O T T O : 3 8 0 0

M A N U A L

Test Automation Software recommends highly that you read these installation instructions on a computer different from the one being configured, or that you print these instructions before starting. If you print these instructions, use the check boxes to keep track of your work.

Table Of Contents

1 Welcome to OTTO:3800

1.1 Introduction

OTTO:3800 is a combined hardware and software package that performs fully automatic measurement of modem reliability and performance. Working in conjunction with telephone network simulator equipment from Telecom Analysis Systems (Eatontown, NJ), OTTO:3800 performs all tests without operator intervention and with a minimum of manual steps. This manual for OTTO:3800 guides you through installation, operation, and customization of the product. It is broken into three parts.

The first section of this manual describes how to install OTTO:3800 on your computer, how to connect and configure the TAS Network Simulator, and how to configure the software and hardware to work together.

The second section describes how to use the software provided. Included in this section is a checklist of common errors operators make when running this software package and how to avoid them.

The third section describes how to create your own test suites.

OTTO:3800 is designed to, by itself on a single computer, test a pair of external modems. Internal modems can be tested if the products to be tested include full controllers and appear to MS–DOS as serial port interfaces.

To test controllerless (so–called “winmodems”) and to test host–signal–processing modems (so–called “soft modems”), you need a total of three personal. You run OTTO:PASS II (provided on the CD–ROM as part of the OTTO:3800 product) to link the modems under test with OTTO:3800.

To test client modems with Ethernet–connected modem banks and router access servers such as the Ascend MAX, you need the companion product OTTO:ESB, sold separately.

Also included on your CD–ROM are two Microsoft Excel templates, one to display the results of TSB 37A analog–analog testing and one to display the results of PN 3857 analog–to–digital testing.

1.2 Installation Overview

Test Automation Software recommends highly that you read these installation instructions on a computer different from the one being configured, or that you print these instructions before starting. If you print these instructions, use the check boxes to keep track of your work.

The steps for installing and configuring OTTO:3800 and your network simulator are:

- Collecting all the parts and tools you will need
- Preparing, connecting, and configuring the TAS network and loop simulators
- Setting switches and jumpers on the IEEE–488 adapter board

- Setting jumpers on the dual serial–port adapter board
- Installing the adapter boards into your computer
- Connecting the TAS equipment to your computer
- Installing the OTTO:3800 software onto your computer hard disk
- Configuring the network simulator control software
- Configuring the OTTO:3800 software

For people experienced in hardware and software installation, see the Quick Install Guide on this CD–ROM.

1.3 Equipment you need

You will need one computer to run OTTO:3800 when testing two modems connected via RS–232 cables.

If you are testing controllerless or Host Signal Processing (HSP) modems that require Windows to operate, you will need at least one additional computer to run the modem, or two additional computers if you are performing pair testing as described in TSB 38 and TSB 37A.

If you are running with an Ethernet–connected modem and using OTTO:ESB, the OTTO:ESB product must run on an additional computer. See the wiring diagrams at the end of the manual for more information.

1.4 Personal computer system requirements — OTTO:3800

The following describes the minimum system requirements for running OTTO:3800:

- ❑ Pentium processor, 200 Mhz or faster. Similar AMD or Cyrix processors are acceptable.
- ❑ 128 kilobytes of Level 2 Cache
- ❑ Four megabytes of RAM
- ❑ VGA adapter
- ❑ CD-ROM drive
- ❑ Bus mouse - Serial pointing devices are strongly discouraged

NOTE: OTTO:3800 does not utilize a mouse device.

- ❑ MS-DOS 5.0 or higher. Be sure that the utility EDIT is on the system and functioning. (The EDIT program is loaded automatically when Windows 95 is installed on your system.)

NOTE: This program has been tested using MS-DOS 5.0 and using the underlying MS–DOS that comes with Windows 95 and Windows 98. You must boot into the command–line mode to use this package; you cannot use the “DOS box” feature of Windows 95 to run OTTO:3800.

OTTO:3800 ***will not run*** on a system with any version of Windows NT installed. (OTTO:PASS II will; see next section.)

1.5 Personal computer system requirements — OTTO:PASS II

You will need two of these computers when testing pairs of modems.

The following describes the minimum system requirements for running OTTO:PASS II:

- ❑ Pentium processor, 200 Mhz or faster. Similar AMD or Cyrix processors are acceptable. OTTO:PASS II works on most laptop computers.
- ❑ 128 kilobytes of Level 2 Cache
- ❑ 32 megabytes of RAM
- ❑ VGA adapter
- ❑ CD-ROM drive
- ❑ Bus mouse - Serial pointing devices are strongly discouraged
- ❑ Windows 95, Windows 98, or Windows NT 3.51

NOTE: OTTO:PASS II ***will not run*** on computer systems running MS–DOS, DR DOS, Windows 2.0, 2.1, 3.0, 3.1, or Windows for Workgroups.

2 Installation

2.1 Preparing to install OTTO:3800

Before you begin the installation process, collect everything you need to complete the installation successfully. Use the checklists below:

2.1.1 From the OTTO:3800 box:

- ❑ Dual serial port adapter board
- ❑ IEEE-488 adapter board
- ❑ IEEE-488 cable
- ❑ Two RS-232 cables
- ❑ Null modem adapter
- ❑ CD-ROM

NOTE: The second high–speed dual–port serial card included in your OTTO:3800 box, plus the manual and driver disks for the card, are for your use in a Windows 95 or Windows 98 computer containing an internal UUT modem. Additional serial cards may be ordered from Test Automation Software.

2.1.2 From Telecom Analysis Systems

- ❑ TAS 1200/DFE Series 2 Network Simulator or TAS 1200 Series 2 Digital Network Simulator
- ❑ TAS 240 VLSE Subscriber Loop Simulator

> NOTE: OTTO:3800 does not make any use of Gemini HS-1022, VS-1022, or Warp. The Gemini unit must be disconnected from the rest of the TAS equipment.

- ❑ Hybrid balance cable (cable with four wires, lugs on both ends of each wire)
- ❑ Four eight-wire RJ-45 flat cable
- ❑ IEEE-488 cable
- ❑ Two equipment power cords

2.1.3 Personal computer

- ❑ Personal computer with one open ISA slot, one PCI slot, and VGA adapter

> NOTE: An ISA slot can be identified by its black connector. A PCI slot can be identified by its off-white connector.

- ❑ Monitor, color preferred (the software will work with black-and-white monitors)
- ❑ Keyboard, either AT or Windows configuration
- ❑ Equipment power cords for monitor and computer
- ❑ Bus Mouse

A serial mouse cannot be used with OTTO:3800

- ❑ Windows 95, any version, OSR2 preferred

2.1.4 Tools

- ❑ Phillips–head screwdriver, in many cases this is the only tool required
- ❑ Standard straight-blade screwdriver for #6 screws

2.1.5 Workspace

- Prepare a clear workspace in a static–free area for testing. The computer, the two adapter boards, and the modems being tested are susceptible to static discharge damage. Please exercise every caution to prevent damage during installation and during subsequent operation.

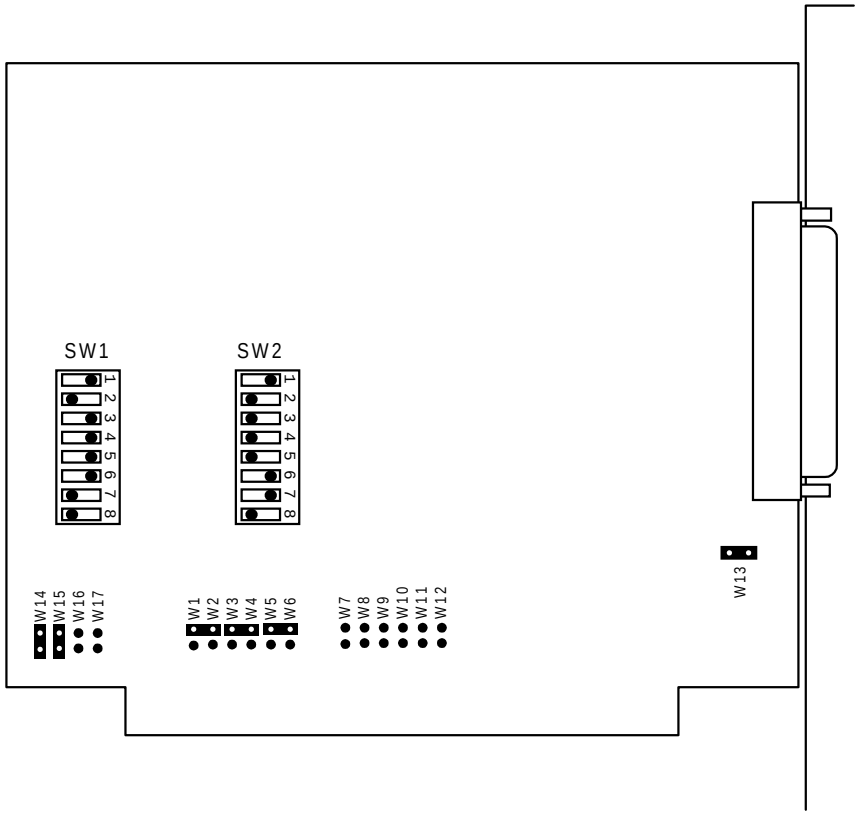
2.2 Preparing Your TAS Network Simulator

If your TAS network simulator has not been set up before, then go to Appendix A at the end of this document and prepare your network simulator for use.

2.3 Installing OTTO:3800 hardware

2.3.1 Configuring the IEEE-488 adapter card

- ❑ Remove the IEEE-488 card from the static protection envelope. You can identify the card by the single large connector on the bracket. Place the static protection envelope on your table, and place the card component-side up on the static protection envelope, with the bracket on the right as shown in the illustration below.



