WANG LABORATORIES, INC.

M E M O R A N D U M

TO: Distribution

FROM: Gail Stanwyck

DATE: November 18, 1981

SUBJECT: Recent Diagnostic Release

The Wangwriter In-System Diagnostic Monitor Package has been released.

The part number for this package is 732-8001, Rev. 2195.

The diagnostics included in this package are as follows:

Z-80 CPU Instruction Test, Rev. 1110
Wangwriter VCO Adjustment Utility, Rev. 8190
Wangwriter Printer Lamp & Status Test, Rev. 6198
Wangwriter Main Memory Diagnostic, Rev. 1192
Wangwriter Display & Keyboard Test, Rev. 613A
Wangwriter Printer Test, Rev. 1175
Wangwriter Soft-Sectored Disk Controller Diagnostic, Rev. 1194
Wangwriter CTC Diagnostic, Rev. 1110

If you have any questions or problems concerning this diagnostic, please send a speedy memo addressed to the Software/Literature Control Center M/S 5222.

1.0 TITLE

Wangwriter In-System Diagnostic Monitor Package

2.0 REVISION/DATE

Date: November 17, 1981
Software Release: 2195
Documentation Release: 9195
Part Number: 732-8001

3.0 REFERENCE DOCUMENTS

Documentation for diagnostics listed in this package. See Section 5.4.

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

A Wangwriter with from 48K to 96K of main memory. The unit may have one or two disk drives. The unit's terminal is the TDC (test display console). The power-up diagnostic prom must be installed, otherwise, the package will not IPL.

4.2 Software

This monitor package is completely self-contained and needs no other software.

5.0 PACKAGE DESCRIPTION

5.1 Uses

* Engineering prototype debugging
* Manufacturing (production and QC) board test, board repair and burn in
* Customer engineering repair at the customer site, board repair and burn-in

5.2 User Interface

The operator controls testing and receives test results at the TDC.

5.3 Hardware Tested

Wangwriter hardware; see Section 5.4 below.

5.4 Programs in the Package

No.	Name (Revision)	Hardware Tested
1	Printer (1175)	Printer mechanism and adjustment.
2	Main Memory (1192)	Main Memory starting at 4000 hex.
3	CRT Memory (1110)	CRT Memory excluding soft character memory.
4	Z80 Instruction (1110)	Z80 CPU.
5	CTC (1110)	CTC (on the 7777 board).
6	Disk (1194)	Disk controller and disk drive(s).
7	Display & Keyboard (613A)	CRT attributes, soft character memory, keyboard logic and keys, CRT alignment.
8	Lamp & Status (6198)	Printer switches, lamps, sensors, character counter; 7-segment display; device type (IN07) switch bank.
9	VCO Adjustment (8190)	VCO (Voltage-Controlled Oscillator)

5.5 Modes of Operation

This package can be run in two distinct modes. One mode is the Automatic Test Sequence. This mode requires minimal operator intervention and allows burn-in (unattended testing for extended periods). The Monitor automatically runs the first six programs listed in Section 5.4. It causes the Printer and Disk Diagnostics to execute special operations. See the documentation for those diagnostics for details. For the Automatic Test Sequence, the Disk Diagnostic is instructed to test only the two innermost tracks on the disk. The other diagnostics run normally.

The other mode is normal operations, where the operator must select one or more programs to be run.

6.0 LOAD PROCEDURE

Power the unit ON, then mount the diagnostic diskette in the system disk drive. The first display should appear in about 20 seconds.

7.0 OPERATING INSTRUCTIONS

7.1 Mode Selection

The first display allowing mode selection should be on display. To select the Automatic Test Sequence, depress EXECUTE. To select normal operations, depress CANCEL. If no key is depressed within 30 seconds, the Automatic Test Sequence will be initiated.

The Automatic Test Sequence can be terminated by depressing CANCEL. When CANCEL is depressed, the program selection menu is displayed and normal operations begin. To resume the Automatic Test Sequence, it is necessary to IPL the system.

All of the monitor features described in the remainder of Section 7, are available during both automatic as well as normal testing.

7.2 Program-Selection Menu

After the normal operations mode has been initiated, the Program-Selection Menu enables the operator to select diagnostic programs.

Programs listed on the Program-Selection Menu can be selected by positioning the cursor next to the desired names and keying INSERT. Similarly, they can be deselected by keying DELETE.

Positioning the cursor next to the desired program names can be accomplished by using the following keys:

a. Up/Down arrows
b. Space (= down)
c. Backspace (= up)

Keying EXECUTE causes the Run-Time Menu to be displayed and selected diagnostic programs to start running in the order that they were shown on the Program-Selection Menu. If testing is not altered by operator action nor by hardware failure, the monitor automatically repeats the set of selected diagnostic programs.

Keying CANCEL causes the monitor to redisplay the Program-Selection Menu.

7.3 Run-Time Menu

The Run-Time menu is entered via the EXECUTE key from the Program-selection menu. It shows:

Those commands which can be keyed for direct operator control of the test programs.

Test descriptors.

An indication of whether the keyboard is LOCKED (not accepting keystrokes) or UNLOCKED (accepting keystrokes).

Error messages and user prompts.

Space is reserved at the bottom of the screen for parameter entry by the operator. Available keys are:

a. RETURN (cause cursor to go to next field)

b. DELETE (deletes everything and goes back to beginning of line)

c. EXECUTE (gives control back to the monitor)

For further details, see the documentation for individual diagnostics.

7.3.1 Command keys

The function of each command available to the operator for diagnostic control is given below. Parentheses following the command names contain the corresponding key names.

ERROR LOOP (INDENT)

INDENT causes looping (continuous repetition of a segment of code) only if there is an error. If the error is not intermittent, the loop consists of all code necessary to generate, detect and report the error. A description of the code contained in an error loop is typically found in the Module Description section of diagnostic program documentation. Non-intermittent errors will thereby be repeated at a constant rate.

If the error is intermittent and goes away after some number of iterations of the error loop, then the loop will be discontinued and any code following the loop will be executed. If no other errors occur before the end of the routine, then routine loop will be put into effect. That is, the entire routine will be executed repeatedly until error loop is deselected, as long as the error does not recur. Any time the error does recur, the looping will occur only on the code to generate, detect and report the error, rather than on the entire routine. When the error goes away, that loop is broken, and loop on test is put back into effect.

INDENT is an alternate action command. The next time it is selected, it will cause the program to discontinue looping on the failure and not loop on any subsequent failure until error loop is selected again.

ROUTINE LOOP (PAGE)

PAGE causes looping on the test routine which is currently running. If no test routine is currently running, looping will start on the next routine. There is a routine loop counter that will display how many times the routine has been repeated.

PAGE is an alternate action command. The next time the PAGE key is hit it will cause the program to discontinue looping on routine, go to the next routine and clear the routine loop counter.

STOP ON ERROR (CENTER)

CENTER causes test performance to stop when the next hardware failure is detected.

CENTER is an alternate action command. The next time the CENTER key is hit it will cause the program to resume testing from where the stop occurred and not stop on any subsequent failures (until Stop on Error is selected again and another failure occurs).

LOOP ON PROGRAM (DECTAB)

DECTAB causes looping on the diagnostic program which is currently running. If no program is currently running, looping will start on the next program. There is a program loop counter that will display how many times the program has been repeated.

DECTAB is an alternate action command. The next time it is keyed, it will cause the monitor to discontinue looping on that program, go to the next program and clear the program loop counter.

CLEAR ALL SETTINGS (STOP)

STOP deselects all the other test control commands, i.e., stop looping if looping, or resume testing if halted, etc.

ERROR LOG (COMMAND)

COMMAND causes error log to be displayed on the TDC screen in place of the Run-Time Menu.

The error log screen shows the most recent errors, up to the time when the command is invoked. Errors are identified by 8 character codes in 22 rows of 8 columns each. The first 2 digits (00-20 hex) of each code identify the channel number, which is always 1. The second 2 digits (00-20 hex) identify the program. See Section 5.4 for the number assigned to each program. The third 2 digits (00-FF hex) identify the test routine, and the fourth 2 digits (00-FF hex) identify the error. Error codes are written from left-to-right, top-to-bottom. They wrap around to the top and start overlaying when the 22 x 8 error log buffer becomes full.

Keying NEXT SCREEN returns the operator to the Run-Time Menu and clears the error log buffer. The next time the error log screen is displayed, it will show only those errors which occurred after NEXT SCREEN was keyed. This feature allows the operator to determine the frequency with which errors are happening.

Keying PREVIOUS SCREEN returns the operator to the Run-Time Menu with no effect upon the error log buffer.

PAUSE (FORMAT)

FORMAT causes the program which is currently running to stop.

FORMAT is an alternate action command. The next time it is keyed, it will cause the program to resume testing from where the halt occurred.

SCOPE LOOP (MERGE)

Scope loop is identical to error loop, except that error reporting is omitted after the first error. Scope loop would be preferred over error loop if an error is to be traced with an oscilloscope. The time required to format and display error information may be very large compared to the time required to generate and test for the error.

MERGE is an alternate action command. The next time it is keyed, it will cause the program to discontinue looping on the failure and not loop on subsequent failures (until MERGE is keyed again).

7.3.2 Test Descriptors

PROGRAM -- the name of the program currently being performed

TEST (RTN) -- the name of the test routine currently being performed

ERROR CODE -- the number of the most recently detected error

ERROR COUNT -- a count in decimal of the number of errors which have been detected. This count is cumulative; it is reset only by re-IPL'ing or by returning to the Program-Selection Menu.

PROGRAM STATUS -- the status of the program currently being performed (e.g., Test in Progress, Stop on Error, Program Pause, etc.)

PROGRAM LOOP COUNT -- a count in decimal of the number of loops which have been made through the program currently being performed. This value is only displayed when the loop-on-program option is in effect. It is cleared when the loop-on-program option is deselected.

PROGRAM SET LOOP COUNT -- a count in decimal of the number of loops which have been made through the set of programs. It is cleared by re-IPL'ing or by returning to the program selection menu.

ROUTINE LOOP COUNT -- identical to PROGRAM LOOP COUNT except that this count applies to routines rather than to programs.

7.3.3 Keyboard Lock/Unlock

The keyboard is normally in a condition where operator keystrokes are accepted immediately after they are made. In this condition, the keyboard is said to be unlocked. During diagnostics that are dependent on very exact timing, the keyboard will be locked and keystroke handling will be temporarily suspended.

If you want to activate a function such as 'loop on error' or 'loop on routine' while the keyboard is locked, depress the appropriate command key. The unit will store the first key struck and process it the next time the keyboard becomes unlocked. There may be a significant delay between the time the key is struck and the time it is acknowledged (by highlighting the selected command).

The unit stores only the first key struck. All subsequent keystrokes made while the keyboard is locked will be ignored.

The condition of keyboard is shown on the TDC near the center of the screen, below the list of command keys.

7.3.4 Error Messages and User Prompts

The test program currently being performed writes error messages and user prompts in the lower half of the Run-Time Menu. If more than one error occurs, only the last error message will be left on display, although the error count is updated for each error.

For further details concerning error messages and user prompts, see the documentation for individual diagnostic programs.

7.3.5 Run-Time Menu Operation

Testing starts when the Run-Time Menu is entered and ends when the Run-Time Menu is exited. Entry is accomplished via the EXECUTE key from the Program-Selection Menu; exit back to the Program-Selection Menu is accomplished via the CANCEL key.

Programs are performed in the order in which their names appear on the Program-Selection Menu. As soon as all selected programs have been performed, testing automatically cycles back to the first selected program.

The operator can control the testing process by utilizing the commands described above and can monitor the results by observing the test descriptors. The 176 most recent errors can be seen by keying COMMAND.

7.3.6 Program Status Prompts

During normal operating condition a number of program status messages will be displayed on the Run-Time Menu next to Program Status. These prompts include:

Test in Progress
Wait for Next Test
Program Paused
Return from Pause
Looping on ERROR
Looping on ROUTINE
Looping on PROGRAM
Scope Loop
Resuming from Loop
Stopped on Error
Resuming from Stop

7.4 Operation Summary

Power the unit ON, then insert the program diskette.

Wait for the Program-Selection Menu to appear and position the cursor using:

Up/Down arrows
Space bar (= down)
Backspace (= up)

Pick one or more diagnostics using:

Insert for selection
Delete for deselection

Key EXECUTE to select the diagnostics and to advance to the Run-Time Menu.

Monitor test results by watching Run-Time Menu.

Control test performance by using the commands listed below. Function key names are in parentheses.

ERROR LOOP (INDENT) -- Loop on routine in which the next failure occurs

LOOP ON PROGRAM (DECTAB) -- Loop on current diagnostic program

PAUSE (FORMAT) -- Halt the program

TEST LOOP (PAGE) -- Loop on current test routine

CLEAR ALL SETTINGS (STOP) -- Stop looping, if looping, or resume testing if halted

SCOPE LOOP (MERGE) -- Loop on next test routine in which a hardware failure occurs

STOP ON ERROR (CENTER) -- Stop the diagnostic program where the next failure is detected

ERROR LOG (COMMAND) -- Display the error log screen. Use NEXT SCRN to return to the Run-Time Menu and clear the error log. Use PREV SCRN to return to the Run-Time Menu and save the error log

8.0 REVISION HISTORY

Rev 2195 -- first release

9.0 MISCELLANEOUS

NOTE: To continue after Stop on Error - select Pause - deselect Stop on Error - Select Stop on Error and deselect Pause. This should allow the user to catch the next error.

1.0 TITLE

Z-80 CPU Instruction Test

2.0 REVISION/DATE

Date: November 17, 1981
Software Release: 1110
Documentation Release: 9110

3.0 REFERENCE DOCUMENTS

Z80 Assembly Language Programming Manual
Z80 Programming Reference Card

4.0 INTRODUCTION

4.1 Purpose

The purpose of this program is to verify the correct execution of all the instructions in the Z-80 Instruction Set.

4.2 Hardware Tested

The only hardware tested in this program is the Z-80 CPU.

4.3 User

This program may be used by anyone who wishes to verify that a Z-80 CPU can correctly perform all the instructions it was designed to.

5.0 CONFIGURATION REQUIREMENTS

5.1 Hardware

Any VS, OIS, WP or Wangwriter Master with a floppy disk drive, and the (TDC).

5.2 Software

This is a System Diagnostic program which interfaces with the VS, OIS, WP, or Wangwriter Monitor.

6.0 PROGRAM DESCRIPTION

6.1 General

This program tests:

The accumulator with logical and rotate commands.
8-bit multi-purpose registers.
The remainder of the rotate and shift commands.
The 16-bit register pairs using indirect addressing.
The exchange commands
Index registers
The stack pointer
The Push and Pop commands.
The arithmetic group
The bit test group
The compare and block transfer group
The Call and Ret commands
The indexed addressing mode

Each instruction is performed at least once, and some are performed several times. Error checking is done at each stage during the test of an instruction. If the data and/or flags are incorrect, an error message will be generated. At this time the operator can select any of the error handling options available.

TEST NO.	TEST NAME
01	AND
02	RLARRA
2A	RLCARRCA
04	BREG
05	CREG
06	DREG
07	EREG
08	HREG
09	LREG
0A	RLRR
0B	RLCRRC
0C	SLASRA
0D	SRL
0E	RLDRRD
0F	HLREG
10	BCREG
11	DEREG
12	EXDEHL
13	EXX
14	SPIYIX
15	PUSH POP

TEST NO.	TEST NAME
16	CCFSCF
17	ADC8
18	ADC16
19	ADD8
1A	ADD16
1B	SUB
1C	DAA
1D	BIT
1E	RES
1F	SET
20	CPI
21	CPD
22	CPDR
23	CPIR
24	LDI
25	LDD
26	LDIR
27	LDDR
28	CALLRET
29	IXYD

6.2 Module Descriptions

Logic Function Routine
Error Code (01)

Test 1

Screen Name: AND

The AND, OR & XOR instructions are tested. Each instruction is performed on the Accumulator and one other register. The results of the operations go to the Accumulator. First, the instruction is performed with both registers containing the same data, i.e. (1111) AND'ed with (1111). Then complementary data is used, i.e. (1010) AND'ed with (0101). If the results found in the Accumulator are incorrect, error (01) is called.

