Move the cursor to the next position within the current field.

<u>Cursor left</u> (C1 key) - Move the cursor to the previous position within the current field.

<u>Character insert</u> (C4 key) - Insert a space into the field at the current cursor position.

<u>Character delete</u> (C4 key, shifted) - Delete the character at the current cursor position.

User alteration of the eight default parameters enables testing of non-standard device assignments and the logical subdivision of the double density diskette subsystem for fault detection and fault isolation. A detailed description of each parameter follows.

MODE (Default - Double density) Sets controller mode to any of Double density, Single density, or IBM mode.

FORMAT TRACKS (Default - Yes) Formats tracks 0-76 regardless of which tracks the user selected to be tested. If the user selected one or more sectors located on the left surface (sectors 32-63) both surfaces will be formatted. The diskette will only be formatted once, regardless of how many loops the diagnostic is run. After formatting the diskette, the format utility changes the format parameter from Y to N. Any test except Test 3 will call the format utility if the user requested the diskette be formatted. User must specify N if Single density mode selected.

VERIFY FORMAT (Default - Yes) Verifies the format of tracks 0-76 regardless of which tracks the user selected to be tested. If the user selected one or more sectors located on the left surface (sectors 32-63) then both surfaces will be verified. The format will only be verified once, regardless of how many loops the diagnostic is run. After verifying the format, the format utility changes the verify format parameter from Y to N. Any test except Test 3 will call the format utility and verify the format if user requested. User must specify N if Single density mode selected.

COMPATIBILITY TESTS (Default N) If requested the compatibility test option halts Tests 1 and 3 (regardless of the run-mode option) immediately before each test attempts to recover a previously recorded data pattern. The diagnostic then prompts the user to exchange diskettes between drives to be tested for compatibility. After exchanging diskettes the user strikes the space bar and the diagnostic attempts to recover the data recorded by the alternate drive. The drives being tested for compatibility do not have to be part of a common system but the diagnostic must be run simultaneously on all systems whose drives are to be tested for compatibility.

SECTORS (Default 0-63) Specifies the sector(s) to be tested. If one or more sectors located on the left surface (sectors 32-63) are selected the diagnostic assumes the drive is double sided. Sectors in Single density mode run from 0-15 mode 1-26.

TRACKS TO BE TESTED (Default 0-76) Specifies the track(s) to be tested.

DRIVES (Default - 1) Specifies the drive(s) to be tested.

DEVICE NUMBER (Default - 3) Specifies the device number of the MPDC Controller to be tested. In the OP-1 or the OP-1/70 the device numbers are slot dependent (refer to the OP-1 Reference Manual). In the OP-1/50 the device numbers are set by the header plugs. If uncertain check the header plugs to determine the device number.

SELECT ADDRESS (Default - 59) The hardware select address of the MPDC Controller to be tested.

After altering the default parameters the user must type RETURN. The diagnostic will clear the parameter menu and display the run-mode prompt. Legal responses to the run-mode prompt are, a hexadecimal test number, the R key, SPACE, RETURN or the S key. Refer to APPENDIX A of the 4K Diagnostics Manual for a detailed description of the run-mode options.

The diagnostic can be restarted at any time by typing the PROG key. Restarting the diagnostic initializes all test statistics, the error log and the test device.

Test Description

While the diagnostic is running each test displays test and drive statistics at line 5 of the display. The statistics displayed from left to right are, the diagnostic loop count, the total error count, the test number, the drive command, the drive number, the surface, the track number, the sector number, and the status byte contents. All numerical data are displayed in hexadecimal.

The sequence of tests is designed to test the least significant function first. Each successive test assumes the reliable operation of previously tested subassemblies. A detailed description of each test follows.

T0: (Double Density and IBM only). Test 0 is a head positioning test. After issuing a RESTORE command the test reads all headers on track 0. If the user selected one or more sectors on the left surface (sectors 32-63) all headers on both the left and right surfaces are read. After reading all headers on track zero the test checks that the actual and expected track, sector and pad fields are equal. The test verifies that the heads are properly positioned after a RESTORE command and all headers on track 0 can be read.

If an error is detected the test displays the headers at lines 8-11. The display consists of four columns, each column contains four headers. The format of the header is track number, pad field, sector number, pad field and CRC character (the pad field is 00).

T1: (Double Density only). Test 1 is a worst case data test. The test randomly selects a track from the set of user selected tracks. The test then writes the worst case data pattern (hexadecimal B6 is the worst case data pattern for MFM recording technique) in each of the user selected sectors. The sequence is repeated until the worst case data pattern has been recorded at all user selected tracks and sectors. The test recovers the worst case data pattern in the same manner it was recorded. After each read command the contents of the read and write buffers are compared to check that they are equal.

If the user responded yes to the compatibility test option the test will halt and request that the user exchange diskettes between drives to be tested for compatibility prior to reading the diskette.

If the test detects an error it displays the contents of the write buffer at lines 8-14 and the contents of the read buffer at lines 16-22 when appropriate.

T2: Test 2 is a random data test. The test randomly selects a sector from the set of user selected sectors. The test then randomly selects each of the tracks in the set of user selected tracks. A unique random data pattern is recorded in the selected sector at each of the randomly selected tracks. A read command is issued after each write command and the contents of the read and write buffers are compared to verify they are equal. This sequence is repeated until all tracks and sectors have been tested.

If the test detects an error it displays the contents of the write buffer at lines 8-14 and the contents of the read buffer at lines 16-22 when appropriate.

T3: Test 3 is a read only test. The test randomly selects a sector from the set of user selected sectors. The test then randomly selects each of the tracks in the set of user selected tracks. The selected sector is read at each of the selected tracks. This sequence is repeated until all tracks and sectors have been read. Test 3 does not require a specific data pattern be recorded on the diskette and will read any properly formatted write enabled diskette.

A diskette cannot be read immediately after having been formatted because the format command does not generate a data mark. To prevent errors caused by the improper use of the diagnostic Test 4 does not call the format utility.

If the user responded yes to the compatibility test option the test will request the user exchange diskettes between drives to be tested for compatiblity prior to starting.

If the test detected an error it displays the contents of the read buffer at lines 16-22.

Errors

All errors are flagged by a one second audible tone. The current test and drive statistics are displayed at line 5 of the display. The contents of the status byte indicates the type of error detected for all operational errors (refer to the OP-1 or OP-1/70 Reference Manual for the status byte bit definitions). Some errors are accompanied by an error message. A detailed description of each error message follows.

HEADER ERROR: HEADER NOT FOUND FOR SELECTED SECTOR - Test 0 was unable to locate the first sector of the left or right surface at track 0 (sector 0 or 32). This is not an operational error. All IFL flags were reset at the completion of the sequence of READID commands but the desired header could not be found.

HEADER MATCH ERROR: ACT= EXP= - After a succession of successful READID commands Test 0 verifies that the actual and expected header data are equal. The error message is displayed with the actual and expected header data in the appropriate fields. This is not an operational error. All IFL flags were reset at the completion of the sequence of READID commands but the header data is in error.

DATA MATCH ERROR: ACT= EXP= - This message is generated by Test 1 and Test 2. After reading a previously recorded data pattern the test determined that the read and write data are not equal. The read data is displayed as the actual data and the write data is displayed as the expected data. This is not an operational error. All IFL flags were reset at the completion of the read command but the read data is in error.

This error can be caused by a previous write error. Check the error log to determine whether or not the data pattern was successfully recorded at the indicated track and sector.

All errors and associated error messages are logged in the error log. The contents of the error log can be displayed by typing 'L'. The cursor up (C5 key) and the cursor down (C0 key) keys to allow the user to scroll the display up or down. Typing cursor home (C2 key) returns the current test and drive statistics to the display screen. The error log is cleared whenever the diagnostic is restarted.

If the user did not elect the continuous run-mode option (typing 'S' in response to the run-mode prompt selects continuous run-mode) the diagnostic will stop after an error. The user can pass commands to the failing drive, retry the operation, continue the test or restart the diagnostic. A detailed description of the commands follows.

S-RESTORE - Typing 'S' will pass a RESTORE command to the failing drive. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

F-FORMAT - (Double Density and IBM only). Typing 'F' will pass a FORMAT command to the failing drive. The track at which the error occurred will be formatted. If the user selected one or more sectors located on the left surface (sectors 32-63) both surfaces will be formatted. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

The FORMAT command does not create a data mark and will cause Tests 1 and 3 to fail if the test attempts to read a track the user just formatted. To prevent errors caused by the improper use of the diagnostic the user should not issue the FORMAT command if the diagnostic failed after a READ command.

V-VERIFY - (Double Density and IBM only). Typing 'V' passes the VERIFY TRACK command to the failing drive. The format of the track at which the error occurred is verified. If the user selected one or more sectors located on the left surface (sectors 32-63) both surfaces are verified. At the completion of the VERIFY command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

C-CHECK CRC - Typing 'C' passes a CHECK command to the failing drive. The CRC will be checked at the track and sector at which the error was detected. At the completion of the command the contents of the status byte is displayed but neither test nor drive statistics are entered in the error log.

R-RETRY TEST - Typing 'R' causes the diagnostic to retry the sequence of events which caused the error. If the retry fails the diagnostic returns to the user with the current test statistics, drive statistics, and any associated error messages. Unsuccessful retries are logged in the error log. If the retry is successful the diagnostic continues the test sequence.

SPACE-CONTINUE - Typing SPACE causes the diagnostic to continue the test sequence.

PROG-RESTART - Typing PROG initializes test statistics, the error log, the test device and restarts the diagnostic.

PDCIFL

SUB-ASSEMBLY PROGRAMMABLE DISKETTE CONTROLLER TEST

PDCIFL - MPDC2, MPDC3 & MINI-DISKETTE CONTROLLER TEST

Applicable Assemblies

5000-11110	MPDC2 Controller
5000-11136	MPDC3 Controller
5000-11118	Mini-Diskette Controller
5000-4033	Shugart Diskette Drive
5000-4062	CDC Diskette Drive
685-00100-001	Mini-Diskette Drive (48 TPI)
685-00100-004	Mini-Diskette Drive (96 TPI)

General Description

PDCIFL is a comprehensive test of the MPDC2, MPDC3 and Mini-Diskette Controller. The diagnostic tests the controller in a programmed I/O and interrupt driven environment. At least one diskette drive must be connected to the controller. Although the diagnostic is not designed to test the diskette drives, certain diskette drive functions are implicitly tested.

The diagnostic consists of a sequence of tests. Each test consists of one or more subtests. The user can run the entire test sequence, an individual test or an individual subtest. The user can logically divide the diagnostic for fault detection and fault isolation.

The diagnostic also incorporates a mini-diskette alignment test. This test can only be used during a mini-diskette alignment procedure to align or verify the alignment of a mini-diskette drive.

The diagnostic assumes the CPU, the IOM and memory are functioning properly. PDCIFL requires at least 16K of memory to run.

Loading Procedure

PDCIFL can be loaded into memory using any standard loading procedure. If PDCIFL loads properly it will identify itself as indicated in Figure 1.

```
PDCIFL V??.??LP00 ERRS00 T00

CNT'LR RESPONSE TIME LIMIT (1 - 999)        030
MANUAL INTERVENTION TESTS (Y or N)          Y
CONTROLLER TYPE (MDC, MPDC2 or MPDC3)       3
MODE (Single, Ibm, Double, Quadruple)       D
ALIGNMENT TEST (Y or N)                     N
DOUBLE SIDED (Y or N)                       N
DRIVE (0,1,2 or 3)                          1
DEVICE NUMBER (1 - 4)                       3
SELECT ADDRESS                              59
```

Figure -1-

Operator Action

The default parameters tests an MPDC3 in double density mode. The default parameters require that drive 1 is available and the MPDC3 is configured with standard device assignments. To begin testing using the default parameters insert a formatted, write enabled diskette in drive 1, type RETURN and enter the desired run-mode information.

PDCIFL - MPDC2, MPDC3 & MINI-DISKETTE CONTROLLER TEST

Applicable Assemblies

5000-11110	MPDC2 Controller
5000-11136	MPDC3 Controller
5000-11118	Mini-Diskette Controller
5000-4033	Shugart Diskette Drive
5000-4062	CDC Diskette Drive
685-00100-001	Mini-Diskette Drive

General Description

PDCIFL is a comprehensive test of the MPDC2, MPDC3 and Mini-Diskette Controller. The diagnostic tests the controller in a programmed I/O and interrupt driven environment. At least one diskette drive must be connected to the controller. Although the diagnostic is not designed to test the diskette drives, certain diskette drive functions are implicitly tested.

The diagnostic consists of a sequence of tests. Each test consists of one or more subtests. The user can run the entire test sequence, an individual test or an individual subtest. The user can logically divide the diagnostic for fault detection and fault isolation.

The diagnostic also incorporates a mini-diskette alignment test. This test can only be used during a mini-diskette alignment procedure to align or verify the alignment of a mini-diskette drive.

The diagnostic assumes the CPU, the IOM and memory are functioning properly. PDCIFL requires at least 16K of memory to run.

Loading Procedure

PDCIFL can be loaded into memory using any standard loading procedure. If PDCIFL loads properly it will identify itself as indicated in Figure 1.

```
PDCIFL V??.??LP00 ERRS00 T00

CNT'LR RESPONSE TIME LIMIT (1 - 999)          030
MANUAL INTERVENTION TESTS (Y or N)            Y
CONTROLLER TYPE (MDC, MPDC2 or MPDC3)         3
MODE (Single, Ibm, or Double Density)         D
ALIGNMENT TEST (Y or N)                       N
DOUBLE SIDED (Y or N)                         N
DRIVE (0,1,2 or 3)                            1
DEVICE NUMBER (1 - 4)                         3
SELECT ADDRESS                                59
```

Figure -1-

Operator Action

The default parameters tests an MPDC3 in double density mode. The default parameters require that drive 1 is available and the MPDC3 is configured with standard device assignments. To begin testing using the default parameters insert a formatted, write enabled diskette in drive 1, type RETURN and enter the desired run-mode information.

User alteration of the default parameters allows the user to test any configuration of the three applicable controllers. A limited number of editing commands are provided to facilitate altering the default parameters. These are:

Cursor Up (C5 key) - Move the cursor to the beginning of the previous field.

Cursor Down (C0 key) - Move the cursor to the beginning of the next field.

Cursor Right (C3 key) - Move the cursor to the next position within the current field.

Cursor Left (C1 key) - Move the cursor to the previous position within the current field.

By altering the default parameters the user can test any configuration of the MDC, MPDC2 and MDPC3 or the user can call the mini-diskette alignment test. A detailed description of each parameter follows.

CNT'LR RESPONSE TIME LIMIT (Default - 030) If the controller does not respond within 30 seconds the test times out. The controller response time limit applies to every test in the test sequence and the mini-diskette alignment test. The controller response timer is driven by the real time clock and assumes a 1 second time interval. If the user's real time clock generates a non-standard time interval the controller response time limit should be adjusted appropriately.

MANUAL INTERVENTION TESTS (Default - Yes) The manual intervention test option determines whether or not those tests which require user intervention will be run. This parameter does not apply to the mini-diskette alignment test.

CONTROLLER TYPE (Default - MPDC3) Specifies the type of controller to be tested. The Mini-Diskette alignment tests can only be run on an MDC.

MODE (Default - Double Density) Specifies the mode in which the controller is to be tested. The mini-diskette alignment test must be run in double density mode.

ALIGNMENT TEST (Default - No) If requested the alignment test option will call the alignment test. This test can only be used to align or verify the alignment of a mini-diskette drive. The diagnostic menu default parameters must be altered to reflect the configuration of the mini-diskette subsystem to be tested. The PDCIFL diagnostic test sequence cannot be entered from the alignment test sequence. To enter the diagnostic test sequence the user must return to the diagnostic menu and respond "N" to the alignment test prompt.

DOUBLE SIDED (Default - No) Specifies whether the attached drive is single or double sided.

DRIVE (Default - 1) Specifies the drive to be used by the test.

DEVICE NUMBER (Default - 3) Specifies the device number of the controller to be tested.

SELECT ADDRESS (Default - 59) Specifies the hardware select address of the controller to be tested. The default is the standard select address for an MPDC2 or MPDC3. If the MDC is to be tested the select address must be changed. The standard select address for the MDC is 55.

After altering the default parameters the user must insert a formatted, write enabled diskette in the test drive, type RETURN and enter the desired run-mode information. If the user responded "Y" to the alignment test prompt the user must insert an alignment diskette in the test drive, type RETURN and enter the desired run-mode information.

The user can enter a run-mode option, a test number or a subtest in response to the run-mode prompt. The run-mode options determine how the diagnostic will be run and how it will respond to errors. The test option facilitates the logical division of the diagnostic.

The user enters a test number by typing a hexadecimal test number. An invalid test number is flagged with an audible alarm (beep). Each test is divided into one or more subtests. A subtest is the smallest possible subdivision of a particular test. A subtest allows the user to execute one command or a sequence of related commands while troubleshooting the test device. The user enters the subtest by typing a shifted alphabetic. Subtest "a" is the only valid subtest entry in response to the run-mode prompt. An invalid subtest is flagged with an audible alarm.

The test number and subtest is displayed in a three character field. The test number is displayed as a two digit hexadecimal number. The subtest is displayed as a lower case alphabetic. If the user enters a test number without entering a subtest the continue and repeat run-mode options apply to the entire test. If the user enters a test number and subtest the continue and repeat run-mode options apply to the subtest.

The user can also enter one of five run-mode options. The five run-mode options are described below.

Loop continuously ("S" key) - Run the test sequence continuously. Each test and subtest is run in sequence. The loop count and error count statistics are appropriately updated but the diagnostic does not stop after an error or at the end of a loop.

Loop continuously until an error is encountered ("N" key) - This run-mode option is identical to the loop continuously run-mode option except the diagnostic will stop after an error.

Run one loop (RETURN key) - Run one loop of the test sequence. Each test and subtest is run in sequence. The loop count and error count statistics are appropriately updated. The diagnostic stops after an error or after one loop.

Run one test or subtest (SPACE) key - Run the current test or subtest. If the user entered a test number without entering a subtest the current test will be run. If the user entered a test number and subtest the current subtest will be run. The diagnostic stops after the test or subtest is executed or an error is encountered. The loop count and error count statistics are updated appropriately.

Repeat one test or subtest ("R" key) - Repeat the current test or subtest. If the user entered a test number without entering a subtest the current test will be repeated. If the user entered a test number and subtest the current subtest will be repeated. The diagnostic stops after the test or subtest is executed or an error is encountered. The loop count and error count statistics are updated appropriately.

If the user did not select the "loop continuously" run-mode option the diagnostic will stop after an error, one loop, one test or one subtest depending on the run-mode. The diagnostic stops and displays the continue prompt. The legal responses to the continue prompt are a hexadecimal test number, a subtest, the SPACE key or the "R" key.

After the diagnostic stops the user can enter any valid test number. The user can only enter subtest "a" or the current subtest. This prevents the user from running a subtest without running all prerequisite subtests.

Typing SPACE will cause the next subtest to be run. The next subtest will be run regardless of whether or not the previous subtest failed. This facility is provided for troubleshooting purposes. The next subtest may fail only because the previous subtest failed. If the user changed the test number or subtest, the new test or subtest will be run. The diagnostic will continue running depending upon the previously selected run-mode option.

Typing "R" will cause the test or subtest to be repeated. If the user entered a test number without entering a subtest the entire test will be repeated. If the user entered a test nunber and subtest, the subtest will be repeated. If the user changed the test number or subtest the new test or subtest will be repeated.

The diagnostic can be restarted by typing the PROG key. Restarting the diagnostic initializes the loop count and error count.

Test Description

PDCIFL contains a diagnostic test sequence and a mini-diskette alignment test sequence. The PDCIFL diagnostic test sequence is described first.

The diagnostic test sequence is designed to test the least significant function first. Each successive test assumed the reliable operation of previously tested functions.

TEST 00-02 - CONTROLLER ADDRESS TESTS

T00a Test that no I/O device responds to an invalid select address. If the test fails it displays "A CNT'LR ACKNOWLEDGED AN INVALID SEL ADR" followed by the invalid select address.

T01a Test that the controller responds to its select address. An error is flagged with an audible alarm.

T02a Test that no device remains selected after an INIT. If the test fails it displays the actual and expected device status.

TESTS 03-07 - CONTROLLER NOT BSY AND NOT RDY STATUS TESTS

T03a Test that the NOT BSY flag is reset during device initialization. This test only applies to the MPDC3. If the test fails it displays the actual and expected device status.

T04a Test that the NOT BSY flag is reset when the device is active. If the test fails it displays the actual and expected device status.

T05a Test that the NOT BSY flag is set after an I/O command is completed. The test waits until the controller response time limit has elapsed before reporting an error. If the test fails, it displays "THE CNT'LR RESPONSE TIME LIMIT ELAPSED".

T06a Test that the NOT RDY flag is reset when the diskette drive is ready. If the test fails it displays the actual and expected device status.

T07a Test that the NOT RDY flag is set when the diskette drive is not ready. This test requires user intervention and is not run if the user responded "N" to the "MANUAL INTERVENTION TESTS" prompt. The test prompts the user to remove the diskette from the test drive. The test then verifies that the NOT RDY flag is set.

TESTS 08-0B - THE DRIVE SELECTION TESTS - Each test prompts the user to insert a diskette in a specific drive. It then verifies that the specified drive is ready and no other drive is ready. These tests require user intervention and will not be run if the user responded "N" to the "MANUAL INTERVENTION TESTS" prompt.

T0Ba Test that drive 0 can be selected and drive 0 does not respond to any other drive address. If the test fails it displays the actual and expected device status.

T09a Test that drive 1 can be selected and drive 1 does not respond to any other drive address. If the test fails it displays the actual and expected device status.

T0Aa Test that drive 2 can be selected and drive 2 does not respond to any other drive address. If the test fails it displays the actual and expected device status. This test does not apply to the MDC.

T0Ba Test that drive 3 can be selected and drive 3 does not respond to any other drive address. If the test fails it displays the actual and expected device status. The test does not apply to the MDC.

TESTS 0C-14 - THE DEVICE STATUS TESTS

T0Ca Prompt the user to insert a write protected diskette in the test drive. Verify that the write protect flag is set. If the test fails it displays the actual and expected device status. This test requires user intervention and is not run if the user responded "N" to the "MANUAL INTERVENTION TEST" prompt.

T0Da Prompt the user to write enable the diskette in the test drive. Verify that the write protect flag is reset. If the test fails it displays the actual and expected device status. This test requires user intervention and is not run if the user responded "N" to the "MANUAL INTERVENTION TEST" prompt.

T0Ea Test that the device can successfully complete a restore command. If the test fails it displays the actual and expected device status.

T0Fa Test that the track 0 flag is set after a restore command. This test does not apply to double density or IBM mode. If the test fails it displays the actual and expected device status.

T10a Test that the controller recognizes an illegal command. If the test fails it displays the actual and expected device status.

T11a Test that successive commands clear the encoded flags. The test generates an illegal command error and then issues a restore command to the test device. The encoded flags should be reset at the completion of the restore command. If the test fails it displays the actual and expected device status.

T12a Test that the controller generates interrupts. If the controller fails to generate an interrupt within the controller response time limit the test displays "THE CNT'LR RESPONSE TIME LIMIT ELAPSED". If the device generates an interrupt and an error flag is set in the device status byte the test displays the actual and expected device status.

T13a Test that the device can successfully complete a write command. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. An error is flagged by an audible alarm and the device status in the device status display field.

T14a Test that the main channel current address vectors (FCAL and FCAH) are properly updated. The test issues a write command to the test device. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. A write error is flagged by an audible alarm and the device status in the device status display field. An incorrect main channel current address vector is flagged by an audible alarm.

TESTS 15-1C - READ/WRITE TESTS - The read/write test section is a sequence of tests which record, recover and verify a unique data pattern. Each test consists of three subtests. Subtest "a" records the data pattern on the first sector of track 0. Subtest "b" recovers the previously recorded data pattern. Subtest "c" verifies that the read and write data are identical. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. The write data is displayed at lines 6-12. The read data is displayed at lines 14-20. A read or write error is flagged by an audible alarm and the device status in the device status display field. If the read and write data are not identical the incorrect byte is tagged in the read buffer display.

T15a Test that the device can write an all 0's data pattern.
T15b Test that the device can read an all 0's data pattern.
T15c Verify that the read and write data are identical.

T16a Test that the device can write a rotating 1's data pattern.
T16b Test that the device can read a rotating 1's data pattern.
T16c Verify that the read and write data are identical.

T17a Test that the device can write alternating bytes of alternating 1's and 0's (55 and AA hexadecimal).
T17b Test that the device can read alternating bytes of alternating 1's and 0's.
17c Verify that the read and write data are identical.

T18a Test that the device can write an alternating 1's and 0's pattern.
T18b Test that the device can read an alternating 1's and 0's pattern.
T18c Verify that the read and write data are identical.

T19a Test that the device can write a rotating 0's data pattern.
T19b Test that the device can read a rotating 0's data pattern.
T19c Verify that the read and write data are identical.

T1Aa Test that the device can write an all 1's data pattern.
T1Ab Test that the device can read an all 1's data pattern.
T1Ac Verify that the read and write data are identical.

T1Ba Test that the device can write alternating bytes of 1's and 0's.
T1Bb Test that the device can read alternating bytes of 1's and 0's.
T1Bc Verify that the read and write data are identical.

T1Ca Test that the device can write the worst case data pattern.
T1Cb Test that the device can read the worst case data pattern.
T1Cc Verify that the read and write data are identical.

TESTS 1D-1E - PDCIFL BLOCK CHECK CHARACTER TESTS

T1Da Record the worst case data pattern at the first sector on track 0. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. A write error is flagged by an audible alarm and the device status in the device status display field.

T1Db Test that the device can successfully complete a check command. The device statistics are presented in the same fashion as T1Da. An error is flagged by an audible alarm and the device status in the device status display field.

T1Ea Test that the device detects an invalid block check character. This test attempts sixteen retries before it reports an error. If the test fails it displays the actual and expected device status.

TESTS 1F-22 - FORMAT TESTS

T1Fa Tests that the device can successfully execute a verify command. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. An error is flagged by an audible alarm and the device status in the device status display field. This test is not applicable to single density mode.

T20a Test that the device can successfully execute a format command. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. An error is flagged by an audible alarm and the device status in the device status display field. This test is not applicable to single density mode.

T20b Verify the previously formatted track. T20b is identical to T1Fa.

T21a Erase the data sync mark at each sector on track 0. The format command erases the data sync mark. A format error is flagged by an audible alarm. This test is not applicable to single density mode.

T21b Test that the device can detect a missing data sync mark. A check command is executed at the previously formatted track. If the test fails it displays the actual and expected device status. This test is not applicable to single density mode.

T22a Test that the device can successfully execute a write deleted data command. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. An error is flagged by an audible alarm and the device status in the device status display field. This test only applies to IBM mode.

T22b Verify that the previous write deleted data command recorded a deleted data mark. The test issues a check command and expects the deleted data mark flag to be set. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. An error is flagged by an audible alarm and the device status in the device status display field. This test only applies to IBM mode.

TEST 23 - THE COM3 LATCH (SECTOR NO.) TESTS - This test tests various bit patterns in the COM3 latch. The number of bit patterns tested is dependent on the controller mode and whether or not the attached drive is double sided. Each sector address is recorded in the corresponding sector. After all the applicable sector addresses have been recorded, each sector is read and the sector address is verified. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. The write data is displayed at lines 6-12. The read data is displayed at lines 14-20. An error is flagged by an audible alarm and the device status in the device status display field.

T23a	Record the sector address in sector 0.
T23b	Record the sector address in sector 1.
T23c	Record the sector address in sector 2.
T23d	Record the sector address in sector 4.
T23e	Record the sector address in sector 8.
T23f	Record the sector address in sector 0F hexadecimal.
T23g	Record the sector address in sector 10 hexadecimal.
T23h	Record the sector address in sector 1F hexadecimal.
T23i	Record the sector address in sector 20 hexadecimal.
T23j	Record the sector address in sector 3F hexadecimal.
T23k	Read sector 0.
T23l	Read sector 1.
T23m	Read sector 2.
T23n	Read sector 4.
T23o	Read sector 8.
T23p	Read sector 0F hexadecimal.
T23q	Read sector 10 hexadecimal.
T23r	Read sector 1F hexadecimal.
T23s	Read sector 20 hexadecimal.
T23t	Read sector 3F hexadecimal.