IEEE–488 adapter

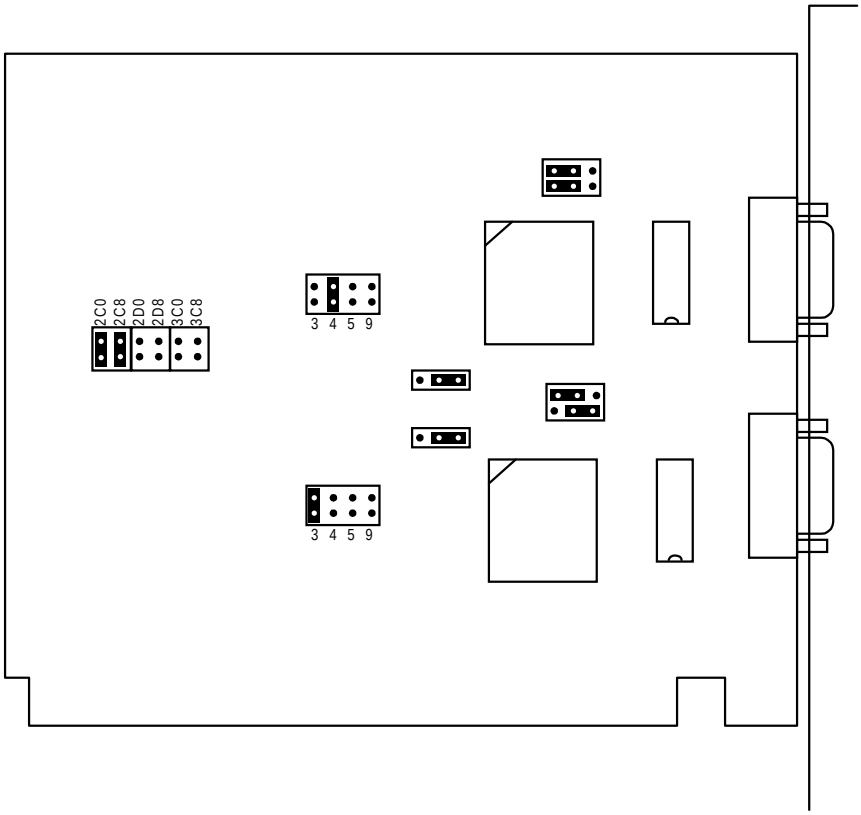
- ❑ Compare the positions of the six jumper blocks on the board with the six jumper blocks shown in the illustration above. If the jumper blocks on the board do not match the illustration above, make the necessary changes so they match.

  The following is the jumper block list, looking left to right along the bottom edge of the card:

  - ❑ W14 jumpered
  - ❑ W15 jumpered
  - ❑ W16 open
  - ❑ W17 open
  - ❑ W1 – W6 left sides jumpered only; right sides open
  - ❑ W7 – W12 open
  - ❑ W13 jumpered

- The eight switches on the DIP switch pack marked “SW1” should be correctly set to the illustration above, counting from top to bottom, 1, 3, 4, 5, 6 to the right (on) and 2, 7, 8 to the left (off).
- The eight switches on the DIP switch pack marked “SW2” should be correctly set to the illustration above, counting from top to bottom, 1, 6, 7 to the right (on) and 2, 3, 4, 5, 8 to the left (off).

2.3.2 Configuring the dual serial port card



Dual–Port Serial Adapter Card

- Remove the dual-port serial card from the static protection envelope. You can identify the card by the two 9-pin connectors on the bracket. Place the static protection envelope on your table, and place the card component-side up on the static protection envelope, with the bracket on the right as shown in the illustration above.
- At the top of the board, there are a pair of 3-pin headers marked “Serial A Select.” Following the illustration above set to COM1 (jumper blocks connect the left and center pins, leaving the right pins open).
- Between the two square chips is a similar pair of 3-pin headers marked “Serial B Select.” Following the illustration above set to COM2 (jumper blocks connect the top-left to the top-center pins and bottom-center to the bottom-right pins).

- To the left of the "Serial B Select" headers you find two separate 3-pin headers. These are marked "Serial A Enable" and "Serial B Enable". Following the illustration above set to enable both ports (jumper blocks connecting the center and right pins on both headers).
- To the upper-left the enable header is a series of pins marked "IRQ Serial A". Following the illustration above set to connect the two vertical pins just above "4".
- Below the IRQ Serial A block is a similar set of header pins marked IRQ Serial B." Following the illustration above set to connect the two vertical pins just above "3".
- Finally, at the left edge of the board there is a block of header pins marked "Clock Set". Connect the vertical pair of pins below "2C0" with one jumper block, and the vertical pair of pins below "2C8" with the second jumper block.

  This completes the configuration of the dual serial port board.

2.3.3 Installing hardware into your personal computer

> NOTE: When the instructions below for installing adapter boards conflict with the instructions for installing adapter boards that came with your computer, use your computer's instructions.

- Turn off your computer and monitor. (If you are running Windows, be sure to use the Shut Down feature to preserve your hard disk data.)
- Unplug the power cords from your computer and monitor.
- Remove the case cover from your computer.
- Find two open ISA slots suitable for the adapter boards provided in the package. ISA slots are identified by looking for large black printed circuit board edge connectors near the slot openings in your computer. (PCI slots are typically off-white circuit board edge connectors; these cannot be used.)
- Remove and save the two slot covers. In most computers, the covers are held by a single screw.
- Insert the serial port card and the IEEE-488 card into the two ISA slots.
- Fasten down the boards with the screws that originally held the slot covers.
- Replace the case cover of your computer.
- If you unplugged all cables earlier, reconnect the cables *except the power cables* at this time.
- Plug in the power cords from your computer and monitor.

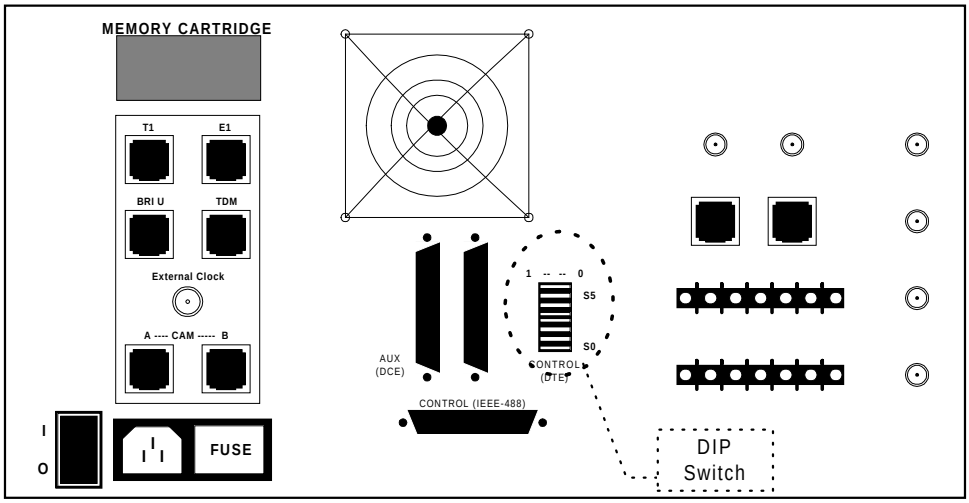
Note: STOP! Read the next step carefully and completely before going on.

- ❑ Turn on power to your monitor, then after 15 seconds turn on your computer. When the power–on self–test (POST) is running, press the key indicated on the screen that takes you to BIOS setup. This may be the DEL or ESC key. *There is a very short window of time to press the indicated key to enter the BIOS setup routines.*
- ❑ Follow the menu prompts to locate how to configure "built–in peripherals". Locate the entries for the two serial ports. Configure the built–in serial ports to "disabled".
- ❑ Tell the BIOS setup routine to exit and save changes.
- ❑ Let the computer boot and start the operating system.
- ❑ Turn off your computer and monitor. (If you are running Windows, be sure to use the Shut Down feature to preserve your hard disk data.)

This completes the preparation of your personal computer.

2.4 Connecting the TAS units to the PC

In this section, you connect the TAS network simulator to your personal computer, and determine the current IEEE–488 addresses for the TAS 240 and TAS 1200.

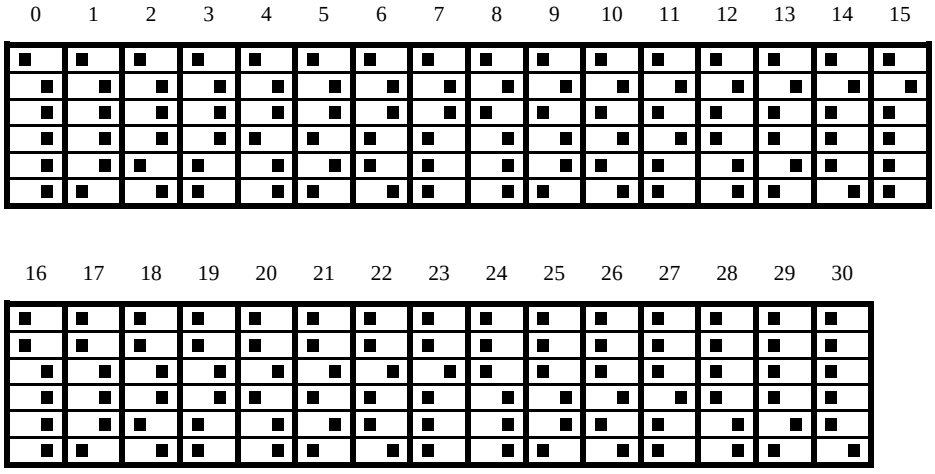


TAS 1200

- ❑ Install one end of the IEEE-488 cable (supplied by OTTO:3800) on top of the first IEEE-488 cable connection to the TAS 1200 control port. Tighten the lock screws finger tight.
- ❑ Install the other end of the IEEE–488 cable to the IEEE–488 connector on the IEEE–488 adapter card installed in your personal computer. Tighten the lock screws finger tight.

2.5 Determining TAS unit IEEE–488 addresses

❑ Find the diagram below which matches the positions of the switches on the DIP switch pack. Circle it.

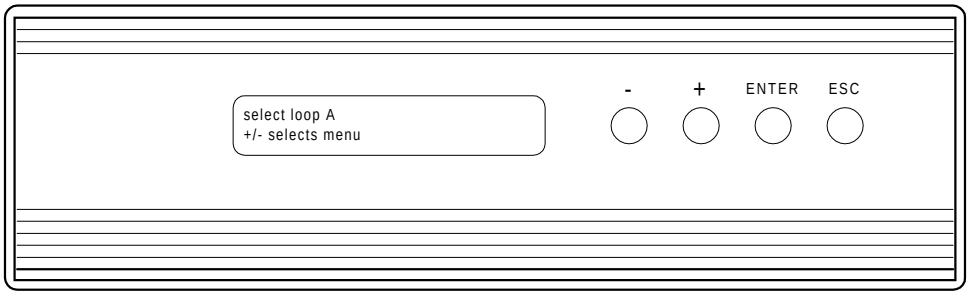


The number above the matching diagram you circled indicates the IEEE–488 address set on the TAS 1200.

❑ Write down the IEEE-488 address for the TAS 1200 here: _____

❑ Turn the 240 and 1200 so the front panels are facing you.

❑ Turn on the TAS 240 and the TAS 1200 units. The switches are located on the rear panels near the equipment power cable plugs into the unit. The TAS 1200 will indicate that it is executing the self–test sequence by flashing the front–panel lights and clicking relays for about a minute. The TAS 240 will indicate that it is executing the self-test sequence by displaying the fact on the front–panel LCD screen and clicking relays for about 90 seconds. This is normal. After both units become quiet, continue with the next step.



TAS 240

❑ On the 240 front panel, press the ESC button five times. This ensures that the unit is at the top–most menu.

❑ On the 240 front panel, press the plus (+) key until the following is displayed:

```
auxiliary parameters↵
+/- selects menu
```

- ❑ Press the ENTER key. Then press the plus (+) key until the following is displayed:

```
configuration↵
+/- selects auxiliary parameter menu
```

- ❑ Press the ENTER key. Then press the plus (+) key until the following is displayed:

```
REMOTE PROTOCOL: gpib↵
+/- selects remote protocol
```

- ❑ Press the ENTER key. A display appears that is similar to the one below:

```
GPIB ADDRESS: 2
+/- selects GPIB address
```

- ❑ Write down the IEEE-488 ("GPIB") address for the TAS 240 here: _____
- ❑ Press the ESC button four (4) times.