Test Accumulator with RRA & RLA Instructions
Error Code (02)

Test 2

Screen Name: RLARRA

This routine will check the integrity of the Accumulator and determine whether the carry flag is operating properly. The Accumulator is loaded with all ones, and a rotate left command (RLA) is performed. This will cause each bit to be copied into the place immediately to its left. The digit from bit 7 will be copied into the carry bit which should cause the carry flag to set. If the carry bit does not set, error (02) will be called. Also, since the carry flag was clear prior to the rotate command, a zero will be copied into bit 0 of the Accumulator. With each successive rotate command, this zero will move one place to the left. After the ninth (RLA) command, this zero should be copied into the carry bit, which should cause the carry flag to clear. If it does not clear, or if it clears at the wrong time, error (02) will be called. To test the rotate right command, the precedure is reversed. Bit 0 will be copied into the carry bit and the zero from the carry flag will be copied into bit 7.

Test Accumulator with RLCA & RRCA Instructions
Error (02)

Test 2A

Screen Name: RLCARRCA

The rotate circular commands, (RLCA) and (RRCA), are similar to the ones in the previous routine. The difference is that when a rotate left circular command is given, and bit 7 is copied into the carry bit, it is also copied into bit 0 of the accumulator. Therefore, if the Accumulator started off with all ones, it should remain unchanged, regardless of how many rotate circular commands are given. This routine tests the instruction by checking the carry flag for set after the first rotate command, and then testing for even parity. If the carry flag doesn't set, or the parity is ever odd, error (02) will be called. To test the rotate right circular command, the procedure is exactly the same. The Accumulator remains unchanged, the carry flag should set, and the parity should always be even.

Register B, C, D, E, H, & L Tests
Error Code (04)

Tests 4, 5, 6, 7, 8, & 9

Screen Names: BREG, CREG, DREG, EREG, HREG, LREG

Since the Accumulator has already proved to be in working order, it is used here as a reference, against which, all the remaining 8-bit registers will be tested. This is accomplished by the following procedure. The Accumulator and the register under test are both loaded with ones. They are then compared, and if they are not the same, error (04) will be called. The rotate left command is performed on both the register and the Accumulator. The carry flag should set, and both registers should agree. The rotate command is repeated, and the registers compared until the carry flag clears. This test is repeated for the B, C, D, E, H, & L registers.

RL & RR Instruction Test
Error Code (05)

Test A

Screen Name: RLRR

These instructions are tested using register B. Register B is loaded with ones, and the rotate left instruction (RL) is performed. The carry flag should set, and the zero from the carry bit should be copied into bit 0 of register B. The rotate command is repeated until the zero has been moved through the register and back into the carry bit. This will cause the carry flag to clear. The (RR) command is tested in the same way, but in the opposite direction. A count is made of the number of times the rotate commands are performed. If the carry flag doesn't clear, or if it clears at the wrong time, error (05) will be called. If the carry flag clears at the right time, but, register B contains the wrong data, error (05) will be called.

RLC & RRC Instruction Test
Error Code (06)

Test B

Screen Name: RLCRRC

These instructions are tested using register C. Register C is loaded with (11111110), and the rotate left circular instruction (RLC) is performed. The one from bit 7 of register C should be copied into both the carry bit (setting the carry flag) and bit 0 of register B. The rotate command is repeated until the zero has been moved through the register and into the carry bit, causing the carry flag to clear. The (RRC) command is tested by loading register B with (10000000). Then performing a rotate right circular until the one in bit 7 has moved through the register into the carry bit, setting the carry flag. A count is made of the number of times the rotate commands are performed. And if the carry flag doesn't clear, or if it clears at the wrong time, error (06) is called. If the carry flag clears at the right time, but register C contains the wrong data, error (06) is called.

SLA & SRA Instruction Test
Error Code (07)

Test C

Screen Name: SLASRA

These will be tested using Register D. Register D is loaded with all ones. The shift left arithemetic (SLA) command will cause each bit to be copied into the place immediately to its left. Bit 7 of register D will be copied into the carry bit. Bit 0 will be loaded with a zero. The carry flag should set after the first shift command, and should not clear until the eighth shift command. If it does clear before the eighth time, or the data in register D is wrong, error (07) will be called. The (SRA) command is tested by loading register D with (10000000). The (SRA) instruction is performed until the one at bit 7 has been moved through the register and into the carry bit, causing the carry flag to set. A count is kept of the number of times each instruction is to be performed. If the carry flag sets or clears at the wrong time, or if register D contains the wrong data, error (07) will be called.

SRL Instruction Test
Error Code (08)

Test D

Screen Name: SRL

The shift right logical (SRL) instruction is tested using Register E. Register E is loaded with all ones. The (SRL) instruction is performed causing all bits to be shifted one place to the right. Bit 0 should be copied into the carry bit, causing the carry flag to set. Bit 7 of Register E will be loaded with a zero. The (SRL) instruction will be repeated until the zero from bit 7 has moved through the register and into the carry bit, clearing the carry flag. At this time, Register E should contain all zeros. If it does not, or if the carry flag cleared too soon, error (08) will be called.

RLD & RRD Instruction Test
Error Code (09)

Test E

Screen Name: RLDRRD

The rotate left/right digit (RLD) & (RRD) instructions are performed on the Accumulator and a memory location. The Accumulator is loaded with the number (AA)hex, and the memory location is loaded with the number (55)hex. For the (RLD) instruction, the lower four bits of the memory location are copied into the upper four bits of the memory location. The upper four bits of the memory location are copied into the lower four bits of the Accumulator. The lower four bits of the Accumulator are copied into the lower four bits of the memory location. The Accumulator should now equal (A5)hex, and the memory should equal (5A)hex. The (RRD) instruction is tested in the same way, using the same initial data. The Accumulator should equal (A5)hex, and the memory location should equal (A5)hex. If the data is incorrect, error (09) will be called.

Check Register Indirect Address Using HL, BC, and DE Pair
Error Code (0A)

Tests F, 10, 11

Screen Names: HLREG, BCREG, DEREG

Load the HL register with the read test address. Load the Accumulator from the address specified by the HL register. If the wrong data was read, call error (0A). Load the HL register with the write test address. Load the address specified by HL from the Accumulator. Clear the Accumulator and read the data back. If it is incorrect, call error (0A). Restore test locations to initial values and repeat the test for the BC, and DE registers.

Check the EX DE,HL Command
Error Code (0B)

Test 12

Screen Name: EXDEHL

When this command is performed, the contents of the DE register are copied into the HL register and the contents of the HL register are copied into the DE. Both registers are loaded with a different number to test this command. The command is performed and the contents are tested for correctness. If they are incorrect, error (0B) will be called.

Check EXX Command
Error Code (0B)

Test 13

Screen Name: EXX

This command exchanges the contents of the BC, DE, and HL registers with the alternate BC, DE, and HL registers. Load the main registers with zeros. The EXX command is performed, which loads the zeros into the alternate registers. The main registers are now loaded with non-zero numbers. The EXX command is performed, which should load the non-zero numbers into the alternate registers and bring the zeros back to the main registers. If all zeros are not present, error (0B) will be called. The EXX is performed again, to bring the non-zero numbers back to the main registers. If the wrong numbers are found, error (0B) will be called.

Check the IX, IY, & SP Register
Error Code (0C)

Test 14

Screen Name: SPIYIX

The IX register is tested with four different bit patterns (binary numbers). After each pattern is loaded, the register is tested for the correct number. If the number is incorrect, error (0C) will be called. This test is repeated for the IY and SP registers.

Check the PUSH & POP Commands
Error Code (OD)

Test 15

Screen Name: PUSHPOP

The stack is an arbitrary place in memory, controlled by the stack pointer (SP register), which acts as a first in, last out memory device. The SP register contains the first, or top address of the stack. For the test, a 16 bit number is PUSH'ed onto the stack. This means that the number will be stored in memory at the address specified by the stack pointer (SP register). First, the SP is decremented; this will give us the address where the upper 8 bits of the number is to be stored. The SP is decremented again, giving us the address for the lower 8 bits. To test the POP command, the procedure is just reversed. The lower 8 bits of our test number are retrieved from the present address of the SP. The SP is incremented, and the upper 8 bits of our number are retrieved from this address. The SP is incremented again, which concludes the POP command. The number is tested, and if not correct, error (OD) will be called.

CCF and SCF Routine
Error Code (OE)

Test 16

Screen Name: CCFSCF

Load the Accumulator with ones. Perform a rotate left instruction on the Accumulator, which will cause a one to be copied into the carry bit, setting the carry flag. Execute a CCF (clear carry flag) instruction. If the carry flag is still set, error (OE) will be called. "Exclusive Or" the Accumulator with itself to make sure the carry flag is clear. Perform a SCF (set carry flag) instruction. If the carry flag does not set, error (OE) will be called.

8 Bit ADC Routine
Error Code (OF)

Test 17

Screen Name: ADC8

The ADC is the add with carry command. It automatically increments the present sum if the carry flag was set from a previous operation. A number is added to the Accumulator with the carry flag clear. The Accumulator is tested, and if it is not correct, error (OF) is called. Next, a number is added to the Accumulator with the carry flag set. The Accumulator is tested again, and if it is not correct, error (OF) is called.

16-Bit ADC Routine
Error Code (10)

Test 18

Screen Name: ADC16

The HL register and the DE register are loaded with two different 16-bit numbers. The carry flag is cleared, and the ADC command is performed. The HL register contains the answer. It is tested, and if found incorrect, error (10) will be called. The carry flag is set, and the add with carry (ADC) command is performed again. The HL register is tested, and if it is found to be incorrect, error (10) will be called.

8-Bit ADD Routine
Error Code (11)

Test 19

Screen Name: ADD8

The ADD command is not affected by the carry flag. The contents of the H register are added to the Accumulator. The Accumulator should contain the answer. If it is not correct, error (11) will be called.

16-Bit ADD Routine
Error Code (12)

Test 1A

Screen Name: ADD16

The contents of the DE register are added to the contents of the HL register. The sum will be in the HL register. The contents of the BC register are added to the contents of the IX register. The sum will be in the IX register. The contents of the SP register are added to the contents of the IY register. The sum will be in the IY register. All sums will be checked, and if not correct, error (12) will be called.

Subtract Routine & Subtract with Carry
Error Code (13)

Test 1B

Screen Name: SUB

The SUB (subtract) and SBC (subtract with carry) commands are tested in the following way:

Subtract a number from the Accumulator. The Accumulator contains the answer. If it is not correct, error (13) will be called. Set the carry flag and subtract with carry (SBC) a number from the Accumulator. If the answer is not correct, error (13) will be called. Subtract without carry (SWB) the DE register from the HL register. The HL register contains the answer. If it is not correct, error (13) will be called.

DAA Routine
Error Code (14)

Test 1C

Screen Name: DAA

The DAA (decimal adjust accumulator) instruction is used when it is necessary that the results of an arithemetic operation be in decimal form rather that hexadecimal form. It takes into account the carries and borrows that might have occured from previous arithemtic operation, and leaves these flags in the correct mode after DAA. The Accumulator is loaded with some number, and the carry flag is cleared. DAA is performed, and the results are checked. If it is not correct, error (14) will be called. Next, the carry flag is set, and the DAA instruction is performed again. The Accumulator is checked, and if it is not correct, error (14) will be called.

Bit Test Routine
Error Code (15)

Test 1D

Screen Name: BIT

The BIT command tests the condition of a specific bit in a register (i.e. whether it is a one or a zero). Initially, Register B is loaded with (00000001). For this test, starting with bit 0, each bit of Register B is tested for a one. The Accumulator keeps a count of the number of bit tests made. When a 1 is found, the Accumulator is checked to see if the right bit was detected. If a bit was skipped (i.e. the BIT command failed in some way), error (15) will be called. After the 1 in bit 0 is detected, it is rotated left one place and the BIT test is begun again from bit 0. When the one from bit 0 of register B has been shifted to bit 7, and has been properly detected. The Accumulator count is checked, and if it is correct, the routine is exited.

RESET Test Routine
Error Code (16)

Test 1E

Screen Name: RES

The RES command will reset a specific bit of the register being acted upon. Register D is loaded with ones. Then each bit is tested for a one, starting with bit 0. Every time a one is detected, the Accumulator should contain the number corresponding to that bit. If they do not equal, error (16) will be called. The correctly detected one is then reset. The bit count in the Accumulator is incremented, and the bit test is repeated from the beginning. When bit 7 has been reset, a final pass of the bit test is made. No ones should be detected, and the Accumulator should equal eight. If it doesn't, error (16) will be called.

SET Test Routine
Error Code (17)

Test 1F

Screen Name: SET

The SET command will set a specific bit of the register being acted upon. Register B is loaded with all zeros. A SET command is performed on bit 7 causing it to go to a one. Each bit is tested for a one, starting with bit 0. When a one is detected, the Accumulator is checked to see if it's the right bit. If it is not, error (17) will be called. The Accumulator is incremented; the next bit in the sequence is set; and the test is repeated. When a one has been detected at bit 0, and the Accumulator is correct, the routine is exited.

CPI Routine
Error Code (18)

Test 20

Screen Name: CPI

CPI (compare & increment) is a block search command. The byte value one wishes to find is loaded into the Accumulator. The memory address where the search is to start, is loaded into the HL register. The number of locations to be checked are loaded into the BC register. When the instruction is performed, the zero flag will set if a match has been found. If not, the HL register is incremented to the next address. The BC register is decremented, and if it does not equal zero, the instruction is ready to be repeated. The first of (16) bytes are loaded into the Accumulator. The number of bytes to be searched through are loaded into the BC Register. The starting address of the search is loaded into the HL register. A search will be made through all (16) bytes until either a match is found, or the BC Register equals (0). The alternate Accumulator will contain the number that the BC Register should equal when a particular match is found. If it is not correct, error (18) will be called. When a correct match is found, the next byte in memory will be loaded into the Accumulator, and a search will be made for it. If the BC Register reaches zero, and no match has been found, error (18) will be called. When the last byte has been found, and the BC Register count is correct, the routine is exited.

CPD Routine
Error Code (19)

Test 21

Screen Name: CPD

CPD (compare & decrement) is a block search command. The byte value one wishes to find is loaded into the Accumulator. The memory address where the search is to start, is loaded into the HL register. The number of locations to be checked are loaded into the BC register. When the instruction is performed, the zero flag will set if a match has been found. If not, the HL register is decremented to the next address. The BC register is decremented, and if it does not equal zero, the instruction is ready to be repeated. To test, the last of (16) bytes are loaded into the Accumulator. The number of bytes to be searched through are loaded into the BC Register. The starting address of the search is loaded into the HL register. A search will be made through all (16) bytes until either a match is found or the BC Register equals zero. The alternate Accumulator will contain the number that the BC Register should equal when a particular match is found. If it is incorrect, error (19) will be called. When a correct match is found, the next byte in memory will be loaded into the Accumulator, and a search will be made for it. If the BC Register reaches zero, and no match has been found, error (19) will be called. When the top byte has been found, and the BC Register count is correct, the routine is exited.

CPDR Routine
Error Code (1A)

Test 22

Screen Name: CPDR

CPDR (compare, decrement & repeat) is a block search command. The byte value one wishes to find is loaded into the Accumulator. The memory address where the search is to start is loaded into the HL register. The number of locations to be checked are loaded into the BC register. When the instruction is executed, it continues until either a match is found, or the BC register equals zero. Each time a match is not found, the HL register is decremented to get the address of the next location. The BC register is decremented to update the count of the number of locations left to be checked. To test, the last of (16) bytes are loaded into the Accumulator. The number of locations to be searched are loaded into the BC Register. The starting address of the search is loaded into the HL register. A search will be made through all (16) locations until either a match is found, or the BC Register equals zero. If no match has been found, and the BC equals zero, error (1A) will be called. When a correct match has been found, the next byte in memory will be loaded into the Accumulator and a search will be made for it. When a correct match has been found for the last byte, the routine is exited.