TEST 24 - THE COM2 LATCH (TRACK NO.) TEST - This test tests various bit patterns in the COM2 latch. The number of bit patterns tested is dependant on the controller type. Each track address is recorded at the first sector of the corresponding track. After all the applicable track addresses have been recorded, the first sector of each track is read and the track address is verified. The I/O command, mode, drive number, surface, track number, sector number, and device status are displayed from left to right. The write data is displayed at lines 6-12. The read data is displayed at lines 14-20. An error is flagged by an audible alarm and the device status in the device status display field.

T24a	Record the track address at the first sector of track 0.
T24b	Record the track address at the first sector of track 1.
T24c	Record the track address at the first sector of track 2.
T24d	Record the track address at the first sector of track 4.
T24e	Record the track address at the first sector of track 8.
T24f	Record the track address at the first sector of track 0F hexadecimal.
T24g	Record the track address at the first sector of track 10 hexadecimal.
T24h	Record the track address at the first sector of track 20 hexadecimal.
T24i	Record the track address at the first sector of track 3F hexadecimal.

T24j Record the track address at the first sector of track 40 hexadecimal.
T24k Read the first sector at track 0.
T24l Read the first sector at track 1.
T24m Read the first sector at track 2.
T24m Read the first sector at track 4.
T24o Read the first sector at track 8.
T24p Read the first sector at track 0F hexadecimal.
T24q Read the first sector at track 10 hexadecimal.
T24r Read the first sector at track 20 hexadecimal.
T24s Read the first sector at track 3F hexadecimal.
T24t Read the first sector at track 40 hexadecimal.

TESTS 25-27 - THE WORST CASE SITUATION TESTS - These tests read and write the worst case data pattern in worst case situations. The I/O command, mode, drive number, surface, track number, sector number and device status are displayed from left to right. The write data is displayed at lines 6-12. The read data is displayed at lines 14-20. An error is flagged by an audible alarm and the device status in the device status display field.

T25a Test that the worst case data pattern can be recorded without precompensation. The test selects the last track at which precompensation is disabled. The worst case data pattern is recorded on all sectors at the predetermined track. This subtest only applies to double density mode.

T25b Test that the worst case data pattern can be recovered without precompensation. The previously recorded worst case data pattern is recovered from the predetermined track. This subtest only applies to double density mode.

T26a Test that the worst case data pattern can be recovered with precompensation. The test records the worst case data pattern on all sectors at the last track (precompensation is enabled at the last track). This subtest only applies to double density mode.

T26b Test that the worst case data pattern can be recovered with precompensation. The previously recorded worst case data pattern is recovered from the last track. This subtest only applies to double density mode.

T27a Test the read/write head settle down during a write operation. The test issues a sequence of write commands. Each write command is interleaved with an INIT and a read/write head unload delay. If the test is running on an mini-diskette controller the heads are not unloaded but the spindle is brought to a stop. The next write command reloads the heads or restarts the spindle.

T27b Test the read/write head settle down during a read operation. This test is identical to T27a except a read command is used rather that a write command. The contents of the read and write buffers are compared to verify they are identical. If the read and write data are not identical the incorrect byte is tagged in the read buffer display.

The mini-diskette alignment test sequence can only be used to align or verify the alignment of a mini-diskette drive. These tests are used in conjunction with the mini-diskette alignment procedure. The mini-diskette alignment test sequence cannot be entered from the diagnostic test sequence. To enter the mini-diskette alignment test sequence the user must respond "Y" to the "ALIGNMENT TEST" prompt in the diagnostic menu.

T00a Radial alignment test. This test provides the facilities necessary to perform the radial alignment procedure. The test allows the user to select any of the three alignment tracks and either surface. The legal keyboard commands are:

Function key 0 - Track 01 seek;
Function key 1 - Track 16 seek;
Function key 2 - Track 34 seek;
"0" - Select surface 0;
"1" - Select surface 1.

Initially the user should select a track prior to selecting a surface. After the initial track selection the keyboard commands can be entered in any order.

T01a Track 0 switch test. This test provides the facilities necessary; to perform the track 0 switch test. The test allows the user to select either track 0 or track 1 while monitoring the status of the track 0 flag. The legal keyboard commands are:

Function key 0 - Track 0 seek;
Function key 1 - Track 1 seek.

Errors

All errors are flagged by a one second audible alarm. Some errors are accompanied by an explanatory error message. Those error messages which only apply to one test are described in the test description section of the documentation. A description of the error messages which apply to more than one test or to events external to the test follows.

STATUS ERROR: ACT=   EXP=   - This message displays the actual and expected device status after a status error. The actual and expected device status is displayed in binary. The expected device status may contain one or more "X"'s to indicate that the status of the indicated bit position is ignored during this test.

THE CNT'LR RESPONSE TIME LIMIT ELAPSED - This message indicates that the user specified controller response time limit expired before the device set the NOT BSY flag or generated an interrupt. This message can be generated at any time during the diagnostic or mini-diskette alignment test sequence.

THE PROGRAM DETECTED AN ILLEGAL INTRPT - This message indicates that the interrupting device was not interrupt enabled at the time it generated the interrupt. The message is followed by the hexadecimal representation of the interrupt status byte contents. This message can be generated at any time during the diagnostic or mini-diskette alignment test sequence.

DATA MATCH ERROR: ACT= EXP= - This message indicates that after reading a previously recorded data pattern the test determined that the read and write data are not equal. The actual and expected data are displayed in hexadecimal. The read data is displayed as actual data and the write data is displayed as expected data.

DSPSW

DISPLAY SWITCH TEST

DSPSW - DISPLAY SWITCH TEST

Applicable Assemblies

5300-1102	OP-1/15 Logic Board

General Description

The purpose of the DSPSW Program is to determine if the cursor function, emphasis and attribute display switches are working properly. The program requires assistance from the operator to continue the test.

Loading Procedure

DSPSW can be loaded into memory using any conveniently available loading method. It is a completely self-contained program. If DSPSW loads properly, it will identify itself and immediately tell the operator which switches turn off.

Operator Action

To proceed with the test the operator must type the space bar. (When a message appears alongside a switch description, this is the switch to be tested.) The operator at this point of time will turn on the switch. All switches must be off before continuing on to test the next switch.

Errors

There are no program-detectable errors. Errors are recognized by operator inspection of the display screen.

Test Description

The switch bank number will be displayed on the screen with its corresponding switches. Each switch will specify what it represents. During the testing a message will appear alongside the switch to be tested (except during the cursor testing). The message text will be the only response to the switch when it is turned on (except when the intensity is tested). There will also be an asterisk that will appear alongside the corresponding switch to be tested.

LFI15

OP-1/15 LINE FREQUENCY INTERRUPT TEST

LFI15 - OP-1/15 LINE FREQUENCY INTERRUPT TEST

Applicable Assemblies

5300 - 1102	OP-1/15 Logic Board

General Description

The purpose of the LFI15 Program is to determine if the line frequency interrupts (6.5 & 7.5) are functioning correctly, and, if not, to give an indication of which functions are incorrect. This diagnostic does not require operator assistance. This diagnostic assumes the asynchronous or synchronous I/O adapter is functioning correctly.

8K of memory is the minimum requirement to run LFI15.

Loading Procedure

LFI15 can be loaded into memory using any conveniently available loading method. If LFI15 loads properly, it will identify itself and will wait for operator input. At this point one of the two I/O adapter types must be inputted (A = asynchronous or S = synchronous OP-1/15 configuration). After inputting the correct I/O adapter type, the program will wait for the line frequency hertz input. If 50Hz is the OP-1/15 configuration type - 'Y', otherwise type 'N' for 60Hz. Finally an audible beep will be sounded and the program will await for run-mode information. (Reference Appendix A for program run modes).

Errors

All errors are program detectable.

Test Description

All test descriptions are described in this section. If an error is detected, the appropriate error message will be displayed. On the following pages each test is listed with a brief description of what it is testing, with a list of error messages in which the test has the capability of displaying. There will be an audible click between tests signifying the program is still running.

Test 00 Tests that when 6.5 and 7.5 interrupt have been masked from interrupting, no interrupt will generate at either of these two vectors. If an interrupt is generated, the vector that had an interrupt will be displayed.

IE: Error message
'Interrupt Responded to Vector - X.5'

Test 01 Tests that when only the 6.5 interrupt is enabled, a level triggered interrupt does respond to the 7.5 vector. If no interrupt responds within a period of time, a time out message will be displayed. An incorrect interrupt will also be displayed.

IE: Error messages 'No Interrupt Occurred'
'Interrupt Responded to Vector - X.5'

Test 02 Tests that after A 6.5 interrupt has been generated and the 6.5 pending interrupt flag goes high, the flag can be reset by the issuing of a COM3 command. If the 6.5 pending flag is stuck high, an error message will appear.

IE: Error message '6.5 Didn't Reset'

Test 03 Tests that when only the 7.5 interrupt is enabled, an edge triggered interrupt does respond to the 7.5 vector. If no interrupt responds within a period of time, a timeout message will be displayed. An incorrect interrupt will also be displayed.

IE: Error messages 'No Interrupt Occurred'
'Interrupt Responded to Vector - X.5'

Test 04 Tests that after a 7.5 interrupt has been generated and the 7.5 pending interrupt flag goes high, the flag can be reset. If the 7.5 pending flag is stuck low an audible beep will be sounded. If the 7.5 pending flag is stuck high, an error message will appear.

IE: Error message '7.5 Didn't Reset'

Test 05 Tests the interrupt line frequency time for either 50Hz or 60 Hz (+ or - 10%) for the level triggered interrupt. If the interrupt has been timed out or the timing falls outside of the boundary limitations, an audible beep will be sounded.

Test 06 Tests the interrupt line frequency time for either 50Hz or 60Hz (+ or - 10%) for the 7.5 edge triggered interrupt. If the interrupt has been timed out or the timing falls outside of the boundary limitations, an audible beep will be sounded.

RAMSWI5
ROM CONFIGURATION SWITCH TEST

Test Description

All tests are described in this section. Expected and actual values will be displayed, as well as, the address the error was encountered. On the following pages each test is listed with a brief description of how it will be testing each switch. Each switch will be tested individually. During a test, three different modes will appear in the right hand upper corner of the screen (only a 'W' and 'R' will appear during test 00).

ie., 'W'
'R'
'Testing ROM'

The 'W' represents a write to the specified amount of RAM. The 'R' represents a read to the specified amount of RAM. The 'testing ROM' message shows the operator that ROM is being accessed. Each test will take approximately 30 seconds to complete execution of the test. If the operator specifies that the OP-1/15 is configured with low prom(s) and a RAM size greater than 16K, the program will relocate from the lower 16K area of RAM. The screen will be blanked during relocation. The operator will be instructed when to turn on and off each switch throughout the desired test.

TEST 00 Tests that with all the RAM switches off, all of RAM can be written to and read from using opposite values. (Reference description 1 for the testing procedure). If an error occurs, a switch is not off or is not functioning properly.
If, high prom and RAM is less than 64K, the program will display the message 'no test is possible'. (Reference description 1 for the testing procedure.)

TEST 01 Tests that with only switch 1 on, the program cannot read what has been written to 2K of RAM, if so, switch 1 is not functioning properly.
High prom and 64K of RAM, F800-FFFF=ROM
low prom, 0000-0800=ROM
If,low prom and RAM is greater than 16K, the program will be relocated out of the lower 16K area.
If, high prom and RAM is less than 64K, the program will display the message 'no test is possible'. (Reference description 1 for the testing procedure.)

TEST 02 Tests that with only switch 2 on, the program cannot read what has been written to 4K of RAM, if so, switch 2 is not functioning properly.
High prom and 64K of RAM, F000-FFFF=ROM
low prom, 0000-1000=ROM
If, low prom and RAM is greater than 16K, the program will be relocated out of the lower 16K area.
If, high prom and RAM is less than 64K, the program will display the message 'no test is possible'. (reference description 1 for the testing procedure.

TEST 03 Tests that with only switch 3 on, the program cannot read what has been written to 8K of RAM, if so, switch 3 is not functioning properly.
high prom and 64K of RAM, E000-FFFF=ROM
low prom, 0000-2000=ROM
If, low prom and RAM is greater than 16K, the program will be relocated out of the lower 16K area.
If, high prom and RAM is less than 64K, the program will display the message 'no test is possible'. (reference description 1 for the testing procedure)

TEST 04 Tests that with only switch 4 on, the program cannot read what has been written to 16K of RAM, if so, switch 4 is not functioning properly.
high prom and 64K of RAM, C000-FFFF=ROM
low prom, 0000-4000=ROM
If, low prom and RAM is greater than 16K, the program will be relocated out of the lower 16K area.
If, low prom and RAM is less than 16K, the program will display the message 'no test is possible'.
If, high prom and RAM is less than 64K, the program will display the message 'no test is possible'. (reference description 1 for the testing procedure)

DESCRIPTION 1

While testing each switch, RAM will have two totally opposite values written and read. Since RAM is assumed good, an error will appear if the value cannot be read from a location that has been written to. These test patterns will not be written into the 800-807 vectors or where the program resides. (00's will be tested first, then FF's will be tested). Each test (except test 00) will then test ROM. It does so, by writing to the ROM overlaying RAM area. It first writes 00's to a location and if this value is read back, FF's are then written. If the FF's are not read the appropriate switch is working properly. If the second value is read, an error will be shown.

SERPR15
SERIAL PRINTER ADAPTER TEST

SERPR15 - OP-1/15 SERIAL PRINTER ADAPTER TEST

Applicable Assemblies

5300 - 1119
5300 - 1120

Required Test Assemblies

COMTST14 Diagnostic Cable

General Description

The purpose of SERPR15 program is to determine if the Serial Printer Adapter (for either -1 or -2 assembly) is functioning correctly, and if not, to give an indication of which functions are incorrect. This diagnostic does not require operator assistance. This diagnostic also assumes the Asynchronous I/O Adapter is functioning properly.

8K of memory is the minimum requirement to run SERPR15.

Loading Procedure

SERPR15 can be loaded into memory using any conveniently available loading method. If SERPR15 loads properly, it will identify itself and will wait for the operator to insert both plugs of the COMTST14 diagnostic cable into their respective connectors, as shown in Figure 1. After the correct insertion of the COMTST14 cable, the operator must make any keystroke (other than Prog), then input the -1 or -2 assembly type configured in the OP-1/15 to be tested. (Key 1 for -1 assembly and Key 2 for -2 assembly). Finally the program will wait for program run-mode information. (Reference Appendix A for program run modes).

Errors

All errors are program detectable.

COMTST14 Diagnostic Cable

The COMTST14 diagnostic cable is necessary for testing the Serial Printer Adapter (for either -1 or -2 assemblies). The cable must be inserted into the Asynchronous I/O Adapter connector and the Serial Printer Adapter connector (as shown in Figure 1) on the rear panel of the OP-1/15 prior to program execution. The schematic of the COMTST14 diagnostic cable is shown in Figure 2 of this section.

Test Description

All test descriptions are described in this section. Expected and actual byte values will be displayed during the necessary tests (when an error is detected). On the following pages each test is listed with a brief description of what it is testing. There will be an audible click between tests signifying the program hasn't locked up.

Refer to Appendix A for run-mode options.

Test 00	Tests that the Serial Printer Adapter cannot be selected with an incorrect select address. Incorrect select addresses are all one byte configurations of three or less bits high. The address that selected the Serial Printer Adapter is displayed in the upper right hand corner of the screen.
Test 01	Tests that the Serial Printer Adapter can be selected with the proper select address.
Test 02	Tests that a selected Serial Printer Adapter can be de-selected with the issuing of an INIT command.
Test 03	Tests that IFL bit 1 (-1 assembly = printer not ready, and -2 assembly = input B) is not stuck high.
Test 04	Tests that IFL bit 1 (-1 assembly = printer not ready, and -2 assembly = input B) is not stuck low.
Test 05	Tests that IFL bit 0 (-1 assembly = motor status, and -2 assembly = input A) is not stuck high.
Test 06	Tests that IFL bit 0 (-1 assembly = motor status, and -2 assembly = input A) is not stuck low.
Test 07	Tests that OFL bit 7 is not stuck high.
Test 08	Tests that OFL bit 7 is not stuck low.
Test 09	Tests that OFL bit 6 is not stuck high.
Test 0A	Tests that OFL bit 6 is not stuck low.
Test 0B	Tests that the not busy flag can go low when an output command is issued. IFL bit 7 is stuck high if this test fails.
Test 0C	Tests that the not busy flag will go high after a DVCL command is issued. IFL bit 7 is stuck low if this test fails.
Test 0D	Tests that the not busy flag will go high after an INIT command is issued. IFL bit 7 is stuck low if this test fails.
Test 0E	Tests that by turning on RS-232 Pin 14 (DNC line), IFL bit 7 goes high (after executing an output instruction). This test is only executed for A -1 assembly.

Test 0F Tests that by turning off RS-232 pin 11 (Input A), IFL bit 7 goes low. This test is only executed for A -2 assembly.

Test 10 Tests that by turning on RS-232 pin 11 (Input A), IFL bit 7 goes high. This test is only executed for A -2 assembly.

Test 11 Tests that an interrupt is generated when the not busy flag is high. The interrupt mask is set to allow an interrupt only from the Serial Printer Adapter.

Test 12 Tests that when the Serial Printer Adapter interrupts, it is the Serial Printer Adapter interrupting the processor.

Test 13 Tests that no interrupt is generated when the not busy flag is low. The interrupt mask is set to allow an interrupt only from the Serial Printer Adapter.

Test 14 Tests that when the SMSK is set to 00000000B, no interrupts are generated.

Test 15 Tests that the transmitter is working properly at 50 baud. 22 characters are transmitted with the communication parameters set as 7 data bits, 2 stop and odd parity. After each character is transmitted an upper case 'T' will appear along the right hand side of the screen.

Test 16 Tests that the transmitter is working properly at 75 baud, 22 characters are transmitted with the communication parameters set as 7 data bits, 1 stop and odd parity. After each character is transmitted an upper case 'T' will appear along the right hand side of the screen.

Tests 17-24 Test the 14 baud rates, by transmitting 8 data bits, one stop and odd parity. The transmit and receive buffers will be displayed on the screen.

Test 18 Tests 134.5 baud rate.

Test 19 Tests 150 baud rate.

Test 1A Tests 300 baud rate.

Test 1B Tests 600 baud rate.

Test 1C Tests 1200 baud rate.

Test 1D Tests 1800 baud rate.

Test 1E Tests 2000 baud rate.

Test 1F Tests 2400 baud rate.

Test 20 Tests 3600 baud rate.

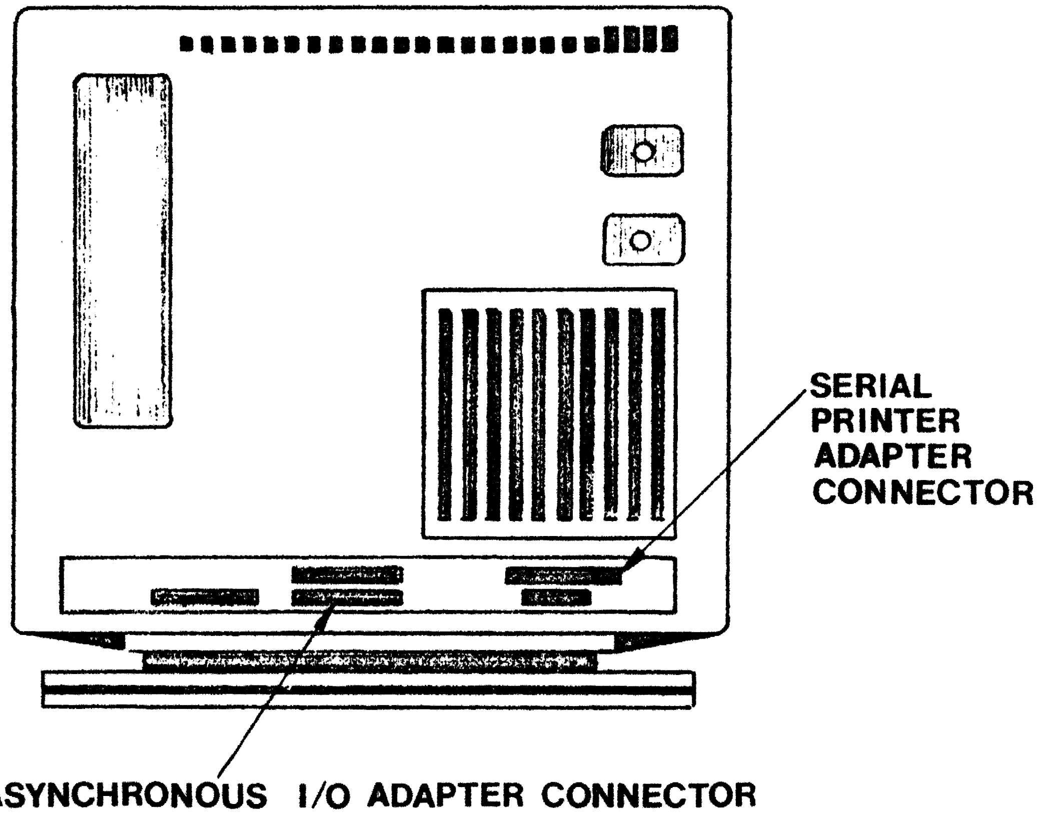
Test 21 Tests 4800 baud rate.

Test 22 Tests 7200 baud rate.

Test	23	Tests 9600 baud rate.
Test	24	Tests 19,200 baud rate.

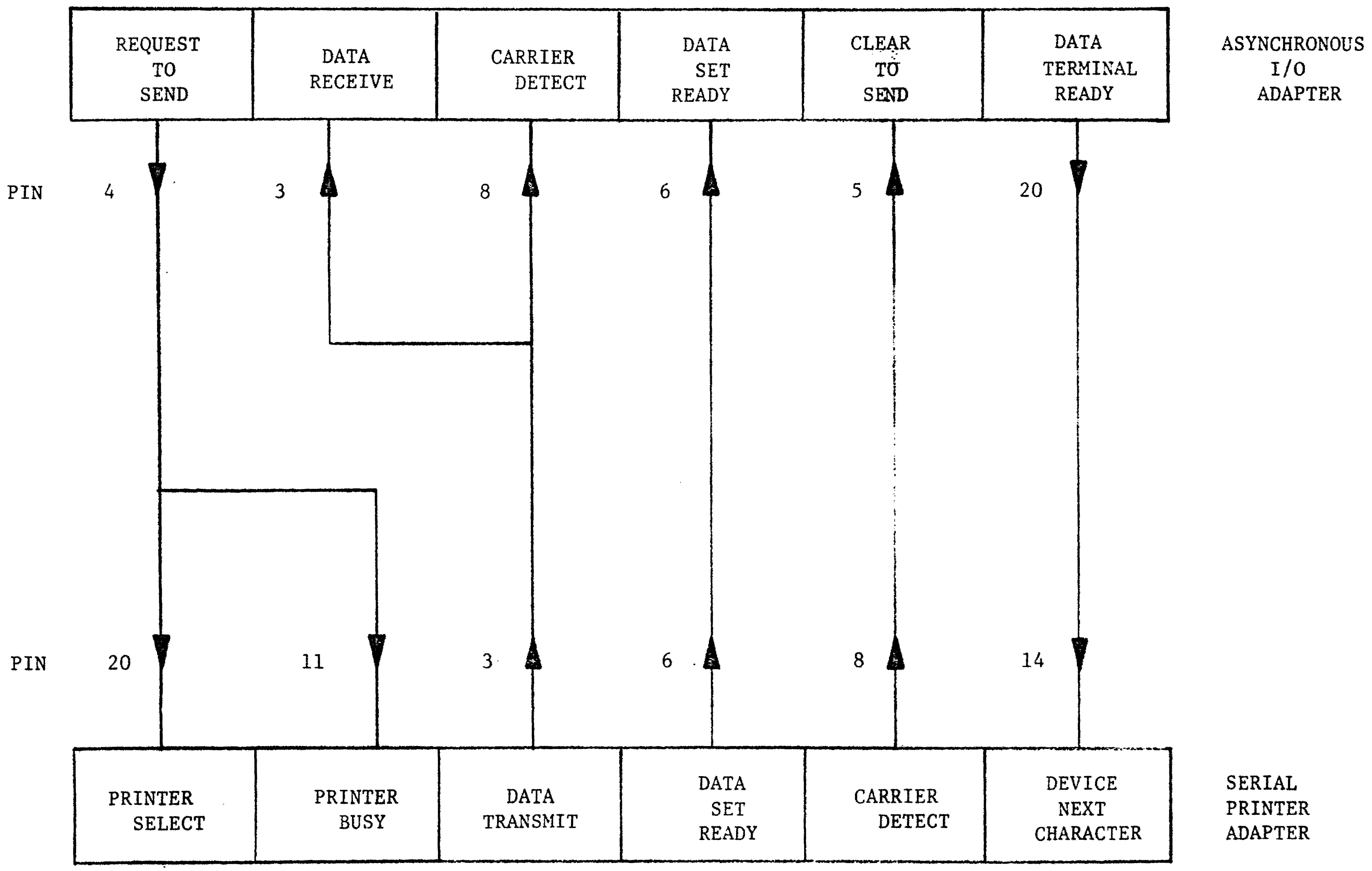
Tests 25-30 Tests all combinations of data bits, stop bits and parity bits. The transmit and receive buffers will be displayed on the screen. A bad transmitted byte will be marked with a reversed 'X' in the transmit buffer and the actual and expected byte values will be displayed in the upper right hand corner of the screen.

Test	25	Tests transmitting 7 data bits, 1 stop bit and no parity.
Test	26	Tests transmitting 7 data bits, 1 stop bit and odd parity.
Test	27	Tests transmitting 7 data bits, 1 stop bit and even parity.
Test	28	Tests transmitting 7 data bits, 2 stop bits and no parity.
Test	29	Tests transmitting 7 data bits, 2 stop bits and odd parity.
Test	2A	Tests transmitting 7 data bits, 2 stop bits and even parity.
Test	2B	Tests transmitting 8 data bits, 1 stop bit and no parity.
Test	2C	Tests transmitting 8 data bits, 1 stop bit and odd parity.
Test	2D	Tests transmitting 8 data bits, 1 stop bit and even parity.
Test	2E	Tests transmitting 8 data bits, 2 stop bits and no parity.
Test	2F	Tests transmitting 8 data bits, 2 stop bits and odd parity.
Test	30	Tests transmitting 8 data bits, 2 stop bits and even parity.


SERIAL PRINTER ADAPTER CONNECTOR
ASYNCHRONOUS I/O ADAPTER CONNECTOR

COMTST14 DIAGNOSTIC CABLE INSERTION GUIDE

FIGURE 1



COMTST14 DIAGNOSTIC CABLE

FIGURE 2

VIDTST4

4K DISPLAY MICROPROCESSOR TEST

VIDTST4 - 4K DISPLAY MICROPROCESSOR TEST

Applicable Assemblies

1122-1	24 Line Display Microprocessor
OP-1/R	Display Microprocessor
1133-1	24/25 Line Display Microprocessor

General Description

VIDTST4 is a completely self-contained operator-controlled diagnostic program with a minimum RAM requirement of 4K. Each separate test displays a video presentation with which the operator must visually compare an expected result. There is no software-controlled error checking.

Loading and Operation

VIDTST4 may be loaded into memory using any available method (see Appendix B). When the program loads correctly the following question is displayed on the screen:

256 CHARACTER SET (PCO 261, 308)? TYPE (Y)es, (N)o

The operator should respond "Yes" if the Assembly under test is an OP-1/R incorporated with PCO 261, and "No" otherwise. Responding to the question produces an audible signal, displays an identification message on the top line of the screen, and awaits operator input from the keyboard. Initially, and throughout the program as it proceeds, a prompter message will also appear on the screen to cue the operator. Typing the "PROG" key at any time will restart VIDTST4. The first prompter message is:

TYPE TEST NO. / SPACE

It is displayed; (a) upon loading, (b) upon completion of the last test, and (c) upon restarting the program. The message instructs the operator as follows:

1. Typing the SPACE bar will present the first sequential test (T00).
2. Typing a hexadecimal test number followed by typing the SPACE bar will present the specified test. (Attempting to enter a non-existent test number will restart the program.)