  This completes the configuration of the TAS 240 Voice Line Subscriber Emulator.

2.6 Installing the OTTO:3800 software

The OTTO:3800 package runs as a MS–DOS program, not as a Windows 95 program or within a Windows 95 "DOS box". Installation, though, may be performed in a Windows 95 "DOS box" without problems. These procedures assume that you have Windows 95 loaded onto your system, and that Windows 95 boots into its "GUI" mode by default.

- ❑ Turn on your monitor, then after 15 seconds turn on your computer. Allow your computer to boot normally.
- ❑ Wait for the boot sequence to complete (the normal arrow cursor has been displayed for at least 15 seconds).
- ❑ Open your CD-ROM drive and insert the OTTO:3800 CD-ROM. Close your CD-ROM drive.
- ❑ Open the "My Computer" icon.

- ❑ Open the CD-ROM disk icon.
- ❑ Launch “INSTALL” (this may be labeled “INSTALL.EXE” depending on how Explorer is configured).
- ❑ The installer will open a “DOS box” window and ask you onto which drive do you wish to install the software. Type the drive letter for the partition onto which you wish to install the software, then press the ENTER key.

  NOTE: Do not select a network drive. When running in the command–line mode, MS–DOS cannot see your network or access any other computer system on your network.

  The installer will copy all information from the CD-ROM drive to the drive you specify. All information and programs are copied into a directory named “\OTTO” — this name cannot be changed in the installer. When the installer is finished, it will display a messge saying that the installation has been completed successfully.
- ❑ Press ENTER. The installer will close. If the “DOS Box” window stays on the screen and has the word “Finished” in the title bar, use your mouse to click the Close box (box with the “X”) located in the upper right hand corner of the window.
- ❑ Turn off your computer and monitor. (If you are running Windows, be sure to use the Shut Down feature to preserve your hard disk data.)

  This completes the software installation for testing using RS-232 interfaces only.

2.7 Install additional OTTO:3800 packages

If you have purchased OTTO:3800 add–ins such as OTTO:PASS or OTTO:ESB (EtherSerial Bridge), install them at this time.

Refer to the instruction manuals that came with each product for system requirements, installation procedures, and configuration instructions.

2.8 Configuring OTTO:3800

2.8.1 Configuring NETSIM, the network simulator control software

NOTE: Read the following direction *before* performing the task.

- ❑ Turn on your monitor, then 15 seconds later turn on your computer. Let your computer start its boot process. Immediately start pressing the F8 key about three times a second until your computer displays a menu of options as shown below:

```
Microsoft Windows 95 Startup Menu
==================================

   1.  Normal
   2.  Logged (\BOOTLOG.TXT)
   3.  Safe mode
   4.  Safe mode with network support
   5.  Step by step confirmation
   6.  Command prompt only
   7.  Safe mode command prompt only
   8.  Previous version of MS-DOS

Enter a choice:  1
```

NOTE: The actual menu displayed may be different from the one pictured here.

- Select “Command prompt only” by typing the number corresponding to the option, then press Enter. (Using the display above, you would press “6”, then Enter.) The computer will then enter MS-DOS mode.

NOTE: If you accidentally boot into Windows, you can recover after Windows completes its startup by using the Start menu to select Shutdown, then select the option “Restart the computer in MS–DOS mode” and click the “Yes” button.

- At the C: prompt type “cd \otto\netsim” and then Enter. The file contents displayed on your screen should look like this:
- At the C: prompt type “edit ztas.def” and then Enter. The EDIT program will start and display the contents of the definitions file. The default file looks like this:

```
NS          01
LS          02
```

The file consists of device names and the IEEE–488 address assigned to them. Each line contains one association. Each association consists of a name, one or more spaces, then the IEEE-488 address.

In the default file, the device “NS” is assigned to address 1 and the device “LS” is set to address 2.

- The first line should contain “NS” followed by the decimal value of the IEEE–488 address for the TAS 1200. The default value is “1”. If the address of your TAS 1200 is different (look back to the section on setting up the TAS hardware where you recorded the address) change the address to the recorded value.

- The second line should contain “LS” followed by the decimal value of the IEEE–488 address for the TAS 240. The default value is “2”. If the address of your TAS 240 is different (look back to the section on setting up the TAS 240 configuration where you recorded the address) change the address to the recorded value.

  NOTE: If you have other devices you wish to control from OTTO:3800, add the additional lines to this table. Make up a two-character name and add lines in the same format as the NS and LS lines to indicate the address of the additional equipment. For example, if you have two modem switches with addresses 7 and 10, you might add lines to your file so it looks like this:

```
NS    01
LS    02
S1    07
S2    10
```

- Select File from the menu bar, and within the menu select Save. This saves the changed file.
- Select File from the menu bar, and select Exit. EDIT will cease to run and the cursor will be at a C> prompt.

  This completes the configuration of the software.

2.8.2 Configuring OTTO:3800, the test control software

This section details the steps required to insert a test station identification into the OTTO:3800 configuration files. Within your shop, this marks each test with the identification of the test stand used. When you need technical support, this identification assists our support staff when you send in log files.

Test Automation Software highly recommends you perform this procedure.

- If necessary, reboot your computer into MS-DOS mode as described at the beginning of the previous section.
- Type “cd \OTTO” and then press Enter.
- Type “edit OTTO.DEF” and then press Enter. The contents of the file is displayed on your screen. The default file looks like this:

```
[stand]      = No test stand information configured
[histlevel]  = 0     ; do not change
;[port1adrs] = 288   ; base address in hex
;[port1irq]  = 5     ; IRQ in decimal (2-7)
;[port2adrs] = 288   ; base address in hex
;[port2irq]  = 5     ; IRQ in decimal (2-7)
```

NOTE: Do not change any line other than the top line. If you have a special condition requiring you to address serial ports other than COM1 and COM2, read the reference manual.

- ❑ Replace the text “No test stand information configured” with a unique identification string less than 80 characters. Examples: “HCL cell station”; “My Labs station 3”; “XYZ Corp, Omaha DVT #2”.
- ❑ Select File from the menu bar, and within the menu select Save. This saves the changed file.
- ❑ Select File from the menu bar, and select Exit. EDIT will cease to run and the cursor will be at a C> prompt.

2.9 Testing your Installation

Now that you have completed your installation, it’s time to check it. Perform the following steps:

- Locate the two serial cables and the null modem adapter that came in your OTTO:3800 box. The null adapter is the small black rectangular box with two 25-pin connectors on each end, and is marked “NULL MODEM (full handshake).”
- Connect the 9–pin end of each serial cable to the two ports onto the two ports of the dual–port serial card.
- Connect the 25–pin end of each serial cable to the null adapter.
- Type at the C: prompt “loopback” and then touch the ENTER key.

During the execution of the loopback test, you should see throughputs reported on the screen of 11,515 characters per second or higher.

The test runs four times, twice in the A–to–B direction, and twice in the B–to–A direction.

If this test runs successfully, then the installation of your dual–port serial card is correct and the software is working with the serial card correctly.

Your installation is now complete!

3 Checklist

This section lists some of the things to watch for when using this product.

- The base name (name part to the left of the dot) of the .MDM file and the log file to be created as declared in the [sumfile] parameter should be five characters or fewer. The names should be identical. For example, the file “43-99.mdm” should have the line “[sumfile] = 43-99.xls” in it when using the supplied batch files, definition files, table files, OTTO:POST post–processing script files, and Excel templates.
- OTTO:3800 always appends data to its log files. When restarting a test from scratch, be sure to delete the log files before starting. Failure to do this will result in data from multiple test runs mixing together. This is especially true when you change modem software or firmware versions.
- *Always always ALWAYS* test a modified .MDM file before launching a multi–hour test, even if you “only changed one character.” Using the standard tests, use the programs “CHK288” (for analog–analog testing) and “CHKD” (for analog–digital testing) to be sure that the modems behave as you intend.
- OTTO:3800 runs under MS–DOS, while Excel requires Windows. It is recommended that all data from a run be saved on a floppy disk or other media and taken to another system running Windows to generate the report. This increases the amount of testing that can be done on a given modem testing station, and it removes any potential interference from a network card in the OTTO:3800 system that could bias test results.

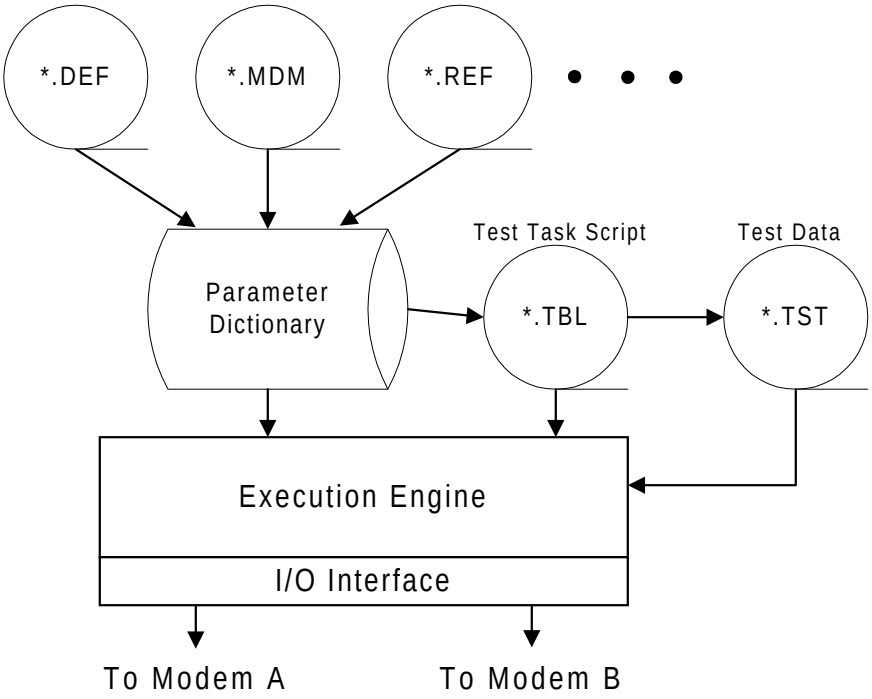
4 Operation Principles and Concepts

4.1 Overview of this section

This section describes how to operate of the OTTO:3800 program.
The first part provides a functional block–diagram description of the product.
The remainder of this section details the files and directories created by the installer.