CPIR Routine
Error Code (1B)

Test 23

Screen Name: CPIR

CPIR (compare, increment & repeat) is a block search command. The byte value one wishes to find is loaded into the Accumulator. The memory address where the search is to start is loaded into the HL register. The number of locations to be checked are loaded into the BC register. When the instruction is executed, it continues until either a match is found, or the BC register equals zero. If no match is found, the HL register is incremented to the next address. The BC register is decremented to update the byte count. The instruction will repeat itself automatically until either a match is found, or the BC register goes to zero. The first of (16) bytes are loaded into the Accumulator. The number of locations to be checked are loaded into the BC Register. A search will be made through all (16) bytes until either a match is found, or the BC Register equals zero. If no match has been found, or the BC Register went to zero, error (1B) will be called. When a correct match is found, the next byte in memory will be loaded into the Accumulator, and a search will be made for it. When a correct match has been found for the last byte, the routine is exited.

LDI Routine
Error Code (1C)

Test 24

Screen Name: LDI

LDI (load & increment) is a block transfer command. The HL Register is loaded with the source starting address. The DE Register is loaded with the destination starting address. The BC Register is loaded with the number of bytes to be transferred. The instruction loads the location specified by the DE register with the contents of the location specified by the HL register. It increments the HL and DE registers to get the new addresses, and then decrements the BC to update the byte count. If the instruction is repeated, each byte will be copied into its new location until the BC Register equals zero. A transfer of 16 bytes is made. The first byte that should have been transferred is loaded into the Accumulator, and a block search instruction is used to find it. If no match is found, error (1C) will be called. If a match is found, the next byte is loaded into Accumulator and the search is performed again. This is repeated until the last byte that was transferred has been found.

LDD Routine
Error Code (1D)

Test 25

Screen Name: LDD

LDD (load & decrement) is a block transfer command. The HL Register is loaded with the source starting address. The DE Register is loaded with the destination starting address. The BC Register is loaded with the number of bytes to be transferred. The instruction loads the location specified by the DE register with the contents of the location specified by the HL register. It decrements the HL and DE registers to get the new addresses, and then decrements the BC to update the byte count. If the instruction is repeated, each byte will be copied into its new location until the BC Register equals zero. A transfer of 16 bytes is made. The first byte that should have been transferred is loaded into the Accumulator, and a block search instruction is used to find it. If no match is found, error (1D) will be called. If a match is found, the next byte is loaded into Accumulator and the search is performed again. This is repeated until the last byte that was transferred has been found.

LDIR Routine
Error Code (1E)

Test 26

Screen Name: LDIR

LDIR (load, increment & repeat) is a block transfer command. The HL Register is loaded with the source starting address. The DE Register is loaded with the destination starting address. The BC Register is loaded with the number of bytes to be transferred. The instruction loads the location specified by the DE register with the contents of the location specified by the HL register. It increments the HL and DE registers to get the new addresses, and then decrements the BC to update the byte count. Once the instruction has been executed, it continues automatically. Each byte will be copied into its new location until the BC register equals zero. To test, a transfer of 16 bytes is made. The first byte that should have been transferred is loaded into the Accumulator, and a block search instruction is used to find it. If no match is found, error (1E) will be called. If a match is found, the next byte is loaded into Accumulator and the search is performed again. This is repeated until the last byte that was transferred has been found.

LDDR Routine
Error Code (1F)

Test 27

Screen Name: LDDR

LDDR (load, decrement & repeat) is a block transfer command. The HL Register is loaded with the source starting address. The DE Register is loaded with the destination starting address. The BC Register is loaded with the number of bytes to be transferred. The instruction loads the location specified by the DE register with the contents of the location specified by the HL register. It decrements the HL and DE registers to get the new addresses, and then decrements the BC to update the byte count. Once the instruction has been executed, it continues automatically. Each byte will be copied into its new location until the BC register equals zero. A transfer of 16 bytes is made. The first byte that should have been transferred is loaded into the Accumulator, and a block search instruction is used to find it. If no match is found, error (1F) will be called. If a match is found, the next byte is loaded into Accumulator and the search is performed again. This is repeated until the last byte that was transferred has been found.

CALL & RET Routine
Error Code (20)

Test 28

Screen Name: CALLRET

A CALL (to subroutine) instruction is similiar to a jump instruction with one exception; the address of the instruction immediately after the CALL is stored in the stack. As in a jump, program control is now turned over to the address that the CALL was made to. The RET (return) instruction is used to exit the subroutine. It allows program control to be returned to the instruction immediately following the CALL instruction. This can be done because the address of that location was stored in the stack. The Accumulator is cleared, setting the zero flag. A CALL is made to subroutine(1). If it is not executed, and the Accumulator is zero, error (20) will be called. Subroutine(1) executes a CALL if zero instruction to Subroutine(2). If this instruction fails, a call to error (20) will be made. Subroutine(2) loads the accumulator with ones, clearing the zero flag. It then executes a Return if not zero instruction. Should this fail, the next instruction will call error (20). After returning from Subroutine(2), if no errors have been detected, the Call if zero instruction will be skipped, and the return from Subroutine(1) will be executed. If the zero flag is clear, the routine will be exited.

IX+d, IY+d Addressing Routine
Error Code (21)

Test 29

Screen Name: IXYD

This routine tests the index addressing mode. The IX or IY registers contain the reference address. The (+d) represents the displacement by which the reference address is to be modified. Load the IX Register with the write reference address. Load the location specified by the IX Register plus the displacement value with ones. Test that location and see if the data is correct. If it is not, error (21) will be called. Load the IY Register with the read reference address. Load the Accumulator from the address specified by the IY Register plus the displacement value. If the Accumulator is not correct, error (21) will be called.

7.0 LOAD PROCEDURE

Refer to VS, OIS, WP, and Wangwriter Monitor Load Procedures.

8.0 ERROR CODES & ERROR MESSAGES

The following is a list of error codes and error messages:

ERROR 01) AND, OR, XOR COMMAND FAILURE
ERROR 02) ACCUMULATOR FAILURE
ERROR 04) 8-BIT REGISTER FAILURE
ERROR 05) RL RR COMMAND FAILURE
ERROR 06) RLC RRC COMMAND FAILURE
ERROR 07) SLA SRA COMMAND FAILURE
ERROR 08) SRL COMMAND FAILURE
ERROR 09) RLD RRD COMMAND FAILURE
ERROR 0A) HL, BC, DE REGISTER FAILURE
ERROR 0B) EX DE,HL OR EXX COMMAND FAILURE
ERROR 0C) IX, IY, OR SP REGISTER FAILURE
ERROR 0D) PUSH OR POP ERROR
ERROR 0E) CCF OR SCF COMMAND FAILURE
ERROR 0F) 8-BIT ADC COMMAND FAILURE
ERROR 10) 16-BIT ADC COMMAND FAILURE
ERROR 11) 8-BIT ADD COMMAND FAILURE
ERROR 12) 16-BIT ADD COMMAND FAILURE
ERROR 13) SUBTRACT CMD FAILURE
ERROR 14) DAA COMMAND FAILURE
ERROR 15) BIT TEST FAILING
ERROR 16) RES CMD FAILURE
ERROR 17) SET CMD FAILURE
ERROR 18) CPI CMD FAILURE
ERROR 19) CPD CMD FAILURE
ERROR 1A) CPDR CMD FAILURE
ERROR 1B) CPIR CMD FAILURE
ERROR 1C) LDI CMD FAILURE
ERROR 1D) LDD CND FAILURE
ERROR 1E) LDIR CMD FAILURE
ERROR 1F) LDDR CMD FAILURE
ERROR 20) CALL OR RET CMD FAILURE
ERROR 21) INDEXED ADDRESSING FAILURE

9.0 REVISION HISTORY

Rev. 1110 -- Revision number was changed from 1.1 to 1110. No software changes were made.

1.0 TITLE

Wangwriter Printer Lamp & Status Test

2.0 REVISION/DATE

Date: November 17, 1981
Software Release: 6198
Documentation Release: 9198

3.0 REFERENCE DOCUMENTS

Documentation for the Wangwriter In-System Diagnostic Monitor Package

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This program runs under the control of the Wangwriter In-System Diagnostic Monitor. The monitor and all of its programs are supplied together on a single diskette. No other software is required.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging.
* Manufacturing (production and QC) board test, board repair and burn in.
* Customer Engineering repair at the customer site, board repair and burn-in.

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

Printer panel switches, lamps, sensors, character counter; 7-segment display; slave type (IN07) switch bank; associated cables.

5.4 Tests in the Program

This program has one test, described below.

5.5 Test Description

The test creates a screen display (shown below), which shows the operator the condition of printer switches and sensors, and of the slave type (IN07) switch bank.

********** MESSAGE AREA OF THE MONITOR **********

Panel Switches	SEL	RU	RD	FU	FD	TOF	PL
	ON	OFF	OFF	OFF	OFF	OFF	OFF
Status Switches	RM	LM	RO	PO	CO	DH	RT
	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Option Switch 0 0 1 0 0 0 0 0
(Bits 1 2 3 4 5 6 7 8)

5.5.1 Panel Switches

The meanings of the abbreviations used in the panel switches section of the menu are:

Panel Switch Name	Menu ID	Associated Indicator
Select Key	SEL	Select Lamp
Roll Up	RU	Change Ribbon Lamp
Roll Down	RD	Change Paper Lamp
Line Feed Up	FU	Error Indicator
Line Feed Down	FD	Change Daisy Lamp
Top of Page	TOF	Character Counter
Paper Load	PL	N O N E

The indication of OFF or ON in the menu refers to the expected condition of the 'associated indicator' in the table above. Thus, the 'ON' shown below the 'SEL' in the menu indicates that the select lamp should be ON; the 'OFF' shown below 'RU' indicates that the change ribbon lamp should be OFF; etc. Each time a panel switch is depressed both the menu and the associated indicator should change condition from OFF to ON or vice versa.

There are two exceptions to this general pattern. The TOF switch controls the character counter visible at the rear bottom of the unit. Each depression of the TOF switch should cause the counter to increment by one.

The other exception is the FU switch. Depressing it causes a sequential pattern of segments to become illuminated in the 7-segment error display (visible from the rear of the unit). First, it blanks the error indicator and then it sequences as illustrated below, turning on the first element, then the second, etc. until all the segments are turned on, including the decimal point. Then it starts over again by blanking the indicator.

```
  111
 2   3
 2   3
  444
 5   6
 5   6
*  777
```

5.5.2 Status Switches

The meanings of the abbreviations in the status switches section of the menu are:

Sensor Name	Sensor Menu Id Code
Right Margin	RM
Left Margin	LM
Ribbon Out	RO
Paper Out	PO
Cover Open	CO
Daisy Home	DH
Ribbon Type	RT

When the appropriate sensor is blocked either by an opaque or reflective material the condition of the sensor will be displayed on the menu. If it is blocked, the indicator will say ON, if it is not blocked then it will indicate OFF.

5.5.3 Option Switch

This section of the menu displays the condition of the slave type switches. There are eight bits displayed on the screen. If a switch is in the OFF position (as shown on the plastic case of the switch bank), then a '1' will be displayed in that particular bit location, and vice versa. Note the reversal of bits: OFF is 1 and ON is 0.

6.0 LOAD PROCEDURE

For a description of how to load and start a program, see documentation for the diagnostic monitor.

7.0 OPERATING INSTRUCTIONS

This program requires operator intervention and observation. The operator is required to depress a desired switch and observe on the menu that in fact this switch is functioning properly. Associated with most switches there is some associated function that must be observed to make sure that it is also functioning properly.

8.0 REVISION HISTORY

Software Rev 6110 (1/14/81) - First release (preliminary).
Documentation Rev 9110 (1/14/81) - First release (preliminary).
Software Rev 6150 (5/18/81) - Second Release
    Reason for modification - Installation of ribbon type switch on the carriage assembly.
Documentation Rev 9150 (5/18/81) - Second Release
    Same as Software Above

9.0 MISCELLANEOUS

This program relies on the operator to detect errors and does not itself call out errors. For this reason, Appendix A (Test and Error Information Table) and Appendix B (Module Descriptions) are omitted.

1.0 TITLE

Wangwriter CRT Memory Diagnostic

2.0 REVISION/DATE

Date: November 17, 1981 Software Release: 1110 Documentation Release: 9110

3.0 REFERENCE DOCUMENTS

Documentation for Wangwriter In-System Diagnostic Monitor.

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This diagnostic runs under control of the Wangwriter In-System Diagnostic Monitor.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging
* Manufacturing (production and QC) board test, board repair and burn-in
* Customer engineering repair at the customer site, board repair and burn-in

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

Addressing and data reliability on all of CRT memory except the soft character font.

5.4 Tests in the Program

No.	Name	Hardware Tested
01	Bank Addressing test	Bank address and page select lines
02	Chip Addressing test	Chip address lines
03	Moving Inversion test	Memory data test

The tests are run in the order listed above.

6.0 LOAD PROCEDURE

For description of how to load and start a diagnostic, see the documentation for the Wangwriter diagnostic monitor.

7.0 OPERATING INSTRUCTIONS

Once started this program will run until done or an error is detected. If the user is using module swap as the repair technique, refer to Appendix A for information in locating the failing module. If the user is repairing faulty memory boards, refer to both appendices. Appendix B contains descriptions of the tests.

8.0 REVISION HISTORY

Rev 1110 (5/15/81) -- first release.

9.0 MISCELLANEOUS

None.

APPENDIX A -- Test and Error Information Table

The following information is intended as a guide in interpreting and making use of the error messages.

In cases where it is likely that the fault is a single memory chip enough information is will be supplied to isolate that chip.

All error messages will begin in this form:

(type of error) at (address) and page xx

Received data = xx

Expected data = yy

XOR data = zz

The important details of this message are the address of the fault, and the XOR data. The address will enable the user to determine which bank has the fault; the XOR data indicates which chip within the bank is faulty.

In the case of addressing errors, the failing address line(s) are identified, along with the apparent type of failure (stuck, open or shorted address line(s)). In most cases only the open address lines can be fixed in the field.

Error	Error Type	Suggested Repair Action
01	Data Error	Replace Memory Chip(s) or Board
02	Address (short, open)	Replace Memory Board
03	Chip address (stuck, open)	Replace Memory Chip(s) or Board

APPENDIX B

MODULE DESCRIPTIONS

BANK ADDRESSING TEST — Test # 1

```
Fill memory with zeros
Read each memory location
    ERROR (01) if data not equal zero (data error)          ERROR 01
Set bank pointer to bank 0 ('C000')
Write FF to location zero of bank under test
Read location zero of bank under test
    ERROR (01) if data not equal FF (data error)            ERROR 01
Read location zero of each higher bank
    ERROR (02) if data equal FF (bank address
    lines shorted)                                          ERROR 02
Repeat for all higher banks

Set bank pointer to bank 0 (C000)
Clear first location of higher banks
Write FF to C000
Read first location from each higher bank
    ERROR (03) if data not equal 00                         ERROR 03
```

CHIP ADDRESSING TEST — TEST # 2

```
Fill bank with zeros
Read each location to verify zeros
    ERROR (01) if not zero                                  ERROR 01
Set address pointer to location 1 of bank
Write FF to location
Read higher addresses with chip address lines asserted
    ERROR (02) if not 00 (chip address line(s)
    short or open)                                          ERROR 02
Repeat to test with each chip address line asserted

Fill bank with zeros
Read each location to verify zeros
    ERROR (01) not zero                                     ERROR 01
Set address pointer to location 00 of bank
Write FF to location
Read higher addresses with chip address lines asserted
    ERROR (03) if not zero (chip address line(s) stuck)     ERROR 03
```

MINI WP MOVING INVERSION TEST TEST #3

```
Set initial increment to 1
Set starting address of bank
Set initial OFFSET to 0
Flood CRT RAM with 00 pattern and verify data in each
location
    ERROR(01) if wrong data                          ERROR 01

Set ADDRESS pointer to begining of bank
Add offset to address
    Read pattern
        ERROR(01) if wrong data                      ERROR 01
    Write new PATTERN
    Read new PATTERN
        ERROR(01) if wrong data                      ERROR 01
    Add increment to address
    Loop until out of range

Add 1 to offset
All done if OFFSET = INCREMENT
Loop for all offsets

Set OFFSET to 0
Get next PATTERN from table
Loop for all patterns

Set ADDRESS pointer to end of bank
Subtract offset from address
    Read pattern
        ERROR(01) if wrong data                      ERROR 01
    Write new PATTERN
    Read new PATTERN
        ERROR(01) if wrong data                      ERROR 01
    Subtract increment from address
    Loop until out of range

Add 1 to offset
All done if OFFSET = INCREMENT
Loop for all offsets

Set OFFSET to 0
Get new PATTERN from table
Loop for all patterns

Set new INCREMENT
Loop for all INCREMENT values
```

1.0 TITLE

Wangwriter Main Memory Diagnostic

2.0 REVISION/DATE

Date: November 17, 1981
Documentation Release: 9192
Software Release: 1192

3.0 REFERENCE DOCUMENTS

Documentation for Wangwriter In-System Diagnostic Monitor.