The prompter message:

TESTING

is displayed during the presentation of a test. The test is in progress as long as it is visible. The operator exits from the test by typing the SPACE bar. The command line will then be returned to the top line of the screen (if it had been moved) and the following prompter message is displayed:

TYPE SPACE OR "R"

VIDTST4 - 4K DISPLAY MICROPROCESSOR TEST

Applicable Assemblies

1122-1	24 Line Display Microprocessor
OP-1/R	Display Microprocessor
1133-1	24/25 Line Display Microprocessor

General Description

VIDTST4 is a completely self-contained operator-controlled diagnostic program with a minimum RAM requirement of 4K. Each separate test displays a video presentation with which the operator must visually compare an expected result. There is no software-controlled error checking.

Loading and Operation

VIDTST4 may be loaded into memory using any available method (see Appendix B). When the program loads correctly the following question is displayed on the screen:

256 CHARACTER SET (PCO 261)? TYPE (Y)es, (N)o

The operator should respond "Yes" if the Assembly under test is an OP-1/R incorporated with PCO 261, and "No" otherwise. Responding to the question produces an audible signal, displays an identification message on the top line of the screen, and awaits operator input from the keyboard. Initially, and throughout the program as it proceeds, a prompter message will also appear on the screen to cue the operator. Typing the "PROG" key at any time will restart VIDTST4. The first prompter message is:

TYPE TEST NO. / SPACE

It is displayed; (a) upon loading, (b) upon completion of the last test, and (c) upon restarting the program. The message instructs the operator as follows:

1. Typing the SPACE bar will present the first sequential test (T00).

2. Typing a hexadecimal test number followed by typing the SPACE bar will present the specified test. (Attempting to enter a non-existent test number will restart the program.)

The prompter message:

TESTING

is displayed during the presentation of a test. The test is in progress as long as it is visible. The operator exits from the test by typing the SPACE bar. The command line will then be returned to the top line of the screen (if it had been moved) and the following prompter message is displayed:

TYPE SPACE OR "R"

The message instructs the operator as follows:

1. Typing 'R' will cause the program to repeat the test just presented (the test number in question is shown in the command line).

2. Typing the SPACE bar will cause the program to present the test which sequentially follows the displayed test number. If the displayed test number is the last test in the sequence, the program will restart.

Description of Tests

T00 Character Generation

The characters generated for each hex value (00-FF) are displayed on the screen for inspection. If OP-1/R with PCO 261 has been specified, the SPACE bar should be typed again, setting bit 0 of display location 802H which should enable a 256 character set display.

T01 Bit 0, Location 802

The cursor should be positioned in the first column of the third line on the screen. Bit 0 of location 802 is made high. The cursor should blink for both OP-1 and OP-1/R configurations. The test is skipped for OP-1/R with PCO 261.

T02 Bit 0, Location 802

The cursor should be positioned as in T01. Bit 0 of location 802 is made low. For OP-1 configurations, the cursor should not blink. For OP-1/R configurations, the cursor should be blinking. The test is skipped for OP-1/R with PCO 261.

T03 Cursor Horizontal (Location 800)

The display is filled with characters as in T00. The cursor is put at the first position of the second line by writing 0 to location 800. Location 800 is then incremented by one up to 50 hex. This should cause the cursor to move horizontally across the line until it disappears from the screen.

T04 Cursor Vertical (Location 801)

The display is filled as in T00 and the cursor is initialized as in T03. Location 801 is then incremented by one up to 18 hex. This should cause the cursor to move down vertically until it disappears from the screen.

T05 Reverse Video (Bit 6, Location 802)

The display is filled as in T00. Bit 6 of location 802 is then set, which should cause the screen to be displayed in reverse video (black on white).

T06 Blank Screen (Bit 7, Location 802)

The display is filled as in T00. Bit 7 of location 802 is then set, which should cause the entire screen to become blank. (Typing the SPACE bar will restore the video display.)

Explanation for Tests T07-T11

In each test (T07 - T11), three strings of characters are written to the screen:

1. A Normal (untagged) ASCII string
2. A Tagged (bit 7 high) ASCII string
3. A Control string (00-1F hex)

Location 802 is then written with a value to test a particular visual enhancement or combination of enhancements. The results for certain tests will vary according to what hardware configuration is being tested.

T07 No Enhancements

Location 802 = 00 hex. The three characters strings should appear similarly un-enhanced.

T08 Blink Tagged Characters

Location 802 = 02 hex. The Tagged string should blink while the Normal and Control strings appear un-enhanced.

T09 Reverse Tagged Characters

Location 802 = 04 hex. The Tagged string should appear in reverse video while the Normal and Control strings appear un-enhanced.

T0A Half-Intensity Tagged Characters

Location 802 = 08 hex. The Tagged string should appear half-intensified while the Normal and Control strings appear un-enhanced.

T0B Blink, Reverse Tagged Characters

Location 802 = 06 hex. The Tagged string should blink in reverse video while the Normal and Control strings appear un-enhanced.

T0C Blink, Half-Intensity Tagged Characters

Location 802 = 0A hex. For OP-1: The Tagged string should blink and be half-intensified while the Normal and Control strings appear un-enhanced. For OP-1/R: The Tagged string should appear half-intensified (no blink). The Normal and Control strings should appear un-enhanced.

T0D Reverse, Half-Intensity Tagged Characters

ocation 802 = 0C hex. The Tagged string should appear in alf-intensified, reverse video while the Normal and Control strings ppear un-enhanced.

T0E Reverse, Blink, Half-Intensity Tagged Characters

Location = 0E hex. For OP-1: The Tagged string should blink in half-intensified, reverse video. The Normal and Control strings should appear un-enhanced. For OP-1/R: The Tagged string should appear in half-intensified, reverse video (no blink). The Normal and Control strings should appear un-enhanced.

T0F Reverse Control Characters

Location 802 = 10 hex. For OP-1 configurations, the Control string should appear in reverse video while the Normal and Tagged strings appear un-enhanced. For OP-1/R configurations, there should be no enhancement for any string.

T10 Half-Intensity Control Characters

Location 802 = 20 hex. For OP-1 configurations, the Control string should appear half-intensified while the Normal and Tagged strings should appear un-enhanced. For OP-1/R configurations, there should be no enhancement for any string.

T11 Reverse, Half-Intensity Control Characters

Location 802 = 30 hex. For OP-1 configurations, the Control string should appear in half-intensified, reverse video while the Normal and Tagged strings appear un-enhanced. For OP-1/R configurations, there should be no enhancement for any string.

<u>Explanation for Tests T12 - T21</u>

In tests T12-T21, the HOME vectors (locations 803 and 804) are written with various values to move HOME throughout RAM. In each test the following message is written into memory after the HOME vectors are changed:

***ERROR IF VISIBLE** ONE STAR-TOP LINE TESTING**

For all tests (T12-T21) the message must appear left-justified on the top line of the display as follows:

*** ONE STAR-TOP LINE TESTING**

Any other characters appearing in the upper left position of the screen denote an error in the operation of the HOME vectors.

T12 HOME Low (Bit 0, Location 804) (EXPECTED FAILURE FOR 1133-1 BOARD)

Configuration:	804-01 hex	(00000001B)
	803-09 hex	(00001001B)

T13 HOME Low (Bit 1, Location 804) (EXPECTED FAILURE FOR 1133-1 BOARD)

Configuration:	804-02 hex	(00000010B)
	803-09 hex	(00001001B)

T14 HOME Low (Bit 2, Location 804) (EXPECTED FAILURE FOR 1133-1 BOARD)

Configuration:	804-04 hex	(00000100B)
	803-09 hex	(00001001B)

T15 HOME Low (Bit 3, Location 804) (EXPECTED FAILURE FOR 1133-1 BOARD)

Configuration:	804-08 hex	(00001000B)
	803-09 hex	(00001001B)

T16 HOME Low (Bit 4, Location 804)

Configuraion:	804-10 hex	(00010000B)
	803-09 hex	(00001001B)

T17 HOME Low (Bit 5, Location 804)

Configuration:	804-20 hex	(00100000B)
	803-09 hex	(00001001B)

T18 HOME Low (Bit 6, Location 804)

Configuration:	804-40 hex	(01000000B)
	803-09 hex	(00001001B)

T19 HOME Low (Bit 7, Location 804)

Configuration:	804-80 hex	(10000000B)
	803-09 hex	(00001001B)

T1A HOME High (Bit 0, Location 803)

Configuration:	803-0E hex	(00001110B)
	804-00 hex	(00000000B)

T1B HOME High (Bit 1, Location 803)

Configuration:	803-0D hex	(00001101B)
	804-00 hex	(00000000B)

T1C HOME High (Bit 2, Location 803)

Configuration:	803-0B hex	(00001011B)
	804-00 hex	(00000000B)

T1D HOME High (Bit 3, Location 803)

Configuration:	803-07 hex	(00000111B)
	804-20 hex	(00100000B)

Special Note for Tests T1E-T21

Tests T1E - T21 tests the high order 4 bits of HOME high (Location 803). If RAM does not exist at these addresses, the tests in question are not presented.

T1E HOME High (Bit 4, Location 803)

Configuration:	803-1F hex	(00011111B)
	804-00 hex	(00000000B)

T1F HOME High (Bit 6, Location 803)

Configuration:	803-2F hex	(00101111B)
	804-00 hex	(00000000B)

T20 HOME High (Bit 6, Location 803)

Configuration:	803-4F hex	(01001111B)
	804-00 hex	(00000000B)

T21 HOME High (Bit 7, Location 803)

Configuration:	803-8F hex	(10001111B)
	804-00 hex	(00000000B)

T22 FLEOL (ASCII 9F Hex)

A string of untagged ASCII characters is written to the screen. Imbedded in the string are 6 tagged characters which together form the word "TAGGED". When the SPACE bar is typed, a FLEOL (9F hex) is written to the screen to the left of the ASCII string. At this time all but the tagged characters in the string should be blanked out.

T23 LEOL (ASCII 8F Hex)

A string of ASCII characters is written to the screen as in T22. When the SPACE bar is typed, a LEOL (8F hex) is written to the screen to the left of the string. At this time, all the characters in the string should be unconditionally blanked out.

Explanation for Tests T24 - T2D

Tests T24-T2D test the WRAP capabilities of locations 805 and 806 as follows:

1. The physical end of RAM is found, and a section of memory equal to 11 screen lines (880 memory locations) is filled with spaces (20 hex), ending at the physical end of RAM.

2. The HOME and WRAP vectors are both made to point to an address that is equal to an integral number of screen lines before the physical end of RAM. Beginning at this location, the following message is written:

EVERY XX LINES TESTING

where "XX" denotes the number of lines before the end of RAM. If the WRAP vectors are functioning properly, the message should appear left-justified on the top line of the screen, and should be repeated every "XX" lines until the end of the screen.

T24 WRAP, 2 lines

T25 WRAP, 3 lines

T26 WRAP, 4 lines

T27	WRAP, 5 lines
T28	WRAP, 6 lines
T29	WRAP, 7 lines
T2A	WRAP, 8 lines
T2B	WRAP, 9 lines
T2C	WRAP, 10 lines
T2D	WRAP, 11 lines

Explanation for Tests T2E - T3D

Tests T2E - T3D test WRAP locations 805H and 806H as follows:

1. The HOME vectors are set to a value equal to 23 screen lines before the end of RAM.

2. The WRAP vectors are set to a test value, beginning at which is written the message:

   ***<--XXXX TESTING.**

   where XXXX equals the hex value of the WRAP address being tested. For each test (T2E - T3D), the message should appear left-justified on the bottom line of the display. If the test location does not exist in RAM, the test is skipped.

T2E	WRAP, to location 0001H	(EXPECTED FAILURE FOR 1133-1 BOARD)
T2F	WRAP, to location 0002H	(EXPECTED FAILURE FOR 1133-1 BOARD)
T30	WRAP, to location 0004H	(EXPECTED FAILURE FOR 1133-1 BOARD)

T31	WRAP, to location 0008H (EXPECTED FAILURE FOR 1133-1 BOARD)
T32	WRAP, to location 0010H
T33	WRAP, to location 0020H
T34	WRAP, to location 0040H
T35	WRAP, to location 0080H
T36	WRAP, to location 0100H
T37	WRAP, to location 0200H
T38	WRAP, to location 0400H
T39	WRAP, to location 0808H
T3A	WRAP, to location 1000H
T3B	WRAP, to location 2000H
T3C	WRAP, to location 4000H
T3D	WRAP, to location 8000H

RTCTST4

REAL TIME CLOCK TEST

RTCTST4 - REAL TIME CLOCK TEST

Description

RTCTST4 is a completely self-contained diagnostic program which tests the Real Time Clock Flag (Keyboard IFL Bit 6). Prior to running RTCTST4, the Asynchronous I/O Adapter must be tested with IOTST4 and be known good.

The minimum memory requirement for RTCTST4 is 4K.

Loading

RTCTST4 can be loaded into memory using any convenient method (see Appendix B). When RTCTST4 loads properly it will identify itself and await run-mode input by the operator (see Appendix A).

Initialization

After RTCTST4 has loaded and standard initialization completed, the question "OP-1/R 1101? TYPE Y or N" will appear on the screen. The user should type a 'Y' if the machine that RTCTST4 is running on is an OP-1/R 1101. However, if the system under test is an OP-1/R 1101 with PCO338, then an 'N' should be typed. PCO338 when implemented on an OP-1/R will tie the 4 high order interrupt register bits to ground. In all other cases an 'N' should be typed in response to the prompt.

Description of Tests

All tests operate automatically and without operator intervention. No key should be depressed while a test is executing. The tests are ordered in such a way that a subsequent test relies on what is demonstrated in a previous test, i.e., if a test fails, the results of following tests are questionable.

RTCTST4 - REAL TIME CLOCK TEST

Description

RTCTST4 is a completely self-contained diagnostic program which tests the Real Time Clock Flag (Keyboard IFL Bit 6). Prior to running RTCTST4, the Asynchronous I/O Adapter must be tested with IOTST and be known good.

The minimum memory requirement for RTCTST4 is 4K.

Loading

RTCTST4 can be loaded into memory using any convenient method (see Appendix B). When RTCTST4 loads properly it will identify itself and await run-mode input by the operator (see Appendix A).

Initialization

After RTCTST4 has loaded and standard initialization completed, the question "OP-1/R 1101? TYPE Y or N" will appear on the screen. The user should type a 'Y' if the machine that RTCTST4 is running on is an OP-1/R 1101, and a 'N' in all other cases.

Description of Tests

All tests operate automatically and without operator intervention. No key should be depressed while a test is executing. The tests are ordered in such a way that a subsequent test relies on what is demonstrated in a previous test, i.e., if a test fails, the results of following tests are questionable.

TEST 00 - KEYBOARD SELECT TEST

Tests that the Keyboard does not respond to an incorrect select address, does respond to the correct address, and INIT deselects a selected Keyboard. The correct device select address is 0E1H hex. The incorrect select addresses are those single byte addresses which have 3 or less bits high. This is derived from the fact that a 4 input AND gate performs the device selection from the address bits of the select address. Thus a select address of 0FF hex would attempt to select all devices.

KEYBOARD SELECTED WITH INCORRECT ADDRESS (XX)

IFL or INP to the select address XX hex gave a result other than 0FF hex (open bus).

KEYBOARD IS NOT SELECTED WITH CORRECT ADDRESS (E1)

IFL to the correct Keyboard address (0E1 hex) gave a result of 0FF hex (open bus).

TEST 01 - TEST THAT INIT DESELECTS KEYBOARD

The Keyboard is selected and an INIT is issued. The test fails if IFL did not get a result of 0FF hex (open bus).

TEST 02 - TEST THAT RTC FLAG CAN BE RESET

The Keyboard is selected and the commands DVCL, COM1, and COM2 are issued. The test fails if the RTC Flag has not been reset by these commands.

TEST 03 - TEST THAT RTC FLAG IS SET BY COM1

The Keyboard is selected and the RTC Flag is reset as in Test 02. A COM1 is issued to attempt to set the RTC Flag. The test fails if the flag has not been set after a nominal 10 seconds.

TEST 04 - TEST THAT INIT RESETS RTC FLAG

The RTC Flag is set as in Test 03, and an INIT is issued. The test fails if the flag has not been reset.

TEST 05 - TEST THAT DVCL RESETS RTC FLAG

The RTC Flag is set as in Test 03, and a DVCL is issued. The test fails if the flag has not been reset.

TEST 06 - TEST THAT COM1 RESETS RTC FLAG

The RTC Flag is set as in Test 03, and a COM1 is issued. The test fails if the flag has not been reset.

TEST 07 - TEST THAT COM2 RESETS RTC FLAG

The RTC Flag is set as in Test 03, and a COM2 is issued. The test fails if the flag has not been reset.

TEST 08 - INTERRUPT TEST, RTC FLAG RESET

The Keyboard is selected and the RTC Flag is reset by DVCL. The test fails if an interrupt is sensed from the keyboard.

TEST 09 - INTERRUPT TEST, RTC FLAG SET

The Keyboard is selected and the RTC Flag is set by COM1. The test fails if an interrupt is not sensed from the keyboard.

TEST 0A - MEASURE RTC TIME-OUT

The RTC is made to time-out six times in succession, and the results in milliseconds are displayed on the screen. In addition, the last five times are averaged and this value is also displayed. There are no program-detectable errors.

TEST 0B - TEST DE-SELECT DEVICE

The RTC timeout is measured once as in Test 0A, and the value is saved. Another measurement is then made during which time the keyboard is continually selected and then de-selected. This value is also saved. The two values are displayed on the screen as abstract hexadecimal character counts and then compared. The test fails if the 2nd value does not equal the 1st, + or - 12.5%.

FIXTST4

FIXED DATA SWITCH TEST

FIXTST4 FIXTST15 - FIXED DATA SWITCH TEST

Applicable Assemblies

5000-1141-X	8080 CPU Board
5000-EOXYZABCD	OP-1/R Main Logic Board
5000-1172-X-Y	Alternate I/O Adapter Board
5100-1106-XYZ	Alternate I/O Adapter Board
5300-1102	OP-1/15 Main Logic Board

Description

The FIXTSTs were designed to test the FIXED DATA SWITCHES (FIX1 and FIX2) and the FIXED DATA SWITCHES on an ALTERNATE I/O ADAPTER (SWITCH A and SWITCH B). They are completely self-contained diagnostic programs with a minimum RAM requirement of 4K.

Loading

The diagnostics can be loaded into memory using any convenient method (see Appendix B). Upon loading properly execution will begin, and the command line will be displayed (see Appendix A).

User Interaction

Immediately after the command line is displayed a question about which FIXED DATA SWITCHES are to be tested will appear. FIXTST4 and FIXTST15 default to testing FIX1 and FIX2 only. To change the defaults type a "Y" or "N" next to the desired switch message. This will cause the appropriate letter to appear in the box pointed to by the greater than sign and will advance the greater than sign to the next box in cyclic order. FIXTST4 and FIXTST15 will only respond to "Y" or "N" for yes or no respectively, the Prog key to restart the program, and the return key when the user decides that the information on the display is correct.

If the user indicated that SWITCH A or SWITCH B is to be tested the address of the ALTERNATE I/O ADAPTER is required. At this time the keyboard will only respond to a valid address (XD where X is 1, 2, 4 or 8), the depression of the return key to indicate that the address displayed is correct, or the Prog key to restart the program. After this, the program will wait for run-mode information (see Appendix A).

In the right corner of the display the original status, in hexadecimal, of the FIXED DATA SWITCHES will appear in the order of FIX1, FIX2, SWITCH A, and SWITCH B. If a switch is not being tested X's will appear instead. Once a valid value appears it will not be changed.

FIXTST4 - FIXED DATA SWITCHES TEST

Applicable Assemblies

5000-1141-X	8080 CPU Board
5000-EOXYZABCD	OP-1/R Main Logic Board
5000-1172-X-Y	Alternate I/O Adapter Board
5100-1106-XYZ	Alternate I/O Adapter Board

Description

FIXTST4 was designed to test the FIXED DATA SWITCHES (FIX1 and FIX2) and the FIXED DATA SWITCHES on an ALTERNATE I/O ADAPTER (SWITCH A and SWITCH B). FIXTST4 is a completely self-contained diagnostic program, with a minimum RAM requirement of 4K.

Loading

FIXTST4 can be loaded into memory using any convenient method (see Appendix B). Upon loading properly execution will begin, and the command line will be displayed (see Appendix A).

User Interaction

Immediately after the command line is displayed a question about which FIXED DATA SWITCHES are to be tested will appear. FIXTST4 defaults to testing FIX1 and FIX2 only. To change the defaults type a "Y" or "N" next to the desired switch message. This will cause the appropriate letter to appear in the box pointed to by the greater than sign and wll advance the greater than sign to the next box in cyclic order. FIXTST4 will only respond to "Y" or "N" for yes or no respectively, the Prog key to restart the program, and the return key when the user decides that the information on the display is correct.

If the user indicated that SWITCH A or SWITCH B is to be tested the address of the ALTERNATE I/O ADAPTER is required. At this time the keyboard will only respond to a valid address (XD where X is 1, 2, 4 or 8), the depression of the return key to indicate that the address displayed is correct, or the Prog key to restart the program. After this, the program will wait for run-mode information (see Appendix A).

In the right corner of the display the original status, in hexadecimal, of the FIXED DATA SWITCHES will appear in the order of FIX1, FIX2, SWITCH A, and SWITCH B. If a switch is not being tested X's will appear instead. Once a valid value appears it will not be changed.

Once a test has been entered, a command message on line two will appear:

T0:	PUT ALL SWITCHES LOW (FIX1 and FIX2)
T1:	PUT FIX1 SWITCH 1 HIGH, ALL OTHERS LOW
T2:	PUT FIX1 SWITCH 2 HIGH, ALL OTHERS LOW
T3:	PUT FIX1 SWITCH 3 HIGH, ALL OTHERS LOW
T4:	PUT FIX1 SWITCH 4 HIGH, ALL OTHERS LOW
T5:	PUT FIX1 SWITCH 5 HIGH, ALL OTHERS LOW
T6:	PUT FIX1 SWITCH 6 HIGH, ALL OTHERS LOW
T7:	PUT FIX1 SWITCH 7 HIGH, ALL OTHERS LOW
T8:	PUT FIX1 SWITCH 8 HIGH, ALL OTHERS LOW
T9:	PUT FIX2 SWITCH 1 HIGH, ALL OTHERS LOW
T0AH:	PUT FIX2 SWITCH 2 HIGH, ALL OTHERS LOW
T0BH:	PUT FIX2 SWITCH 3 HIGH, ALL OTHERS LOW
T0CH:	PUT FIX2 SWITCH 4 HIGH, ALL OTHERS LOW
T0DH:	PUT FIX2 SWITCH 5 HIGH, ALL OTHERS LOW
T0EH:	PUT FIX2 SWITCH 6 HIGH, ALL OTHERS LOW
T0FH:	PUT FIX2 SWITCH 7 HIGH, ALL OTHERS LOW
T010H:	PUT FIX2 SWITCH 8 HIGH, ALL OTHERS LOW

T011H:	PUT ALL SWITCHES HIGH (FIX1 and FIX2)
T012H:	PUT ALL SWITCHES LOW (SW A and SW B)
T013H:	PUT SW A SWITCH 0 HIGH, ALL OTHERS LOW
T014H:	PUT SW A SWITCH 1 HIGH, ALL OTHERS LOW
T015H:	PUT SW A SWITCH 2 HIGH, ALL OTHERS LOW
T016H:	PUT SW A SWITCH 3 HIGH, ALL OTHERS LOW
T017H:	PUT SW A SWITCH 4 HIGH, ALL OTHERS LOW
T018H:	PUT SW A SWITCH 5 HIGH, ALL OTHERS LOW
T019H:	PUT SW A SWITCH 6 HIGH, ALL OTHERS LOW
T01AH:	PUT SW A SWITCH 7 HIGH, ALL OTHERS LOW
T01BH:	PUT SW B SWITCH 0 HIGH, ALL OTHERS LOW
T01CH:	PUT SW B SWITCH 1 HIGH, ALL OTHERS LOW
T01DH:	PUT SW B SWITCH 2 HIGH, ALL OTHERS LOW
T01EH:	PUT SW B SWITCH 3 HIGH, ALL OTHERS LOW
T01FH:	PUT SW B SWITCH 4 HIGH, ALL OTHERS LOW
T020H:	PUT SW B SWITCH 5 HIGH, ALL OTHERS LOW
T021H:	PUT SW B SWITCH 6 HIGH, ALL OTHERS LOW
T022H:	PUT SW B SWITCH 7 HIGH, ALL OTHERS LOW
T023H:	PUT ALL SWITCHES HIGH (SW A and SW B)

Set the FIXED DATA SWITCHES that are being tested to the appropriate position, and depress the space bar to continue, or the Prog key to restart.

If at this time an error was encountered a message will appear on lines 3 and 4:

```
FIX1 EXP: XX ACT: XX
FIX2 EXP: XX ACT: XX
```

or

```
SW A EXP: XX ACT: XX
SW B EXP: XX ACT: XX
```

where X is any hexidecimal number.

Execution is now transferred back to the test module (see Appendix A).

When all the tests are complete, to get the original values of the FIXED DATA SWITCHES depress the return key.

Set the FIXED DATA SWITCHES that are being tested to the appropriate position, and depress the space bar to continue, or the Prog key to restart.

The OP-1/15 switch configurations are:

Switch #	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
Data Bit #	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
	FIX 1	FIX 2

Rear View

The switch configurations for all other machines are:

Switch #	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
Data Bit #	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

If at this time an error was encountered a message will appear on lines 3 and 4:

```
FIX1 EXP: XX ACT: XX
FIX2 EXP: XX ACT: XX
```

or

```
SW A EXP: XX ACT: XX
SW B EXP: XX ACT: XX
```

where X is any hexidecimal number.

Execution is now transferred back to the test module (see Appendix A).

When all the tests are complete, to get the original values of the FIXED DATA SWITCHES depress the return key.

PRNTST4

PRINTER ADAPTER TEST

PRNTST4

Alternate I/O Printer Adapter Test for Centronics/Okidata/Printronix

Applicable Assemblies

5000-1172	Alternate I/O Printer Adapter
5100-E-XYZAB2D	OP-1/R Main Logic Board with Printer
5100-1106-1Y2	OP-1/R Alternate I/O Printer Adapter

General Description

The purpose of the PRNTST4 program is to determine if the Centronics, Okidata or Printronix Printer and the Alternate I/O Printer Adapter are working properly, and, if not, to give an indication of which functions are incorrect. The program requires operator assistance to perform certain functions and to analyze the PRINTOUT (Figures 1-6) at the completion of the last test.

4K of memory is required to run PRNTST4.

Loading Procedure

PRNTST4 can be loaded into memory using any convenient method (see Appendix B). When PRNTST4 loads properly, it will identify itself and await run-mode input by the operator (see Appendix A).

Errors

Even if PRNTST4 proceeds through all the tests without a displayed error, there could still be a printed error since PRNTST4 has no way to examine characters printed by the Printer. After the final test the operator must compare the print-out produced by the tests to the correct corresponding PRINTOUT shown in Figures 1-6. Also, during Test 20, the operator must listen for the Printer to beep as the word "BELL" is printed.*

* This test is not applicable on the Printronix.

User Initialization

When the program is loaded, a four part user initialization dialog is executed as follows:

1. Since the Printer Adapter has more than one select address, the message:

   "Enter Hex Select Address"

   is displayed. A two digit hexadecimal number is expected.

2. Since the Printer Adapter may be inserted in any one of the slots from 7-10, the message:

   "Enter Priority Level"

   is displayed. A one digit number ranging from 3-6 is expected. This number represents the interrupt priority. If this test is being run on the OP-1/R main board Printer Adapter, enter 3.

3. The user is asked to input the type of printer being used. When the message:

   "Type P for Printronix, O for Okidata, or C for Centronics"

   is displayed. The user should enter the proper code.

4. Finally the user is asked:

   "Type 0 for 80 Column Printer, 1 for 132 Column Printer

   and should enter the proper code.

Initialization will not advance to the next part until a valid entry is made on the current part.