4.2 Test Program Architecture

The OTTO:3800 system consists of an execution engine, a task script executor, and a parameter dictionary. The execution engine performs each testing task as described in the task script, using local test parameters defined in the task description line and global parameters defined in the parameter dictionary. The following diagram illustrates the relationship:



OTTO:3800 Architecture

The parameter dictionary is built by OTTO when it reads one or more parameter files, as specified on OTTO's command line. The name of the file (even the extension) has no meaning to the program, so the different types of files are identical in construction. For example, the command line

```
OTTO  INIT.DEF  31-02.MDM  TC.REF
```

invokes the OTTO program and directs the program to read the contents of three files, `INIT.DEF`, `31-02.MDM`, and `TC.REF` in that order. (A fourth file, `OTTO.DEF`, is also read after all files indicated in the command line have been processed.)

The use of the extensions DEF, MDM, and REF are to permit the operator to keep track what data should be contained in that file. There would be no difference if the command above were replaced by the following two commands:

```
COPY INIT.DEF+31-02.MDM+TC.REF STUFF
OTTO STUFF
```

The result would be exactly the same.

One of the parameter dictionary entries contains the name of the file that contains the task execution script. This script indicates how to perform a specific test using the capabilities of the execution engine, what test–specific parameters are to be applied, and for external programs when to invoke them and with what parameters.

Regularly invoked from task execution scripts are calls to the interface subsystem which sets up the network simulator. For the OTTO:3800 distribution that works with Telecom Analysis Systems hardware, the interface subsystem consists of an interface program and a series of command scripts.

4.3 Types of Files — OTTO directory

Within the OTTO directory, the standard distribution has the following types of files:

- (no extension) special data files
- .BAT Executable batch files
- .DEF Test definition files
- .EXE Programs
- .MDM Modem definition files
- .LST OTTOPOST program file listings (debug information)
- .PGM OTTOPOST program files for post-parsing data
- .REF Reference modem definition files
- .TBL Test task scripts, describing the steps for executing a test
- .TXT Text files
- .XLS Test results files

4.4 Types of Files — OTTO\EIA directory

The files in this directory have the same extension — .TST — and all files are exactly 32,768 bytes. These are the data files used in Throughput versus file type testing.

4.5 Types of Files — OTTO\GOOD and OTTO\BAD directories

These two directories are used by OTTOPOST to hold intermediate results of results parsing. The files in these directories have the same extension — .XLS — and the files contain test results with no errors (OTTO\GOOD) and test results with errors (OTTO\BAD).

In particular, the copies of the log files saved in the OTTO\BAD directory contain the results of tests that did not succeed. The tester can examine the data contained in the log to determine where and how a particular test trial failed. This examination can improve the tester's insight into what the modem(s) did or did not do.

4.6 Types of Files — OTTO\NETSIM directory

Within the OTTO\NETSIM directory, the standard distribution has the following types of files:

- No extension Network Simulator setup command files
- .EXE Programs
- .DEF Program parameter files

4.7 Types of Files — OTTO\SAMPLES directory

This directory contains samples of .MDM and .REF files. An index of the files is named INDEX.

5 Using the Provided Batch Files

OTTO:3800 in the standard distribution contains a number of pre–defined batch files for testing. These files make it easy to perform standard testing chores. This section describes how to use these standard batch files.

Before you can use these batch files, you need to make Modem Definition Files for the units you are testing. This is described in the section following.

5.1 CHK288.BAT

This batch file is used when performing tests on two V.34 modems. For the majority of tests, you invoke it as follows:

```
CHK288 31-03.MDM
```

This will display the name from the .MDM file and let you verify that you have the correct modem in test. The batch file will then conduct a short sample of each of the various tests, so that you can verify that you have the correct modes selected for each test.

The purpose of performing this checkout test, which takes about 20 minutes, is to save you from having to re–execute the full suite of tests if you make an error in the settings in the Modem Definition File.

5.2 STD288.BAT

This batch file is used when performing tests on two V.34 modems. For the majority of tests, you invoke it as follows:

```
STD288 31-03.MDM
```

This will display the name from the .MDM file and let you verify that you have the correct modem in test. The batch file will then conduct the following tests: 100 percent Throughput versus Network Model Coverage, Throughput versus File type, Call Connect Reliability, Character Echo Delay, Block Acknowledge Delay, and a scan of all the ATI commands. The results are then post–processed automatically. The output of this test can be copied directly into the TSB37A.XLS template (found on the CD–ROM) to generate the OTTO reports.

5.3 CHKD.BAT

This batch file is used when performing tests on a V.34 or V.90 modem with a digital router access server. For the majority of tests, you invoke it as follows:

```
CHK288 31-03.MDM TC.REF
```

This will display the name from the .MDM and .REF file and let you verify that you have the correct modem in test. The batch file will then conduct a short sample of each of the various tests, so that you can verify that you have the correct modes selected for each test.

The purpose of performing this checkout test, which takes about 20 minutes, is to save you from having to re–execute the full suite of tests if you make an error in the settings in the Modem Definition File and the Reference file.

5.4 STDD.BAT

This batch file is used when performing tests on two V.34 modems. For the majority of tests, you invoke it as follows:

```
STD288 31-03.MDM TC.REF
```

This will display the name from the .MDM file and the .REF file and let you verify that you have the correct modem in test. The batch file will then conduct the following tests: 100 percent Throughput versus Network Model Coverage (I model only), Throughput versus File type, Call Connect Reliability, Character Echo Delay, Block Acknowledge Delay, and a scan of all the ATI commands. The results are then post–processed automatically. The output of this test can be copied directly into the 3857I.XLS template (found on the CD–ROM) to generate the OTTO reports.

6 Modem Definition Files

Before you can perform any testing, you have to describe a number of parameters needed by the modems to the program. This is customarily done in the .MDM file. This section describes how to create .MDM files for various testing situations.

6.1 MDF Files for V.34 Pair Testing

Modem settings files drive OTTO:3800. A sample file “13-42.MDM” is reproduced below:

```
;Modem definition file 13-42.MDM
[barcode] = 13.42
[product] = Joes Bar and Modem V.34 (Rockwell RC288ACi)
[sumfile] = 13-42.xls
;
[speeda]       = 115200
[i1a]          = AT &F M0 W1 &C1 &D2 &G0 \K1 \N4 %C2 -K0
[i2a]          =
[nmc-eca]      = AT %C0
[echo-directa] = AT \N1 %E0 +MS=11,0,28800,28800
[echo-normala] = \N0 AT %E0 +MS=11,0,28800,28800
[echo-eca]     = AT %C0 %E0 +MS=11,0,28800,28800
[echo-dca]     = AT %E0 +MS=11,0,28800,28800
;
[speedb]       = 115200
[i1b]          = AT &F M0 W1 &C1 &D2 &G0 \K1 \N4 %C2 -K0
[i2b]          =
[nmc-ecb]      = AT %C0
[echo-normalb] = \N0 AT %E0 +MS=11,0,28800,28800
[echo-ecb]     = AT %C0 %E0 +MS=11,0,28800,28800
[echo-dcb]     = AT %E0 +MS=11,0,28800,28800
;
[dialb]   = ATDT1
[answera] = ATA
;
;  The following are optional
;
[stripparity]      = N
[autoanswer]       = N
[interchar-delaya] = 2
[interchar-delayb] = 2
[setupdelay]       = 0
[calldelay]        = 0
[busy-delay]       = 20
[answerdelay]      = 0
[test-ms-delay]    = 0
[hangupdelay]      = 2
[at-timeout]       = 2
[commandtimeout]   = 10
[dialtimeout]      = 20
[answer-timeout]   = 45
[hangup-timeout]   = 10
[retry-limit]      = 4
[busy-retry-limit] = 4
[at-retry-limit]   = 4
```

Each line of a modem definition file has the same basic structure: an identifier enclosed in square brackets, an equal sign, and a settings string. A semicolon introduces a comment and is never part of a settings string.

The **[barcode]** line defines a shorthand identifier (fourteen characters maximum) for the modem that appears in all test output. This identifier can be a test article sequence number, a serial number, or a “job” number.

> The developer used a scheme "*mm-uu*", where *mm* is the month in which the modem was received and *uu* is the sequential ID assigned during the month for that modem. Thus, "30-04" indicated the modem under test was the fourth modem received during the 30th month.

The **[sumfile]** line defines the suffix when building file names. This suffix is coded in the form "xxxxx.XLS" in order to work with the standard files. Note that the length of the file name ("xxxxx") is limited to five characters or fewer. It is highly recommended that the name of the .MDM file match, so that if the .MDM file is named "12-34.mdm" this line should be coded "[sumfile] = 12-34.xls".

The **[product]** line defines the name of the product. This field may be up to 80 characters long.

The following lines define specific setup parameters to each of the two modems.

> NOTE: In the following descriptions, an asterisk (*) indicates that the name has either an "a" or a "b" appended to it. The suffix indicates whether the parameter applies to the reference (A) modem or the UUT (B) modem.

The **[speed*]** line indicates the fastest modem-to-DTE interface rate the modem is capable without dropping characters or otherwise causing errors.

The **[i1*]**, **[i2*]**, ... , **[i9*]** lines describes the modem setup string(s) sent before every test. The command to be specified in each line includes the AT prefix; a carriage return is appended to each line by the software. The first string should always start with a command to reload factory settings (while &F is shown here, &F1 &F2 are also common) so that the modem is always returned to a known state. The remaining commands on this line set the modem as required for the Throughput vs. File Type (TFT), and Call Connect Reliabilty (CCR). In some modems the system has to issue multiple setup strings; in those cases the [i2*], [i3*], etc. parameters are specified.

> NOTE: The compression option shall be set to compress data in both directions.
>
> In the sample file above, an "empty" specification for [i2a] and [i2b] appear. The reason for this is discussed in section 2.2 when testing of differing modems is discussed.

The **[nmc-ec*]** line defines the modem setup string to send after the basic string (defined in [i1*] et. al.) has been sent to the modem when performing a standard Throughput versus network model coverage test. This string just turns off compression.

The **[echo-normal*]** line defines the command string that kills flow control, and sets up non-error-controlled buffered ("normal") mode.

The **[echo-ec*]** line defines the command string that kills flow control, and sets up V.42 error-control mode (no compression).

The **[echo-dc*]** line defines the command string that kills flow control, and sets up compression mode — V.42 error control with V.42 bis compression.

The **[dial*]** line defines the command to use to dial the modem. This command includes the number to be dialed.

The **[answer*]** line defines the command to use to cause the modem to answer. This parameter is not used when [autoanswer] is set to Y.

NOTE: A given .MDM file can have one or both [diala] and [dialb] commands, and have one or both [answera] and [answerb] commands. As long as there is one of either [diala]/[answerb] or [dialb]/[answera] the test can function properly. When all four commands are present, the test script commands "fdial" and "rdial" determine which modem becomes the dialing modem and which modem the answer modem.

There are other commands which can appear in a modem settings file:

- The **[stripparity]** = Y line, when present, means that the modem does not send result codes with the 8th bit turned off. There is no "a" or "b" appended; the entry affects both modems.
- The **[autoanswer]** = Y line, when present, means that the answering modem will answer automatically; OTTO:3800 does not have to detect the RING message nor send the answer command [answer*]. There is no "a" or "b" appended; the entry affects both modems.