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This diagnostic runs under control of the Wangwriter In-System Diagnostic Monitor.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging.
* Manufacturing (production and QC) board test, board repair and burn-in.
* Customer Engineering repair at the customer site, board repair and burn-in.

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

Addressing and data reliability on all of main memory except the first bank of 16K.

5.4 Tests in the Program

No.	Name	Hardware Tested
01	CPM Addressing test	Special addressing feature for CPM
02	Bank Addressing test	Bank address and page select lines
03	Chip Addressing test	Chip address lines
04	Moving Inversion test	Memory data test

The tests are run in the order listed above.

6.0 LOAD PROCEDURE

For description of how to load and start a diagnostic, see the documentation for the Wangwriter Diagnostic Monitor.

7.0 OPERATING INSTRUCTIONS

Once started this program will run until done or an error is detected. If the user is using module swap as the repair technique, refer to Appendix A for information in locating the failing module. If the user is repairing faulty memory boards, refer to both appendices. Appendix B contains descriptions of the tests.

8.0 REVISION HISTORY

Rev 1192 (11/17/81) -- first release.

9.0 MISCELLANEOUS

None.

APPENDIX A

TEST AND ERROR INFORMATION TABLE

The following information is intended as a guide in interpreting and making use of the error messages.

In cases where it is likely that the fault is a single memory chip, enough information is will be supplied to isolate that chip.

All error messages will begin in this form:

(type of error) at (address) and page xx

Received data = xx

Expected data = yy

XOR data = zz

The important details of this message are the address of the fault, the page number, and the XOR data. The address will enable the user to determine which bank of chips contains the fault (see table below), the page number indicates one of the banks that reside at location 8000 (hex), and the XOR data isolates the failing bit(s).

Bank #	Page #	Address
0	N/A	0000-3FFF
1	N/A	4000-7FFF
2	0	8000-BFFF
2	1	8000-BFFF
2	2	8000-BFFF
2	3	8000-BFFF

In the case of addressing errors, the failing address line(s) are identified, along with the apparent type of failure (stuck, open or shorted address line(s)). In most cases, only the open address lines can be fixed in the field.

Error	Error Type	Suggested Repair Action
01	Data Error	Replace Memory Chip(s) or Board
02	Data Error	Replace Memory Chip(s) or Board
03	Bank address (short, open)	Replace Memory Board
04	Paged Bank address (short, open)	Replace Memory Board
05	Chip address (short, open)	Replace Memory Chip(s) or Board
06	Chip address (stuck)	Replace Memory Board
07	Unable to read PROM	Replace memory board
08	CPM feature not recognized	Replace or update memory board
09	CPM feature not working	Replace memory board

APPENDIX B

MODULE DESCRIPTIONS

CPM ADDRESSING FEATURE TEST — TEST # 1

```
Select RAM bank 2/3
Clear and verify location X'8000'
Select PROM and RAM bank 2/0
Write/Verify 00 to location X'8000'
Verify location 00 of PROM
Select RAM bank 2/4
Check for data = 00
        Set flag 1 if not zero
Read location X'C000'
        Set flag 2 if data is X'C3'

Both flags set means ADDRESSING FEATURE NOT PRESENT
One flag set means ADDRESSING FEATURE NOT FUNCTIONING PROPERLY
No flags set means ADDRESSING FEATURE FUNCTIONING PROPERLY
```

BANK ADDRESSING TEST — TEST # 2

```
Write and Verify 00 to first loacation of each bank
        ERROR (01) if data not equal zero (data error)          ERROR 01
    Set bank pointer to bank 0 ('0000')
    Write FF to location zero of bank under test
    Read location zero of bank under test
    ERROR (01) if data not equal FF (data error)                ERROR 01
    Read location zero of each higher bank
    If bank under test is not at address  x'8000' then
        ERROR (03) if data equal FF (bank address lines
        shorted)                                                ERROR 03

If bank under test is at address x'8000' then
    ERROR (04) if data equal FF (page select lines shorted)     ERROR 04
    Repeat for all higher banks
Repeat for bank zero
```

CHIP ADDRESSING TEST TEST # 3

Fill memory with zeros
Read each location to verify zeros
ERROR (01) if not zero ERROR 01
Set address pointer to location 1 of bank
Write FF to location
Read higher addresses with chip address lines asserted
bank address + (2, 4, 8, 10, 20, 40, 80, 100, 200, 400, 800, 1000, 2000)
ERROR (05) if not 00 (chip address line(s) short or open) ERROR 05
Repeat to test with each chip address line asserted

Fill accessed bank locations with zero
Read each location to verify zeros
ERROR (01) not zero ERROR 01
Set address pointer to location 00 of bank
Write FF to location
Read higher addresses with chip address lines asserted
bank address + (1, 2, 4, 8, 10, 20, 40, 80, 100, 200, 400, 800, 1000, 2000)
ERROR (06) if not zero (chip address line(s) stuck) ERROR 06
Repeat for all banks

MINI WP MOVING INVERSION TEST TEST # 4

```
Set initial increment to 1
Set starting address of bank
Set initial paged memory for out (00)
Set initial OFFSET to 0
Flood bank with AA pattern and verify data in each location
    ERROR (01) if wrong data                                  ERROR 01

Set ADDRESS pointer to beginning of bank
Add offset to address
    Read pattern
        ERROR (02) if wrong data                              ERROR 02
    Write new PATTERN
    Read new PATTERN
        ERROR (01) if wrong data                              ERROR 01
    Add increment to address
    Loop until out of range

Add 1 to offset
All done if OFFSET = INCREMENT
Loop for all offsets

Set OFFSET to 0
Complement PATTERN
Loop if PATTERN = 55

Set ADDRESS pointer to end of bank
Subtract offset from address
    Read pattern
        ERROR (02) if wrong data                              ERROR 02
    Write new PATTERN
    Read new PATTERN
        ERROR (01) if wrong data                              ERROR 01
    Subtract increment from address
    Loop until out of range

Add 1 to offset
All done if OFFSET = INCREMENT
Loop for all offsets

Set OFFSET to 0
Complement PATTERN
Loop if PATTERN = 55

Set new INCREMENT
Loop for all INCREMENT values
```

1.0 TITLE

Wangwriter Display & Keyboard Test

2.0 REVISION/DATE

Date: November 17, 1981
Software Release: 613A
Documentation Release: 913A

3.0 REFERENCE DOCUMENTS

Documentation for Wangwriter In-System Diagnostic Monitor

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This diagnostic runs under control of the Wangwriter In-System Diagnostic Monitor.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging
* Manufacturing (production and QC) board test, board repair and burn-in
* Customer engineering repair at the customer site, board repair and burn-in

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

7775 board -- display attribute logic, soft character memory
keyboard -- key contacts, keyboard logic and UART
7777 board -- support logic for CRT and keyboard
CRT alignment circuitry

6.0 LOAD PROCEDURES

For description of how to load and start a diagnostic, see documentation for the diagnostic monitor.

7.0 OPERATING INSTRUCTIONS

7.1 Display Attributes

The purpose of the test is to allow the operator to detect faults in the logic on the 7775 board that generates display attributes and that addresses the soft character generator memory. Any error detected in this test probably indicates a failure in the 7775 board; although the 7777 board may also be responsible.

Upon selection of this test, the UUT screen will be outlined and will display lines of text. Each line should display one attribute affecting one character at a time from the left of the screen to the right. Check that each attribute is visible on the screen.

Each line of display, except the last two, is meant to show the effect of turning on exactly one attribute. The next to last line, labeled 'SCRIPT COMPARISON' sequentially invokes subscript, then no attribute, then superscript, then double superscript in order to produce the effect of a smoothly rising line of characters. The line is then moved smoothly down so that the overall effect is a wavy line of degree symbols.

The last line, labeled 'SOFT CHAR GENERATOR' shows various characters generated by the soft character memory. The pattern line from left to right should consist of:

* 1 rectangle
* 1 blank
* A sequence of 8 descending horizontal lines, the first should be at the top of the character space; the last at the bottom.
* 1 blank
* A sequence of 8 ascending horizontal lines, starting at the bottom and going up.
* 1 blank
* 1 rectangle
* A sequence of 7 vertical lines, the first located at the left edge of the character space; the last at the right edge.
* 1 blank
* A sequence of 8 ascending horizontal lines
* 1 blank
* 4 rectangles

Note that a character space on the CRT, consists of a dot matrix that is 8 dots high and 7 dots wide. The character space is thus 56 dots, each of which may be on or off, independently. Selectively turning these dots on or off will create a display character or symbol.

The rectangle mentioned above is a character space with all dots on. The blank is a space with all dots off. A horizontal line is a space with all 7 dots in one row turned on. A vertical line is a space with all 8 dots in one column turned on.

Observe the last line of display. A fault exists if any dots are on when they should be off or vice versa. It is nearly impossible to determine whether the vertical lines are correctly positioned. This determination is not necessary. The only important visual effect to check for is dots appearing where they should not.

To make any adjustments to the controls located under the screen, depress the shift lock. This will cause any keystrokes, except for CANCEL, to be disregarded. Shift-CANCEL or any un-shifted keystroke will cause the UUT to enter the keyboard test.

7.2 Keyboard Test

The purpose of the test is to allow the operator to detect faults in the keyboard assembly and the logic on the 7777 board that handles keystrokes. Any error detected in this test probably indicates a failure in the keyboard assembly; although the 7777 board may also be responsible.

This test is entered by depressing shift-CANCEL or any unshifted key during the display attributes test. The screen should display a graphic representation of the Wangwriter keyboard.

The highlighted key is the one that is to be depressed. The key in the upper left hand corner is first to be depressed and consequently it is the first to be highlighted. After depressing this key twice, the key to its immediate right will be highlighted. The route through the keyboard was chosen because it is consistent with all other keyboard tests that existed at the time this program was written.

If an incorrect keystroke is detected, the key that corresponds to the incorrect keycode will be flahsed and the "expected", "received", and "xor" data will be displayed in hex on the UUT. If the incorrect keycode does not correspond to an existing key, no key will be flashed; however, the data line will be displayed on the UUT.

7.3 Soft Character Generator Memory Test

The purpose of the test is to allow the operator to detect stuck bits in the soft character generator memory. Any fault detected in this test probably indicates a failure in the 7775 board; although the 7777 board may also be responsible.

This test is entered by finishing the keyboard test or by depressing shift-CANCEL during the keyboard test.

The screen should display eight groups of fifteen character spaces. Each group is labeled so that the memory addresses of the character spaces can be determined.

Until a keystroke is detected, the portion of memory that is dedicated to the Soft Character Generator (ie. D800 through DFFF) will atlernately be written with 'FF's and '00's. The visual effect should be a set of 120 rectangles alternating every 5 seconds with a set of 120 blanks. When the rectangles are being displayed, check that there are no missing dots in any of the rectangles. When the blanks are being displayed, check that there are no dots displayed in any of the blanks.

Users who are interested only in testing the unit can terminate the test at this point by keying shift-CANCEL. Those who wish to toggle individual bits (dots) in the soft character generator memory for purposes of isolating faults to a chip or of generating customized fonts should depress any key other than shift-CANCEL. It is best to depress the key while the rectangles are on rather than when the blanks are on display.

Depressing a key will cause the display to freeze and the upper leftmost rectangle will be double superscripted to indicate that it is the one currently selected for modification. Different rectangles can be selected for modification by use of the North, East, South, and West cursor control arrow keys. The currently selected rectangle is always elevated above its neighbors.

Each rectangle consists of 8 lines, each containing 7 dots. Only one line of a rectangle is selected at a time. Different lines can be selected for modification by use of the space bar. Each time the space bar is depressed, the next lower line is selected. If the bottom line is currently selected, the top line becomes selected. No indication is given as to which line is selected. The only way to tell is for the operator to toggle a bit and visually determine which line was affected.

Bits can be toggled in the selected line of the selected rectangle by using the leftmost 8 command keys at the top of the keyboard. Depressing the INDENT key causes bit 7 (the leftmost of the line) to be complemented. Bits 6 through 0 are complemented by depressing PAGE through STOP, respectively.

No visual indication is made if bit 0 is complemented. This is because the hardware does not display this bit. The character lines are only 7 bits wide.

To terminate this test, depress shift-CANCEL.

7.4 CRT Alignment Pattern

The purpose of this display pattern is to facilitate alignment of the CRT. The display is self-explanatory. Adjust each attribute so that it meets the specification given on the screen.

This display pattern is entered by depressing SHIFT-CANCEL during the Soft Character Generator Memory Test.

To terminate the pattern, depress SHIFT-CANCEL.

8.0 REVISION HISTORY

Revision 613A -- first release

9.0 MISCELLANEOUS

This program relies on the operator to detect errors and does not itself call out errors. For this reason, Appendix A (test and error information table) and Appendix B (Module Descriptions) are omitted.

1.0 TITLE

Wangwriter Printer Test

2.0 REVISION/DATE

Date: November 17, 1981
Documentation Release: 9175
Software Release: 1175

3.0 REFERENCE DOCUMENTS

Documentation for the Wangwriter In-System Diagnostic Monitor Package.

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This program runs under the control of the Wangwriter In-System Diagnostic Monitor. The monitor and all of its programs are supplied together on a single diskette. No other software is required.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging
* Manufacturing (production and QC) board test, board repair and burn-in

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

All printer print functions are tested to a limited degree. Items such as carriage servo motor, platen servo motor, ribbon motors, daisy motor, carriage position, left margin sensor position, and daisy home sensor position are checked.

5.4 Functional Description

The printer diagnostic functions are:

Test all operation functions both electrical and mechanical.
Augment standard trouble shooting procedures.
Permit operator to make necessary mechanical alignments.

Through the use of the Monitor, features such as: halt testing, loop on test routine, and count loops, can be implemented. For further information on how to operate the monitor, refer to the documentation for the Wangwriter In-System Diagnostic Monitor Package. The TDC workstation will be utilized to display all messages, and input controls necessary to operate this diagnostic.

Printer Menu Selection:

There are three menus required to operate this printer diagnostic. If the operator has selected a mode of operation called BURNIN, there is no menu operation required. The program defaults to the following conditions and performs the burnin program (described at a later point in this documentation).

Master Menu:

The master menu supplies five different options. These options are as follows:

1) Test Selection
   (0=ALL) Runs all print and electrical tests.
   (1=Mechanical) select mechanical and alignment tests.
   (2=Print) select print sample tests.
   (3=BurnIn) Limited non-interrupted burnin tests.

2) Type Test
   (0=Conf.) Performs limited (Short) printer tests.
   (1=QA Acceptance) Performs in-depth printer tests.

3) Pitch Selection
   (0=10) Perform the printer test in 10 pitch
   10 Characters per inch.
   75 characters per line in Portrait Mode
   100 characters per line in Landscape Mode
   (1=12) Performs the printer test in 12 pitch
   12 characters per inch.
   90 characters per line in Portrait Mode
   120 characters per line in Landscape Mode
   (2=15) Performs the printer test in 15 pitch
   15 characters per inch.
   112 characters per line in Portrait Mode
   150 characters per line in Landscape Mode

4) Page Type
(0=Portrait) Performs test with standard 8 1/2" X 11 1/2" sheet inserted in upright position
(1=Landscape) Performs test with standard 8 1/2" X 11 1/2" sheet inserted in sideways position

5) Paper Form
(0 = Single Sheet) Test performed as if only a single sheet of paper was to be installed, at the end of each 55 or so lines of text were printed.
(1 = Cont. Form) Test performed as if it was using continuous feed type paper.