If the user enters a valid code and discovers that it is incorrect, typing PROG will restart the initialization.

At the conclusion of part four, the Diagnostic program will begin execution.

Test 00	Test that the Printer Adapter cannot be selected with an incorrect select address. Incorrect select addresses are all one byte bit configurations of three or less bits high.
Test 01	Tests that the Printer Adapter can be selected with the proper select address (entered during user initialization). If this test fails the user should check the select address of the Printer Adapter with that entered during initialization.
Test 02	Tests that a selected Printer Adapter can be deselected with the issuing of an INIT instruction.
Test 03	Tests that a selected printer can be detected. If this test fails, IFL bit 6 is stuck low.
Test 04	Tests that a deselected printer can be detected. If this test fails, IFL bit 6 is stuck high.
Test 05	Tests that a not ready printer can be detected. If this test fails, IFL bit 1 is stuck low.
Test 06	Tests that a ready printer can be detected. If this test fails, IFL bit 1 is stuck high.
Test 07	Tests that the Not Busy flag is not stuck low by issuing an INIT and a DVCL to the selected Printer Adapter. If this test fails, IFL bit 7 is stuck low.
Test 08	Tests that the Not Busy flag goes low after an OUT command is issued. If this test fails, IFL bit 7 is stuck high.
Test 09	Tests that after issuing an OUT command to reset the Not Busy flag, that issuing an INIT causes the Not Busy flag to set. If this test fails, IFL bit 7 is bad.
Test 0A	Tests that after issuing an OUT command to reset the Not Busy flag, that issuing a DVCL causes the Not Busy flag to set. If this test fails, IFL bit 7 is bad.
Test 0B	Tests that the Not Busy flag goes high after an OUT, when given sufficient time. If this test fails, IFL bit 7 is bad.
Test 0C	Tests that after issuing an INIT and DVCL to the selected Printer Adapter, the Printer Busy flag is not stuck high. If this test fails, IFL bit 0 is stuck high. This test is only applicable with the use of the Okidata Printer. The other two printers will cause no error to occur.*
Test 0D	Tests that the Printer Busy flag goes high after an OUT. If this test fails, IFL bit 0 is bad.

Test 0E Tests that after issuing an OUT command, the issuing of an INIT resets the Printer busy flag. If this test fails, IFL bit 0 is bad. This test is only applicable with use of the Okidata. The other two printers will not cause an error to occur.*

Test 0F Tests that after issuing an OUT command, the issuing of a DVCL resets the Printer Busy flag. If this test fails, IFL bit 0 is bad. This test is only applicable with use of the Okidata. The other two printers will not cause an error to occur.*

Test 10 Tests that the Printer Busy flag will go low after an OUT, when given sufficient time. If this test fails, bit 0 is bad.

Test 11 Test that after issuing an OUT command, the issuing of a DVCL clears the printer internal line buffer. An error is indicated by the message, "TEST 11 ERROR" on the printer. This test is not applicable with use of a Printronix Printer. No error condition will be made with use of Printronix.*

Test 12 Tests that after issuing an OUT command, the issuing of an INIT clears the printer internal line buffer. An error is indicated by the message, "TEST 12 ERROR" on the printer. This test is not applicable with use of a Printronix Printer. No error condition will be made with use of Printronix.*

Test 13 Tests that a selected printer is deselected when sent a deselect command. If this test fails, Tests 03 and 04 should be run again to test IFL bit 6. If those two tests pass, then the printer does not respond to a deselect command. This test is not applicable with use of the Printronix Printer. No error indication will be made with a Printronix.*

Test 14 Tests that a selected printer is selected when sent a select command. If this test fails, Tests 03 and 04 should be run again to test IFL bit 6. If those two tests pass, then the printer does not respond to a select command. This test is not applicable with use of the Printronix Printer. No error indication will be made with a Printronix.*

Test 15 Tests that the Not Busy flag goes low when sent a deselect command during an OUT. If this test fails, Tests 07 - 0B should be run. If any of these tests fail, then bit 7 is bad. Otherwise, the printer does not respond to a deselect command during a print. This test is not applicable with use of the Printronix Printer. No error indication will be made with the Printronix.*

Test 16 Tests that an interrupt is generated when the Not Busy flag is high, the interrupt mask is set to allow an interrupt only from the Printer Adapter, and the interrupts are enabled. If this test fails, the interrupt priority for the slot that the board is in should be checked with that entered during initialization. If the interrupt priorities match then the interrupt is bad.

Test 17 Tests that no interrupt is generated when the Not Busy flag is low, the interrupt mask is set to allow an interrupt only from the Printer Adapter, and the interrupts are enabled - disabled. If this test fails, the interrupt priority for the slot that the board is in should be checked with that entered during initialization. If the interrupt priorities match then the interrupt is bad.

Tests 18-21 are visual tests.

Tests 18-1B should be run in sequence.

Test 18 Test of Form Feed. Paper should advance to top of form. "TOP OF FORM" should be printed on top line.

Test 19 Test of carriage return/line feed. Paper should advance one line "NEXT LINE DOWN" should be printed.

Test 1A Test of Carriage Return. On Okidata Printer the issuing of a Carriage Return also causes a line feed to occur. The message printed should be:

")))))) ERROR IF X's BEFORE THIS"

On Centronics Printers the internal buffer is printed and the print head is positioned at the beginning of that line. The message printed should be "XXXXXX ERROR IF NOT X's BEFORE THIS".

On the Printronix Printer the internal line buffer is not printed and the internal buffer pointer is positioned to the start of the buffer. The message: "////// ERROR IF X's BEFORE THIS".

Test 1B Test of Vertical Tab. The paper should advance to the sixth line and the message printed should be: "6 LINES BELOW TOP OF FORM". The vertical tab on the Printronix requires the use of the VFO therefore, three carriage return/line feeds are issued and the Vertical Tab is not tested.

Test 1C Tests that all characters are printable.

SECTION 1 6 slewed lines of upper case characters ASCII codes 20-5F hex.

SECTION 2 6 slewed lines of lower case characters ASCII codes 60-7E hex, repeated.

If the printer does not support lower case characters, Section 2 will be printed in upper case.

Test 17

Tests that no interrupt is generated when the Not Busy flag is low, the interrupt mask is set to allow an interrupt only from the Printer Adapter, and the interrupts are disabled. If this test fails, the interrupt priority for the slot that the board is in should be checked with that entered during initialization. If the interrupt priorities match then the interrupt is bad.

Tests 18-21 are visual tests.

Tests 18-1B should be run in sequence.

Test 18

Test of Form Feed. Paper should advance to top of form. "TOP OF FORM" should be printed on top line.

Test 19

Test of carriage return/line feed. Paper should advance one line "NEXT LINE DOWN" should be printed.

Test 1A

Test of Carriage Return. On Okidata Printer the issuing of a Carriage Return also causes a line feed to occur. The message printed should be:

"⟩⟩⟩⟩⟩⟩ ERROR IF X's BEFORE THIS"

On Centronics Printers the internal buffer is printed and the print head is positioned at the beginning of that line. The message printed should be "XXXXXX ERROR IF NOT X's BEFORE THIS".

On the Printronix Printer the internal line buffer is not printed and the internal buffer pointer is positioned to the start of the buffer. The message: "////// ERROR IF X's BEFORE THIS".

Test 1B

Test of Vertical Tab. The paper should advance to the sixth line and the message printed should be: "6 LINES BELOW TOP OF FORM". The vertical tab on the Printronix requires the use of the VFO therefore, three carriage return/line feeds are issued and the Vertical Tab is not tested.

Test 1C

Tests that all characters are printable.

SECTION 1	6 slewed lines of upper case characters ASCII codes 20-5F hex.
SECTION 2	6 slewed lines of lower case characters ASCII codes 60-7E hex, repeated.

If the printer does not support lower case characters, Section 2 will be printed in upper case.

Test 1D Test double width characters. The characters with ASCII codes 20-48 hex will be printed double width. The Printronix does not support double width characters.*

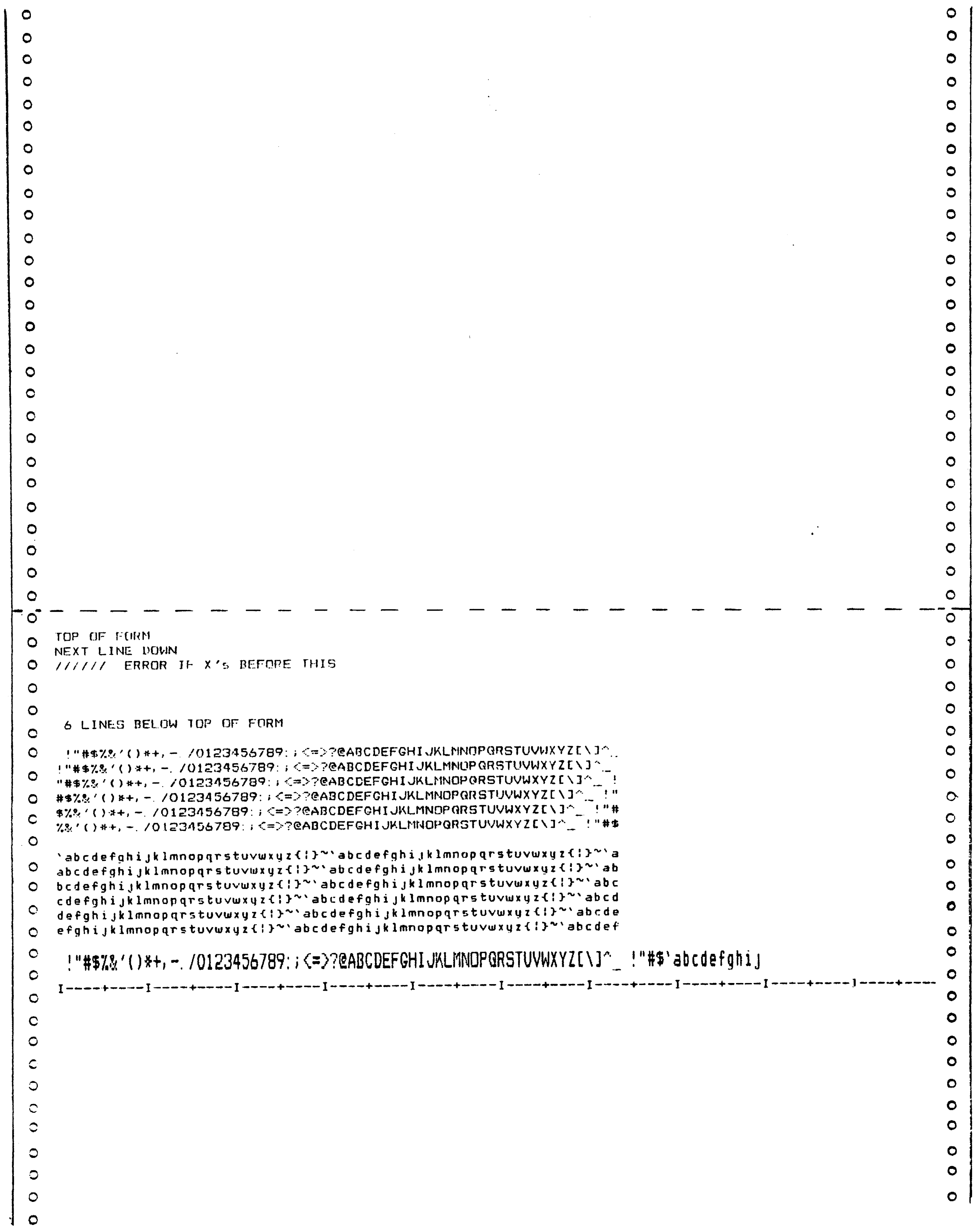
Test 1E Test double height characters. Characters with ASCII codes 20-6A hex will be printed double height. Centronics does not support double height.*

Test 1F Test expanded characters. Characters with ASCII codes 20-48 hex will be printed expanded. Centronics and Printronix do not support expanded characters.*

Test 20 Test Printer bell. The printer bell should be audible. The message "BELL" should be printed. Printronix does not support the Bell.*

Test 21 Test all colums printable. A line of length 80 characters or 132 characters (depending on the entry during user initialization) of dashes with tick marks every five places starting with the first column will be printed.

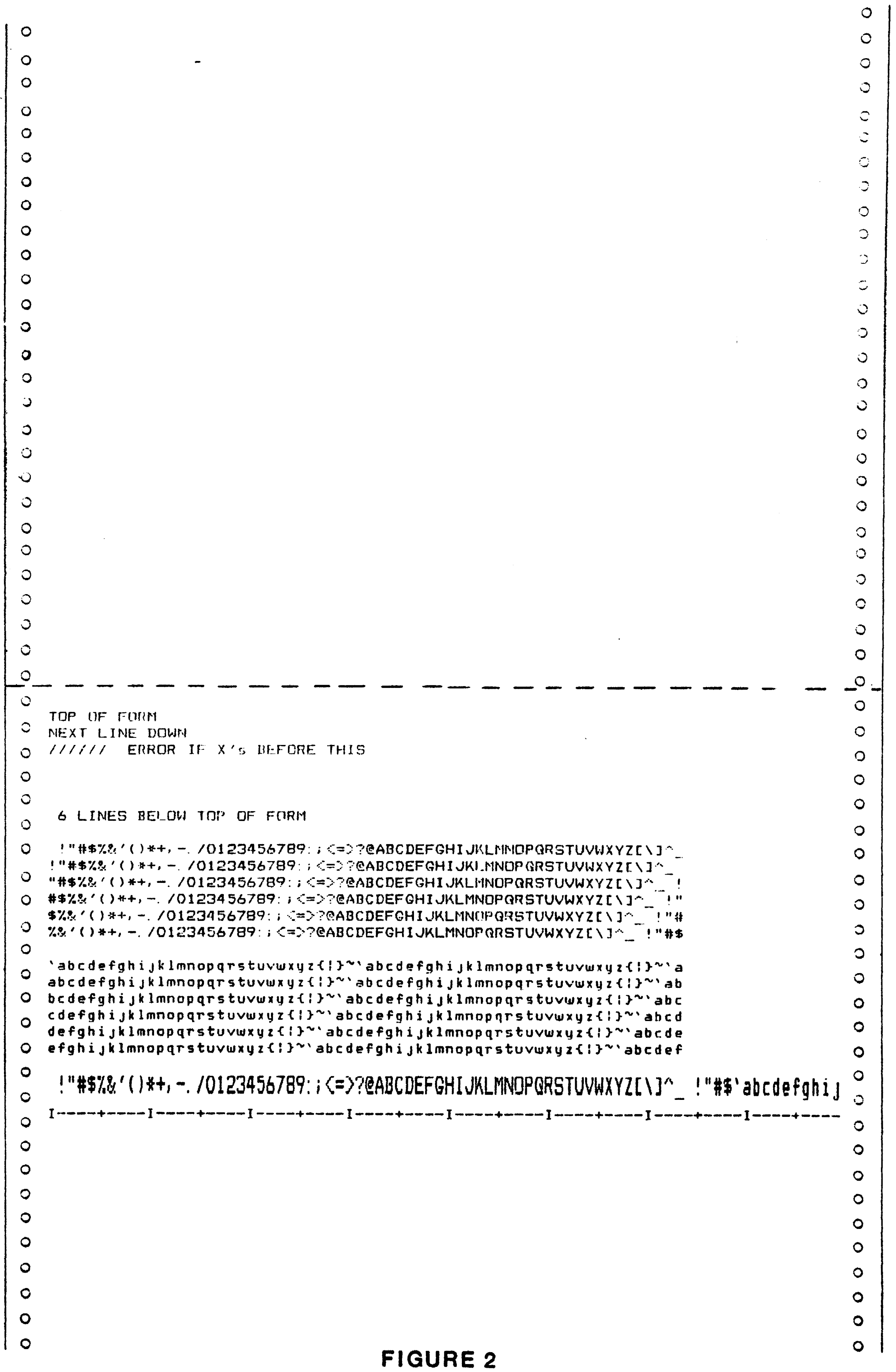
* An error is not generated because this test is not executed for non-applicable printers.



```
TOP OF FORM
NEXT LINE DOWN
//////  ERROR IF X's BEFORE THIS



 6 LINES BELOW TOP OF FORM

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"
$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"#
%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"#$

`abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`a
abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`ab
bcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abc
cdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcd
defghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcde
efghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcdef

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"#$`abcdefghij
I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----
```

FIGURE 1

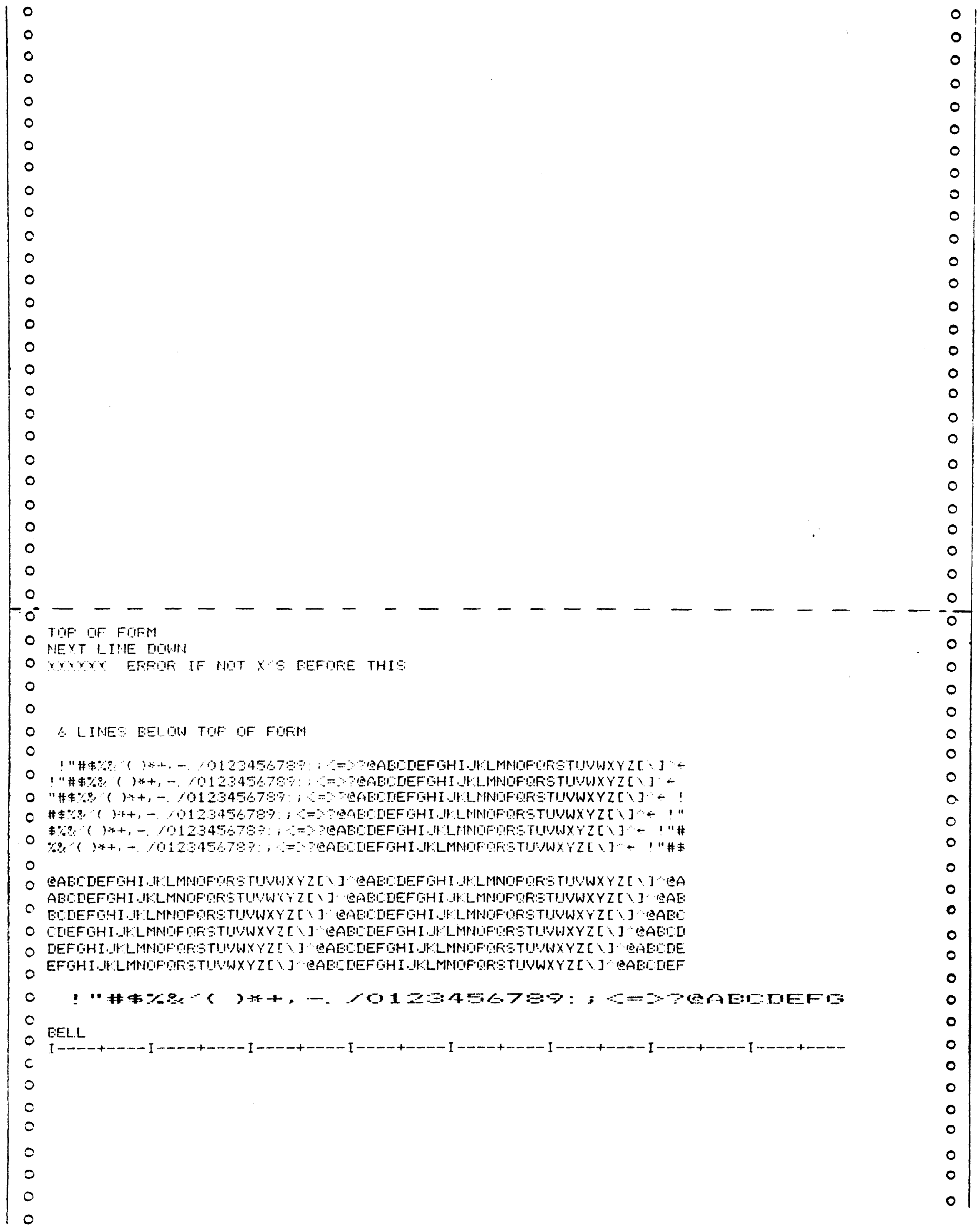
132 COLUMN PRINTRONIX



```
TOP OF FORM
NEXT LINE DOWN
//////  ERROR IF X's BEFORE THIS


 6 LINES BELOW TOP OF FORM

  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
 "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !
 #$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"
 $%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"#
 %&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"#$

 `abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`a
 abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`ab
 bcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abc
 cdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcd
 defghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcde
 efghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcdef

  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_ !"#$`abcdefghij
 I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----
```

FIGURE 2

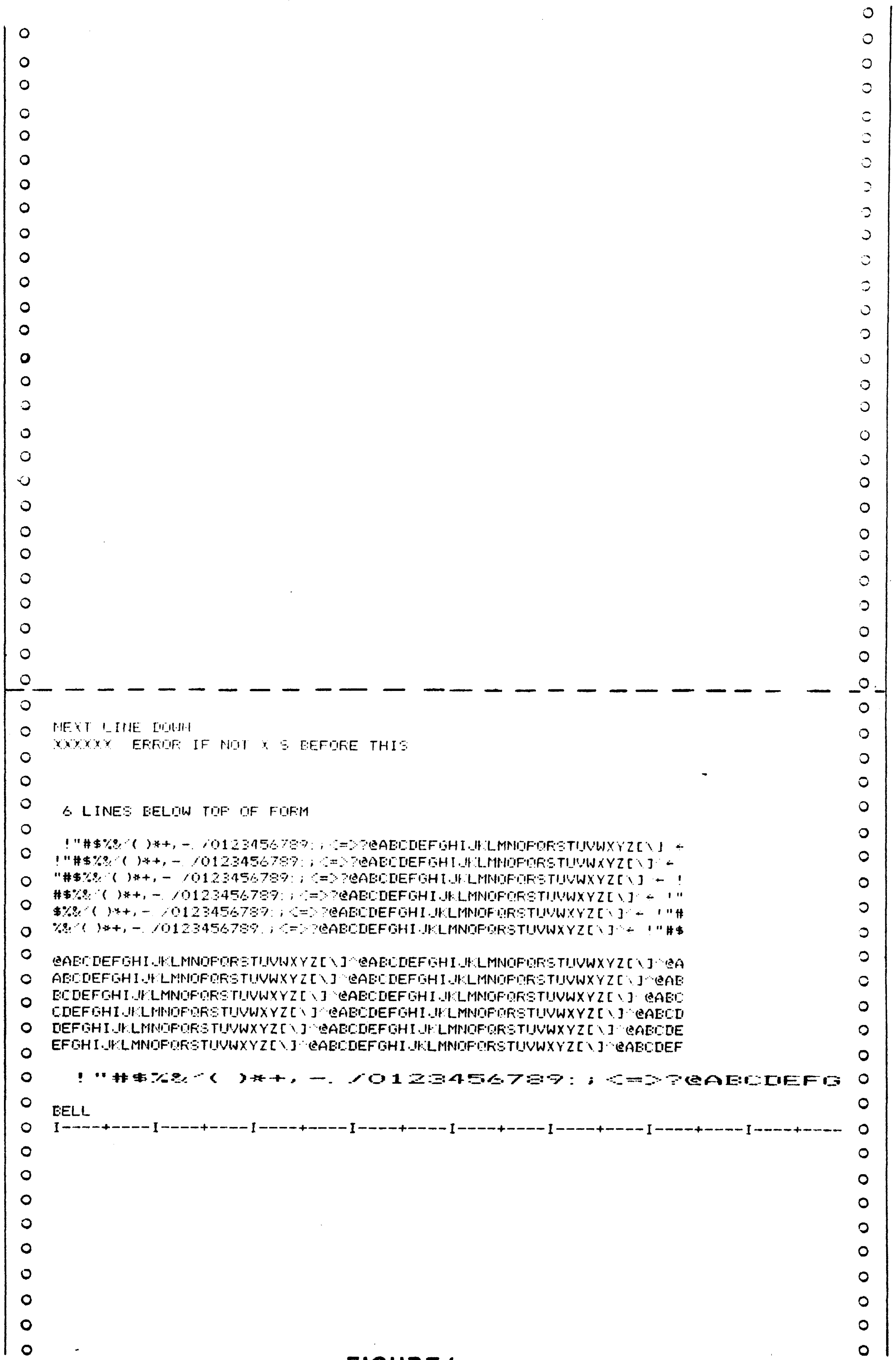
80 COLUMN PRINTRONIX



```
TOP OF FORM
NEXT LINE DOWN
XXXXXX  ERROR IF NOT X'S BEFORE THIS


 6 LINES BELOW TOP OF FORM

  !"#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^←
 !"#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^←
"#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !
#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !"
$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !"#
%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !"#$

@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@A
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@AB
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABC
CDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCD
DEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDE
EFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEF

  ! " # $ % & ' (   ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G

BELL
I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----
```

FIGURE 3

132 COLUMN CENTRONICS



```
NEXT LINE DOWN
XXXXXX  ERROR IF NOT X S BEFORE THIS


 6 LINES BELOW TOP OF FORM

 !"#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^←
!"#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^←
"#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !
#$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !"
$%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !"#
%&'( )*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^← !"#$

@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@A
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@AB
BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABC
CDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCD
DEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDE
EFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^@ABCDEF

  ! " # $ % & ' (  ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G

BELL
I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----
```

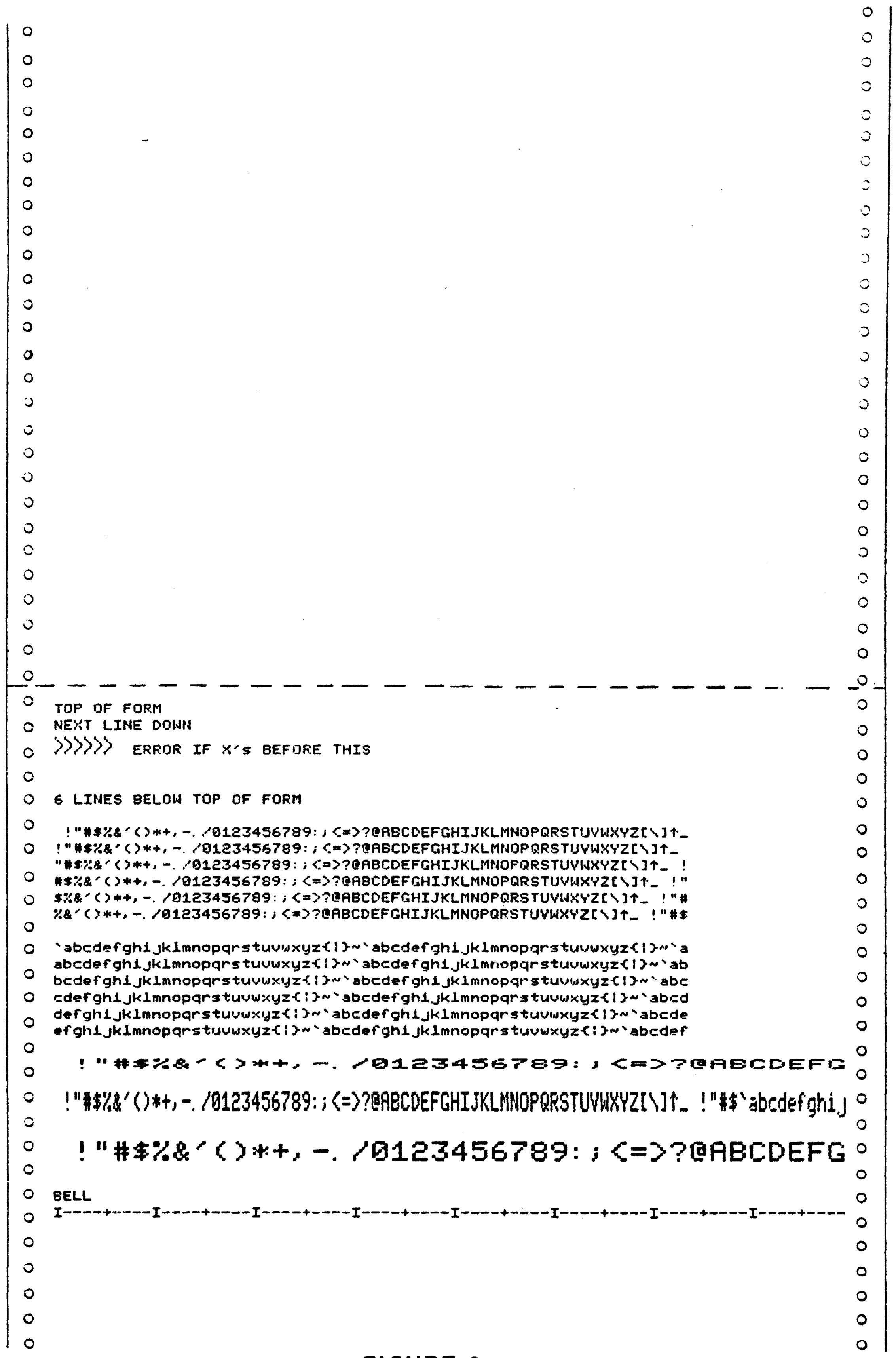
FIGURE 4

80 COLUMN CENTRONICS

```
TOP OF FORM
NEXT LINE DOWN
>>>>>>  ERROR IF X's BEFORE THIS


6 LINES BELOW TOP OF FORM

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"
$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"#
%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"#$

`abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`a
abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`ab
bcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abc
cdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcd
defghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcde
efghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcdef

  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"#$`abcdefghij

  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G

BELL
I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----
```

FIGURE 5

132 COLUMN OKIDATA



```
TOP OF FORM
NEXT LINE DOWN
>>>>>>  ERROR IF X's BEFORE THIS


6 LINES BELOW TOP OF FORM

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"
$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"#
%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"#$

`abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`a
abcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`ab
bcdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abc
cdefghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcd
defghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcde
efghijklmnopqrstuvwxyz{|}~`abcdefghijklmnopqrstuvwxyz{|}~`abcdef

  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_ !"#$`abcdefghij

  ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G

BELL
I----+----I----+----I----+----I----+----I----+----I----+----I----+----I----+----
```

FIGURE 6

80 COLUMN OKIDATA

IOTST4/IOTSTM

ASYNCHRONOUS I/O ADAPTER TEST

ASYNCHRONOUS I/O ADAPTER TEST

Applicable Assemblies			Applicable Program
5000-1141-X	8080 CPU Board	(X=1-2)	IOTST4
5000-E0XYZABCD	OP-1/R Main Logic Board	(B=1-3)	IOTST4
5000-11114-X	8085 CPUM Board	(X=1)	IOTSTM
	8085	(X=3)	IOTST15
	8085 CPUM Board	(X=3)	IOT2WM
	8085		IOT2W15

Required Test Assemblies

IOTST80 Test Plug (RS232 Compatible, X or B=1)
COMTST2 Test Plug (TTY Compatible, X or B=2)
no plug required (2 Wire Direct, X=3)

General Description

The purpose of the test program is to determine if the Asynchronous I/O Adapter is working properly and, if not, to give an indication of which functions are incorrect. The corresponding test plug is needed to run. The program requires no operator interaction unless an error is encountered.