There are controls that affect timing for the various phases of call setup and teardown:

- The **[interchar-delay*]** line should be added in those rare cases where a particular modem cannot handle the default 2-millisecond intercharacter delay which the software inserts when transmitting a setup string. It defines, in milliseconds, the inter-character delay to be used for that particular modem; zero is acceptable.
- The **[setupdelay]** line defines the number of seconds the software is required to wait before sending the first AT command in a sequence. Default is no setup delay.

- The **[calldelay]** line specifies the number of seconds the software is required to wait before sending the DIAL command. Default is no call delay.
- The **[busy-delay]** line specifies the number of seconds the software is required to wait after seeing a BUSY or NO DIAL TONE response code from the modem before re–attempting the call. Default is 20 seconds. This is usually sufficient time for a router access server to recover from a modem disconnection.
- The **[answerdelay]** line specifies the number of seconds the software is required to wait before sending the ANSWER command after seeing the ring signal. Default is one second.
- The **[test-ms-delay]** line specifies the number of milliseconds the software is required to wait after receiving CONNECT response codes from both modems and seeing data carrier detect from both modems before starting the test. The default value is zero.
- The **[hangupdelay]** line defines the number of seconds the software is required to wait after V.24 circuit 109 (DCD, or RLSD) turns off. Some modems use a

connection cleardown procedure which cannot be disabled, so the [hangupdelay] parameter indicates how long after the call is completed to wait before starting the next call. There is no “a” or “b” appended; the entry affects both modems.

- The **[at-timeout]** line defines the number of seconds the software waits after sending an “AT <CR>” command for the “OK” response code before declaring an error. The default is two seconds.
- The **[commandtimeout]** line defines the number of seconds the software waits after sending an initialization command (ia*, ib*, or the test–specific initialization commands such as nmc-ec*) for the “OK” response code before declaring an error. The default is ten seconds.
- The **[dialtimeout]** line defines the number of seconds the software waits after sending the dial command [dial*] for the RING response code from the answering modem before declaring an error. The default is 20 seconds. This parameter is not used when [autoanswer] is set to Y.
- The **[answer-timeout]** line specifies the number of seconds the software waits for a complete connection (both modems reporting a CONNECT message, data carrier detect [DCD] asserted, and clear to send [CTS] asserted) before assuming the call has failed. The default is 45 seconds.
- The **[hangup-timeout]** line specifies the number of seconds the software waits for data carrier detect to be deasserted in response to DTR being turned off. No error is declared if this timeout expires. The default is 10 seconds.
- The **[retry-limit]** line specifies how many times a call will be attempted before the software gives up on the test and moves on. The default value is four. A value of zero or one tells the software not to retry failed connections. Note that a data mismatch or carrier loss counts as two tries, not one, to minimize the test time extension that such errors would otherwise cause.
- The **[busy-retry-limit]** line specifies how many times a call will be attempted when the modem reports BUSY or NO DIAL TONE before the software gives up on the test and moves on. This retry count is separate from the count in [retry-limit]. A value of zero or one tells the software not to retry failed connections.
- The **[at-retry-limit]** line specifies how many times the software will issue initialization strings before giving up on the test and moving on. A value of zero or one tells the software not to retry failed attempts.
    - Serial port tuning parameters are covered in a separate document.

TIA TSB–38 remains silent about many of the options available in modems today. Most users don’t understand the options, how to set them, and how setting them will affect modem performance. Henderson Communications Laboratories has established a policy that default values for options shall be used unless the default conflicts with requirements in the standard, or requirements dictated by OTTO:3800.

6.2 REF Files for V.34 Interoperability Testing

OTTO:3800 has been designed to permit testing between two modems of different make and model with a minimum of fuss. The modem definition file for the modem under test remains the same; the modem definition file for the reference modem is slightly different. A sample file “13-42.REF” is shown below:

```
; Reference modem definition file 13-42.REF
[barcode-ref] = 13.42
[product-ref] = Joes Bar and Modem V.34 (Rockwell RC288ACi)
;
[speeda]       = 115200
[i1a]          = AT &F M0 W1 &C1 &D2 &G0 \K1 \N4 %C2 -K0
[i2a]          =
[nmc-eca]      = AT %C0
[echo-directa] = AT \N1 %E0 +MS=11,0,28800,28800
[echo-normala] = \N0 AT %E0 +MS=11,0,28800,28800
[echo-eca]     = AT %C0 %E0 +MS=11,0,28800,28800
[echo-dca]     = AT %E0 +MS=11,0,28800,28800
```

The principal difference between the two types of modem is that only the “a”–suffix parameters are specified, and the product name and barcode number use a different label. The information in [barcode-ref] and [product-ref] are recorded in the results file.

To use reference files, you run OTTO:3800 in the following manner:

C:\OTTO: OTTO tvf.def 13-42.ref 17-02.mdm

The name of the reference modem file must come first, followed by the name of the UUT modem to be tested. If you reverse these names, the infomation in “13-42.ref” is masked, and the reference modem will be treated as though it were the same make and model of modem as the UUT modem.

6.3 REF Files for Ethernet–Connected Modem Testing

Please read this section in the OTTO:ESB manual for a description of the construction of REF files for Ethernet–connected modems.

6.4 MODEM SETUP OPTIONS AND SETTINGS SUMMARY

The following pages gives a summary of all modem command options and settings required by TSB–38 or OTTO:3800:

	(typical command)	TVF, CCR, DSC [i1]	NMC [nmc-ec]	CED1 [echo-direct]	CED2 [echo-normal]	CED3 [echo-ec]
TSB-38 mandated						
Mode		Cmprssn	**Err Ctrl**	**Direct**	**Buffrd**	**Err Ctrl**
Line rates active		all	all	**max**	**max**	**max**
Adaptive rate changes		on	on	**off**	**off**	**off**
Retrains		on	on	**off**	**off**	**off**
Hardware flow control		on	on	**off**	**off**	**off**
Required by OTTO:3800						
Factory reset	&f	yes	yes	yes	yes	yes
300-bps	b	103 type	103 type	103 type	103 type	103 type

	(typical command)	TVF, CCR, DSC [i1]	NMC [nmc-ec]	CED1 [echo-direct]	CED2 [echo-normal]	CED3 [echo-ec]	CED4 [echo-dc]
RTS follows CTS	&r	default	default	default	default	default	default
DSR always on	&s	yes	yes	yes	yes	yes	yes
Grant RDL	&t	default	default	default	default	default	default
Trellis coding	&u	on	on	on	on	on	on
Modem xmit clock	&x	default	default	default	default	default	default
User profile	&y	default	default	default	default	default	default
Stored phone number	&z	default	default	default	default	default	default
Max MNP block size	\a	default	default	default	default	default	default
Auto-reliable buffer	\c	default	default	default	default	default	default
DSR and CTS	\d	forced on	forced on	forced on	forced on	forced on	forced on
data echo	\e	off	off	off	off	off	off
DCE-to-DCE flow ctrl	\g	off	off	off	off	off	off
HP ENQ/ACK	\h	off	off	off	off	off	off
Interface protocol	\i	off	off	off	off	off	off
BPS adjust	\j	off	off	off	off	off	off
Break control	\k	expedited destructive	expedited destructive	expedited destructive	expedited destructive	expedited destructive	expedited destructive
MNP link type	\l	default	default	default	default	default	default
Mode	\n	Cmprssn	**Err Ctrl**	**Direct**	**Normal**	**Err Ctrl**	Cmprssn
RTS/CTS flow control	\q	yes	yes	**no**	**no**	**no**	**no**
RING indicator	\r	default	default	default	default	default	default
Inactivity timer	\t	off	off	off	off	off	off
EC result codes	\v	maximum	maximum	maximum	maximum	maximum	maximum
XON/XOFF passthru	\x	default	default	default	default	default	default
Autorel. fallback char	%a	default	default	default	default	default	default
Modem bps rate	%b	max	max	max	max	max	max
Compression (both ways)	%c	on	on	**off**	**off**	**off**	on
Disconnect delay	%d	2 sec	2 sec	2 sec	2 sec	2 sec	2 sec
Retrain	%e	on	on	**off**	**off**	**off**	**off**
V.23 mode	%f	off	off	off	off	off	off
Serial/modem speed	%g	serial	serial	serial	serial	serial	serial
Speed matching	%l	default	default	default	default	default	default
V.23 equalizer	%o	default	default	default	default	default	default

	(typical command)	TVF, CCR, DSC [i1]	NMC [nmc-ec]	CED1 [echo-direct]	CED2 [echo-normal]	CED3 [echo-ec]	CED4 [echo-dc]
Pulse digit comman	%w	default	default	default	default	default	default
Connect msg control	@c	default	default	default	default	default	default
V.42 detection phase	-j	on	on	on	on	on	on
MNP extended services	-k	default	default	default	default	default	default
Extended connect msgs	-m	on	on	on	on	on	on
Modulation fallback	-q	on	on	**off**	**off**	**off**	**off**
Auto answer	s0	off	off	off	off	off	off
carriage return	s2	default	default	default	default	default	default
line feed	s3	default	default	default	default	default	default
Allow 2-sec delay of dialtone	s6	yes	yes	yes	yes	yes	yes
DTMF timing	s11	default	default	default	default	default	default
DTR change detect time	s25	default	default	default	default	default	default
Inactivity timeout	s30	off	off	off	off	off	off
LAPM fallback	s36	hang up	hang up	hang up	hang up	hang up	hang up
Maximum DCE speed	s37	default	default	default	default	default	default
Hangup delay	s38	2 sec	2 sec	2 sec	2 sec	2 sec	2 sec
Error-control protocol	s46	Cmprssn	**Err Ctrl**	—	—	**Err Ctrl**	Cmprssn
Error control negotiation	s48	default	default	default	default	default	default
Break signalling	s82	expedited destructive	expedited destructive	expedited destructive	expedited destructive	expedited destructive	expedited destructive
Connect message options	s95 =45	maximum, DCE rate	maximum, DCE rate	maximum, DCE rate	maximum, DCE rate	maximum, DCE rate	maximum, DCE rate
Signal quality action threshold	s108	default	default	default	default	default	default
Calling tone	s113	on	on	on	on	on	on
Security		off	off	off	off	off	off
Remote configuration		off	off	off	off	off	off
Cellular options		default	default	default	default	default	default
File transfer protocol spoofing		off	off	off	off	off	off
All other options		default	default	default	default	default	default

7 TAS 3508 Modem Switch Support

7.1 Introduction

OTTO:3800 includes as part of its standard distribution support for the TAS Model 3508 Modem Test Switch. This switch is an eight–port switch that selects in tandem an RS–232 port and a telephone port. OTTO:3800's distribution supports two such switches, one to be connected to the A side of the network simulator and one to be connected to the B side of the network simulator.

The A–side switch can be used to select one of several reference modems, or in tandem with OTTO:ESB it can be used to perform TSB 37A testing and PN 3857 testing in one execution cycle.