Test Selection Menu #1

The operator has the ability to run all print tests from the first to the last test or the option to select any single test that they require to be performed. These options are as follows:

0 = All
1 = Mechanical
2 = Print
3 = BurnIn

The following is a list of the tests that will appear on the Mechanical Menu.

1. Ribbon Advance
2. Hammer Impact
3. Mechanical Alignment
4. Ribbon Adjust
5. Daisy H-Sensor
6. LM Sensor
7. Paper Ejection
8. BurnIn
9. Print Menu

The following is a list of the tests that will appear on the Print Test Menu:

1. Data Buffer
2. Spiral Pattern
3. Underscore
4. Overstrike
5. Character Set
6. Line Buffer
7. Worst Case
8. Paper Tension
9. Horizontal Motion
10. Vertical Motion
11. Std Page
12. Mechanical Menu

If the operator selects ALL, then the program will start by performing the Data Buffer Test then continue through all the printing tests.

If the operator selects an individual MECHANICAL test, the test selected will run and only the test selected.

If the operator selects an individual PRINTING test, the test selected will run and then each test after that test to the end.

When the Operator selects Burnin. The program automatically defaults to the Confidence Test, in 10 pitch with a single strike ribbon. The test is semiautomatic at the very beginning. It gives the operator the opportunity to load paper into the machine. A special printer test is performed during the burnin cycle. The sequence of operation is as follows:

The printer will print one line of text. It will indicate the number of times the printer burnin program has been performed from the time it was selected. (All printer functions are normal at this time).
Then the hammer force is eliminated, ribbon advance is eliminated, and vertical paper motion is also eliminated.
The Carriage Motion Test is performed (no printing).
The Worst Case Pattern Test is performed (no printing).
The Paper Tension Test is performed (no printing).
The program then return to the Monitor and other Wangwriter Diagnostics are performed.

TYPE OF TEST Menu #1

The operator has the ability to run the short form (Confidence Test) or to run the long form of the test (Q.A. Quality Assurance Test). These options are as follows:

0 = Confidence Test
1 = Q.A. Quality Assurance Test

Pitch Selection Menu #1

The diagnostic is normally performed with the 10 pitch selected, but the operator has the ability to perform it in either 10, 12, or 15 pitch.

0 = 10 Pitch
1 = 12 Pitch
2 = 15 Pitch

PAGE TYPE Menu #1

The operator has the ability to run the diagnostic in either the portrait or landscape positions. The test will default to Portrait Mode.

(0 = Portrait)
(1 = Landscape)

PAPER FORM

This diagnostic has the ability to be performed with two different types of printer paper. Continuous Form which is connected end to end with a perforated line separating the individual pages. As with a single sheet of paper. NOTE: If an individual test is selected, the program automatically selects the continuous form option.

0 = Cont. Form
1 = Single Sheet

5.5 Functional Operations of Mechanical Tests

5.5.1 Test 1 - Ribbon Advance Test

INPUT: None.

OUTPUT: There are seven different patterns impressed onto the ribbon.

PURPOSE: This test is designed to verify that the electrical and mechanical requirements for ribbon motion and control is operating correctly.

DESCRIPTION: The program creates seven different patterns onto the ribbon. Each pattern has a given space between each mark on the ribbon. A period is used to indicate that a new distance has been selected. There are seven different amounts of travel that the printer can program. See appendix X for an example of the impression that should be seen on the ribbon. (The print sample is used only as an example at this time and should not be used as a comparison with the ribbon sample).

5.5.2 Test 2 - Hammer Impact Test

INPUT: None.

OUTPUT: Five lines of code. The first line of code being the lighter of the five lines. The last line of code being the darker of the four lines.

PURPOSE: Verify the electrical, mechanical and associated software is functioning properly.

DESCRIPTION: The operator is advised to position the print control to the lighter mode or (down). The program will then print one line of data with the lightest value in the hammer. Then it will print a second line with the 2/5's higher value. Then it will print the third line with 3/5's power and the fourth line with 4/5's power and on the last line it will print with the maximum power available.

5.5.3 Test 3 - Mechanical Head Alignment

INPUT: None.

OUTPUT: Four lines of code. The first character of each line is the character "T". The first line of code is altered between "Hj". Then the paper is advanced a half line down and the line is totally underscored. The third line is advanced another half line and the line is totally underscored again. The fourth line is composed of a full line of the cent sign (The small character c with an overstrike slash /).

PURPOSE: Used as an aid in the initial manufacturing states of the Wangwriter.

DESCRIPTION: When selected via the Mechanical Menu and the printer is selected, the program runs continuously until the operator deselects the printer. The operator has the ability to loop on routine and stay in this program area.

5.5.4 Test 4 - Ribbon Adjust Test

INPUT: None.
OUTPUT: One line of characters (The letter E).
PURPOSE: To verify that all ribbon carriage assembly is properly positioned to use the centermost portion of the ribbon.
DESCRIPTION: The program prints one line of characters then deselects the printer and tells the operator to remove the ribbon from the machine and observe where the character impression is placed on the ribbon. He is instructed to put it back into the machine, make any necessary adjustments and if necessary rerun the test again. To rerun the test the operator simply selects the printer. To Exit the test the operator depresses the Line Feed Down button then select.

5.5.5 Test 5 - Daisy H-Sensor Test

INPUT: Keyboard commands
OUTPUT: Correct alignment of the Daisy Home Sensor.
PURPOSE: To check and adjust as necessary the Daisy Home sensor.
DESCRIPTION: Operator action is required in this test. The program places the carriage mechanism in the center of the carriage. It then homes the daisy to the little (o) petal. Then it deselects itself. The operator can make any necessary adjustments. He can rerun the daisy home test by simply depressing the select button on the printer console. He must depress cancel and reselect the printer test to get out of this mode of test.

Special Features - There are special features associated with this test and the left margin sensor adjust test. There are three modes of operation to this test.

1) Step Mode - the program will be performed in three steps. A single 0 on the lower left portion of the screen indicates that the item under test was moved from the sensor and then walked back onto where it first detected the sensor, and that's where it is waiting for the next command. The next phase of this step test is the individual 1/2 stepping of the stepping motor across the sensors. This is indicated by two 0's in the lower left portion of the screen. When three zero's appear in the lower left portion of the screen, the test is completed. Now the test will loop back to the beginning. To advance to the next phase of operation simply depress EXECUTE on the key board. To access this mode of operation depress the PAGE button on the keyboard. Page can be depressed at any time during this mode of operation.

2) Normal Mode - When selected via the menu, the program will perform the entire test and stop by deselecting the printer at the end of the test. To perform the test again, select the printer again.

SIMULATED DAISY HOME SCREEN DISPLAY

4 3 2 1 8 7 6 5 4 3 2 1 H 7 6 5 4 3 2 1 8 7 6 5 4 (Fixed Display)

0 0 0 0 0 0 0 0 0 0 A 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 (Adjustable Display)

A (Fixed Display)

Align the letter (A's)

Motor Table Value is (XX)

N O T E :

The Adjustable Display above will position itself according to the mechanical adjust was performed. The operator need only make the mechanical adjustment and reselect the printer and this line will be redisplayed.

The motor table value displayed (XX) was the value sent to the motor to position it at it present location.

3) Full automatic - This will run the NORMAL MODE of operation explained in item 2 above. In this mode of operation it will not stop after the test is over, it will just run continuously. Once in this mode of operation the only way out is to depress cancel on the keyboard. To enter this mode you must be in the step mode of operation. Depress PAGE on the keyboard and wait for the single zero to appear in the lower left portion of the screen. NOTE: it may be necessary to depress EXECUTE and step through some step phases until the single zero is displayed. When the single zero is displayed depress any numeric pad on the keyboard and then EXECUTE. This will start the full automatic mode of operation.

NOTE: THE COVER MUST BE CLOSED DURING THE MENU SELECTION. The cover status will be ignored after the actual program is entered.

5.5.6 Test 6 - Left Margin Sensor Adjust Test

INPUT: Keyboard commands.

OUTPUT: Correct alignment of the left margin sensor.

PURPOSE: To check and adjust as necessary the left margin sensor.

DESCRIPTION: Operator action is required in this test. The program moves the carriage from left to right, then turns around and moves from right to left until if finds the sensor. At that point, it starts moving into the sensor by 12 half steps from right to left. It then changes direction and moves the carriage 24 half step from left to right, during this time displaying the condition of the sensor. If a zero is displayed, then the sensor was not blocked. If a 1 appears, then the sensor was blocked. It then continues moving another 8 half steps away from the sensor. At this point, the program will deselect the printer. Please read the instructions for the Daisy Home Adjust. This program has the same functions.

SIMULATED LEFT MARGIN SCREEN DISPLAY

```
4 3 2 1 8 7 6 5 4 3 2 1 H 7 6 5 4 3 2 1 8 7 6 5 4    (Fixed Display)
1 1 1 1 1 1 1 1 1 1 1 1 1 B 0 0 0 0 0 0 0 0 0 0 0    (Adjustable Display)
                          B                          (Fixed Display)
```

Align the letter (B's)

Motor Table Value is (XX)

See the note under Daisy Home Adjust for additional display information on the this test.

5.5.7 Test 7 - Paper Eject

INPUT: Paper

OUTPUT: Paper ejection from the printer into the collection hopper under the printer housing.

PURPOSE: To verify all the functions, that are required to eject the paper from the platen, after the paper out sensor is detected.

DESCRIPTION: The printer is deselected under program control. The terminal message informs the operator to insert a piece of 8 1/2" by 11" sheet of paper into the printer. (The operator has five different switches at their disposal to load paper into the printer.) It also requests the operator to select the printer to perform the rest of the Ejection Test. Once the printer is reselected the test is under program control again and advances the paper down to a point where the bottom edge of the paper is released by the platen pinch rollers. At that time, the program reverses the direction of the paper motion and moves the paper down until it looses contact with the paper tension rollers on the platen and falls into the paper bin under the printer.

NOTE This test may be difficult to perform under the following conditions:

1) The paper thickness (paper being too thick).
2) The mechanical adjustment between the platen and the ribbon quide is not adjusted properly.

5.5.8 Test 8 - BurnIn Test

INPUT: None.

OUTPUT: One line of code is printed. This line of code will display the number of times that the printer test was performed.

PURPOSE: This test is designed to allow the printer to be exercised without operator intervention so that the machine can be repeatedly tested for approximately 20 hours.

DESCRIPTION: The program automatically defaults to the Confidence Test, in 10 pitch with a single strike ribbon. The test is semi-automatic at the very beginning. It gives the operator the opportunity to load paper into the machine. A special printer test is performed during the burnin cycle. The sequence of operation is as follows:

The printer will print one line of text. It will indicate the number of times the printer burnin program has been performed from the time it was selected. (All printer functions are normal at this time).

Then the hammer force is eliminated, ribbon advance is eliminated, and vertical paper motion is also eliminated.

The Carriage Motion Test is performed (no printing).
The Worstcase Pattern Test is performed (no printing).
The Paper Tension Test is performed (no printing).
The program then return to the monitor and other Wangwriter Diagnostics are performed.

5.5.9 Test 9 - Select Print Type Menu

INPUT: None.

OUTPUT: None.

PURPOSE: Inter-menu changing without cancelling out of the program if a new option is required.

DESCRIPTION: None.

5.6 Functional Operations of Print Type Tests

5.6.1 Test 1 - Data Buffer Test

INPUT: None.

OUTPUT: Several lines of characters are printed under each of the seven subheadings depicting the various data bit patterns possible and are required to be visually inspected by the operator.

PURPOSE: To verify that all major bit patterns print out the correct character.

DESCRIPTION: The test will print five lines of each of the seven bit patterns for the long version, or one line of each pattern for the short version and proceed to the next test.

5.6.2 Test 2 - Spiral Pattern Test

INPUT: None.

OUTPUT: Several lines of a complete set of ninety-six alphanumeric characters in a wraparound spiral pattern will be printed on the printer and are required to be visually inspected by the operator.

PURPOSE: To check to be sure there is no interaction between the internal print buffer, the data bus, and the printed character ASCII code.

DESCRIPTION: The test will print five lines if short test or fifty lines if long test of the pattern, then proceed to the next test.

5.6.3 Test 3 - UnderScore Test

INPUT: None.

OUTPUT: One line of single underscore, then one line of double underscore for short test, or five lines of single underscore, then five lines of double underscore for long test of the underscored character.

PURPOSE: To verify that the underscore & the double underscore characters are functioning correctly, and that the character overlaps and makes a continuous line across the page.

DESCRIPTION: Instructions for this test are displayed on the TDC console. The program prints out first, the single underscored character and then the double underscored character.

5.6.4 Test 4 - Overstrike Data

INPUT: None.

OUTPUT: Several lines of various types of overstricken character text to be examined by the operator.

PURPOSE: To verify the printers mechanics for the ability to overstrike in all modes correctly.

DESCRIPTION: In the long version the printer prints out lines of '=', 'c', 'O', and 'S' overstricken by either an '/' as in the first three cases or a vertical line as in the last case. In 'S' and vertical line overstrike will be performed for each mode. In the short version only the 'S' overstriken by a vertical bar when printed.

5.6.5 Test 5 - Character Set Test (Upper Case/Lower Case)

INPUT: None.

OUTPUT: There are six lines of text printed. Each line is displaying 16 of the available characters on the daisy print wheel.

CHARACTER 40 through 4F @ A B C D Etc,

40 = @
41 = A
42 = B etc.

PURPOSE: To verify the ability to print all the characters on the printer daisy wheel.

DESCRIPTION: Since there are 96 petals on the daisy print wheel, we are attempting to display all of the characters and the associated hex codes required by the hardware, to print this character on paper.

5.6.6 Test 6 - Line Buffer Test

INPUT: None.

OUTPUT: Two lines for short test and five for long test of text that must be verified by the operator.

PURPOSE: This test verifies the line length of the printer, and prints the number of characters on the line for easy identification as to the line length.

DESCRIPTION: The diagnostic prints one line for the short version of the test and five lines for the QA version. The text line consists of printing a series of 123456789-'s repeatedly for the full length of the paper. The operator is required to count the number of dashes and multiply this value by 10, and then add the highest number after the last dash on the right side of the printer to determine the printer width, and this number should agree with the printer width specification specified in the hardware manual. An example would be as follows: The operator has counted 13 dashes and multiplied this number by 10, and then has added 2 to this number for a value of 132 as shown in the example below:

123456789-123456789-12.
* is the last dash and the highest number after that is 2, so 13 X 10 = 130 + 2 for 132 character line.

5.6.7 Test 7 - Worst Case Pattern Test

INPUT: None.

OUTPUT: One line for short test and five lines for long test of data hopefully worst case character selected print code.

PURPOSE: This test is designed to verify and check the speed of recovery for the printing of characters on the opposite side of the print wheel, and other places along the wheel deemed to be a time factor. This test is designed at this time for a (Tile 10/12 daisy print wheel). The next version of this diagnostic will ask for the wheel and will print the Worst Case Test for that particular wheel, but it is not available for this diagnostic at this time.

DESCRIPTION: The test prints out what it thinks is a worst case condition for the daisy printer wheel.

5.6.8 Test 8 - Paper Tension Test

INPUT: None.

OUTPUT: Five lines of text. First character on the first line, the second a quarter line down from the first and then the third character down another. This continues until the fifth character is printed then it starts back up again. The following effect is given at a quarter line feed increments.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

PURPOSE: This test is designed to verify that all the paper holding functions are working properly.

DESCRIPTION: The test prints a full line of characters from left to right and then overprints from right to left. But after each character is printed the line is either advanced or backed up by a quarter line feed to give the above effect on the printer.

5.6.9 Test 9 - Carriage Motion Data Test

INPUT: None.

OUTPUT: Ten lines of data for the short test, or twenty-four lines of data for the long test to be verified by the operator with the use of a predetermined template to verify the accuracy of the printer.

PURPOSE: This test is used to verify the accuracy of the stepping motor, the electrical and mechanical hardware and to determine if they are within specifications set for printer hardware. This test requires a template for the verification.