4K (or more) of memory is required to run IOTST4/IOTSTM.

Loading Procedure (IOTST4)

IOTST4 can be loaded into memory using any convenient method (see Appendix B). When IOTST4 loads properly it will display the following message on the screen:

OP-1/R 2008? TYPE (Y)es; (N)o

The operator should type 'Y' or 'N' depending on whether or not the unit being tested is an OP1-R 2008 logic board. When the question has been responded to, IOTST4 will identify itself and await run-mode input (see Appedix A).

Loading Procedure (IOTSTM/IOTST15/IOT2WM/IOT2W15)

IOTSTM can be loaded into memory using any convenient method (see Appendix B). IOTSTSM will identify itself and await run-mode input (see Appendix A).

Errors

All errors are indicated by an error indicator on the display screen and the simultaneous activation of the bell. The error displayed on the screen refers to the detailed description of the specific test to determine the purpose and expected results for the displayed error indicator.

After an error indicator is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing at the next test, typing the R key (without the SHIFT key) will repeat the current test, and depressing the PROG key will restart the program.

Because of the inverted pyramid test strategy used, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages in subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

IOTST80 Test Plug

The IOTST80 Test Plug is necessary for testing the Asynchronous I/O Adapter RS232 compatible. The plug must be inserted into the Asynchronous I/O Adapter connector (as shown in Figure 1) on the rear panel of the OP-1 or OP-1/R prior to program execution. The schematic of the IOTST80 Test Plug is shown in Figure 2.

COMTST2 Test Plug

The COMTST2 Test Plug is necessary for testing the Asynchronous I/O Adapter TTY compatible (20ma). The plug must be inserted into the Asynchronous I/O Adapter connector (as shown in Figure 1) on the rear panel of the OP-1 or OP-1/R prior to program execution. The schematic of the COMTST2 Test Plug is shown in Figure 3, as well as the Logic Flow diagram.

Test Description

All test operations are described in this section. The program will halt and the specified error indicator is displayed if expected results are not obtained.

On the following pages, each test is listed with a description of what is being tested, with an explanation of the cause of the error.

Initialization

When IOTST4 loads, it will determine whether it is running on an OP-1 or an OP-1/R and make the appropriate timing adjustments.

The diagnostic will then prompt the user in order to determine if the board supports 38,400 and 50,000 baud. After responding to this prompt, the diagnostic will inquire as to whether the unit under test is an OP-1/R 1101, an OP-1/RW 2008, or something else. In the case of an OP-1/R 1101, the user will be further prompted in order to determine if PCO338 is on the board.

If IOTSTM is run, the diagnostic will prompt the user in order to determine if 38,400 and 50,000 baud are supported.

T00: Tests that the Asynchronous I/O Adapter does not respond to an incorrect select address. Addresses of all combinations of three bits or less high are used to try to select the board.

T01: Tests that the Asynchronous I/O Adapter does respond to the correct select address. The address F0 Hex is used to try to select the board.

T02: Tests that the selected Asynchronous I/O Adapter can be deselected by issuing an INIT.

T03: Tests that, by issuing an INIT, DVCL, and an OFL with a command byte of 0 Hex, the Data Set Ready flag is not stuck high. For 2-Wire direct versions tests that bit 7 of Rim, transmit receive signal can be reset.

T04: Tests that, by issuing an INIT, DVCL, and an OFL with a command byte of 0 Hex, the Clear To Send Flag is not stuck high. Omitted for 2-wire.

T05: Tests that, by issuing an INIT, DVCL, and OFL with a command byte of 0 Hex, the Carrier Detect signal flag is not stuck high. Executed only for OP1-R. Omitted for 2-wire.

T06: Tests that, by issuing an OFL with a command byte set to Break, the Data Set Ready flag is not stuck low. For 2-wire direct tests that bit 7 of Rim, transmit receive signal can be set.

T07: Tests that, by issuing an OFL with a command byte set for Request to Send, the Clear to Send flag is not stuck low. Omitted for 2-wire direct.

T08: Tests that, by issuing an OFL with a command byte set for Data Terminal Ready, the Carrier Detect Signal flag is not stuck low. Executed only for OP1-R. Omitted for 2-wire.

T09: Tests that, after issuing an OFL with command byte set to Break, to set the Data Set Ready flag, an INIT resets the Data Set Ready flag. For 2-wire tests that after setting bit 7 of Rim, issuing an INIT will reset it.

T0A: Tests that, after issuing an OFL with command byte set for Request to Send, to set the Clear to Send flag, an INIT resets the Clear to Send flag. Omitted for 2-wire.

T0B: Tests that, after issing an OFL with command byte set for Data Terminal Ready, to set the Carrier Detect Signal flag, and INIT resets the Carrier Detect Signal flag. Executed only for OP1-R.
For 2001: Tests that INIT does not reset Carrier Detect. Omitted for 2-wire.

T0C:	Tests that, after issuing an OFL with command byte set to Break, to set the Data Set Ready flag, a DVCL resets the Data Set Ready flag. Tests that after setting Bit 7 of Rim, issuing a DVCL will reset it.
T0D:	Tests that, after issuing an OFL with command byte set for Request to Send, to set the Clear to Send flag, a DVCL resets the Clear to Send flag. Omitted for 2-wire.
T0E:	Tests that, after issuing an OFL with command byte set for Data Terminal Ready, to set the Carrier Detect Signal flag, a DVCL resets the Carrier Detect Signal flag. Executed only for OP1-R. For 2008: Tests that DVCL does <u>not</u> reset Carrier Detect. Omitted for 2-wire.
T0F:	Tests that, after issuing an OFL with command byte set to Break, to set the Data Set Ready flag, an OFL with command byte of 0 Hex resets the Data Set Ready flag. Tests that after setting Bit 7 of Rim, issuing an OFL will reset it.
T10:	Tests that, after issuing an OFL with command byte set for Request to Send, to set the Clear to Send flag, an OFL with command byte of 0 Hex resets the Clear to Send flag.
T11:	Tests that, after issuing an OFL with command byte set for Data Terminal Ready, to set the Carrier Detect Signal flag, an OFL with command byte of 0 Hex resets the Carrier Detect signal flag. Executed only for OP1-R.
T12:	Tests that, by issuing an INIT and DVCL, the Character Needed for Transmission flag is not stuck low.
T13:	Tests that, by issuing two successive OUT commands, the Character Needed for Transmission flag is not stuck high.
T14:	Tests that, given sufficient time, the Character Needed for Transmission flag goes high after one OUT command.
T15:	Tests that, given sufficient time, the Character Needed for Transmission flag goes high after two OUT commands.
T16:	Tests that, the Character Needed for Transmission flag does not go high during tranmission.

T17: Tests that, a DVCL sets the Character Needed for Transmission flag during a transmission.

T18: Tests that, an INIT sets the Character Needed for Transmission flag during a transmission.

T19: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 19,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T1A: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 9,600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T1B: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 7,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T1C: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 4,800 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T1D: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 3,600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T1E: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 2,400 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T1F: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 2,000 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T20: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 1,800 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T21: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 1,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T22: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T23: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 300 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T24: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 150 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T25: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 135.5 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T26: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 110 baud and checking that the time needed to trasmit the 10 bits does not take less than the time needed to transit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T27:	Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 75 (50,000 if so configured) baud and checking that the time needed to transit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).
T28:	Tests that the transmitter is working by transmitting 8 data bits,1 stop bit and 1 start bit, at 50 (38,000 if so configured) baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).
T29:	Tests that, with the interrupts enabled and the Character Needed for Transmission flag high, a SMSK with command byte of 0 Hex holds off the transmit interrupt.
T2A:	Tests that, with the interrupts enabled and the Character Needed for Transmission flag high, a priority 2 interrupt is generated.
T2B:	Tests that, with the interrupts enabled and the Character Needed for Transmission flag low, no interrupt is generated.
T2C:	Tests that the transmitter correctly transmits 7 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. all characters 00..7F Hex are transmitted at 300 baud and displayed on the screen.
T2D:	Tests that the transmitter correctly transmits 8 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. all characters 00.0F Hex are transmitted at 300 baud and displayed on the screen.
T2E-37:	Tests that character length, parity, and stop bits are transmitted correctly by comparing the Data Set Ready flag fluctuations with the expected bit pattern of the transmitted character. The even parity character has an ASCII value of 55 Hex and the odd parity character has an ASCII value of 52 Hex.

T27: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 75 baud and checking that the time needed to transit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T28: Tests that the transmitter is working by transmitting 8 data bits,1 stop bit and 1 start bit, at 50 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (9) or more than the maximum amount of bits (12).

T29: Tests that, with the interrupts enabled and the Character Needed for Transmission flag high, a SMSK with command byte of 0 Hex holds off the transmit interrupt.

T2A: Tests that, with the interrupts enabled and the Character Needed for Transmission flag high, a priority 2 interrupt is generated.

T2B: Tests that, with the interrupts enabled and the Character Needed for Transmission flag low, no interrupt is generated.

T2C: Tests that the transmitter correctly transmits 7 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. all characters 00..7F Hex are transmitted at 300 baud and displayed on the screen.

T2D: Tests that the transmitter correctly transmits 8 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. all characters 00.0F Hex are transmitted at 300 baud and displayed on the screen.

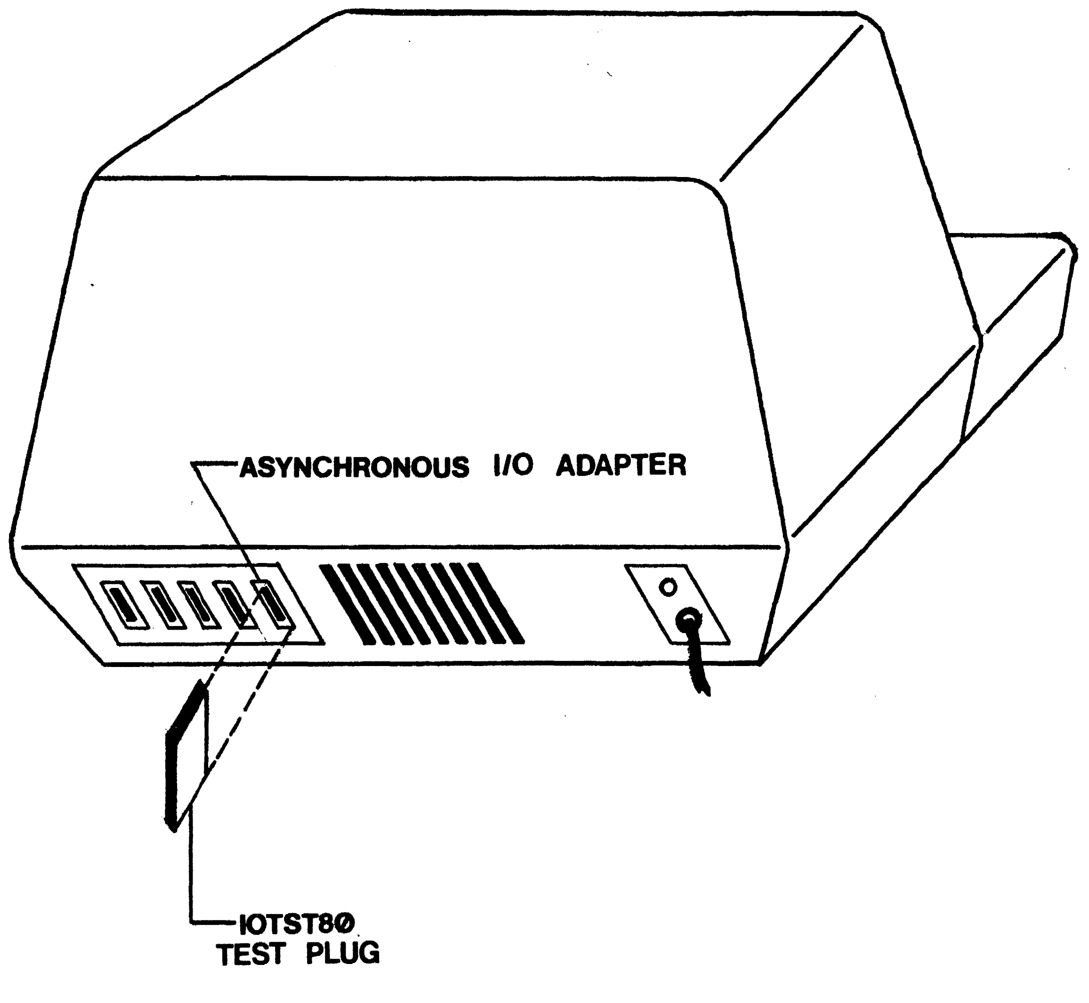
T2E-37: Tests that character length, parity, and stop bits are transmitted correctly by comparing the Data Set Ready flag fluctuations with the expected bit pattern of the transmitted character. The even parity character has an ASCII value of 55 Hex and the odd parity character has an ASCII value of 52 Hex.

T2E:	Tests 8 Data bits, 1 stop bit, and no parity, with the transmission of a character of odd parity.
T2F:	Tests 8 Data bits, 1 stop bit, and no parity, with the transmission of a character of even parity.
T30:	Tests 8 Data bits, 2 stop bits, and no parity, with the transmission of a character of odd parity.
T31:	Tests 8 Data bits, 2 stop bits, and no parity, with the transmission of a character of even parity.
T32:	Tests 8 Data bits, 1 stop bit, and even parity with the transmission of a character of odd parity.
T33:	Tests 8 Data bits, 1 stop bit, and even parity with the transmission of a character of even parity.
T34:	Tests 8 Data bits, 1 stop bit, and odd parity with the transmission of a character of odd parity.
T35:	Tests 8 Data bits, 1 stop bit, and odd parity with the transmission of a character of even parity.
T36:	Tests 7 Data bits, 1 stop bit, no parity, with the transmission of a character of odd parity.
T37:	Tests 7 Data bits, 1 stop bit, no parity, with the transmission of a character of even parity.
T38-T47:	Tests that the time boundaries are within +/- 10% of the actual time needed for transmission of 12 bits.
T38:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 19,200 baud.

T39:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 9,600 baud.
T3A:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 7,200 baud.
T3B:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 4,800 baud.
T3C:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 3,600 baud.
T3D:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 2,400 baud.
T3E:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 2,000 baud.
T3F:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 1,800 baud.
T40:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 1,200 baud.
T41:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 600 baud.
T42:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 300 baud.
T43:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 150 baud.
T44:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 135.5 baud.
T45:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 110 baud.
T46:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 75 baud (50,000 if so configured).

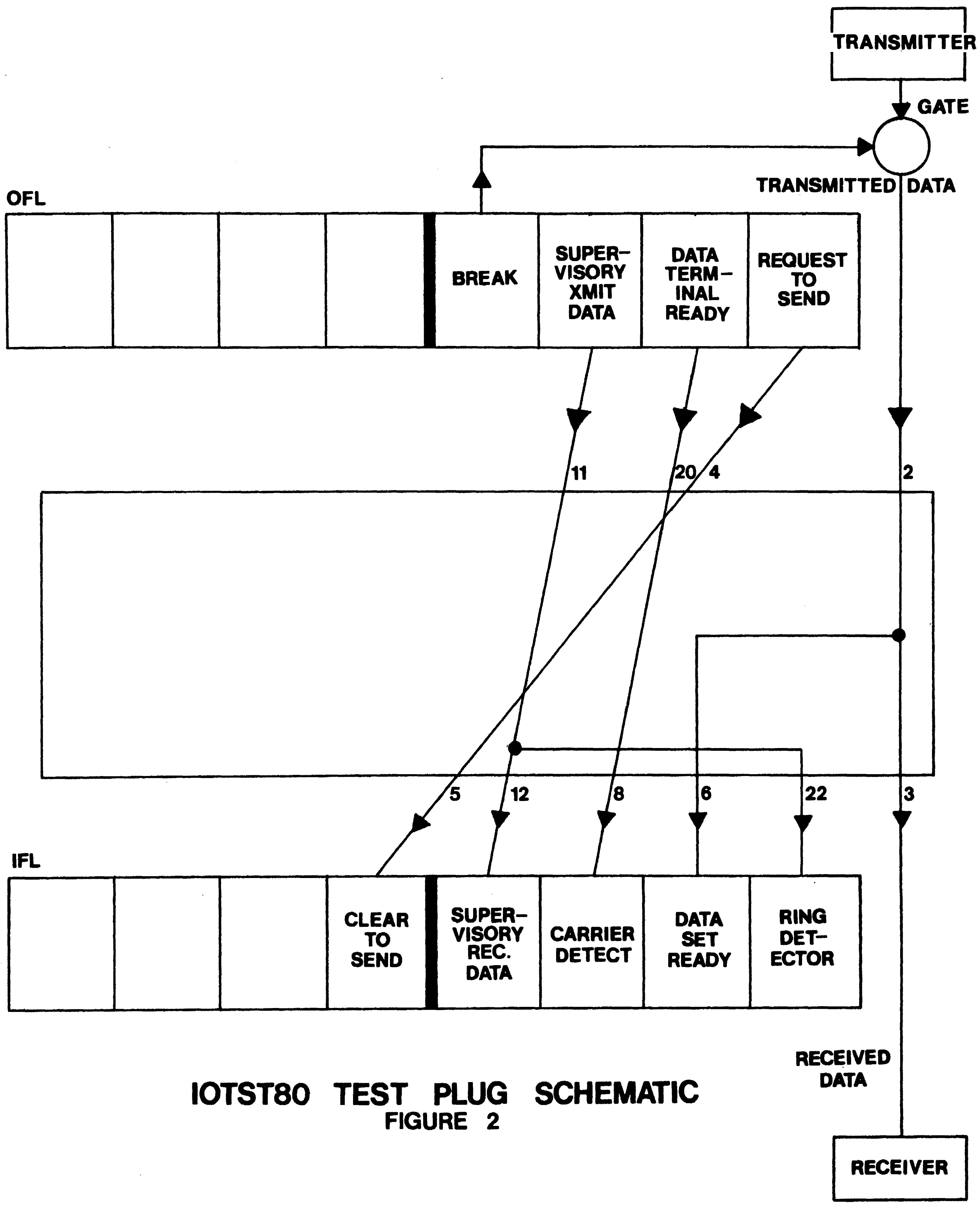
T47:	Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 50 baud (38,400 if so configured).
T48:	Tests that, by issuing an INIT, DVCL and INP, the Character Received and Available flag is not stuck high.
T49:	Tests that, by issuing an OUT, the Character Received and Available flag is not stuck low.
T4A:	Tests that, after setting the Character Received and Available flag, it can be reset by issuing an INIT, DVCL and INP.
T4B:	Tests that, after setting the Character Received flag, it can be reset by issuing an INIT.
T4C:	Tests that, after setting the Character Received and Available flag, it can be reset by issuing a DVCL.
T4D:	Tests that, after setting the Character Received and Available flag, it can be reset by issuing an INP.
T4E:	Tests that, for all COM1 values, an OUT command, given sufficient time, sets the Character Received and Available flag. The current COM1 value is displayed on the screen.
T4F:	Tests that, for all COM1 values, the Character Received and Available flag does not go high during reception. The current COM1 value is displayed on the screen.
T50:	Tests that, with the interrupts enabled, a SMSK with command byte of 0 Hex does not allow a Receive interrupt.
T51:	Tests that, with the interrupts enabled and the Character Received and Available flag high, a interrupt priority 1 is generated.
T52:	Tests that, with the interrupts enabled and the Character Received and Available flag low, an interrupt is not generated.
T53:	Tests that, at all combinations of stop bits, parity, and 7 or 8 data at 100 and 19200 (38,400 if so configured) baud bits, that the character transmitted is the character received. Current COM1 values and transmitted characters are displayed on the screen.

T54:	Tests that, by issuing an INIT and DVCL, the Parity, Overrun and Framing Error flag is not stuck high.
T55:	Tests that, at all COM1 values, the Parity, Overrun and Framing Error flag is not set on good reception. The current COM1 value is displayed on the screen.
T56:	Tests that, by issuing three OUT commands without an INP, the Parity, Overrun and Framing Error flag shows an Overrun error.
T57:	Tests that, by using the OFL Break to generate a Framing Error during transmit, that the Parity, Overrun and Framing Error flag shows a Framing Error.
T58:	Tests that, by transmitting a character and using the OFL Break to change a high bit to a low bit, the Parity, Overrun and Framing Error flag shows a Parity error.
T59:	Tests that, after setting the Parity, Overrun and Framing Error flag, an INIT resets it.
T5A:	Tests that, after setting the Parity, Overrun and Framing Error flag, a DVCL resets it.

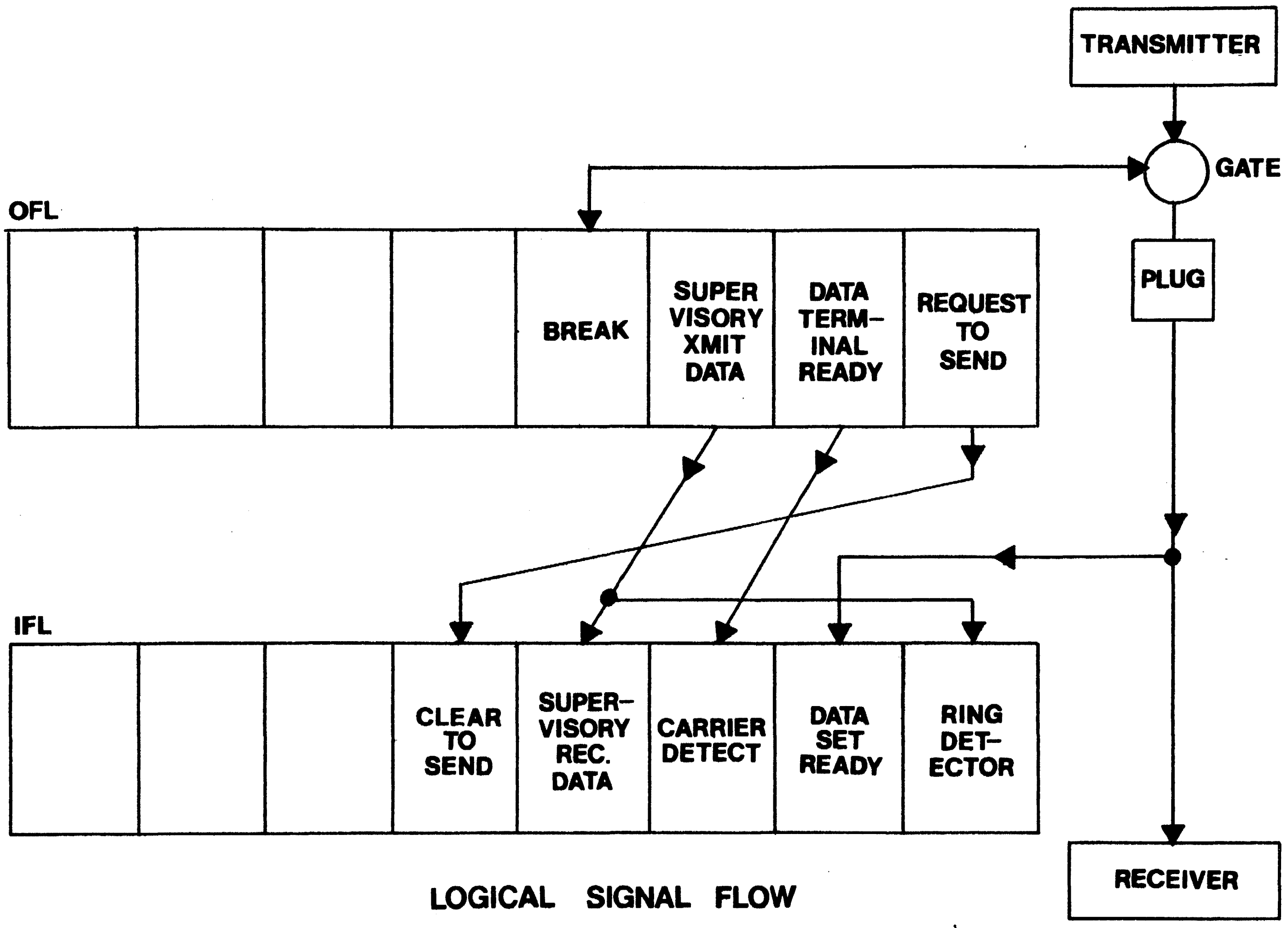


IOTST8Ø TEST PLUG INSERTION GUIDE

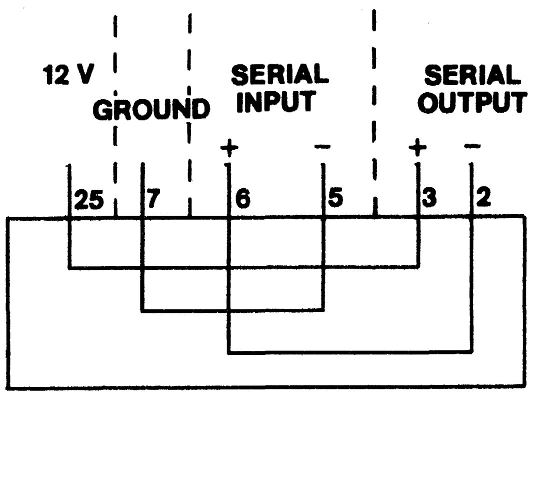
FIGURE 1



IOTST80 TEST PLUG SCHEMATIC
FIGURE 2



LOGICAL SIGNAL FLOW



COMTST2 TEST PLUG SCHEMATIC
FIGURE 3

AIOTST4

ALTERNATE ASYNCHRONOUS I/O ADAPTER TEST

AIOTST4 - ALTERNATE ASYNCHRONOUS I/O ADAPTER TEST

Applicable Assemblies

5000-1172-X-Y	Alternate I/O Adapter Board	(X=1-3, Y=1-3)
5100-1106-XYZ	Alternate I/O Adapter Board	(X=2-4, Y=1&2, Z=1)

Required Test Assemblies

IOTST80 Test Plug (RS232 Compatible, X=1)
COMTST2 Test Plug (TTY Compatible, X=2)
no plug required (2 Wire Direct, X=3)

Note: 5000-1172-X-Y (X-2,3 Y=1-3) Rev. A requires a carrier Part No. 5000-1377-1.