The eighteen files contained in the standard distribution set a specified switch to a port. The following table describes the names of the files and the effect using the file has on a switch:

File	Effect	File	Effect
swa-1	set switch "a" to port 1	swb-1	set switch "b" to port 1
swa-2	set switch "a" to port 2	swb-2	set switch "b" to port 2
swa-3	set switch "a" to port 3	swb-3	set switch "b" to port 3
swa-4	set switch "a" to port 4	swb-4	set switch "b" to port 4
swa-5	set switch "a" to port 5	swb-5	set switch "b" to port 5
swa-6	set switch "a" to port 6	swb-6	set switch "b" to port 6
swa-7	set switch "a" to port 7	swb-7	set switch "b" to port 7
swa-8	set switch "a" to port 8	swb-8	set switch "b" to port 8
swa-cas	set switch "a" to cascade	swb-cas	set switch "b" to cascade

(The terms "switch a" and "switch b" refer to a switch connected to the TAS 240 and TAS Series 2 telephone port A and B respectively.)

7.2 Using the 3508 switch manually

The simplest use of the modem switch is to permit an operator to hook up a modem to be tested while another modem is already being tested. You use the modem switch commands from the console to switch to the next modem to be tested. The following is the series of commands that you might issue to effect manual testing:

```
netsim\netsim  netsim\swb-1
stdd   52-10.mdm  ras.ref
netsim\netsim  netsim\swb-2
stdd   52-11.mdm  ras.ref
```

The first command instructs the switch to select port 1. The second command launches a complete digital-modem (PCM) test suite execution for the modem identified as 52-10. The third line instructs the switch to select port 2. The fourth line launches a complete digital-modem (PCM) test suite execution for the modem identified as 52-11.

While it seems silly to key these commands manually, it is one way that the facility can be used to minimize delays in using the equipment.

7.3 Using the 3508 in automated testing

The more interesting use of the modem switch is to permit unattended multiple modem tests to be performed (such as in weekend testing). Taking the example above, the batch file needed to execute multiple tests looks like this:

```
rem batch file to run two modem tests unattended
rem using a TAS 3508 switch
command   /c   netsim\netsim  netsim\swb-1
command   /c   stdd   52-10.mdm  ras.ref
command   /c   netsim\netsim  netsim\swb-2
command   /c   stdd   52-11.mdm  ras.ref
```

(The reason you prefix each command with the string “command /c” is that some of the commands in OTTO are implemented in batch files – and some versions of DOS can’t handle one batch file calling another properly. This includes the versions of MS-DOS included with Windows 95 and Windows 98.)

The above will indeed perform the tests, but some diagnostic information might be lost if the first test encounters problems. To fix this problem, Test Automation Software suggests the batch file be augmented as follows:

```
rem   batch file to run two modem tests
rem   unattended using a TAS 3508 switch
del   *.xls
command   /c   netsim\netsim  netsim\swb-1
command   /c   stdd   52-10.mdm  ras.ref
mkdir    52-10
copy    *.xls  52-10
copy    log.txt   52-10
del   *.xls
command   /c   netsim\netsim  netsim\swb-2
command   /c   stdd   52-11.mdm  ras.ref
mkdir   52-11
copy   *.xls   52-11
copy   log.txt  52-11
rem    end batch file.
```

This ensures that only the latest version of the test is saved, so that old tests are automatically replaced.

If you have a pair of switches and performing analog testing (for V.34 and V.22 bis modems) the corresponding batch file might look like this:

```
rem   batch file to run two modem tests
rem   unattended using a TAS 3508 switch
del   *.xls
command   /c   netsim\netsim  netsim\swa-1  netsim\swb-1
command   /c   std   52-10.mdm  ras.ref
mkdir    52-10
copy    *.xls  52-10
copy    log.txt   52-10
del   *.xls
command   /c   netsim\netsim  netsim\swa-2  netsim\swb-2
command   /c   stdd   52-11.mdm  ras.ref
mkdir   52-11
copy   *.xls   52-11
copy   log.txt  52-11
rem    end batch file.
```

The only differences between this file and the one previously is that both A and B switches are set, and the “STD” (standard analog-modem suite) batch file is used instead of the “STDD” (standard digital-modem suite) batch file.

The standard switch lets you select one of up to nine modems. If you wish to cascade multiple switches, contact Test Automation Software for assistance.

8 Reference — OTTO:3800

8.1 Introduction

The OTTO:3800 program consists of three major sections: the parameter dictionary, the test engine, and the device interface. The dictionary, described in section 8.3 below, contains the global test parameters. The test engine is driven by the task description file, described in detail in section 8.4 below. The device interface consists of the serial port drivers and the associated buffer management routines.

8.2 Invocation

8.2.1 General

The following are the run commands for OTTO:3800:

OTTO <switch> ... <file> ...

The total length of the invocation line is limited by MS–DOS to 128 characters. There is no mechanism to logically extend the length of the invocation line.

If this limitation is exceeded, simply combine some or all of the files into a single file by using the COPY command of the form

COPY FILE1+FILE2+FILE3+FILE4 FILE5

The file FILE5 can then replace FILE1, FILE2, FILE3, and FILE4 in the invocation line. Any number of files can be so combined into a single file for the purposes of supplying information to OTTO:3800.

8.2.2 Switches

The switches are:

–b	Batch mode (start test immediately, exit when done
–n	Don't read OTTO.DEF
–w	Condition the serial drivers to work just like Windows driver.

Each switch must be specified separately with its own flag. If you are using the –b and –w flags together in the command line, they must be coded as "–b –w" and not "–bw".

8.2.3 File Names

OTTO:3800 will read the files listed in the command line from left to right. When all the files have been read (and if the –n switch wasn't specified) OTTO:3800 then reads the file "OTTO.DEF" from the current directory.

> NOTE: If you combine files as described above, the files FILE1 through FILE4 are concantinated into FILE5 in left–to–right order as well.

There is no restriction to the names of the files used, other that the restrictions on file names placed on the program by MS–DOS. File names must consist of letters, digits, and the following graphics characters:

~ ! @ $ % ^ * ( ) - _ { } [ ] < >

NOTE: use of the special characters listed above is discouraged.

In general, the OTTO program recognizes the standard file path convention, which consists of the following:

[<drive> :] [\] [<directory> [. <ext>] \] <name> [. <ext>]

where <drive> is a single alphabetic character, <directory> (and any associated <ext>) identifies the directory to use, and <name> (and any associated <ext>) identifies the file to use. The program recognizes the two special directories "." and ".." and will process them appropriately. The total length of a file name is restricted to 64 characters by MS-DOS.

In the standard distribution of this program, the types of files that should be included in the invocation line have the extensions .MDM, .REF, and .DEF, referring to modem definition files, reference definition files, and test task definition files. Files in the standard distribution with the extension .TBL should *not* be included in the invocation line, but instead should be mentioned within a test definition file as part of the **[taskfile]** dictionary entry. No reference should be made to .PGM (OTTO:POST post–processing scripts), .XLS (raw log files and summary files), or .XLT (Excel templates for report generation).

8.2.4 Batch mode return codes

If the –b switch wasn't specified, OTTO:3800 enters into terminal–emulation mode; otherwise it goes directly to test mode. When batch mode (–b) is specified, the error code returned by the program is one of the following:

0	Normal return
1	Operater aborted test
2	Operator pressed CTRL-BREAK
3	Bad specification in definition files
4	Missing file
5	Serial port failure or internal error

8.3 Dictionary

The dictionary is created by OTTO:3800 at program initiation and the information in the dictionary comes from the contents of the files named on the invocation line. The data is stored in the dictionary as name/value pairs. Information contained in the dictionary is extracted by referencing the symbol, and the value replaces the symbol.

8.3.1 General

Each file, called a definition file elsewhere in this document, consists of one or more lines of ASCII text separated by a newline sequence CR-LF. Each line defines a dictionary entry, or it can be blank, or consist solely of a comment as indicated by the semi-colon character as the first non-space character on the line.

Each dictionary entry consists of a symbol followed by its definition. The general form for a dictionary entry is:

[symbol] = value ; comment

where “symbol” is any string of 18 or fewer printable non-space characters enclosed within square brackets, and value is any string containing printable characters other than a semicolon. Characters between the first semicolon on the line and the end of the line are ignored. If there is no semicolon after the equal sign, then all characters up to the end of line are used as the definition.

Unprintable characters are silently dropped.

Any string of white space (spaces and tabs) within “value” are converted to a single space before storing the value in the dictionary.

Optional white space before and after the equal sign, and white space before the semicolon or end of line, are removed before the value is stored in the dictionary.

If there are no non–whitespace characters between the equal sign and either the semicolon introducing the comment or the end of line, the value of the symbol is the empty string.

8.3.2 Special symbols

OTTO:3800 has a small number of symbols with specific meanings within the program. The following table describes those symbols and the allowed values. Unless otherwise indicated, the maximum length of the value is 80 characters. The order of the symbols correspond to symbol classes within the OTTO:3800 program, and are placed here in that order to facilitate maintaining the list. See the note at the end of the section regarding those symbols marked with an asterisk (*).

[stripparity]	when value is “y” strips 8th bit from all modem responses
[autoanswer]	when value is “y” suppresses [answer*] output
[loopback]	when value is “y” suppresses initialization and dialling
[aggressivenessa]	reference serial port tuning tool
[aggressivenessb]	UUT serial port tuning tool

[port1adrs]	hexidecimal port address for reference modem
[port2adrs]	hexidecimal port address for UUT modem
[port1irq]	interrupt line for reference modem
[port2irq]	interrupt line for UUT modem
[overspeeda]	reference serial port tuning tool
[overspeedb]	UUT serial port tuning tool
[retry-limit]	Maximum attempts after connection
[busy-retry-limit]	Maximum attempts before connection
[at-retry-limit]	Maximum attempts to send AT<CR>
[speeda]	DTE/DCE interface speed for reference modem
[speedb]	DTE/DCE interface speed for UUT modem
[interchardelaya]	intercharacter dial delay for reference modem in ms.
[interchardelayb]	intercharacter dial delay for UUT modem in ms.
[test-ms-delay]	delay in ms between CONNECTs and test start
[setupdelay]*	delay before first initialization AT command sent
[calldelay]*	delay before dial string command sent
[busy-delay]*	delay before retry on BUSY or NO DIAL TONE
[answerdelay]*	delay between RING and answer command sent
[hangupdelay]*	Delay after hangup request to ensure modems on–hook
[fault-delay]	Delay from detection of error until call teardown
[at-timeout]*	Timeout for AT<CR> command
[commandtimeout]*	Timeout for AT…<CR> initialization commands
[dialtimeout]*	Timeout from [dial*] command to RING response code
[answertimeout]*	Timeout from [answer*] to CONNECT response codes
[hanguptimeout]*	Timeout from DTR drop to DCD drop
[diala]	command string to dial out on the reference modem
[dialb]	command string to dial out on the UUT modem
[answera]	command string to answer reference modem
[answerb]	command string to answer UUT modem
[taskfile]	path and name of test task description file, up to 64 characters
[barcode]	product id code, limited to 14 characters
[barcode-ref]	product id code of reference modem; limited to 14 characters
[product]	product description

[product-ref]	product description of reference modem
[stand]	identification of the test stand, project, or other information
[sumpath]	(concatenated with “[sumfile]”)
[sumfile]	path and file name for summary (results) file
[i#a]	(# from 0 to 9) setup strings for reference modem
[i#b]	(# from 0 to 9) setup strings for UUT modem

Items marked with an asterisk (*) above have three different symbols: the unadorned symbol (such as [setupdelay]), the symbol associated with the reference modem (such as [setupdelaya]). Amd the symbol associated with the UUT modem (such as [setupdelayb]). Within OTTO:3800, the maximum value of the three symbols is used for that parameter. This permits separate values to be specified for delays and timeouts for specific modems, and OTTO:3800 will select the maximum value at run time.