DESCRIPTION: There are for the double bit test, ten degrees of horizontal movement available with the daisy printers. Each degree of movement is designated by a certain data bit. The higher the bit value the larger the movement that will occur. The diagnostic will print a reference vertical bar, then it will move the carriage to the next degree of horizontal movement and print a vertical bar. At that time, it will move the carriage back to the original reference position, print another vertical bar, and move the carriage out even farther to the next degree of movement available. The printer, by combinations of these degree bits together, can thereby move the carriage to any spot on the print line, the second part of the test utilizes this feature in the following manner: The diagnostic will print a reference vertical bar then it will move the carriage to the next degree of horizontal movement and print a vertical bar. At that time, it will move the carriage back to the original reference position, print another vertical bar and move the carriage out even farther to the next degree of movement combined with all the degrees done before, in a cumulative pattern.

5.6.10 Test 10 - Vertical Movement Test

INPUT: None.

OUTPUT: Prints one character of data on the line (the character is a horizontal reference character).

PURPOSE: This test is designed to verify the accuracy of the stepping motor and the associated electrical and mechanical specification are within the printer hardware specifications for vertical paper motion. This test requires a template for this verification.

DESCRIPTION: There are ten degrees of vertical movement available with the Daisy printers. Each degree of movement is designated by a certain data bit. The higher the bit value, the larger the movement that will occur. The diagnostic will print a reference horizontal bar, then it will move the paper to the next degree of vertical movement and print another horizontal bar. At that time, it will move the platen back to the original reference position, print another reference horizontal bar, and move the platen down even farther to the next degree of movement available.

5.6.11 Test 11 - Standard Page

INPUT: None.

OUTPUT: A single page reproduction of a Print Quality Standard developed by the New Products Quality Control Engineering Department.

PURPOSE: This test according to the New Products Engineer is to test the PRINT QUALITY of the printer. This page of text does not adequately test the functionality of all the printer hardware or software and there for should not be used to QA accept this printer before it is released for shipment to the customer.

DESCRIPTION: This test is performed functionally like normal operating software. Refer to the New Products Quality Control Engineering Document "Print Quality Standard for the Model 420 (Mini Word Processor) for a description of this test.

Note. (This test is valid only when performed with a PRESTIGE PICA 10 PITCH DAISY WHEEL part # 96-072 .)

5.6.12 Test 12 - Select Mechanical Menu

INPUT: None.

OUTPUT: None.

PURPOSE: Inter-menu changing without cancelling out of the program if a new option is required.

DESCRIPTION: None.

6.0 LOAD PROCEDURE

For a description of how to load and start a program, see documentation for the diagnostic monitor.

7.0 OPERATING INSTRUCTIONS

This program requires operator intervention and observation from time to time during the execution of the program.

The program has the feature to automatically load paper by placing the paper into the feed holder behind the platen and pulling the paper tension bar lever forward. This feature can be exercised after the selection menus whenever the printer is deselected.

Rev. 1150 contains the following changes,

prints bidirectionally wherever possible.
en modified to include a one-half inch margin
d right borders.
w moves in tabulation mode eliminating the
aces individually.
lows the printer to be tested in both 12 and

been modified to allow the testing to be
e Landscape Mode as well as Portrait Mode.
s the ability, in single sheet feed mode to
f the page and automatically eject the paper
e bin found under the printer section.
attern on the Standard Print Page has been
lphabetical order.
vement Test has an additional underscore line
ltle as an additional reference line.
tion Test is now set up on a separate sheet
event the test scrolling out of the platen.
ished by stepping the page until ejected (see
after this test.
Motion Test has been shortened to 12 lines
in QA mode.
aper feed can now be accomplished at any time
deselected and not just when head is at left
e previous release.
Set Test has been modified to print out a
re in place of the space that corresponds
in ASCII. This was done because the space
rriage motion and not a printwheel character
ASCII hex value for a double underscore.
n menus, invalid selections are now ignored.
n displayed again.

· to detect errors and does not itself
hat it will detect, is when carriage
ol and exceeds its boundary on either
carriage assembly. For this reason
ion Table) is omitted.

APPENDIX B

SAMPLE PRINT OUTS

Page 1	Burnin Sample
Pages 2 to 4	Confidence Version in Portrait Mode
Page 5	Standard Print Quality Page
Pages 6 to 11	Long Version in Portrait Mode
Pages 12 to 15	Confidence Version in Landscape Mode
Pages 16 to end	Long Version in Landscape Mode

WANG WRITER BURNIN TESTING

MWP BI PASS 001 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 002 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 003 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 004 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 005 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 006 ¨"#£%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 007 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 008 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 009 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 010 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 011 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 012 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 013 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 014 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 015 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 016 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 017 ¨"#£%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 018 ¨"#£%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 019 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 020 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 021 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 022 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 023 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 024 ¨"#£%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 025 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 026 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 027 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 028 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 029 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
MWP BI PASS 030 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ

PRINTER DIAGNOSTIC OPTIONS SELECTED Wangwriter PRINTER-1175 (07-06-81)

CONFIDENCE TEST (SHORT VERSION)
PRINTING IN 12 PITCH MODE
WITH A SINGLE-STRIKE RIBBON CARTRIDGE

DATA BUFFER TEST

BITS 1 THROUGH 6 HEX(3F=?)
??

BITS 3,4 & 7 HEX(4C=L)
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

BITS 1,3,5 & 6 HEX(35=5)
555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555

BITS 2,4,5 & 6 HEX(3A=§)
§§

BITS 4,5 & 7 HEX(58=X)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

BITS 2,3 & 6 HEX(26=&)
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

BITS 1,3,5 & 7 HEX(55=U)
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

SPIRAL TEST PATTERN

¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxy
¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxy
"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxy ¨
#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxy ¨"
ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxy ¨"#

UNDERSCORE TEST

OVERSTRIKE TEST

Backward Overstrike
$$

Forward Overstrike
$$

Individual Overstrike
$$

CHARACTER SET TEST - UPPER & LOWER CASE

96 Character set identification

CHARACTERS 20 THROUGH 2F _ ¨ " # ß % & ' ( ) * + , - . /
CHARACTERS 30 THROUGH 3F 0̄ 1 2 3 4 5 6 7 8 9 § : ̦ = ° ?
CHARACTERS 40 THROUGH 4F @ A B C D E F G H I J K L M N O
CHARACTERS 50 THROUGH 5F P Q R S T U V W X Y Z [£] ¨ _
CHARACTERS 60 THROUGH 6F ^ a b c d e f g h i j k l m n ō
CHARACTERS 70 THROUGH 7F p q r s t u v w x y z ; ı | ´ `

LINE BUFFER TEST

123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-

WORST CASE PATTERN

eIeaXaiEiXaXeHe*#**HeH#**#eIeaXaiEiXaXeHe*#**HeH#**#eIeaXaiEiXaXeHe*#**HeH#**#eIeaXaiEiXaXeHe*#**

PAPER TENSION TEST

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

HORIZONTAL MOTION DATA TEST

Double Bit Distances

Cumulative Bit Distances

VERTICAL MOVEMENT TEST

T STANDARD TEST DOCUMENT FOR MINI WORD PROCESSOR (REV.3/10) TT
T T

TEST PARAMETERS

1.Print Wheel: PRESTIGE PICA 10 2.Ribbon VENDER CODE 07 3.System Disk: REL. 2.0 4.Left Margin: 00 5.Pitch: 10 6.Lines/Inch: 6 7.Justification: None

DIAGONAL PATTERN

"#%&'()*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWX
#%&'()*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXY
%&'()*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ
&'()*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"
'()*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#
()*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%
)*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%&
*+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%&'
+,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%&'(
,-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%&'()
-./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%&'()*
./0123456789§=?@abcdefghijklmnopqrstuvwxyz[]ABCDEFGHIJKLMNOPQRSTUVWXYZ"#%&'()*+

T CHARACTER SET T

THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG'S BACK 0123456789 T
the quick brown fox jumped over the lazy dog's back 0123456789

HORIZONTAL DENSITY

H___H___H___H___H___H___H___H___H___H___H___H___H___H___H___H___H___H___H___H
"___"

VERTICAL DENSITY TT

[___[___[___[___[___[___[___[___[___[___[___[___[___[___[___[___[___[___[___[
//

VERTICAL SPACING

[[[
[[[
[[[
[[[

UNDERSCORE/FRACTIONAL LINE FEED

T T
T T

OVERSTRIKE CHARACTERS

¢¢
$$

VERTICAL REGISTRATION

ErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2
Z2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mErZ2mE

T T
TThis is the last line (no.59). Last character (no.79) on this line is "T"..T

PRINTER DIAGNOSTIC OPTIONS SELECTED Wangwriter PRINTER-1175 (07-06-81)

QA TEST (LONG VERSION)
PRINTING IN 10 PITCH MODE
WITH A SINGLE-STRIKE RIBBON CARTRIDGE

DATA BUFFER TEST

BITS 1 THROUGH 6 HEX(3F=?)

BITS 3,4 & 7 HEX(4C=L)

BITS 1,3,5 & 6 HEX(35=5)

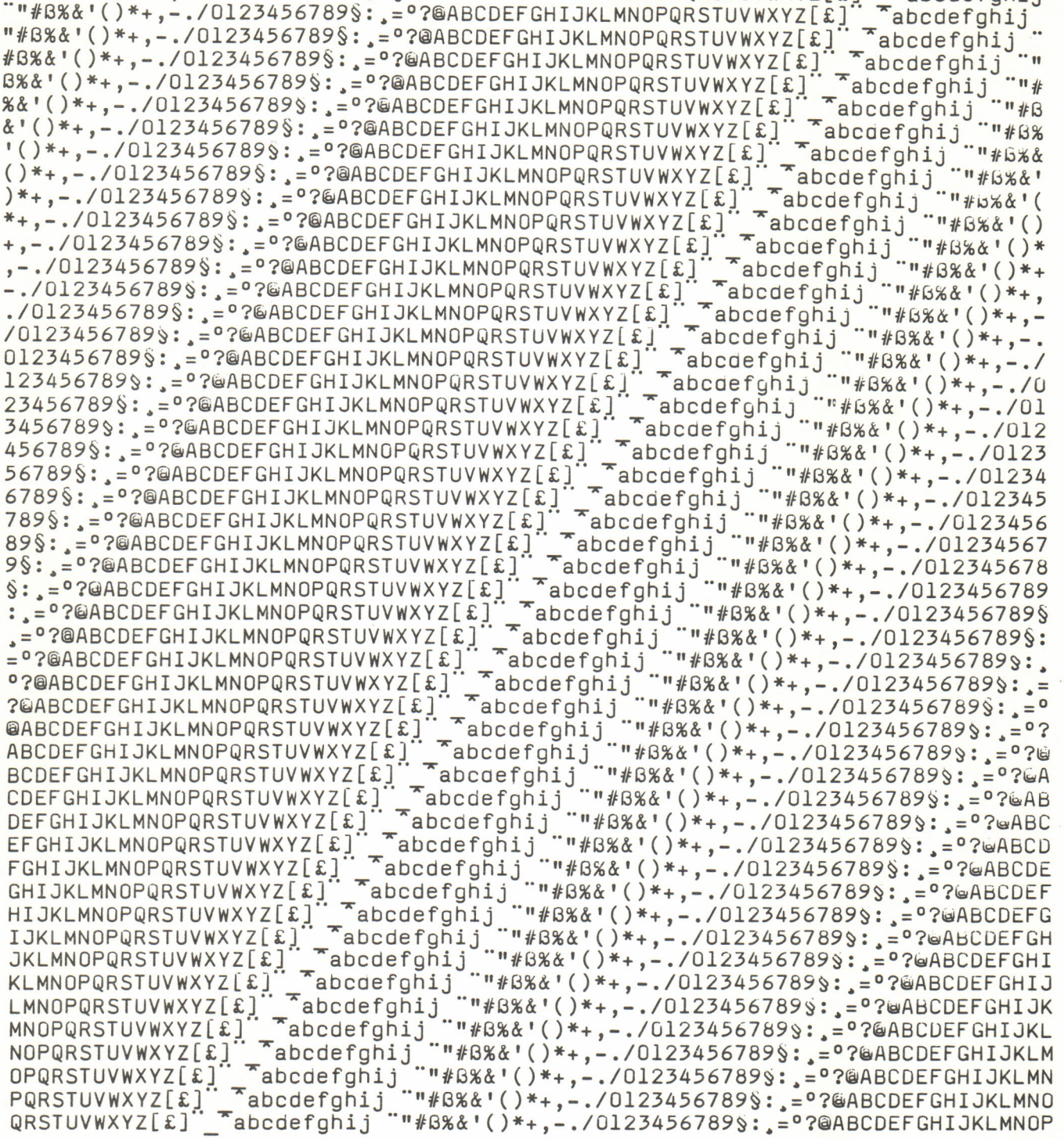
BITS 2,4,5 & 6 HEX(3A=§)

BITS 4,5 & 7 HEX(58=X)

BITS 2,3 & 6 HEX(26=&)

BITS 1,3,5 & 7 HEX(55=U)

SPIRAL TEST PATTERN



UNDERSCORE TEST

_____

_____

_____

OVERSTRIKE TEST

Backward Overstrike

¢¢¢
≠≠≠
ØØØ
$$$

Forward Overstrike

¢¢¢
≠≠≠
ØØØ
$$$

Individual Overstrike

¢¢¢
≠≠≠
ØØØ
$$$

CHARACTER SET TEST - UPPER & LOWER CASE

96 Character set identification

CHARACTERS 20 THROUGH 2F _ ¨ " # ß % & ' ( ) * + , - . /
CHARACTERS 30 THROUGH 3F 0 1 2 3 4 5 6 7 8 9 § : . = ° ?
CHARACTERS 40 THROUGH 4F @ A B C D E F G H I J K L M N O
CHARACTERS 50 THROUGH 5F P Q R S T U V W X Y Z [£] ¨ _
CHARACTERS 60 THROUGH 6F ^ a b c d e f g h i j k l m n o
CHARACTERS 70 THROUGH 7F p q r s t u v w x y z ; ı | ´ `

LINE BUFFER TEST

123456789-123456789-123456789-123456789-123456789-123456789-123456789-12345
123456789-123456789-123456789-123456789-123456789-123456789-123456789-12345
123456789-123456789-123456789-123456789-123456789-123456789-123456789-12345
123456789-123456789-123456789-123456789-123456789-123456789-123456789-12345
123456789-123456789-123456789-123456789-123456789-123456789-123456789-12345

WORST CASE PATTERN

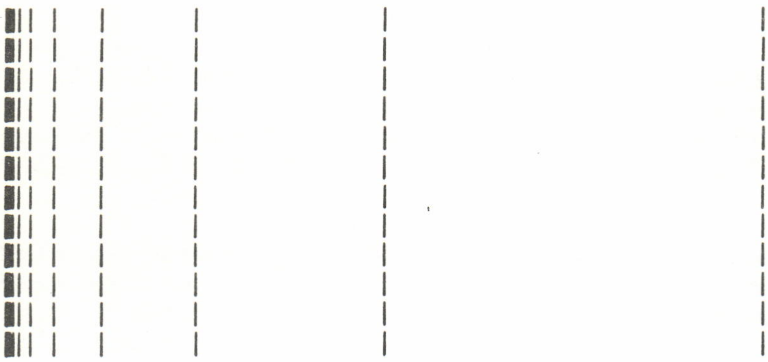
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIe
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIe
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIe
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIe
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIe

PAPER TENSION TEST

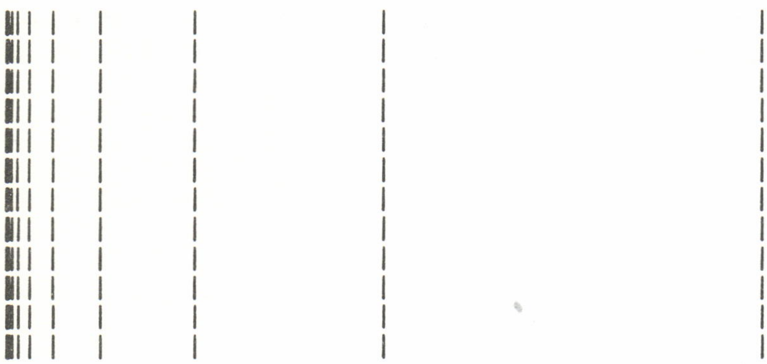
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

HORIZONTAL MOTION DATA TEST

Double Bit Distances



Cumulative Bit Distances



VERTICAL MOVEMENT TEST

PRINTER DIAGNOSTIC OPTIONS SELECTED

Wangwriter PRINTER-1175 (07-06-81)

CONFIDENCE TEST (SHORT VERSION)
PRINTING IN 10 PITCH MODE
WITH A SINGLE-STRIKE RIBBON CARTRIDGE

DATA BUFFER TEST

BITS 1 THROUGH 6 HEX(3F=?)
??