General Description

The purpose of the AIOTST4 test program is to determine if the Alternate Asynchronous I/O Adapter is working properly and, if not, to give an indication of which functions are incorrect. The corresponding test plug is needed to run AIOTST4. The program requires no operator interaction, except for the Initialization, unless an error is encountered.

4K (or more) of memory is required to run AIOTST4.

Loading Procedure

AIOTST4 can be loaded into memory using any convenient method (see Appendix B). When AIOTST4 loads properly it will immediately start an initialization dialogue, afterwards it will identify itself and await run-mode input by the operator (see Appendix A).

Errors

All errors are indicated by an error indicator on the display screen and the simultaneous activation of the bell. The error displayed on the screen refers to the detailed description of the specific test to determine the purpose and expected results for the displayed error indicator.

After an error indicator is displayed, the operator has three ways to proceed. Typing the SPACE bar will continue testing at the next test, typing the R key (without the SHIFT key) will repeat the current test, and depressing the PROG key will restart the program.

Because of the inverted pyramid test strategy used in AIOTST4, it is desirable to service erroneous functions as they occur. An erroneous function discovered in one subtest may cause misleading error messages is subsequent subtests since the function is assumed to be working in all subtests after the one in which it is tested.

IOTST80 Test Plug

The IOTST80 Test Plug is necessary for testing the Alternate Asynchronous I/O Adapter RS232 compatible. The plug must be inserted into the Alternate Asynchronous I/O Adapter connector (as shown in Figure 1) on the rear panel of the OP-1 or OP-1/R prior to program execution. The schematic of the IOTST80 Test Plug is shown in Figure 2.

COMTST2 Test Plug

The COMTST2 Test Plug is necessary for testing the Alternate Asynchronous I/O Adapter TTY compatible (20ma). The plug must be inserted into the Alternate Asynchronous I/O Adapter connector (as shown in Figure 1) on the rear panel of the OP-1 or OP-1/R prior to program execution. The schematic of the COMTST2 Test Plug is shown in Figure 3, as well as the logic flow diagram.

Test Description

All test operations are described in this section. The program will halt and the specified error indicator is displayed if expected results are not obtained.

On the following pages, each test is listed with a description of what is being tested, with an explanation of the cause of the error.

IOTST80 Test Plug

The IOTST80 Test Plug is necessary for testing the Alternate Asynchronous I/O Adapter RS232 compatible. The plug must be inserted into the Alternate Asynchronous I/O Adapter connector (as shown in Figure 1) on the rear panel of the OP-1 or OP-1/R prior to program execution. The schematic of the IOTST80 Test Plug is shown in Figure 2.

COMTST2 Test Plug

The COMTST2 Test Plug is necessary for testing the Alternate Asynchronous I/O Adapter TTY compatible (20ma). The plug must be inserted into the Alternate Asynchronous I/O Adapter connector (as shown in Figure 1) on the rear panel of the OP-1 or OP-1/R prior to program execution. The schematic of the COMTST2 Test Plug is shown in Figure 3, as well as the logic flow diagram.

Test Description

All test operations are described in this section. The program will halt and the specified error indicator is displayed if expected results are not obtained.

On the following pages, each test is listed with a description of what is being tested, with an explanation of the cause of the error.

Initialization

When AIOT514 has finished loading it will ask if the board contains 38,400 and 50,000 baud.

Initialization

When the Alternate I/O Adapter test is loaded a user initialization dialogue is immediately executed. This is a two part dialogue that asks the user to respond to two questions.

The first asks for the select address of the Alternate I/O Adapter. A response of two hexadecimal digits is expected. The dialogue will not advance to the next part until a valid response is made.

The second asks for the interrupt priority. A response of one digit in the range 3-6 is expected. Execution of the Alternate I/O Adapter test will not begin until a valid response is made.

In either of the two parts of the initialization dialogue, if a valid but incorrect response is made, typing the prog key will restart the dialogue.

Once valid responses are made to both parts of the dialogue the program will determine if it is running on an OP-1 or an OP-1/R and make the appropriate timing adjustments before execution begins.

T00: Tests that the Alternate I/O Adapter does not respond to an incorrect select address. Addresses of all combinations of three bits or less high are used to try to select the board.

T01: Tests that the Alternate I/O Adapter does respond to the correct select address. The address entered during initialization is used to try to select the board.

T02: Tests that the selected Alternate I/O Adapter can be deselected by issuing an INIT.

T03: Tests that, by issuing an INIT, DVCL, and an OFL with a command byte of 0 Hex, the Data Set Ready flag is not stuck high.

T04: Tests that, by issuing an INIT, DVCL, and an OFL with a command byte of 0 Hex, the Clear to Send flag is not stuck high.

T05: Tests that, by issuing an INIT, DVCL, and an OFL with a command byte of 0 Hex, the Carrier Detect Signal flag is not stuck high.

T06: Tests that, by issuing an INIT, DVCL, and an OFL with a command byte of 0 Hex, the Supervisory Received Data and Ring Detect Signal flags are not stuck high.

T07: Tests that, by issuing an OFL with a command byte set to Break, the Data Set Ready flag is not stuck low.

T08: Tests that, by issuing an OFL with a command byte set for Request to Send, the Clear to Send flag is not stuck low.

T09: Tests that, by issuing an OFL with a command byte set for Data Terminal Ready, the Carrier Detect Signal flag is not stuck low.

T0A: Tests that, by issuing an OFL with a command byte set for Supervisory Transmit Data, the Supervisory Received Data and Ring Detect Signal flags are not stuck low.

T0B: Tests that, after issuing an OFL with command byte set to Break, to set the Data Set Ready flag, an INIT resets the Data Set Ready flag.

T0C: Tests that, after issuing an OFL with command byte set for Request to Send, to set the Clear to Send flag, an INIT resets the Clear to Send flag.

T0D: Tests that, after issuing an OFL with command byte set for Data Terminal Ready, to set the Carrier Detect Signal flag, and INIT resets the Carrier Detect Signal flag.

T0E: Tests that, after issuing an OFL with command byte set for Supervisory Transmit Data, to set the Supervisory Received Data and Ring Detect Signal flags, and INIT resets that Supervisory Received Data and Ring Detect Signal flags.

T0F: Tests that, after issuing an OFL with command byte set to Break, to set the Data Set Ready flag, a DVCL resets the Data Set Ready flag.

T10: Tests that, after issuing an OFL with command byte set for Request to Send, to set the Clear to Send flag, a DVCL resets the Clear to Send flag.

T11: Tests that, after issuing an OFL with command byte set for Data Terminal Ready, to set the Carrier Detect Signal flag, a DVCL resets the Carrier Detect Signal flag.

T12: Tests that, after issuing an OFL with command byte set for Supervisory Transmit Data, to set the Supervisory Received Data and Ring Detect Signal flag, a DVCL resets the Supervisory Received Data and Ring Detect Signal flag.

T13: Tests that, after issuing an OFL with command byte set to Break, to set the Data Set Ready flag, an OFL with command byte of 0 Hex resets the Data Set Ready flag.

T14: Tests that, after issuing an OFL with command byte set for Request to Send, to set the Clear to Send flag, an OFL with command byte of 0 Hex resets the Clear to Send flag.

T15: Tests that, after issuing an OFL with command byte set for Data Terminal Ready, to set the Carrier Detect Signal flag, an OFL with command byte of 0 Hex resets the Carrier Detect Signal flag.

T16: Tests that, after issuing an OFL command byte set for Supervisory Transmit Data, to set the Supervisory Received Data and Ring Detect Signal flags, an OFL with command byte of 0 Hex resets the Supervisory Received Data and Ring Detect Signal flags.

T17: Tests that, by issuing an INIT and DVCL, the Character Needed for Transmission flag is not stuck low.

T18: Tests that, by issuing two successive OUT commands, the Character Needed for Transmission flag is not stuck high.

T19: Tests that, given sufficient time, the Character Needed for Transmission flag goes high after one OUT command.

T1A: Tests that, given sufficient time, the Character Needed for Transmission flag goes high after two OUT commands.

T1B Tests that, the Character Needed for Transmission flag does not go high during transmission.

T1C: Tests that, a DVCL sets the Character Needed for Transmission flag during a transmission.

T1D: Tests that, an INIT sets the Character Needed for Transmission flag during a transmission.

T1E: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 19,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T1F: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 9,600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more thatn the maximum amount of bits (12).

T20: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 7,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T21: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 4,800 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12)).

T22: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 3,600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T23: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 2,400 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T24: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 2,000 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12)).

~~T25: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 1,800 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).~~

~~T26: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 1,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).~~

~~T27: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12)).~~

~~T28: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 300 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).~~

~~T29: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 150 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).~~

T2A: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 135.5 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12)).

T2B: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 110 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2C: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 75 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2D: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 50 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T23: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 2,400 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T24: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 2,000 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12)).

T25: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 1,800 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T26: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 1,200 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T27: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 600 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T28: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 300 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T29: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 150 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2A: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 135.5 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2B: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 110 baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2C: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 75 (50,000 if so configured) baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2D: Tests that the transmitter is working by transmitting 8 data bits, 1 stop bit and 1 start bit, at 50 (38,400 if so configured) baud and checking that the time needed to transmit the 10 bits does not take less than the time needed to transmit the minimum amount of bits (7) or more than the maximum amount of bits (12).

T2E: Tests that, with the interrupts enabled, a SMSK with command byte of 0 Hex holds off the Transmit and Ring Detect interrupts.

T2F: Tests that, in Ring Detect Mode with the Transmit interrupt on and the Ring Signal and Character Needed for Transmission flags high and the interrupts enabled, the correct priority interrupt is generated.

T30: Tests that, when not is Ring Detect mode with the Transmit interrupt off and with the Ring Signal and Character Needed for Tranmission flags high and the interrupts enabled, no interrupt is generated.

T31: Tests that, with only the Transmit interrupt enabled and the Ring Signal off and the Character Needed for Transmission flag high and the interrupts enabled, an interrupt is generated.

T32: Tests that, when in Ring Detect mode only, and the Ring Signal on and the Character Needed for Transmission flag low and the interrupts enabled, an interrupt is enabled.

T33: Tests that, with only the Transmit interrupt enabled and the Ring Signal on and the Character Needed for Tranmission flag low and the interrupts enabled, that no interrupt is generated.

T34: Tests that, when in Ring Detect mode, with the Ring Signal off and the Character Needed for Tranmission flag high and the interrupts enabled, no interrupt is generated.

T35: Tests that, when in Ring Detect mode with the Transmit interrupt on and the Ring Signal off and the Character Needed for Transmission flag low and the interrupts enabled, no interrupt is generated.

T36: Tests that, when in Ring Detect mode only with the Ring Signal off and the Character Needed for Transmission flag low and the interrupts enabled, no interrupt is generated.

T37: Tests that, with the Transmit interrupt enabled and the Ring Signal off and the Character Needed for Transmission flag low and the interrupts enabled, no interrupt is generated.

T38: Tests that, the transmitter correctly transmits 5 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. All characters 00...1F Hex are transmitted at 300 baud and displayed on the screen.

T39: Tests that the transmitter correctly transmits 6 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. All characters 00...3F Hex are transmitted at 300 baud and displayed on the screen.

T3A: Test that the transmitter correctly transmits 7 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. All characters 00...7F Hex are transmitted at 300 baud and displayed on the screen.

T3B: Tests that the transmitter correctly transmits 8 data bits, by checking the Data Set Ready flag and comparing its fluctuation to see if it matches the bit configuration of the transmitted character. All characters 00...0FF Hex are transmitted at 300 baud and displayed on the screen.

T3C-49:
Tests that character length, parity, and stop bits are transmitted correctly by comparing the Data Set Ready flag fluctuations with the expected bit pattern of the transmitted character. The even parity character has an ASCII value of 55 Hex and the odd parity character has an ASCII value of 52 Hex.

T3C: Tests 8 Data bits, 1 stop bit, and no parity, with the transmission of a character of even parity.

T3D: Tests 8 Data bits, 1 stop bit, and no parity, with the transmission of a character of odd parity.

T3E: Tests 8 Data bits, 2 stop bits, and no parity, with the transmission of a character of even parity.

T3F: Tests 8 Data bits, 2 stop bits, and no parity, with the transmission of a character of odd parity.

T40: Tests 8 Data bits, 1 stop bit, and even parity, with the transmission of a character of even parity.

T41: Tests 8 Data bits, 1 stop bit, and even parity, with the transmission of a character of odd parity.

T42: Tests 8 Data bits, 1 stop bit, and odd parity, with the transmission of a character of even parity.

T43: Tests 8 Data bits, 1 stop bit, and odd parity, with the transmission of a character of odd parity.

T44: Tests 7 Data bits, 1 stop bit, no parity, with the transmission of a character of even parity.

T45: Tests 7 Data bits, 1 stop bit, no parity, with the transmission of a character of odd parity.

T46: Tests 6 Data bits, 1 stop bit, no parity, with the transmission of a character of even parity.

T47: Tests 6 Data bits, 1 stop bit, no parity, with the transmission of a character of odd parity.

T48: Tests 5 Data bits, 1 stop bit, no parity, with the transmission of a character of even parity.

T49: Tests 5 Data bits, 1 stop bit, no parity, with the transmission of a character of odd parity.

T4A-T59:
Tests that the time boundaries are within +/- 10% of the actual time needed for transmission of 12 bits.

T4A: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 19,200 baud.

T4B: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 9,600 baud.

T4C: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 7,200 baud.

T4D: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 4,800 baud.

T4E: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 3,600 baud.

T4F: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 2,400 baud.

T50: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 2,000 baud.

T51: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 1,800 baud.

T52: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 1,200 baud.

T53: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 600 baud.

T54: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 300 baud.

T55: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 150 baud.

T56: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 135.5 baud.

T57: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 110 baud.

T58: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 75 (50,000 if so configured) baud.

T59: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 50 (38,400 if so configured) baud.

T5A: Tests that, by issuing an INIT, DVCL and INP, the Character Received and Available flag is not stuck high.

T4A-T59:
Tests that the time boundaries are within +/- 10% of the actual time needed for transmission of 12 bits.

T4A: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 19,200 baud.

T4B: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 9,600 baud.

T4C: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 7,200 baud.

T4D: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 4,800 baud.

T4E: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 3,600 baud.

T4F: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 2,400 baud.

T50: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 2,000 baud.

T51: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 1,800 baud.

T52: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 1,200 baud.

T53: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 600 baud.

T54: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 300 baud.

T55: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 150 baud.

T56: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 135.5 baud.

T57: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 110 baud.

T58: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 75 baud.

T59: Tests that the transmission of 12 data bits falls within the time limit for 12 data bits at 50 baud.

T5A: Tests that, by issuing an INIT, DVCL and INP, the Character Received and Available flag is not stuck high.

T5B: Tests that, by issuing an OUT, the Character Received and Available flag is not stuck low.

T5C: Tests that, after setting the Character Received and Available flag, it can be reset by issuing an INIT, DVCL and INP.

T5D: Tests that, after setting the Character Received and Available flag, it can be reset by issuing an INIT.

T5E: Tests that, after setting the Character Received and Available flag, it can be reset by issuing an DVCL.

T5F: Tests that, after setting the Character Received and Available flag, it can be reset by issuing an INP.

T60: Tests that, for all COM1 values, an OUT command, given sufficient time, sets the Character Received and Available flag. The current COM1 value is displayed on the screen.

T61: Tests that, for all COM1 values, the Character Received and Available flag does not go high during reception. The Current COM1 value is displayed on the screen.

T62: Tests that, with the interrupts enabled, a SMSK with command byte of 0 Hex does not allow a Receive interrupt.

T63: Tests that, with the interrupts enabled and the Character Received and Available flag high, the correct priority interrupt is generated.

T64: Tests that, with the interrupts enabled and the Character Received and Available flag low, an interrupt is not generated.

T65: Tests that, at all combinations of stop bits, parity, and 7 or 8 data bits at 50 and 19,200 baud, that the character transmitted is the character received. Current COM1 values and transmitted characters are displayed on the screen.

T66: Tests that, at all combinations of stop bits, parity, and 5 or 6 data bits, that the character transmitted is the character received. Current COM1 values and transmitted characters are displayed on the screen.

T67: Tests that, by issuing an INIT and DVCL, the Parity, Overrun and Framing Error flag is not stuck high.

T68: Tests that, at all COM1 values, the Parity, Overrun and Framing Error flag is not set on good reception. The Current COM1 value is displayed on the screen.

T69: Tests that, by issuing three OUT commands without an INP, the Parity, Overrun and Framing Error flag show an overrun error.

T6A: Tests that, by using the OFL Break to make an apparent transmit with Framing Error, that the Parity, Overrun and Framing error flag shows a Framing Error.

T6B: Tests that, by transmitting a character and using the OFL Break to change a high bit to a low bit, the Parity, Overrun and Framing Error flag shows a parity error.

T6C: Tests that, after setting the Parity, Overrun and Framing Error flag, an INIT resets it.

T6D: Tests that, after setting the Parity, Overrun and Framing Error flag, a DVCL resets it.

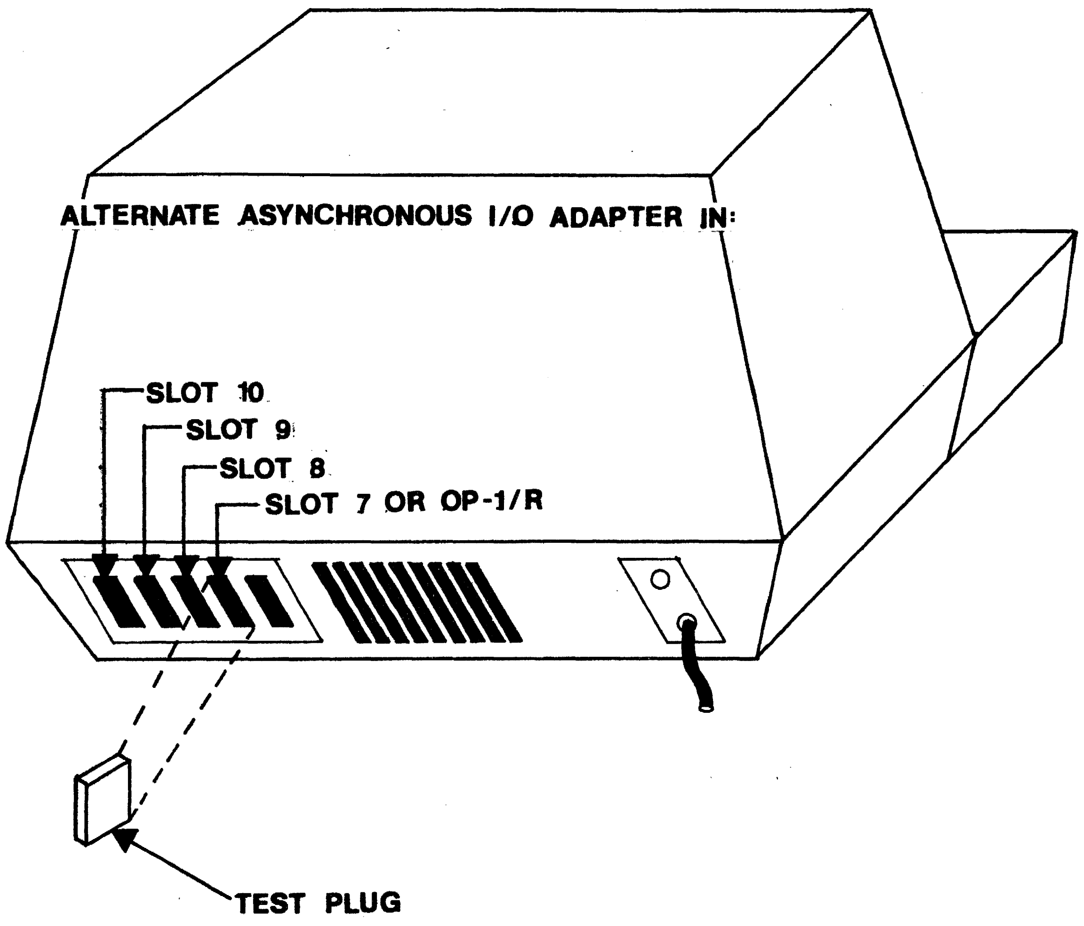
T6E-T6F:
The character used for transmission has an ASCII value of 0AA Hex. This character has an alternating 1's and 0's pattern and when transmitting at twice the rate of reception the receiver will see the appropriate start and stop bits to receive one transmitted character as two characters.

T6E: Tests that the receive baud works independently of the transmit baud by setting the transmit and receive baud with a COM1 and then changing the receive baud with a COM3 then testing that the receiver receives twice as many characters as are transmitted.

T6F: Tests that the receive baud can be reset by a COM1 command by setting the receive baud with a COM3 then issuing a COM1 to set the transmit baud and reset the receive baud and testing that the receiver receives as many characters as are transmitted.

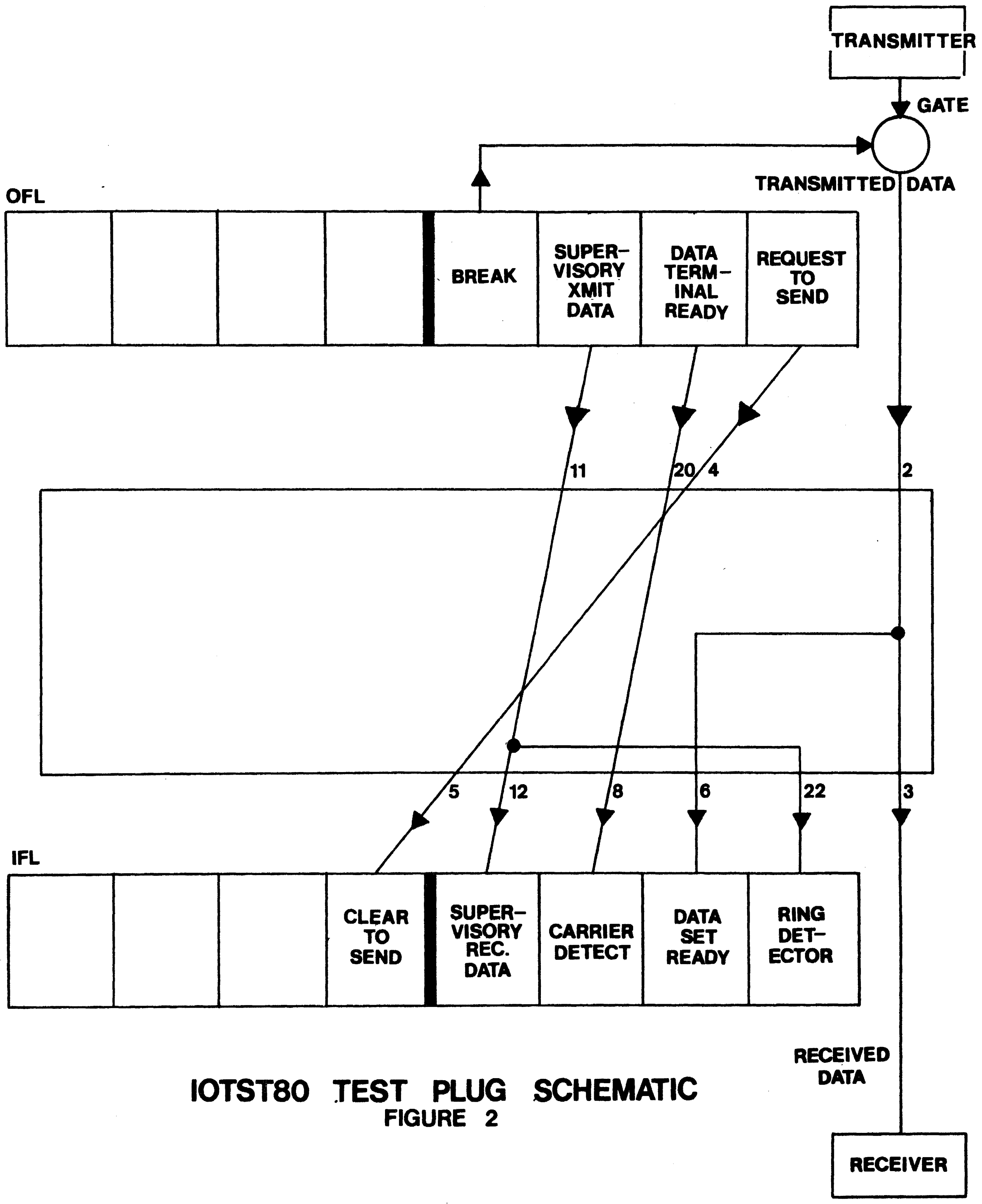
T70: Same as T62, but tests that the appropriate status bit of the IIN register is low when no interrupt is generated. Fails if T62 fails.

T71: Same as T63, but tests that the appropriate status bit of the IIN register is set when an interrupt is generated. Fails if T63 fails.