8.3.3 Treatment of multiple definitions

Definition files are read in the order they appear, left to right, in the invocation line for OTTO:3800. For a given definition, the first definition applies and any symbol redefinition is silently ignored. For example, given two definition files a.def and b.def:

a.def:

```
[speeda]  = 57600
```

b.def:

```
[speeda] = 115200
```

and the invocation line “OTTO A.DEF B.DEF” the parameter [speeda] would be 57600; given the invocation line “OTTO B.DEF A.DEF” the parameter [speeda] would be 115200.

There are no restrictions on the symbols which may be defined. There is a limit to the amount of memory available for the dictionaries, however.

Badly formed dictionary entries are silently ignored.

8.4 Task description file

The task description file contains the testing steps to be performed for a given test sequence. The file contains zero or more tasks to perform, each task described in one line. Any number of tasks may be included in the file.

The following is the “TVF.TBL” file from OTTO:3800

```
; TVF.TBL
s18c2, run      0,  netsim\netsim.exe - netsim\reset netsim\18c2
s18c2, rerun    0,  netsim\netsim.exe - netsim\reset netsim\18c2
s18c2, rerun    0,  alarm.exe
s18c2, rerun    0,  netsim\netsim.exe - netsim\reset netsim\18c2
```

s18c2, rerun 0, log.exe ***tvf_18c2_failed

10, throughput * *, eia\1.tst 30 16384, eia\0.tst 0 16384, 150

01, throughput * *, eia\0.tst 0 16384, eia\1.tst 30 16384, 150

11, throughput * *, eia\1.tst 30 16384, eia\1.tst 30 16384, 150

20, throughput * *, eia\2.tst 10 16384, eia\0.tst 0 16384, 150

02, throughput * *, eia\0.tst 0 16384, eia\2.tst 10 16384, 150

22, throughput * *, eia\2.tst 10 16384, eia\2.tst 10 16384, 150

30, throughput * *, eia\3.tst 6 16384, eia\0.tst 0 16384, 150

03, throughput * *, eia\0.tst 0 16384, eia\3.tst 6 16384, 150

33, throughput * *, eia\3.tst 6 16384, eia\3.tst 6 16384, 150

40, throughput * *, eia\4.tst 4 16384, eia\0.tst 0 16384, 150

04, throughput * *, eia\0.tst 0 16384, eia\4.tst 4 16384, 150

44, throughput * *, eia\4.tst 4 16384, eia\4.tst 4 16384, 150

50, throughput * *, eia\5.tst 16 16384, eia\0.tst 0 16384, 150

05, throughput * *, eia\0.tst 0 16384, eia\5.tst 16 16384, 150

55, throughput * *, eia\5.tst 16 16384, eia\5.tst 16 16384, 150

The general format of a task table entry is:

```
<id>  <tasktype>  <p>  <p>  <p>  ...  <p> ; <comment>
```

where **<id>** is the 1–to–16–character label used to identify the particular test run, **<tasktype>** is the type of test to run (see table below), and the **<p>** fields represent parameters. The **<comment>** field and the semicolon preceeding it is optional.

Fields in the task description entry may be separated by spaces, by commas, or by commas surrounded with white space. Empty parameters must be coded with two commas (although no test to date allows optional parameters).

For those tests which dial the modem, the general format is extended to:

```
<id>  <tasktype>  <mode>  <DTErate>  <p>  <p>  ... <p> ; <comment>
```

where **<mode>** is either "*" or the root name of the two dictionary entries to set up the modems into specific modes, and **<DTErate>** is either "*" or a DTE rate to use with the modems.

A <mode> of "echo–direct" will direct the dialer to issue the contents of "[echo–directa]" to the reference modem and the contents of "[echo–directb]" to the UUT modem when sending initialization strings in response to an INIT test or a dial function.

If the <DTErate> specified is higher than the rate specified in the [speed*] entry in the modem definition file(s), the lower rate is used.

The following is the table of valid test ID names:

init	Performs a modem initialization only.
i_scan	Performs 256 ATI
dial_only	Initializes and dials the modem, then disconnects.
throughput	Performs a throughput test
mer	Performs a message-error-rate test
call_ratio	Performs a call connection ratio test
char_echo	Performs a chracter-echo delay test
block_ack	Performs a block-acknowlegement delay test
measure_delay	Measures the inherent delay in a modem (diagnostic)
run	Runs an external program
rerun	Runs an external program after a prior run fails
fdial	Forward dial (A dial, B answer)
rdial	Reverse dial (B dial, A answer)

8.4.1 INIT

This module sets up the modem, but does not do any dialing. No reports are written to the summary file.

This is useful to initialize modems to a known value before starting tests, and can prevent “NO DSR” reports during regular testing by ensuring that DSR is strapped to be high.

<id> init <mode> <DTE>

<mode> “*” or test–specific setup string parameter name

<DTE> serial port speed or "*"

8.4.2 DIAL ONLY

This module dials a call and then hangs up. No records are written to the summary file.

<id> dial_only <mode> <DTEspeed>

<mode> “*” or test–specific setup string parameter name

<DTE> serial port speed or "*"

8.4.3 CHARACTER ECHO DELAY

This module performs a character echo delay test. A random character is transmitted from A to B, and the transit time measured. The same character is then transmitted from B to A, and the transit time measured. The two transit times are then added to give the round-trip time. The program captures the minimum, average, and maximum of each of these three times.

Then the program will delay for a random time depending on the settings of <min>, <nom>, and <max> such that the average of all delay times is close to the <nom> parameter.

<id>	char_echo <mode> <DTE> <t> <min> <nom> <max>

<mode>	"*" or test–specific setup string parameter name
<DTE>	serial port speed or "*"
<t>	number of character-echo tests, 1 to 10,000
<min>	minimum delay between characters, 5-1000 ms
<nom>	nominal delay between characters, 5-1000 ms
<max>	maximum delay between characters, 5-1000 ms

8.4.4 ACKNOWLEDGEMENT DELAY

This module performs the block–acknowledgment delay test. A block of 133 random characters is transmitted from A to B, and the transit time for the first and last character of the block is measured. A single character with value 6 (ASCII ACK) is then transmitted from B to A and the transit time measured. The last-character transit time and the return-character transit time are then added together to form the round-trip time. The program captures the minimum, average, and maximum of each of these four times.

Unlike the character-echo delay test, there is a minimum delay between trials.

<id>	block_ack <mode> <DTE> <t>

<mode>	"*" or test–specific setup string parameter name
<DTE>	serial port speed or "*"
<t>	number of character-echo tests, 1 to 10,000

8.4.5 DELAY

This measures character delay from modem A to modem B. No report is written to the summary file.

<id>	block_ack <mode> <DTE> <t>

<mode> “*” or test–specific setup string parameter name

<DTE> serial port speed or "*"

<t> number of character-echo tests, 1 to 10,000

8.4.6 MESSAGE ERROR RATE

This module performs a message-error-rate test. After an initial five-second delay, the FOX message is transmitted from A to B. If the message is received at B properly, the count of good blocks is incremented. Then the program will delay for <d> or (260000/DCErate + 3) milliseconds, whichever is greater before sending the next block.

<id>	char_echo <mode> <DTE> <t> <d>

<mode> “*” or test–specific setup string parameter name

<DTE> serial port speed or "*"

<t> number of messages, 1000 to 1,000,000

<d> delay between messages, 0-1000 ms

8.4.7 CALL CONNECT RATIO

This module performs a call-connect reliability test. This test outputs the modem initialization string once. For each call the command “AT” is sent to the answering modem before the dial command is sent to the dialing modem.

<id>	call_ratio <mode> <DTE> <c> <t> <d> <max>

<mode> “*” or test–specific setup string parameter name

<DTE> serial port speed or "*"

<c> number of calls to make, 1 to 1000

<t> number of messages per call, 1 to 10,000

<d> delay between messages, 0-1000 ms

<max> timeout for FOX message receipt, 1500-120000 ms

8.4.8 RUN/RERUN

These tests execute external DOS programs as required to set up a test. A "run" entry is always processed; a "rerun" entry is processed only if the previous “run” or “rerun” command failed. If a RUN/RERUN sequence fails to execute properly, then all tests between the last RERUN and the next RUN or RERUN command are skipped.

If the program does not return a proper error code, then coding "*" for the <errorlevel> field will force this package to accept the result of the program without question. Coding a positive number in this field means that any non-negative return code less than or equal to the number coded indicates a successful completion.

```
<id> run      <errorlvl> <exe> <p1> <p2> ... <p9>
<id> rerun    <errorlvl> <exe> <p1> <p2> ... <p9>
```

<errorlvl> maximum allowed error code to be returned by the running program to be considered a successful run

<exe> name of the program to run. The full path and the extension must be supplied, such as “NETSIM\NETSIM.EXE”

<p1> first parameter

<p2> second parameter, and so forth

8.4.9 THROUGHPUT

This module performs a complete throughput test.

```
<id> throughput <mode> <DTE> <n1> <r1> <h1> < n2> <r2> <h2> <t> <h>
```

<mode> “*” or test–specific setup string parameter name

<DTE> serial port speed or "*"

<n1> A-to-B file segment path and name

<r1> A-to-B repeat count, 0-32767

<h1> A-to-B histogram increment, 64-65535

<n2> B-to-A file segment path and name

<r2> B-to-A repeat count, 0-32767

<h2> B-to-A histogram increment, 64-65535

<t> test timeout, 10-5000 seconds

<h> Histogram level, 0–4.

8.4.10 ATI Scan

This module issue 256 ATI commands (ATI0 through ATI255) and records the results of that scan.

```
<id>      i_scan  *  <DTE>
```

<DTE> serial port speed or "*"

8.4.11 Forward Dial

This module tells the dialing processor that, if there is both a DIALA/ANSWERB pair and a DIALB/ANSWERA pair, to use the DIALA/ANSWERB pair to place the connection.

<id>	fdial

8.4.12 Reverse Dial

This module tells the dialing processor that, if there is both a DIALA/ANSWERB pair and a DIALB/ANSWERA pair, to use the DIALB/ANSWERA pair to place the connection.

<id>	fdial

8.5 Dialing parameters

8.5.1 General

The built-in automatic dialer uses a mixture of RS-232 lead control and AT commands to control the operation of the modem. Commands are specified in the dictionary; responses are hard-coded in the program.