BITS 3,4 & 7 HEX(4C=L)
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

BITS 1,3,5 & 6 HEX(35=5)
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555

BITS 2,4,5 & 6 HEX(3A=§)
§§

BITS 4,5 & 7 HEX(58=X)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

BITS 2,3 & 6 HEX(26=&)
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

BITS 1,3,5 & 7 HEX(55=U)
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU

SPIRAL TEST PATTERN

```
 ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"#
¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"#
"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨
#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"
ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#
```

UNDERSCORE TEST

OVERSTRIKE TEST

Backward Overstrike
$$

Forward Overstrike
$$

Individual Overstrike
$$

CHARACTER SET TEST - UPPER & LOWER CASE

96 Character set identification

CHARACTERS 20 THROUGH 2F	_	¨	"	#	ß	%	&	'	(	)	*	+	,	-	.	/
CHARACTERS 30 THROUGH 3F	0̄	1	2	3	4	5	6	7	8	9	§	:	¸	=	°	?
CHARACTERS 40 THROUGH 4F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
CHARACTERS 50 THROUGH 5F	P	Q	R	S	T	U	V	W	X	Y	Z	[	£	]	¨	_
CHARACTERS 60 THROUGH 6F	^	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
CHARACTERS 70 THROUGH 7F	p	q	r	s	t	u	v	w	x	y	z	;	ı	|	´	`

LINE BUFFER TEST

123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-

WORST CASE PATTERN

eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIea

PAPER TENSION TEST

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

HORIZONTAL MOTION DATA TEST

Double Bit Distances

Cumulative Bit Distances

VERTICAL MOVEMENT TEST

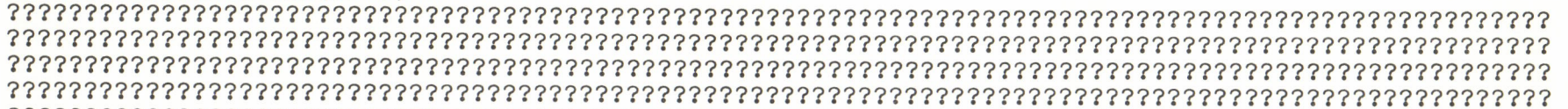
PRINTER DIAGNOSTIC OPTIONS SELECTED

Wangwriter PRINTER-1175 (07-06-81)

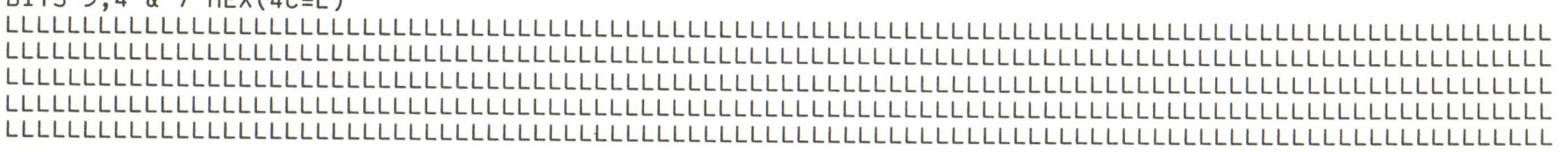
QA TEST (LONG VERSION)
PRINTING IN 10 PITCH MODE
WITH A SINGLE-STRIKE RIBBON CARTRIDGE

DATA BUFFER TEST

BITS 1 THROUGH 6 HEX(3F=?)



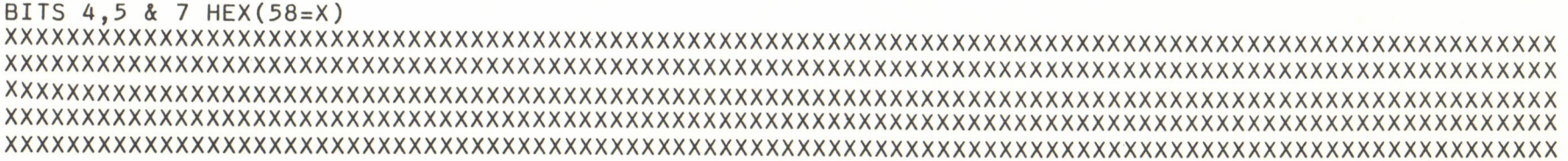
BITS 3,4 & 7 HEX(4C=L)



BITS 1,3,5 & 6 HEX(35=5)

BITS 2,4,5 & 6 HEX(3A=§)

BITS 4,5 & 7 HEX(58=X)



BITS 2,3 & 6 HEX(26=&)

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

BITS 1,3,5 & 7 HEX(55=U)

```
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
```

SPIRAL TEST PATTERN

```
 ¨"#ß%&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#
¨"#ß%&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#
"#ß%&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨
#ß%&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"
ß%&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#
%&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß
&'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%
'()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&
()*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'
)*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'(
*+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()
+,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*
,-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+
-./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,
./0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-
/0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-.
0123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./
123456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./0
23456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./01
3456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./012
456789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./0123
56789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./01234
6789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./012345
789§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./0123456
89§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./01234567
9§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./012345678
§:‸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨_^abcdefghijklmnopqrstuvwxyz;ı|´`  ¨"#  ¨"#ß%&'()*+,-./0123456789
```

```
:¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§
¸=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:
=°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸
°?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=
?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@
BCDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@A
CDEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@AB
DEFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABC
EFGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCD
FGHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDE
GHIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEF
HIJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFG
IJKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGH
JKLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHI
KLMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJ
LMNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJK
MNOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKL
NOPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLM
OPQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMN
PQRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNO
QRSTUVWXYZ[£]¨ _^abcdefghijklmnopqrstuvwxyz;ı|´` ¨"# ¨"#ß%&'()*+,-./0123456789§:¸=°?@ABCDEFGHIJKLMNOP
```

UNDERSCORE TEST

OVERSTRIKE TEST

Backward Overstrike

¢¢¢

≠≠≠

```
ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Forward  Overstrike
¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠
ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Individual Overstrike
¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠≠
ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

CHARACTER SET TEST - UPPER & LOWER CASE

96 Character set identification

```
CHARACTERS 20 THROUGH 2F _ ¨ " # ß % & ' ( ) * + , - . /
CHARACTERS 30 THROUGH 3F Ō 1 2 3 4 5 6 7 8 9 § : ¸ = º ?
CHARACTERS 40 THROUGH 4F @ A B C D E F G H I J K L M N O
CHARACTERS 50 THROUGH 5F P Q R S T U V W X Y Z [ £ ] ¨ _
CHARACTERS 60 THROUGH 6F ^ a b c d e f g h i j k l m n ō
CHARACTERS 70 THROUGH 7F p q r s t u v w x y z ; ı | ´ `
```

LINE BUFFER TEST

```
123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-
123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-
123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-
123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-
123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-123456789-
```

WORST CASE PATTERN

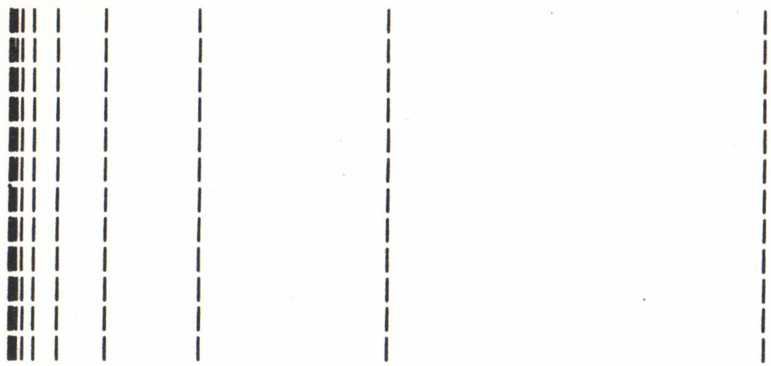
```
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIea
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIea
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIea
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIea
eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIeaXaiEiXaXeHe*#*HeH#*#eIea
```

PAPER TENSION TEST

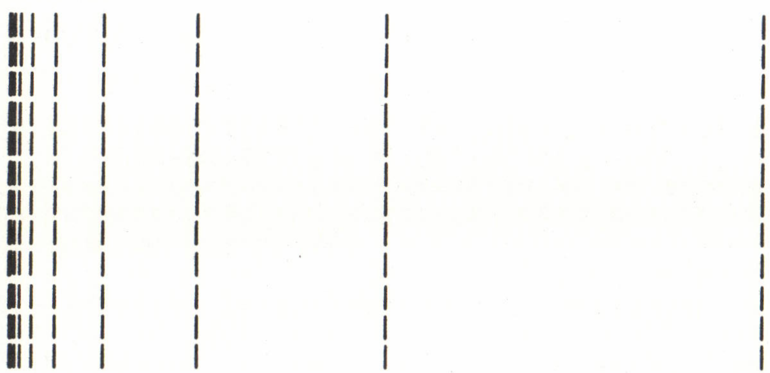
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

HORIZONTAL MOTION DATA TEST

Double Bit Distances



Cumulative Bit Distances



VERTICAL MOVEMENT TEST

APPENDIX X

Ribbon Advance Test

```
. 11111                          Minimum Space
. 1 1 1 1 1                      Minimum + 1 Spaces
. 1  1  1  1  1                  Minimum + 2 Spaces
. 1   1   1   1   1              Minimum + 3 Spaces
. 1    1    1    1    1          Minimum + 4 Spaces
. 1     1     1     1     1      Minimum + 5 Spaces
. 1      1      1      1      1  Minimum + 6 Spaces


.11111.1 1 1 1 1.1  1  1  1  1.1   1   1   1   1  - - - Etc
```

APPENDIX X

Ribbon Advance Test

```
. 11111                          Minimum Space
. 1 1 1 1 1                      Minimum + 1 Spaces
. 1  1  1  1  1                  Minimum + 2 Spaces
. 1   1   1   1   1              Minimum + 3 Spaces
. 1    1    1    1    1          Minimum + 4 Spaces
. 1     1     1     1     1      Minimum + 5 Spaces
. 1      1      1      1      1  Minimum + 6 Spaces
```

.11111.1 1 1 1 1.1 1 1 1 1.1 1 1 1 1 - - - Etc

1.0 TITLE

Wangwriter CTC Diagnostic

2.0 REVISION/DATE

Date: November 17, 1981
Software Release: 1110
Documentation Release: 9110

3.0 REFERENCE DOCUMENTS

Documentation for the Wangwriter In System Diagnostic Monitor

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This diagnostic runs under control of the Wangwriter In-System Diagnostic Monitor.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging
* Manufacturing (production and QC) board test, board repair and burn-in
* Customer engineering repair at the customer site, board repair and burn-in

5.2 User Interface

Wangwriter In-System Diagnostic Monitor

5.3 Hardware Tested

CTC Chip and interrupt circuitry on 7777 CPU board. See also section 5.4 below.

5.4 Tests in the Program

No.	Name	Hardware Tested
1	Timer Mode	CTC, CPU interrupt logic
2	Interrupt Priority	CTC interrupt daisy chain, CPU interrupt logic

6.0 LOAD PROCEDURE

For description of how to load and start a diagnostic, see documentation for the Wangwriter In System Diagnostic Monitor.

7.0 OPERATING INSTRUCTIONS

Once started this program will run until done or an error is detected. If the user is using module swap as the repair technique, refer to Appendix A for information in locating the failing module. If the user is repairing faulty memory boards, refer to both appendices. Appendix B contains descriptions of the tests.

8.0 REVISION HISTORY

Rev 1110 (5/15/81) -- first release.

9.0 MISCELLANEOUS

None.

APPENDIX A

TEST AND ERROR INFORMATION TABLE

TEST NO.	TEST NAME	ERROR CODE	FAILING MODULE or DESCRIPTION
			ALL ERRORS INDICATE FAILURE ON 7777 BOARD CTC CHIP OR INTERRUPT CIRCUITRY
01	CTC Timer Mode	01	Time out too soon with prescaler at 16
01	CTC Timer Mode	02	Time out too late with prescaler at 16
01	CTC Timer Mode	03	Time out too soon with prescaler at 256
01	CTC Timer Mode	04	Time out too late with prescaler at 256
02	Interrupt Priority	05	No interrupt from channel under test
02	Interrupt Priority	06	No interrupt from higher priority ch.
02	Interrupt Priority	07	Interrupt received from low priority ch

APPENDIX B

MODULE DESCRIPTIONS

```
TEST ONE:  CTC TIMER MODE TEST
      Set interrupt mode two
      Load I register with page number
      Load CTC interrupt vector (00)
      Load channel control words to CTC channels 0,1,2,3 (prescaler=16)
      Load channel x time constant
      Delay a little less than expected
          ERROR (01) if interrupt has occurred                ERROR 01
      Delay enough for interrupt to occur
          ERROR (02) if interrupt hasn't occurred             ERROR 02
      Reset channel x
      Loop for all channels

      Clear interrupt counters

      Load channel control words to CTC channels 0,1,2,3 (prescaler=256)
      Clear interrupt counters
      Load channel x time constant
      Delay a little than expected
          ERROR (03) if interrupt has occurred                ERROR 03
      Delay enough for interrupt to occur
          ERROR (04) if interrupt hasn't occurred             ERROR 04
      Reset channel x
      Loop for all channels
      Clear interrupt counters
```

TEST TWO: CTC INTERRUPT PRIORITY TEST

```
Set interrupt mode two
Load interrupt vector to CTC  (04)
Load CTC control words (all channels)
Load channel zero time constant (to generate interrupt)
Enable interrupts
Delay one half millisecond
Check channel zero interrupt count
    ERROR(05) if zero                                    ERROR 05
Check channel three interrupt count
    ERROR(07) if not zero                                ERROR 07
Reset all CTC channels
Clear interrupt counters

Load interrupt vector to CTC  (08)
Load CTC control words (all channels)
Load channel one time constant (to generate interrupt)
Enable interrupts
Delay one half millisecond
Check channel zero interrupt count
    ERROR(06) if zero                                    ERROR 06
Check channel one interrupt count
    ERROR(05) if zero                                    ERROR 05
Check channel three interrupt count
    ERROR(07) if not zero                                ERROR 07
Reset all CTC channels

Load interrupt vector to CTC  (0C)
Load CTC control words (all channels)
Load channel two time constant (to generate interrupt)
Enable interrupts
Delay one half millisecond
Check channel zero interrupt count
    ERROR(06) if zero                                    ERROR 06
Check channel two interrupt count
    ERROR(05) if zero                                    ERROR 05
Check channel three interrupt count
    ERROR(07) if not zero                                ERROR 07
Reset all CTC channels