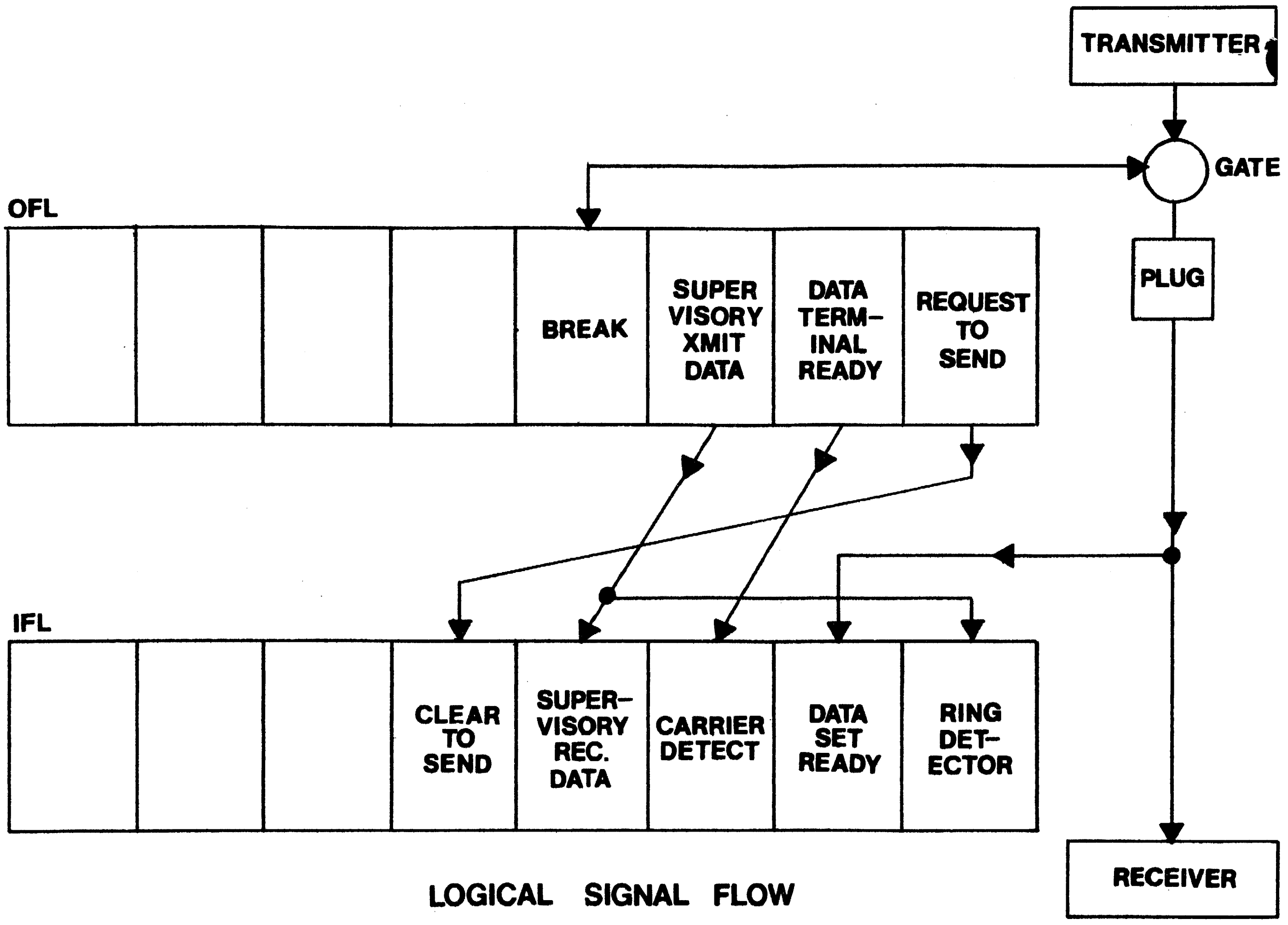


TEST PLUG INSERTION GUIDE

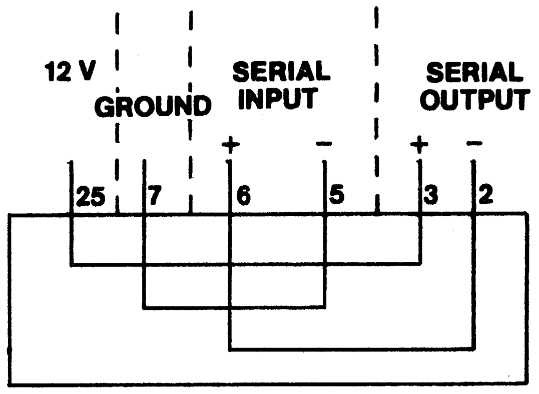
FIGURE 1



IOTST80 TEST PLUG SCHEMATIC
FIGURE 2



LOGICAL SIGNAL FLOW



COMTST2 TEST PLUG SCHEMATIC
FIGURE 3

SIOTST4

SYNCHRONOUS I/O ADAPTER TEST FOR OP-1/RS

ASYNCHRONOUS I/O ADAPTER TEST

Applicable Assemblies			Applicable Program
5000-1141-X	8080 CPU Board	(X=1-2)	IOTST4
5000-E0XYZABCD	OP-1/R Main Logic Board	(B=1-3)	IOTST4
5000-11114-X	8085 CPUM Board	(X=1)	IOTSTM
5000-11114-X	8085 CPUM Board	(X=3)	IOT2WM
5300-1102	8085		IOTST15
5300-1102	8085		IOT2W15

Required Test Assemblies

IOTST80 Test Plug (RS232 Compatible, X or B=1)
COMTST2 Test Plug (TTY Compatible, X or B=2)
no plug required (2 Wire Direct, X=3)

General Description

The purpose of the test program is to determine if the Asynchronous I/O Adapter is working properly and, if not, to give an indication of which functions are incorrect. The corresponding test plug is needed to run. The program requires no operator interaction unless an error is encountered.

4K (or more) of memory is required to run IOTST4/IOTSTM.

Loading Procedure (IOTST4)

IOTST4 can be loaded into memory using any convenient method (see Appendix B).

Loading Procedure (IOTSTM/IOTST15/IOT2WM/IOT2W15)

IOTSTM can be loaded into memory using any convenient method (see Appendix B). IOTSTSM will identify itself and await run-mode input (see Appendix A).

Errors

All errors are indicated by an error indicator on the display screen and the simultaneous activation of the bell. The error displayed on the screen refers to the detailed description of the specific test to determine the purpose and expected results for the displayed error indicator.

SIOTST4 - SYNCHRONOUS I/O ADAPTER TEST FOR OP-1/RS

Applicable Assemblies

5100-E-1110	OP-1/RS
5100-E-1111	OP-1/RS

Required Test Assemblies

SIOTST Diagnostic Test Plug
Exernal Timer (19200)
COMTST10 Diagnostic Cable

General Description

The purpose of the SIOTST4 program is to determine if the Synchronous I/O Adpater on an OP-1/RS is working properly, and if not to give an indication of which functions are incorrect. A SIOTST diagnostic test plug and an external timer set at 19200 bits-per-second are needed to run SIOTST4. No operator interaction is needed to run this program, unless an error is detected.

4K of memory is required to run the BIN80 version of SIOTST4.

Operator Action

The OP-1/RS asks the user to "Enter Select Address: F0 " at this time replace the attached COMTST10 diagostic cable that is connected to the "A" port of the OP-1/RS with a diagnostic test plug. Then the select address of the Synchronous I/O Adapter must be entered. If the select address of this device is F0H just depress return key; if it is anything else depress the appropriate keys.

SIOTST4 now wants to know if PCO 325 was implemented*. To comply with this question key in a "Y" for yes or an "N" for no.

At this time a beep will be generated to let the user know that run-mode options are expected (see Appendix A).

To restart the program during the initializing questions or when a test is not being run, depress the key marked "Cancel".

* PCO 325 stops transmission of SYN characters if RTS or CTS is turned off.

All tests run automatically and without operator action, unless an error is encountered.

When SIOTST4 is being run all preceding tests are expected to have passed with no errors before going to a subsequent test.

Errors

All errors are noted by an audible beep (see Appendix B).

SIOTST Diagnostic Test Plug

A SIOTST diagnostic test plug is needed to run SIOTST4. It must be inserted into the "A" port on the rear pannel of the OP1/RS prior to execution of the tests (refer to figure 1). The pin configuration of this plug is shown in figure 2 of this section (NOTE: An external timer set at 19200 bits-per-second should be attached to pin 17 or pin 15 if an 1110 board is being used).

External Timer

An external timer (clock osilator) is needed when using an OP-1/RS Version I (5100-E-1110 board), but is all ready included in the design of an OP-1/RS Version II (5100-E-1111 board).

Test Description

On the following pages, each test is listed with a brief description of what is being tested:

T00:	Test that the Synchronous I/O Adapter can not be selected with an incorrect select address.
T01:	Test that the Synchronous I/O Adapter can be selected with the correct select address.
T02:	Test that an INIT will deselect the Synchronous I/O Adapter.
T03:	Test that Carrier Detect is not stuck high by issuing an INIT, SEL, DVCL and an OFL with Data Terminal Ready reset.
T04:	Test that Carrier Detect can be set by issuing an OFL with Data Terminal Ready set.
T05:	Test that after setting Carrier Detect by issuing an OFL with Data Terminal Ready set, an INIT will reset it.

T06:	Test that after setting Carrier Detect by issuing an OFL with Data Terminal Ready set, a DVCL will reset it.
T07:	Test that after setting Carrier Detect by issuing an OFL with Data Terminal Ready set, issuing an OFL with Data Terminal Ready reset, will reset Carrier Detect.
T08:	Test that Data Set Ready is not stuck high by setting the transmit and receive SYN character to be 0FFH and issuing an INIT, SEL, DVCL and an OFL with Break Transmitted Data reset.
T09:	Test that Data Set Ready can be set by issuing an OFL with Break Data Transmitted set.
T0A:	Test that after setting Data Set Ready by issuing an OFL with Break Transmitted Data set, an INIT will reset it.
T0B:	Test that after setting Data Set Ready by issuing an OFL with Break Transmitted Data set, a DVCL will reset it.
T0C:	Test that after setting Data Set Ready by issuing an OFL with Break Transmitted Data set, issuing an OFL with Break Transmitted Data reset, will reset Data Set Ready.
T0D:	Test that Clear to Send is not stuck high by issuing an INIT, SEL, DVCL and an OFL with Request to Send reset.
T0E:	Test that Clear to Send can be set by issuing an OFL with Request to Send set.
T0F:	Test that after setting Clear to Send by issuing an OFL with Request to Send set, an INIT will reset it.
T10:	Test that after setting Clear to Send by issuing an OFL with Request to Send set, a DVCL will reset it.
T11:	Test that after setting Clear to Send by issuing an OFL with Reqest to Send set, issuing an OFL with Request To Send reset, will reset Clear to Send.
T12:	Test that Clear to Send, Carrier Detect and Data Set Ready can be set simultaneously by issuing an OFL with Request to Send, Break Transmitted and Data Terminal Ready set.
T13:	Test that the Character Needed for Transmission flag is set when the device is first selected.
T14:	Test that when transmitting a character, the Character Needed for Transmission flag gets reset.
T15:	Test that the Character Needed for Transmission flag will set within a nominal time after a character has been transmitted.

T16:	Test that the Transmitted Data line is not stuck high or low by transmitting a known character with an alternating bit pattern (055H) and monitor the Data Set Ready line for two transitions. (This test will also fail if the character needed for transmission flag is stuck low).
T17:	Test that the Character Received and Available flag is not stuck high by issuing an INIT, SEL and a DVCL.
T18:	Test that by sending SYN characters the Character Received and Available flag will set. If this test fails the complement of the original SYN character will be tried, and a failure will only result if neither SYN character can intiate reception.
T19:	Test that after a character is received the Character Received and Available sets, and by issuing an INIT, the Character Received and Available flag will reset.
T1A:	Test that after a character is received the Character Received and Available flag sets and by issuing a DVCL, the Character Received and Available flag will reset.
T1B:	Test that after a character is received the Character Received and Available flag sets, and by issuing an INP, the Character Received and Available flag will reset.
T1C:	Test that a character cannot be received without a leading SYN character.
T1D:	Test that all 256 characters (from 00-0FFH) can be used as SYN characters.
T1E:	Test transmission of all characters using 8 data bits.
T1F:	Test transmission of all characters using 7 data bits.
T20:	Test transmission of all characters using 6 data bits.
T21:	Test transmission of all characters using 5 data bits.
T22:	Test that the Parity or Overrun flag is not stuck high by issuing an INIT, SEL and a DVCL.
T23:	Test that all characters can be transmitted and received with even parity.
T24:	Test that all characters can be transmitted and received with odd parity.
T25:	Test that receiving two characters without resetting the Character Received and Available flag causes an overrun error.
T26:	Test that after the Parity or Overrun flag was set, that the issuing of an INIT, resets the Parity or Overrun flag.

T16:	Test that the Transmitted Data line is not stuck high or low by transmitting a known character with an alternating bit pattern (055H) and monitor the Data Set Ready line for two transitions. (This test will also fail if the character needed for transmission flag is stuck low).
T17:	Test that the Character Received and Available flag is not stuck high by issuing an INIT, SEL and a DVCL.
T18:	Test that by sending SYN characters the Character Received and Available flag will set. If this test fails the complement of the original SYN character will be tried, and a failure will only result if neither SYN character can intiate reception.
T19:	Test that after a character is received the Character Received and Available sets, and by issuing an INIT, the Character Received and Available flag will reset.
T1A:	Test that after a character is received the Character Received and Available flag sets and by issuing a DVCL, the Character Received and Available flag will reset.
T1B:	Test that after a character is received the Character Received and Available flag sets, and by issuing an INP, the Character Received and Available flag will reset.
T1C:	Test that a character cannot be received without a leading SYN character.
T1D:	Test that the Parity or Overrun flag is not stuck high by issuing an INIT, SEL and a DVCL.
T1E:	Test that all 256 characters (from 00-0FFH) can be used as SYN characters.
T1F:	Test transmission of all characters using 8 data bits.
T20:	Test transmission of all characters using 7 data bits.
T21:	Test transmission of all characters using 6 data bits.
T22:	Test transmission of all characters using 5 data bits.
T23:	Test that all characters can be transmitted and received with even parity.
T24:	Test that all characters can be transmitted and received with odd parity.
T25:	Test that receiving two characters without resetting the Character Received and Available flag causes an overrun error.
T26:	Test that after the Parity or Overrun flag was set, that the issuing of an INIT, resets the Parity or Overrun flag.

T27:	Test that after the Parity or Overrun flag was set, that the issuing of a DVCL resets the Parity or Overrun flag.
T28:	Test that after causing the Parity or Overrun flag to set issuing an INP and receiving another character causes the Parity or Overrun flag to reset.
T29:	Test that the parity hardware is functioning correctly by cauusing a parity error. An error can be forced by setting the BREAK Transmited Data line in the middle of transmitting a character and then resetting it, before the character is finished transmitting.
T2A:	Tests that with interrupts enabled and the SMSK set to zero, no interrupts occur. (Tests character received and available interrupt)
T2B:	Test that with interrupts enabled, and the SMSK set to interrupt at Synchronous receive (04H), after a transmission only the Synchronous receive will interrupt.
T2C:	Test that with interrupts enabled and SMSK set to interrupt at Synchronous receive (04H), a transmission is done and the expected interrupt occurs. After that, a DVCL is done to make sure that the interrupt circuitry will reset.
T2D:	Test that with interrupts enabled and the SMSK set to zero no interrupt occurs (Tests character needed for transmission).
T2E:	Test that with interrupts enabled and SMSK set to interrupt at Synchronous transmit, only the Synchronous transmit will interrupt.
T2F:	Tests that the Synchronous transmit interrupt will not occur if the transmitter is not ready for another character to be transmitted. That is done by setting the SMSK to interrupt at Synchronous transmit (02H), filling up the transmit buffer and then making sure that no interrupt occurs.
T30:	Test that if PCO 325 is implemented, if RTS or CTS gets reset, SYN characters are not transmitted.
T31:	Test that if PCO 325 is not implemented, SYN characters are transmitted even if RTS or CTS gets reset.

T27: Test that after the Parity or Overrun flag was set, that the issuing of a DVCL resets the Parity or Overrun flag.

T28: Test that after causing the Parity or Overrun flag to set issuing an INP and receiving another character causes the Parity or Overrun flag to reset.

~~T29: Test that the parity hardware is functioning correctly by cauusing a parity error. An error can be forced by setting the BREAK Transmited Data line in the middle of transmitting a character and then resetting it, before the character is finished transmitting.~~

~~T2A: Test that with interrupts enabled and the SMSK set to zero the Character Received and Available flag will not set.~~

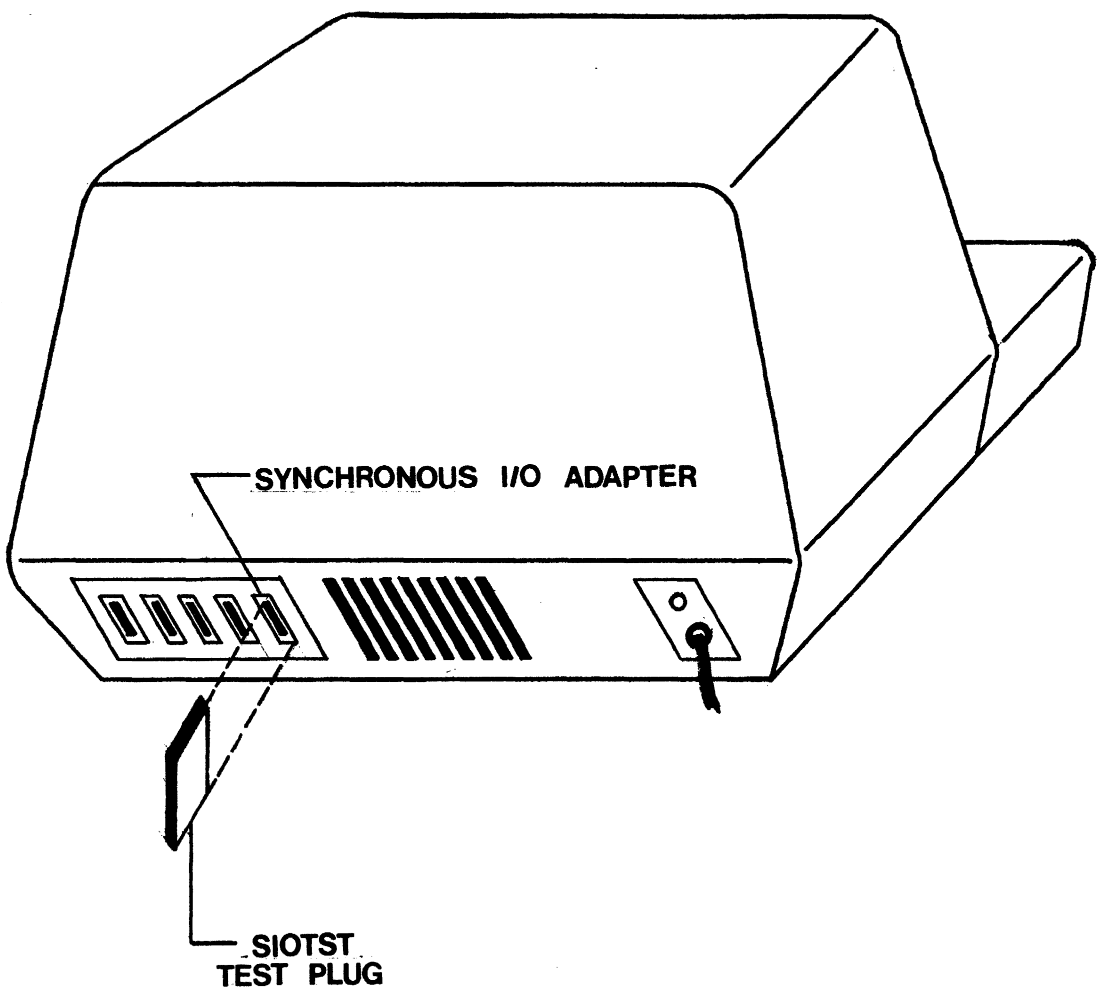
~~T2B: Test that with interrupts enabled, and the SMSK set to interrupt at synchronous receive (04H), after a transmission the Character Received and Available sets.~~

~~T2C: Test that with interrupts enabled and the SMSK set to zero the Character Needed for Transmission flag will not set.~~

~~T2D: Test that with interrupts enabled and the SMSK set to interrupt at synchronous transmit (02H) the Character Needed for Transmission flag will set.~~

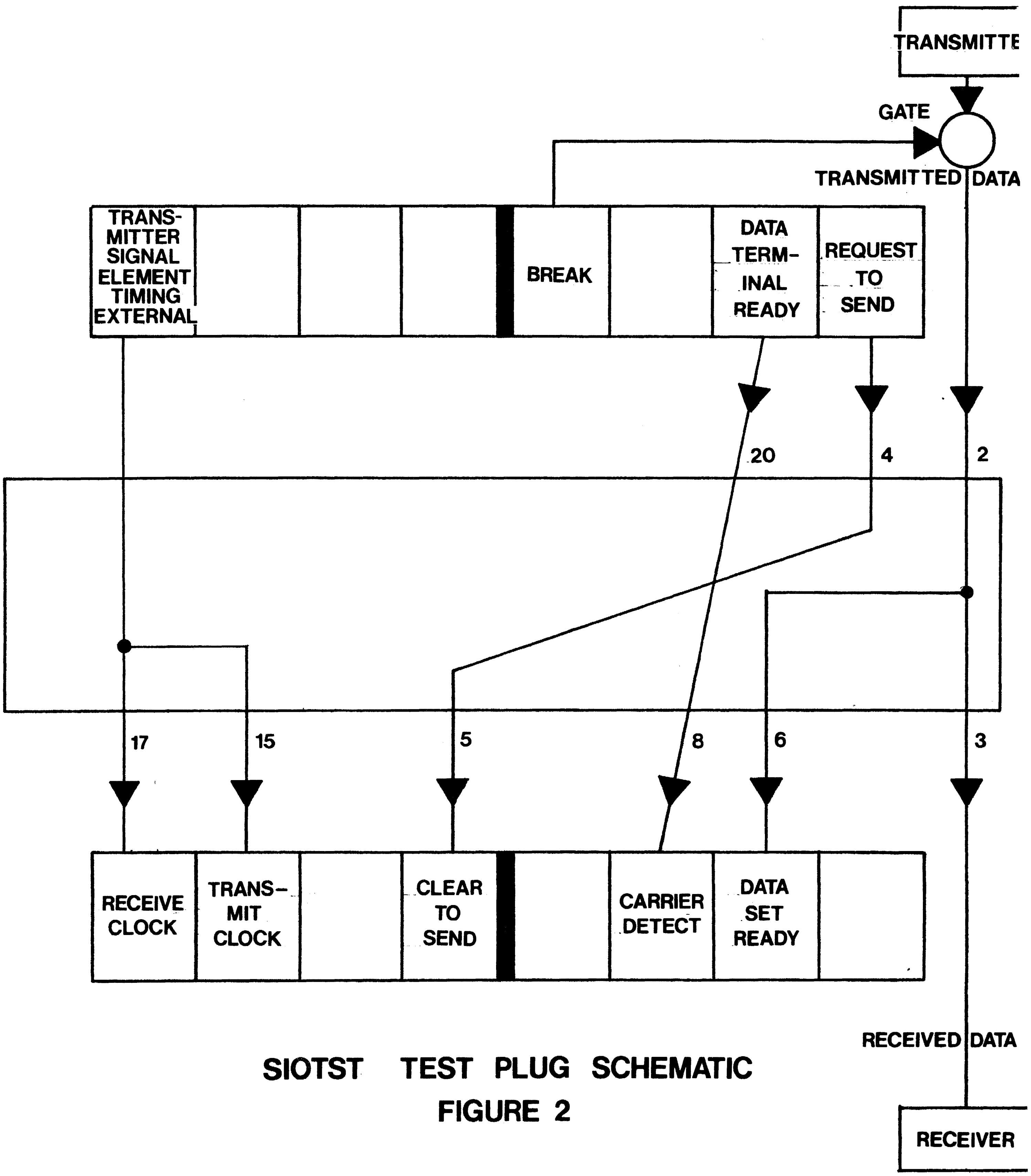
~~T2E: Test that is PCO 325 is implemented if RTS or CTS gets reset, SYN characters are not transmitted.~~

~~T2F: Test that if PCO 325 is not implemented, SYN characters are transmitted even if RTS or CTS gets reset.~~



SIOTST TEST PLUG INSERTION GUIDE

FIGURE 1



SIOTST TEST PLUG SCHEMATIC
FIGURE 2

* SPR8BOOT *

SPR8BOOT is a bootstrap loader designed to enable the operator to load programs from diskette using the standard DOS file structure. The SPR8BOOT may be implemented on either a SUPER8 or any standard OP-1 8080 in lieu of the standard bootstrap.

The SPR8BOOT has four capabilities; load a file, modify memory, begin execution and terminate bootstrap mode. Below is the format of the load file instruction:

	LX/YY	**NAME**	**TYPE**
Example:	LO/35	FIXTST	BIN80

Where X is the drive number, YY is the high address of the temporary one page (256 byte) buffer used for loading, NAME is the actual file name which is separated by a tab from YY, and TYPE is the type of file which also is separated from NAME by a tab.

To modify memory use the following format:

	MXXXX/YY1,YY2,...,YYN
Example:	M0840/10,2B

Where XXXX is the address of the first memory location and YY1 thru YYN is the new data to be entered into consecutive address locations beginning at XXXX.

To begin execution of a program, use the following format:

	JXXXX
Example:	J0900

Where XXXX is the address location to which the program counter is initialized.

To terminate the bootstrap mode just type:

	S
Example:	S

It should be noted that all fields indicated above by 'X' and 'Y' are of fixed length. Therefore, any leading zeros required to fill these fields must be entered. In addition, all of the preceding instructions must be followed by a 'RETURN'.

EIGHT-WAY/TEN-WAY - Diagnostic Bootstrap

Description

Eight-way/Ten-way is a diagnostic bootstrap program which loads all flavors of Ontel Operating Systems and performs a self-test diagnostic. The diagnostic functions test RAM, Display, and Prom.

Operation

Eight-way/Ten-way is initialized by power-up or the simultaneous depression of the CONTROL, SHIFT and PROG keys. At this time, there is an audible signal (beep), and the 4 keyboard lamps (F0-F3) are lighted. The operator may now type one of three function keys to initiate an action:

F0 - Attempt MDOS load from SS-type diskette.
F1 - Attempt MDOS load from SP-type diskette.
F2 - Attempt MDOS load from DP-type diskette.
F3 - Attempt MDOS load from MP-type diskette.
F4 - Attempt DOS load.
F5 - Attempt HDOS load.
F6 - Attempt PASCAL load.
F7 - Execute diagnostic sequence, then re-initialize for next input.
F8 - Boot downline load.
F9 - Minibug.

If F0 through F7 is typed and the Operating System is not loaded for any reason (e.g. diskette not ready), the message:

"Ready device, type PROG"

is displayed on the screen. When PROG is typed, Eight-way/Ten-way is re-initialized for input. The diagnostic sequence consists of the following test segments:

RAM Test - All of RAM excluding the display vector area (800H - 807H) is written to with a test value. Each location in RAM is then read and checked to see if the read value equals the test value. Nine such loops are executed, one for each of the test values 00H, 80H, C0H, E0H, F0H, F8H, FCG, FEH, and FFH.

Display Test - The last 2 lines in memory are written with an ASCII sequence starting with the version number of Eight-way/Ten-way. The HOME and WRAP vectors are set such that the pattern will be repeated on each line until the end of the screen. The cursor is then stopped diagonally up the screen beginning at the lower left corner.

Prom Test - Prom is tested by use of the check-sum written in each PROM by the BURN program.

NOTE: The diskette drive must be powered down before the diagnostic sequence may be run.

To recover from an error, type a single character corresponding to the Test number indicated. The diagnostic sequence will then continue to the next test. When all tests have been executed, Eight-way/Ten-way is re-initialized for input.

NOTE: There are no possible program-detected errors during the Display test.

RAM test is test number '1' or 'p'.
Prom test is test number '2'.

Special error notation for RAMTST.

If a RAM error occurs it will be displayed as follows:

1XY	The 1 signifies there has been a RAM failure The X is the BANK number (0-3) The Y is the Bit in Error (0-7 for data, 8 for parity, and F for transient soft error in unknown bit) If XY are both FF the parity error flag is stuck low even when reading known bad parity written at the highest memory location.
2XY	Prom check sum error contents of prom starting at address XY00, does not agree with contained check sum.
pXY	Run tim parity error trap. Error codes same as Test 1.

SPDPBT - Diagnostic Bootstrap

Description

SPDPBT is a diagnostic bootstrap program which loads MDOS from SP- and DP-type diskettes. The diagnostic functions test RAM, Display, and Asynchronous I/O Adapter.

Operation

SPDPBT is initialized by power-up or the simultaneous depression of the CONTROL, SHIFT and PROG keys. At this time there is an audible signal (beep) and the 4 keyboard lamps (F0-F3) are lighted. The operator may now type one of three function keys to initiate an action:

F1 -	Attempt MDOS load from SP-type diskette.
F2 -	Attempt MDOS load from DP-type diskette.
F3 -	Execute diagnostic sequence, then re-initialize for next input.

If either F1 or F2 is typed an MDOS is not loaded for any reason (e.g. diskette not ready), the message:

"Ready device, type PROG"

is displayed on the screen. When PROG is type, SPDPBT is re-initialized for input. The diagnostic sequence consists of the following test segments:

RAM Test - All of RAM excluding the display vector area (800H - 807H) is written to with a test value. Each location in RAM is then read and checked to see if the read value equals the test value. Nine such loops are executed, one for each of the test values 00H, 80H, C0H, E0H, F0H, F8H, FCH, FEH and FFH.

Display Test - The last 2 lines in memory are written with an ASCII sequence starting with the version number of SPDPBT. The HOME and WRAP vectors are set such that the pattern will be repeated on each line until the end of the screen. The cursor is the stopped diagonally up the screen beginning at the lower left corner.

Asynchronous I/O Adapter Test - If an IOTST80 plug is present, the I/O Adapter is tested by setting and re-setting the static flags, and by loop-back transmission of test characters at various Baud rates.

Error Notation

If an error is detected during the diagnostic sequence, testing is halted and the following indications are given:

1) Continuous audible signal (beep).
2) Test number displayed on screen at HOME position, with the rest of the top line blank.
3) Test number displayed in reversed binary using the keyboard lites.

To recover from an error, type a single character corresponding to the Test number indicated. The diagnostic sequence will then continue to the next test. When all tests have been executed, SPDPBT is re-initialized for input.

NOTE: There are no possible program-detected errors during the Display test.