Load interrupt vector to CTC  (10)
Load CTC control words (all channels)
Load channel three time constant (to generate interrupt)
Enable interrupts
Delay one half millisecond
Check channel zero interrupt count
    ERROR(06) if zero                                    ERROR 06
Check channel three interrupt count
    ERROR(05) if zero                                    ERROR 05
Reset all CTC channels
Clear interrupt counters
```

1.0 TITLE

Wangwriter Soft-Sectored Disk Controller Diagnostic

2.0 REVISION/DATE

Date: November 17, 1981
Documentation Release: 9194
Software Release: 1194

3.0 REFERENCE DOCUMENTS

Documentation for the Wangwriter In-System Diagnostic Monitor.

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This diagnostic runs under control of the Wangwriter In-System Diagnostic Monitor.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging.
* Manufacturing (production and QC) board test, board repair and burn-in.
* Customer Engineering repair at the customer site, board repair and burn-in.

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

Disk Controller Circuitry on 7777 CPU board, disk drive and cable. See also section 5.4 below.

5.4 Tests in the Program

No.	Name	Hardware Tested
1	Reset Interrupt	FDC (floppy disk controller), PIO
2	Disk Ready Test	FDC, ready signal
3	Track Zero Test	FDC, track zero signal
4	Cylinder Address	FDC, seek circuitry on drive
5	Forward Seek Test	FDC, seek circuitry on drive
6	Reverse Seek Test	FDC, seek circuitry on drive
7	Formatting Test	FDC, disk drive
8	Read/Write Tst	FDC, VCO, disk drive

6.0 LOAD PROCEDURE

For description of how to load and start a diagnostic, see documentation for the Wangwriter In-System Diagnostic Monitor.

7.0 OPERATING INSTRUCTIONS

When control is given to the disk diagnostic by the monitor, the diagnostic examines the setting of switch 4 on the CPU board switch bank. If this switch is on, then the diagnostic will output a message indicating the options that are available for testing disk units 0 and 1.

These options are:

0 To test unit 0, hit the 0 key, then hit EXECUTE.
1 To test unit 1, hit the 1 key, then hit EXECUTE.
2 To test both units 0 and 1, hit the 2 key, then hit EXECUTE.

If switch 4 is off, then unit 1 is the only drive on the system and will be selected for test automatically.

Once an option has been selected or the default to unit 1 has occurred, a message will be put on the screen instructing the operator to remove the diagnostic diskette from unit 1 and insert a scratch diskette if unit 1 is to be tested; and to insert a scratch diskette into unit 0 if that unit is to be tested. Then depress the EXECUTE key to start the test.

If unit 1 is being tested, a message will be displayed at the end of the test instructing the operator to replace the scratch diskette in unit 1 with the diagnostic diskette, and depress the EXECUTE key. This message does not appear if the Loop on Program option is in effect.

8.0 REVISION HISTORY

Rev 1194 -- first release.

9.0 MISCELLANEOUS

None.

APPENDIX A

TEST AND ERROR INFORMATION TABLE

TEST NO.	TEST NAME	ERROR CODE	FAILING MODULE or DESCRIPTION
			ALL ERRORS INDICATE FAILURE ON 7777 BOARD OR DISK DRIVE OR CABLE FAILURE
01	Reset Interrupt	01	Int line not active occur after reset
02	Disk Ready	02	No Disk Ready Signal
02	Disk Ready	14	Disk controller hung up
03	Track Zero test	03	Bad RECALIBRATE status
03	Track Zero test	04	Bad SEEK status
03	Track zero test	05	Track zero signal failed to occur
03	Track zero test	06	Track zero signal stuck on
03	Track Zero test	08	Seek completion too slow
03	Track Zero test	15	FDC hung up during RECALIBRATE
03	Track Zero test	16	FDC hung up during SEEK
03	Track Zero test	17	FDC hung up during SENSE INT STATUS
03	Track Zero test	18	FDC hung up during SENSE DRIVE STATUS
04	Cylinder Address	04	Bad status returned after SEEK
04	Cylinder Address	07	Seek completion too fast
04	Cylinder Address	08	SEEK completion too slow
04	Cylinder Address	09	SEEK terminates at wrong track
04	Cylinder Address	15	FDC hung up during RECALIBRATE
04	Cylinder Address	16	FDC hung up during SEEK
04	Cylinder Address	17	FDC hung up during SENSE INT STATUS
05	Forward Seek	03	Bad status returned after RECALIBRATE
05	Forward Seek	04	Bad Sense Interrupt Status
05	Forward Seek	08	Seek completion too slow
05	Forward Seek	09	SEEK terminates at wrong track
05	Forward Seek	0A	Sequential SEEK completion too fast
05	Forward Seek	15	FDC hung up during RECALIBRATE
05	Forward Seek	16	FDC hung up during SEEK
05	Forward Seek	17	FDC hung up during SENSE INT STATUS
06	Reverse Seek	04	Bad Sense Interrupt Status
06	Reverse Seek	08	SEEK completion too slow
06	Reverse Seek	09	SEEK terminates at wrong track
06	Reverse Seek	0A	Sequential SEEK completion too fast
06	Reverse Seek	16	FDC hung up during SEEK
06	Reverse Seek	17	FDC hung up during SENSE INT STATUS
07	Formatting Test	0B	Bad status recieved after FORMAT
07	Formatting Test	0E	Bad status returned after READ ID
07	Formatting Test	0F	Bad header values read after FORMAT
07	Formatting Test	15	FDC hung up during RECALIBRATE
07	Formatting Test	16	FDC hung up during SEEK
07	Formatting Test	19	FDC hung up during FORMAT
07	Formatting Test	1A	FDC hung up during READ ID
08	Read Write Test	10	Bad WRITE status
08	Read Write Test	11	Bad READ status
08	Read Write Test	12	Data read not equal to data written
08	Read Write Test	15	FDC hung up during RECALIBRATE
08	Read Write Test	16	FDC hung up during SEEK
08	Read Write Test	1B	FDC hung up during WRITE
08	Read Write Test	1C	FDC hung up during READ

APPENDIX B

MODULE DESCRIPTIONS

TEST 1: RESET INTERRUPT TEST

Initialize PIO	
Issue reset to FDC	
Delay 3 milliseconds	
Check INT signal from FDC	
ERROR (01) if not active	ERROR 01

TEST 2: DISK READY TEST

Set up FDC's internal timers	
ERROR (14) if controller hung	ERROR 14
Issue SENSE DRIVE STATUS Command	
Check READY bit in status register 3	
ERROR (02) if not ready	ERROR 02

TEST 3: TRACK ZERO TEST

Call RECALIBRATE (move head to track 0)	
ERROR (15) if controller hung	ERROR 15
Call SENSE INTERRUPT STATUS (reads status reg 0)	
ERROR (17) if controller hung	ERROR 17
ERROR (03) if bad status	ERROR 03
CALL SENSE DRIVE STATUS (read status reg 3)	
ERROR (18) if controller hung	ERROR 18
ERROR (05) if no track 0 signal	ERROR 05
Load NCN with 01	
Call SEEK (step off track 0)	
ERROR (16) if controller hung	ERROR 16
ERROR 08 if time out	ERROR 08
Call SENSE INTERRUPT STATUS (read status register 0)	
ERROR (17) if controller hung	ERROR 17
ERROR (04) if bad status	ERROR 04
Call SENSE DRIVE STATUS (read status register 3)	
ERROR (18) if controller hung	ERROR 18
ERROR (06) if track zero signal	ERROR 06

TEST 4: CYLINDER ADDRESS

```
NCN = 1 (new cylinder number)
For tracks 1, 2, 4, 8, 10, 20, D0
    Call RECAL (move head to track 0)
        ERROR (15) if controller hung                    ERROR 15
        ERROR (08) if time out                           ERROR 08
    Call SEEK
        ERROR (16) if controller hung                    ERROR 16
    Wait 10 milliseconds
    Check for SEEK complete
        ERROR (07) if complete                           ERROR 07
    Wait xx milliseconds (worst case time for seek)
    Check for SEEK complete
        ERROR (08) if not complete                       ERROR 08
    Call SENSE INTERRUPT STATUS
        ERROR (17) if controller hung                    ERROR 17
        ERROR (04) if bad status                         ERROR 04
        ERROR (09) if PCN not equal NCN                  ERROR 09
    Shift NCN left one bit position (next track number)
Loop for next track if NCN not out of range
```

TEST 5: FORWARD SEEK TEST

```
Call RECALIBRATE
    ERROR (15) if controller hung                        ERROR 15
    ERROR (08) if time out                               ERROR 08
    ERROR (03) if bad status                             ERROR 03
NEW CYLINDER NUMBER = 1
Call SEEK
    ERROR (16) if controller hung                        ERROR 16
Wait 10 milliseconds
Check for RECAL complete
    ERROR (0A) if complete                               ERROR 0A
Wait 15 milliseconds
Check for SEEK complete
    ERROR (08) if not ready for command                  ERROR 08
Call SENSE INTERRUPT STATUS
    ERROR (17) if controller hung                        ERROR 17
    ERROR (04) if bad status                             ERROR 04
    ERROR (09) if PCN not equal NCN                      ERROR 09
Increment NCN
Loop until NCN   LASTTRACK
```

TEST 6: REVERSE SEEK TEST

NEW CYLINDER NUMBER = LASTTRACK	
Call SEEK	
ERROR (16) if controller hung	ERROR 16
ERROR (08) if time out	ERROR 08
Call SENSE INTERRUPT STATUS	
ERROR (17) if controller hung	ERROR 17
ERROR (04) if bad status	ERROR 04
ERROR (09) if PCN not equal NCN	ERROR 09
Decrement NCN	
Call SEEK	
ERROR (16) if controller hung	ERROR 16
Wait 10 milliseconds	
Check for SEEK complete	
ERROR (0A) if complete	ERROR 0A
Wait 15 milliseconds	
Check for SEEK complete	
ERROR (08) if not complete	ERROR 08
Call SENSE INTERRUPT STATUS	
ERROR (17) if controller hung	ERROR 17
ERROR (04) if bad status	ERROR 04
ERROR (09) if PCN not equal NCN	ERROR 09
Loop until NCN = 0	

TEST 7: FORMATTING TEST

Move head to track 0	
ERROR (15) if controller hung	ERROR 15
Format a pattern on track 0	
ERROR (19) if controller hung	ERROR 19
ERROR (0B) if bad status	ERROR 0B
Move head to next track	
ERROR (16) if controller hung	ERROR 16
Loop to format all tracks	
Read ID from track 39	
ERROR (1A) if controller hung	ERROR 1A
ERROR (0E) if bad status	ERROR 0E
ERROR (0F) if bad data	ERROR 0F
Move head in to next track	
ERROR (16) if controller hung	ERROR 16
Loop for all tracks	
Loop to test both sides	

TEST 8: READ WRITE TEST

Set up values for MFM operations	
Set up values for head 0	
Move head to track 0	
ERROR (15) if controller hung	ERROR 15
Write worst case pattern onto track	
ERROR (1B) if controller hung	ERROR 1B
ERROR (10) if bad status	ERROR 10
Move head in to next track	
ERROR (16) if controller hung	ERROR 16
Loop until all tracks written	
Read data from track	
ERROR (1C) if controller hung	ERROR 1C
ERROR (11) if bad status	ERROR 11
ERROR (12) if data read not equal to data written	ERROR 12
Move head out to next track	
ERROR (16) if controller hung	ERROR 16
Loop to read all tracks	
Loop to test both sides of disk	

APENDIX C

STATUS REGISTER IDENTIFICATION

STATUS REGISTER 0

BIT	NAME	DESCRIPTION
D7 D6	interrupt code	D7=0 and D6=0 Normal Termination of command. Command was completed and properly executed. D7=0 and D6=1 Abnormal termination of command. Execution was started, but was not sucessfully completed. D7=1 and D6=0 Invalid command Issue. Command which was issued was never started. D7=1 and D6=1 Abnormal termination because during command execution the ready signal from the drive changed state.
D5	Seek End	When the FDC completes the SEEK Command, this flag is set to 1.
D4	Equipment Check	If a fault signal is received from the drive, or if the Track 0 signal fails to occur after 77 step pulses then this flag is set.
D3	Not Ready	When the drive not-ready and a READ or WRITE Command is issued to side 1 of a single sided drive, then this flag is set.
D2	Head Address	This flag is used to indicate the state of the head at interrupt.
D1 D0	Unit Select 1 Unit Select 0	These flags are used to indicate a drive unit number at interrupt.

STATUS REGISTER 1

BIT	NAME	DESCRIPTION
D7	End of Cylinder	When the FDC tries to access a sector beyond the final sector of a cylinder, this flag is set.
D6	Not used	This bit is always 0.
D5	Data Error	When the FDC detects a CRC error in either the ID or the data field, this flag is set.
D4	Over Run	If the FDC is not serviced during data transfers within a certain time interval, this flag is set.
D3	Not Used	This bit is always 0.
D2	No Data	During READ DATA, WRITE DELETED DATA, or SCAN Command, if the FDC cannot find the sector specified in the IDR Register, this flag is set. During the READ ID Command, if the FDC cannot read the ID field, then this flag is set. During the READ A CYLINDER Command, if the starting sector cannot be found, then this flag is set.
D1	Not Writable	During of WRITE DATA, WRITE DELETED DATA, or FORMAT A CYLINDER Command, if the FDC detects a write protect signal from the drive, then this flag is set.
D0	Missing Address Mark	If the FDC cannot detect an ID Address Mark after encountering the index hole twice, then this flag is set. If the FDC cannot detect the Data Address Mark or Deleted Data Address Mark, this flag is set. Also at the same time, the Missing Address Mark in Data Field of Status Register 2 is set.

STATUS REGISTER 2

BIT	NAME	DESCRIPTION
D7	Not Used	This bit is always 0.
D6	Control Mark	During executing the READ DATA or SCAN Command if the FDC encounters a sector which contains a Deleted Data Address Mark, this flag is set.
D5	Data Error in Data Field	If the FDC detects a CRC error in the data field, then this flag is set.
D4	Wrong Cylinder	This bit is related to the No Data bit, and when the contents of C (cylinder) on the medium differs from that stored in the IDR register, this flag is set.
D3	Scan Equal Hit	During execution of the SCAN Command, if the condition of EQUAL is satisfied, then this flag is set.
D2	Scan Not Satisfied	During of the SCAN Command, if the FDC cannot find a sector on the cylinder which meets the condition, then this flag is set.
D1	Bad Cylinder	This bit is related to the No Data bit, and when the contents of C (cylinder) on the medium is different from that stored in the IDR, and the content of C is FF, then this bit is set.
D0	Missing Address Mark in Data Field	When data is read from the medium, if the FDC cannot find a Data Address Mark or Deleted Data Address Mark, then this flag is set.

STATUS REGISTER 3

D7	Fault	This bit is used to indicate the status of the fault signal from the drive.
D6	Write Protect	This bit is used to indicate the status of the WRITE PROTECT signal from the drive.
D5	Ready	This bit is used to indicate the status of the READY signal from the drive.
D4	Track 0	This bit is used to indicate the status of the track 0 signal from the drive.
D3	Two Side	This bit is used to indicate the status of the two side signal from the drive.
D2	Head Address	This bit is used to indicate the status of the side select signal from the drive.

D1	Unit Select 1	This bit indicates the status of the Unit Select 1 signal to the drive.
D0	Unit Select 0	This bit indicates the status of the Unit Select 0 signal to the drive.

1.0 TITLE

Wangwriter VCO Adjustment Utility

2.0 REVISION/DATE

Date: November 17, 1981
Documentation Release: 9190
Software Release: 8190

3.0 REFERENCE DOCUMENTS

Documentation for Wangwriter In-System Diagnostic Monitor.

4.0 CONFIGURATION REQUIREMENTS

4.1 Hardware

This program was developed on a Wangwriter having the 7775 (memory/CRT board) at E rev 1, the 7776 (printer driver board) at E rev 3, and the 7777 (CPU/Disk board) at E rev 4.

4.2 Software

This diagnostic runs under control of the Wangwriter In-System Diagnostic Monitor.

5.0 PROGRAM DESCRIPTION

5.1 Uses

* Engineering prototype debugging.
* Manufacturing (production and QC) board test, board repair and burn-in.
* Customer Engineering repair at the customer site, board repair and burn-in.

5.2 User Interface

Wangwriter In-System Diagnostic Monitor.

5.3 Hardware Tested

This program is an adjustment utility and does not test hardware.

5.4 Tests in the Program

This is an adjustment utility and contains no tests.

6.0 LOAD PROCEDURE

For description of how to load and start this utility, see the documentation for the Wangwriter Diagnostic Monitor.

7.0 OPERATING INSTRUCTIONS

Once started this program do a continuous READ ID operation on track 10 and reports the results of the operation to the screen. The operator is to use this information to adjust the potentiometer on the CPU board until the reported status is consistently good. The screen will be constantly updated with status of the READ ID operation with the following messages:

Good status returned from continuous READ ID.
Bad status returned from continuous READ ID.
Disk controller time out.

Once the operator is satisfied that the adjustment is complete, the program may be exited by striking the CANCEL key.

8.0 REVISION HISTORY

Rev 8140 -- first release.

9.0 MISCELLANEOUS

None.