RAM test is test number '1'.
I/O Adapter test is test number '2'.

APPENDIX A

RUN MODIFICATION FEATURES

APPENDIX A

Certain diagnostic programs have run modification features designed to aid in pin-pointing hardware failures. Programs with these features have a slight delay between initial loading and execution of TEST 00 to allow the operator to select an option.

There are two parts to selecting one of the specific options. During the delay after loading the operator may enter the starting test number. Test execution will begin at this test; if no number is entered, execution will default to start at TEST 00.

After the desired test number has been entered, one of three specific test continuation options must be selected. Depressing the "SPACE" bar enters the step mode, causing execution to begin with the starting test. At the conclusion of each individual test rather than proceeding to the next test automatically the message "TYPE SPACE TO CONTINUE" will be displayed (along with an error message and a beep if the test failed). At this time the operator may choose to depress the "R" key to repeat the same test, the "SPACE" bar to continue on to the next test, or the "PROG" key to restart the entire diagnostic program.

The second individual test continuation option is the repeat mode, selected by depressing the "R" key. At the conclusion of the starting test, successful execution will cause the test to automatically restart. If the test fails at any time, the message "TYPE SPACE TO CONTINUE" will be displayed along with an error message and a beep. At this time depressing the "SPACE" bar or "R" key will cause the test to be repeated; depressing the "PROG" key will restart the entire diagnostic program.

The last option is the continuous run mode. By depressing the "C" key, execution will begin with the starting test number and will unconditionally proceed through all tests, not stopping at errors. After the last test, execution will restart from TEST 00 and continue in the same manner. If an error occurs during any test, the displayed error count will be updated and an error message will momentarily be displayed along with a beep.

An additional feature is that any of the test run capabilities may be aborted by depressing the "PROG" key. At the conclusion of the test being executed the entire diagnostic program will be restarted, allowing a new test continuation option to be selected.

APPENDIX B

4K FORMAT TEST MODULE RUN OPTIONS / LOADING THE 4K DIAGNOSTIC PROGRAMS

APPENDIX B

4K Format Test Module Run Options

Initialization

When the 4K format diagnostic program loads successfully, an audible signal (beep) will sound, and a command line will be displayed giving the following information:

1. Program Name
2. Program Version Number
3. Hexadecimal Loop Counter (for Continuous Mode)
4. Hexadecimal Error Counter
5. Hexadecimal Test Number Counter
6. The Prompter Message "MODE"

At this point, all counters will be cleared to 00 hex. The operator may now enter run-mode information as follows:

1. Typing the SPACE bar will present the first sequentially available test, T00. Typing a hexadecimal test number followed by the SPACE bar will present that test. (Attempting to enter a non-existent test number will cause an audible signal and will reset the test number counter to 00.)

2. Typing a hexadecimal test number followed by typing 'R' will present that test repeatedly until the PROG key is typed, at which time the program will re-initialize.

3. Typing the RETURN key will present each test sequentially beginning with T00, and then reset the test number counter to 00 after the last test has been presented.

4. Typing 'S' will present each test sequentially beginning with T00, and then will restart and run continuously until the PROG key is typed, at which time the program will re-initialize. The loop counter is updated each time the program restarts in this mode.

A second prompter message "CONT" is displayed at the completion of a test when running in the "SPACE" mode, and when the program has stopped because of an error. When this message is displayed the operator has two ways with which to proceed:

1. Typing the SPACE bar will present the next sequentially available test following the test number shown in the test number counter. If the number shown in the counter is the last available test, the counter is reset to 00 and the prompter message "MODE" is displayed, signalling that all tests have been presented.

2. Typing 'R' will repeat the test shown in the test number counter.

Error Notation

When the program is running in the continuous ('S') mode and a test fails, there is an audible signal (beep) and the error counter is updated.

In any other run mode, the program signals and updates the counter similarly, but also stops at the test which failed, and displays the prompter message "CONT" described above. At this point, the operator types either 'R' to repeat the current test, or SPACE to present the next sequentially available test. In all run modes, a reversed 'E' is written to the screen to the right of the test number counter whenever a test fails.

Typing the PROG key at any time will re-initialize the program and clear all counters in the command line.

APPENDIX B

Loading the 4K Diagnostic Programs

All 4K Diagnostic Programs are completely self-contained and operating-system independent. They may be down-line loaded through the on-board UART (refer to Appendix D).

To load using DOS or HDOS, the RUN utility must be used since the programs all reside in the lower 4K of RAM.

APPENDIX C

RUN

APPENDIX C

The RUN Command

The RUN Command loads BINARY PROGRAMS into memory. RUN HBIN80 is for HDOS while RUN BIN80 is for DOS. It is useful for loading into areas where the Operating System resides. RUN is initially loaded into the four pages of memory immediately following the Operating System. The program relocates itself to the top four pages of memory, where it is independent of the Operating System, and then loads and executes the argument filename from disk.

```
+------------------------------------------+
|                                          |
|  RUN <filename> [<filetype>] [<device>]  |
|                                          |
+---Figure X.X Syntax of the RUN Command---+
```

The location being loaded appears on the top line of the screen.

<filename>	is the name of the file to be loaded.
<filetype>	specifies the type of the file being loaded. If omitted, HBIN80 is assumed for HDOS and BIN80 for DOS.
<device>	specifies the device from which <filename> is loaded. If omitted, :H0 is assumed for HDOS and :D0 for DOS. For HDOS it must be any one of :H0 - :HF.

NOTE: **Since the largest use of the RUN Utility under HDOS will be to load programs over the HDOS Operating System, the user should make sure that all shared files are closed before using the utility and should log back in on the same terminal and then logoff after using the utility.**

RUN, RUNSS, and RUNXP

USE

These commands load **BINARY** programs into memory. They are useful for loading into areas where MDOS/80, HDOS/80, or DOS/80 resides. RUN is initially loaded into locations 900H-C0CH. RUNSS and RUNXP are initially loaded into the four pages following 1000H. Either of these four programs relocates itself to the top four pages of memory, (or the address specified on the command line, as an option in the case of RUNSS and RUNXP) where it is independent of DOS/80 or MDOS/80, and then loads and executes the argument **FILENAME**, on the command line, from disk. The location presently being loaded is displayed on the first line.

SYNTAX

RUN(SS or XP)	Filename Filetype Device (Option
RUN HBIN80	Compatible for HDOS
RUN BIN80	Compatible for DOS
RUNSS MBIN80	Compatible for an MDOS Standard Diskette Controller. (Accessible through Slots 7-10).
RUNXP MBIN80	Compatible for an MDOS Micro-Programmable Diskette Controller II and a Mini-Diskette Controller. (Accessible through Slots 7-10) Single or Double Density.
FILENAME	Name of the file to be loaded.
FILETYPE	(Optional) Specifies the type of the file being loaded. If omitted, BIN80, HBIN80, or MBIN80 is assumed according to the RUN (SS or XP) program being used.
DEVICE	(Optional) Specifies the device from which FILENAME is loaded.
(OPTION	(Optional) Only compatible with the RUNSS or RUNXP programs. This option specifies to the program that this address is the highest address, either RUNSS or RUNXP will relocate up to. This is useful for programs that are loaded which need higher memory to run. i.e., Display or Buffer Areas.

NOTE: Since the largest use of the RUN utility under HDOS will be to load programs over the HDOS operating system, the user should make sure that all shared files are closed before using the utlity and should log back in on the same terminal and logoff after using the utility.

APPENDIX D

SEND

APPENDIX D

SEND

USE: The Send program is used to down-line load remote terminals from a host running under the DOS or MDOS Operating System. Its function is to accept a command line from the operator, locate the requested file on diskette, and transmit the file to remote terminals (either OP-1 or OP-1/R).

HARDWARE: The host terminal requires a minimum of 4K of memory and at least one I/O Adapter either on the CPU (1142) or on an Alternate I/O Adapter (1172). The number of I/O devices is equal the number of remote terminals the host can transmit to simultaneously.

The remote terminals require one I/O Adapter present on the CPU (1142) to receive the transmitted file from the host. The adapter should be of the same type (RS232, 2-wire direct, or 20 mil loop) as the transmitting adapter in the host terminal. Control over the receiving adapter, loading, and execution are the responsibilities of a bootstrap program. The ASYN80 and MINIBUGR boots are recommended for OP-1 and OP-1/R remotes.

Both host and remote require a FIX1 data switch which is used to set communication parameters (baud, parity, etc.).

SYNTAX: END   file name   file type   diskette device specifier   adapter device specifier.

Filename is the name of the file to be transmitted by the host to the remote.

File type is the type of file to be transmitted. If absent, BIN80 or MBIN80 is assumed.

Diskette device specifier is the specifier for the device on which the file to be transmitted exists. If present it must be the appropriate device specifier for DOS or MDOS operating systems.

The adaptor device specifier contains the select addresses of the adaptors the host is to use for transmission. It must be preceded by (and may be optionally ended with). All select addresses must have exactly four bits high. If absent, (F0 is assumed.

EXAMPLES: The command lines listed below are identical.

```
SEND RAMCOM

SEND RAMCOM BIN80 or MBIN80

SEND RAMCOM BIN80 :D0 or MBIN80 :A0

SEND RAMCOM BIN80 :D0 (F0 or MBIN80 :A0 (F0

SEND RAMCOM (F0)
```

APPENDIX E

SENDS

APPENDIX E

SENDS

USE: The Sends Program is used to down-line load a file from a host (Asynchronous) running under the DOS or MDOS Operating System to remotes (Synchronous) containing a down-line load prom (i.e. from an OP-1 to an OP-1/RS).

HARDWARE: The host terminal requires a minimum of 4K of memory to use the SENDS program. At least one I/O Adapter either on the CPU (1142) or an Alternate I/O Adapter (1172). The number of I/O devices is equal to the number of remote terminals the host can transmit to simultaneously.

The remote terminals require one I/O adapter present on the CPU to receive the transmitted file from the host terminal. The adapter should be of the same type (RS232, 2 wire direct, or 20 mil loop) as the transmitting adapter in the host terminal. Control over the receiving adapter, loading and execution are the responsibilities of the bootstrap program. MINIBUGS boots are recomended for the OP-1/RS .

Both host and remotes require a FIX2 Data Switch which is used to set communication parameters (baud, parity, etc).

SYNTAX: SENDS file name

File name is the name of the file to be transmitted by the host to the remote.

The file type is assumed to be BIN80 or MBIN80.

The device is assumed to be :D0 or :A0.

COMTST10 Diagnostic Cable:

A COMSTS10 Diagnostic cable can be made by attaching two cannon DBC-25S connectors, and attaching the pins as shown in Figure 1.

APPENDIX E

SENDS

USE: The Sends Program is used to down-line load a file from a host (Asynchronous) running under the DOS Operating System to remotes (Synchronous) containing a down-line load prom (i.e. from an OP-1 to an OP-1/RS).

HARDWARE: The host terminal requires a minimum of 4K of memory to use the SENDS program. At least one I/O Adapter either on the CPU (1142) or an Alternate I/O Adapter (1172). The number of I/O devices is equal to the number of remote terminals the host can transmit to simultaneously.

The remote terminals require one I/O adapter present on the CPU to receive the transmitted file from the host terminal. The adapter should be of the same type (RS232, 2 wire direct, or 20 mil loop) as the transmitting adapter in the host terminal. Control over the receiving adapter, loading and execution are the responsibilities of the bootstrap program. MINIBUGS boots are recomended for the OP-1/RS .

Both host and remotes require a FIX2 Data Switch which is used to set communication parameters (baud, parity, etc).

SYNTAX: SENDS file name

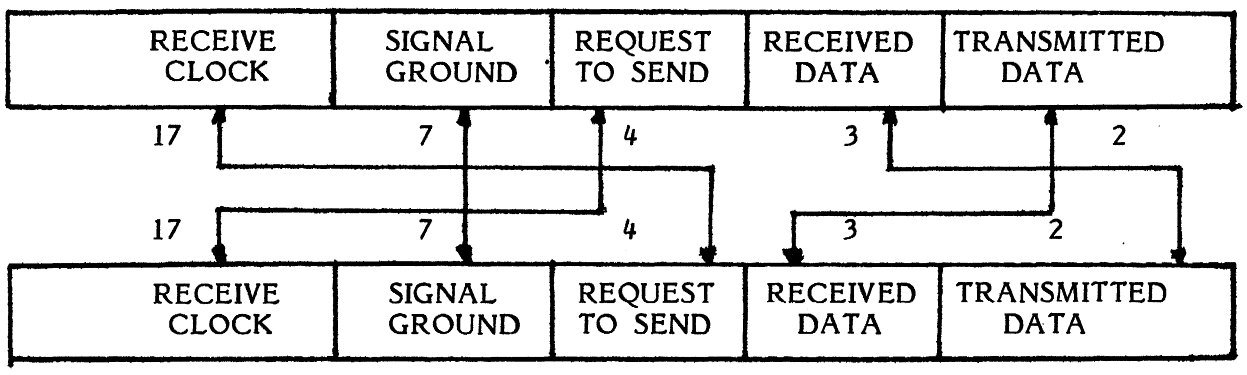
File name is the name of the file to be transmitted by the host to the remote.

The file type is assumed to be BIN80.

The device is assumed to be :D0.

COMTST10 Diagnostic Cable:

A COMSTS10 Diagnostic cable can be made by attaching two cannon DBC-25S connectors, and attaching the pins as shown in Figure 1.



CONNECTOR: CANNON DBC-25S

COMTST10 DIAGNOSTIC CABLE

FIGURE 1

APPENDIX F

LIMITED DISTRIBUTION

APPENDIX F

LIMITED DISTRIBUTION

07/15/80

SECTION	PROGRAM NAME	TITLE
	SECTION I	
1	BYTTST	Byte String Controller Test
4D	RAMPRM	PROM Version of RAMCOM
7A	IOPROM/IOPROMR	PROM Version of IOTST4
8	BSCTST	Binary Synchronous & AEG-T Binary Synchronous Controller Test
10	SYNTST	Synchronous Controller Test
19	ROMTST	Read Only Memory Test
19A	PROMTST/PROMTST2	PROM Test for OP-1/R
24	LOKTST	Lockheed Communications Controller Test
25	TAPTST	9 Track Tape Controller Test
26	MPCTST	Multiprocessor Controller Test
30	SPRNTST	Qume Sprint Printer Test
33	UACTST	Universal Asynchronous Controller
34	LOCM/LOCS	Lockheed Controller Test for OP-1/R
35	SDM/SDS	SDLC Controller Test Programs
36	DSKSTP3	Technicians Troubleshooting Program for File Controller III
38	DMATST	Direct Memory Access Test
	SECTION II	
4	KBTST4	Keyboard Test

APPENDIX G

ARCHIVED DIAGNOSTICS

APPENDIX G

ARCHIVED DIAGNOSTICS

07/15/80

SECTION	PROGRAM NAME	TITLE
2	VIDTST	Video Test
7	IOTST	Asynchronous I/O Adapter Test
9	FIXTST	Fixed Data Switch Test
11	AISWTST	Alternate I/O Adapter Fixed Data Switch Test
12	AIOTST	Alternate Asynchronous I/O Adapter Test
14	RTCTST	Real Time Clock Test
15	APRNTST	Alternate I/O Adapter Centronics/Okidata Printer Test
20	ERPTST	Extended ROM/PROM Test
27	PDCTST	Programmable Diskette Controller Test
32	DBLTST	Double Density Diskette Subsystem Test

APPENDIX H

(A)synchronous or (S)ynchronous

Many of the diagnostics (either standard or 4K) require the operator to enter an "A" or an "S". This is required for all diagnostics that have critical timing tests. The entire question being asked is whether there is an Asynchronous or Synchronous I/O Adapter on the CPU board. These two devices are programmed differently and their presence or absence is not readily detectable through software.

These devices are used by the diagnostic for determining the elapsed time between the start of an action and the completion of it. The necessity for timing events via this circuitry is that CPU timings vary among the different ONTEL terminals. An 8085 in an OP-1/R for example is faster that the 8080 in an OP-1/64, therefore, CPU-dependent code would require a different version of the diagnostic for each Ontel product line. The crystal-controlled Asynchronous or Synchronous I/O Adapters in the terminals are very accurate from one terminal to another.

The tests actually transmit characters of a known length at a specific baud rate and from this can determine the accuracy of a timing circuit to within 500 microseconds. There is no test plug or any other action required when using an Asynchronous I/O Adapter. However, an OP-1/R with 5100-1115 added to it to convert them to synchronous operation, or the OP-1/RS I (5100-1110), require an external test oscillator to drive the clocks of the USRT. This test oscillator also requires +5v, +12v and -12v from the OP-1/R.

The OP-1/RS II, 5100-1111 has a built in test oscillator and only requires a SIOTST test plug to jump the oscillator to the clocks.

APPENDIX X

Recent Enhancements to Diagnostic programs.

DIAGNOSTICS

Diagnostics capable of loading under MDOS are now available. This new diskette is D80-220-01-01.

DSKEX	A Hard Disk exerciser that allows one terminal to contain up to 4 (modified select addresses) disk controllers to thoroughly exercise a file controller and disk drives.
PDCTST	Exerciser for the Micro-Programmable Diskette Controller.
RAMBT	A DOS loader bootstrap that will continuously test RAM until a diskette is inserted in Drive 0. If a RAM error is detected the address and data bit will be displayed.
SYN143	SYNTST modified to test a SYNC COM Board with PCO 143 installed. SYNTST documentation is applicable.
R511	A technicians test for checking for shorts on Pins 5 and 11 of RAM chips.
DSKTST	Revised to allow for varying select addresses and/or slots.
FONTBIN	Revised to allow creation of 256 character fonts for the Word Display board.
TAPTST	Software error corrected that would generate erroneous Read Errors when End of Tape was encountered.
DIATST	Revised to allow for tolerance change enacted by Diablo that should be non-detectable to customer software.
TROUBLE	Revised to allow for unlimited output commands.
DSTTST	Revised to accept wider parameters on Activity Timeout that have no bearing on actual operation. In addition corrected software error that intermittenly generated an error when testing the Track Zero flag. Also, modified tests 8 and 9 to test 256 and 2048 byte writes only which are currently the only two lengths implemented by Ontel Software.

In addition the following documentation that was not complete at the time of the last library release will now be available.

IOTST4	Revised to show recent updates to program.
TAPTST	First release of the Tape Drive Diagnostic documentation.
BSC2	First release of the Binary Synchronous II documentation.

NEW:

SDLCTST	-	SDLC Controller Test.
MPCTST	-	Multi-Processor Controller Test.
PARRAM	-	Multi-Algorithm RAM Test for CPU-M which allows testing of Parity bit and Circuitry.
SENDS	-	Down-line Loader for OP-1/RS using OP-1 I/O Adapter as software USRT.
MINIBUGS	-	MINIBUGR modified for OP-1/RS.

REVISED:

WETTST	-	Revised and enhanced to generally increase testing capabilities, and to allow testing of OP-1/RW. All current Attribute Prom Overlays are available.
IOMTST	-	Modified to eliminate CPU timing-dependent errors.
WIOMTST	-	Modified to eliminate CPU timing-dependent errors.
FIXTST4	-	Entirely rewritten; both switches may be displayed simultaneously; original switch statuses are saved for restoring upon completion of testing.
AIOTST4	-	Tests added to determine if IIN Register bits functioning properly.
PRNB8	-	PRNTST modified for a Printer Controller with Select Address B8.
VIDTST	-	Added 128-Character Matrix Display Test.
VIDTST4	-	Added 128/256-Character Matrix Display Test; Added tests to more completely check WRAP Capabilities.
PRNTST4	-	Fixed bugs to correct intermittent errors in Tests 13, 14, 15 while runing in REPEAT mode.

The following Diagnostic Programs have been upgraded to eliminate CPU Timing-dependence by deploying the Asynchronous or Synchronous I/O Adapter as a clock for timing hardware events:

SRDTST
DBDTST
DSTTST
BYTTST
TAPTST
AIOTST

New Releases

DST5C	DSTTST modified to test Diskette Controller board with Select Address of 5C.
UACTST	Universal Asynchronous Controller diagnostic.
KBDTST	An entire new series of Keyboard Diagnostics. Every Ontel keyboard now has a unique diagnostic capable of running with CPUM and OP-1/R.
SPRNTST	Qume Sprint Printer diagnostic.

Revised Programs

BYTTST	Revised to run with CPUM and OP-1/R, in addition new test module incorporated.
BSCTST	Revised to run with CPUM and OP-1/R.
ASCTST	Revised to run with CPUM and OP-1/R.
WETTST	Revised to run with CPUM under MDOS. In addition overlays added and modified for new WETTST.
PARRAM	Revised to prevent erroneous parity error detection.
RTCTST4	Revised to run under MDOS and HDOS.
FIXTST4	Rewritten to simultaneously display both switches.
AIOTST4	Revised to run under MDOS and HDOS. In addition capable of running with CPUM.

New Releases

BSC2C9	BSC2 Modified for board with Select Address C9.
SYN2	SYNCHRONOUS II Communications Controller Test.
IOTSTM	IOTST4 Revised to test the I/O Adapter on the CPUM board.
ASCTST63	ASCTST modified for board with Select Address 63.
ASCTST65	ASCTST modified for board with Select Address 65.
BSC2C5	BSC2 modified for board with Select Address C5.

Revised Programs

VIDTST	Revised for MDOS and HDOS.
BSCTST	Fixed bugs in Test 4 and 5.
DSKTST	Revised for CPUM.

RTCTST4	Revised to test component failure that will arise when deselecting either the keyboard or I/O Adapter.
DISKEX	Revised for CPUM.

NEW RELEASES

AIOPROM	To enable the running of AIOTST4 from prom in the OP-1/R by moving the code to RAM for execution.
IOPROM	To enable the running of IOTST4 from prom in the OP-1/R by moving the code to RAM for execution.
KBDPROM	To enable the running of KBDTST4 from prom in the OP-1/R by moving the code to RAM for execution.
PRNPROM	To enable the running of PRNTST4 from prom in the OP-1/R by moving the code to RAM for execution.
VIDPROM	To enable the running of VIDTST4 from prom in the OP-1/R by moving the code to RAM for execution.
RTCPROM	To enable the running of RTCTST4 from prom in the OP-1/R by moving the code to RAM for execution.
FIXPROM	To enable the running of FIXTST4 from prom in the OP-1/R by moving the code to RAM for execution.
TENWAY	Tenway bootstrap has all features of the EIGHTWAY plus a down line loader and minibug.
DBLTST	DBLTST tests the double density 8.5 in diskette subsystem. The diagnostic does not support IBM or single density modes. The diagnostic also incorporates a format utility which allows the user to format diskette and a compatibility test which allows the user to verify driver compatibility.
MINTST	MINTST tests the dual sided double density mini-diskette subsystem. The test also incorporates a format utility which allows the user to formate diskettes and a compatibility test which allows the user to verify driver compatibility.
SPRNTST	Qume (Sprint) Printer Test
R2930	A WETTST Overlay for Attribute prom 508-02930-005
W3210	A WETTST Overlay for attribute prom 508-03210-002

REVISED PROGRAMS

DIATST	Modified to run under MDOS, DOS and HDOS.
BSCTST	Updated to give correct error message when activity time out failed.

DSKTST	Revised timing tests
BSC2	Fixed error messages in time out tests
BSC2C9	BSC2 modified for select address of C9 Hex
BSC2C5	BSC2 modified for select address of C5 Hex
FIXTST4	Test 1E was corrected to test the correct switch
DBDTST4	Code added to initialize interrupts
BURN	Modified to prevent premature Time Out when burning 2732 proms. In addition BURN has been modified to work in conjunction with PRO-LOG Prom Burner with Auto Baud option.

New Releases

MPDCTST:	This program is an enhancement of DBLTST. MPDCTST allows an operator to test the MPDC2 or MPDC3 diskette controller boards in any one of the three available modes (IBM, Double, or Single density). With this flexibility of MPDCTST neither PDCTST or DBLTST are needed.
PDCIFL:	Is the subassembly test for the MDC, MPDC2, and MPDC3 diskette controller boards. It allows the operator to test all the commands and status bits of the applicable boards. In addition PDCIFL is also to be used as the alignment program when aligning the heads on the five inch drives. It is intended that an operator first successfully complete running PDCIFL before proceeding to the applicable exerciser program (MPDCTST or MINTST).
ALIGN:	This program is now being made available on all Diagnostic Distribution Diskettes. It is a technicians tool to be utilized when aligning the screen.
WET8A/ WET8B:	These programs are an equivalent of WETTST. They allow the use of the standard WETTST overlays and have one necessary advantage. They require only 8K of RAM (0000 thru 1FFF hex) to test either the WORD, ETED or OP-1/RW display boards.
SIOPROM:	This is a PROM version of SIOTST4 for testing the Synchronous I/O Adapter Port on the OP-1/RS. This program is available both on the Diagnostic Distribution Diskettes and as a PROM release.
KBDCCI/ KBDCCI8:	These programs are new Keyboard Diagnostics for a specific configuration. They are equivalent except that KBDCCI8 will function on a terminal with 8K of RAM (0000 thru 1FFF hex).
R511:	This program has been made available for the first time in an MDOS and HDOS version. It is a technicians program for testing pins 5 and 11 on the OP-1/64 RAM chips for possible shorts and opens.

FONTPREP:	This program has been made available for the first time in an MDOS version. It allows customized design of character sets for the ONTEL display boards.
FONTBIN:	This program has been made available for the first time in an MDOS version. In addition this program combines the facilities of FONTBINR with FONTBIN and allows creating character sets for the new WORD display board employing 2K proms. It is used to convert character sets designed using FONTPREP to binaries that may be burned into PROM using the BURN program.
PFONT:	This program has been made available for the first time in an MDOS version. It allows an operator to print a hard copy of a character set from the character FONT created using FONTPREP. This prevents having to burn PROMS prior to completely designing a satisfactory character set.

Updated Programs

IOTSTM:	Has been enhanced to test actual reception of data at both the high and low baud rates. In addition it will now find shorts between the static flags, function correctly irregardless of 50 or 60 HZ, and test the interrupt status bits.
IOTST4:	Has been enhanced to test actual reception of data at both the high and low baud rates. In addition it will now find shorts between the static flags, function correctly irregardless of 50 or 60 HZ, and test the interrupt status bits.
AIOTST4:	Has been enhanced to test actual reception of data at both the high and low baud rates. In addition it will now find shorts between the static flags, function correctly irregardless of 50 or 60 HZ, and test the interrupt status bits.
SIOTST/ SIOTST4:	Modified to function correctly irregardless of 50 or 60 HZ.
WETTST:	Modified to correctly handle the Stop All Attribute.
RTCTST4:	Has been enhanced to test the Interrupt status bits.
PRNTST4:	Modified to allow testing the new OKIDATA printer. Required slight delay when resetting printer by use of a DVCL.

Updated

KBDTST	The keyboard tests were revised to fix an error that did not clear the test message from the screen if the test was repeated.
BSC2	A bug in Test 08 was fixed to allow proper execution in an OP-1/15.
DIATST	The test module has been added to allow executing any one test. A default timer has been added so no test will hang. An additional test has been included to verify that a WORD MOVE command will not affect printer operation.
WETTST/WET8B	A test has been added to verify that the attributes are visible in the visual enhancement mode (location 802=01) and that the attributes may start away from home or wrap.
WET8A	Revised to maintain version level.
AIOTST4	Program was revised to display the actual and expected results of the algorthim that utilizes Data Set Ready flag as a software UART.
IOTST4/IOTSTM	Revised testing of 7 data bits, and revised the display to include the actual and expected results of the algorithm that utilizes Data Set Ready as a software UART.
BSCTST	Enhanced to allow Bisychronous Controller boards having various select addresses to be tested.
SYNTST	Enhanced to allow Synchronous Controller boards having various select addresses to be tested. In addition the transmit and receive buffers are now displayed on the screen.
WIOMTST	Revised source terminating address tests.
IOMTST	Revised source terminating address tests.
RTCTST	Revision number change.
ASCTST	Enhanced to display buffers on the screen, allow operator specification of the select address, and correct misleading error messages in Tests 04 and 08.
PARRAM	PARRAM has been modified to allow the operator to selectively test various portions of RAM.
PRNTST4	Revised to allow a longer delay which is required to test the new Okidata printer.
VIDTST4	Created an MDOS version.