The speed of the serial port is set up from the dictionary entries [speeda] and [speedb]. Before the dialer begins to do its work, it looks to see if the <DTE> field in the task descriptor file record contains an asterisk ("*") character; if so, it leaves the DCE/DTE speed of the modems alone. Otherwise it sets both modems to the speed indicated, or to the speed in the [speed*] entry, whichever is lower, before starting the modem initialization sequence.

In the special-symbol table, there are two entries for dialing and for answering. The testing standards call for modem A to dial and modem B to answer; in some testing situations these need to be reversed. To keep dialing flexible, the test designer can specify whether the A modem answers or dials by using either [dialb][answera] or [diala][answerb] respectively. If all four strings are present in the parameter dictionary, then the settings of the "fdial" and "rdial" tests are used. If no "fdial" or "rdial" commands have been issued, "fdial" is assumed and [diala] is used.

The dialer will output the initialization strings to both modems. The initialization strings are output alternating UUT and REF, and in increasing order of init string symbol number (I1A, I2A, I3A, and so forth). After the initialization strings are output, any test-specific initialization string (described in the next section) is output to the modems. Then the dial command is output. When OTTO:3800 sees a RING indication, it then sends the answer command. After receiving CONNECT messages from both modems and DCD is asserted by both modems, the connection is considered complete.

> NOTE: If [autoanswer] is "y" then the detection of RING status and the output of the [answer*] command is disabled.

Normally, initialization commands wait until an OK is returned from the modem. If another response (like ERROR) is seen, or no response is seen within 10 seconds, the command is assumed to be in error and is retried at most four times.

If an initialization string begins with a tilda ("~") the dialer does not expect to see an OK from the modem after the command has been output. This is useful for certain modems with questionable autobaud routines; the command "~AA" will send two "A"s followed by a return. A subsequent "AT" should then work properly.

8.5.2 AT Command String Output Details

Each string is output in a uniform manner. Output of space characters is suppressed, although the spaces are counted in the first characters to be delayed.

The string output routine ensures that no data is sent sooner than 250 milliseconds after the last character was received by the modem. This is mandated by the TIA/EIA 602 standards, although in OTTO:3800 the delay is 250 milliseconds instead of the 100 milliseconds specified in that standard.

When this initial command–output delay is satisfied, the first three characters are sent with 110 milliseconds of mark–idle separating them. If any of the first three characters are space codes, both the output of the space and the delay associated with the code are removed.

The fourth and subsequent characters of the string are sent with at least [intercharacterdelaya] or [intercharacterdelayb] milliseconds of mark_idle separating them.

> NOTE — The default value for the intercharacter delay parameters are "2". If a value of zero is specified, the stop bit for the character is extended by a fraction of a millisecond.

A carriage return (decimal value 13) is appended to the end of the initialization string, and that carriage return is sent after a 250–ms delay.

For example, the command string "AT &F &C1" results in the following characters being sent with the timing shown:

<250> 'A' <110> 'T' <110> '&' <2> 'F' <2> '&' <2> 'C' <2> '1' <250> <S2>

where <250>, <110>, and <2> refer to the delay in milliseconds, the characters in single–quotes are themselves, and <S2> refers to the value of the contents of strapping register S2, which for XFR is a carriage return (decimal 13, hex 0x0D).

The command string "~AAAA" results in the following characters being sent with the timing shown:

<250> 'A' <110> 'A' <110> 'A' <2> 'A' <250> <S2>

8.5.3 The <mode> task description file entry field

The <mode> field of the task description file entry is used when a test-specific modem setup is required. This facility is required, for example, when running the character delay tests which must be run in four different modes: direct, normal, error control, and data compression.

After all the initialization strings in the dictionary have been output, the dialer will look at the <mode> field to see if it is coded as an asterisk ("*") character; if so, it skips test-specific initialization. Otherwise it takes the contents of the <mode> field, appends a "b", and looks up the resulting symbol in the dictionary. The value from that dictionary entry is then output to the UUT modem. It then takes the contents of the <mode> field, appends an "a", and looks up the resulting symbol in the dictionary. The value from that dictionary entry is then output to the reference modem.

Example: the task descriptor file record is "test1 char_delay echo-direct 14400". The dictionary will need definitions for the symbols "[echo-directa]" and "[echo-directb]" in order for the test to run. The value for these symbols would be the commands required to place the modems in direct mode for echo tests.

8.6 OUTPUT FILE FORMAT — General

The output file created by OTTO:3800 consists of a number of records intended to be imported into a database or spreadsheet program. Each record consists of a series of fields separated by an ASCII tab character.

Each record shares a common opening format: barcode, followed by the ID from the task description file record, and a tag. The remaining fields are dependent upon the tag.

The first three sections describe those portions of the results file common to all tests. The remainder of the sections describe those portions of the results file which are specific to certain test types.

8.6.1 Preamble

For every test which saves a report in the file, the following preamble is written. If a particular record outputs multiple values after the tag, each value appears on a separate line; see PIP for an example. Note that the tags marked with asterisks ("*") are suppressed when there is no information for that record. Example: if there is no reference barcode—no definition of [barcode-ref] in the dictionary—no record with tag BRF is outout.

<barcode> <id> --- <OTTO:3800 version>
<barcode> <id> TST <testid mode DTE>

<barcode> <id> PIP <date> <time>
<barcode> <id> NAM <[product] setting>
<barcode> <id> BRF* <[barcode-ref] setting>
<barcode> <id> NRF* <[product-ref] setting>
<barcode> <id> EQU* <[stand] setting>
<barcode> <id> LBK* <[loopback] setting>
<barcode> <id> PAR* <[stripparity] setting>
<barcode> <id> DIA* <[diala] setting>
<barcode> <id> ANA* <[answera] setting>
<barcode> <id> I#A* <[i#a] setting for each # defined>
<barcode> <id> ITA* <setting for mode setup>
<barcode> <id> DIB* <[dialb] setting>
<barcode> <id> ANB* <[answerb] setting>
<barcode> <id> I#B* <[i#b] setting for each # defined>
<barcode> <id> ITB* <setting for mode setup>
<barcode> <id> LOG* <message, repeated for each message>

LOG entries are suppressed for call-connect ratio testing from each of the individual calls for no connection and lost-connection messages.

8.6.2 Postamble

The standard postamble consists of a single line:

<barcode> <id> END -30-

8.6.3 Dialling reports

For each test which performs a single dialling operation as part of the test, the following records are output:

<barcode> <id> ERR* <error-code> <error-class-code>
<barcode> <id> CMA* <connect message reported by modem A>
<barcode> <id> CMB* <connect message reported by modem B>
<barcode> <id> CDL* <dial-to-answer delay> <answer-to-connect delay>

The error code is specific to the particular version of OTTO:3800 being run. It is a combination of a mark_error code, a step number, and reports from the serial engine. These change as OTTO:3800 changes, so no table of codes are documented here. The error class code, is defined in the following table:

0 No error

1 OTTO parameter setup error

2 Modem command error

3 Modem error on ATI command

4 No–ring timeout

5 No–connection timeout

6 Test stopped because of modem carrier loss

The CDL (call-delays) record contains two measurements: the time in milliseconds from the end of the dial command to the receipt of RING indication; and the time in milliseconds from the end of the answer command to the receipt of complete connection information from both modems. The answer-to-connect delay can be used to determine if an extended connection sequence occurred during a test.

CDL is not output for call-connect ratio tests. Instead, a summary of the answer-to-connect delay is output as part of the test results.

8.7 OUTPUT FILE FORMAT — Test specific

In each of these record formats, the <barcode> field is “field 0”, the <id> field is “field 1”, the tag is “field 2”, and the parameters following start at “field 3” for the purposes of writing OTTO:POST scripts.

8.7.1 CHARACTER ECHO DELAY

<barcode> <id> INT <specified minimum delay, ms>
<specified nominal delay, ms>
<specified maximum delay, ms>
<actual minimum delay, ms>
<actual average delay, ms>
<actual maximum delay, ms>

<barcode> <id> CED <measurement attempts made>
<successful measurements>
<A-to-B minimum delay, 0.1ms>
<A-to-B average delay, 0.1ms>
<A-to-B maximum delay, 0.1ms>
<B-to-A minimum delay, 0.1ms>
<B-to-A average delay, 0.1ms>
<B-to-A maximum delay, 0.1ms>
<round-trip minimum delay, 0.1ms>
<round-trip average delay, 0.1ms>
<round-trip maximum delay, 0.1ms>

All intervals in the INT record are in milliseconds. All intervals in the CED record are in 0.1-millisecond increments.

8.7.2 BLOCK ACKNOWLEDGMENT DELAY

<barcode> <id> BAK <measurement attempts made>
<successful measurements>
<A-to-B 1st-char minimum delay, 0.1ms>
<A-to-B 1st-char average delay, 0.1ms>
<A-to-B 1st-char maximum delay, 0.1ms>
<A-to-B last-char minimum delay, 0.1ms>
<A-to-B last-char average delay, 0.1ms>
<A-to-B last-char maximum delay, 0.1ms>
<B-to-A minimum delay, 0.1ms>
<B-to-A average delay, 0.1ms>
<B-to-A maximum delay, 0.1ms>
<round-trip minimum delay, 0.1ms>
<round-trip average delay, 0.1ms>
<round-trip maximum delay, 0.1ms>

8.7.3 MESSAGE-ERROR RATE

<barcode> <id> MER <measurement attempts made>
<successful measurements>

The message-error rate is expressed in terms of the ratio of bad attempts to total number of attempts. To avoid problems with floating-point values in the database the raw information is reported instead.

8.7.4 CALL CONNECT RELIABILITY

<barcode> <id> CON <connect speed>
<number of connections at this speed>
<minimum answer-to-connect delay, ms>
<average answer-to-connect delay, ms>
<maximum answer-to-connect delay, ms>

<barcode> <id> CCR <number of call attempts>
<number of successful calls>
<minimum answer-to-connect delay, ms>
<average answer-to-connect delay, ms>
<maximum answer-to-connect delay, ms>

Each CON record shows the call-delay measurements for calls which connect at the speed contained in the record; there may be as many as 20 of these records. The CCR call-delay measurements are for all the calls, regardless of speed.

8.7.5 THROUGHPUT

<barcode> <id> XFR <throughput time limit, seconds>
<actual transfer time, seconds>
<reference modem DTE/DCE interface speed>
<UUT modem DTE/DCE interface speed>
<UUT modem transmit byte count>
<reference modem receive byte count>
<reference modem transmit byte count>
<UUT modem receive byte count>

<barcode> <id> FLO <reference modem transmit pace count>
<reference modem maximum transmit stall, ms>
<UUT modem transmit pace count>
<UUT modem maximum transmit stall, ms>
<tuning parameters (documented separately)>

<barcode> <id> CPS <reference modem DCE rate reported by modem>
<UUT modem DCE rate reported by modem>
<B-to-A modem throughput, char/s>
<A-to-B modem throughput, char/s>

8.7.6 ATI SCAN

<barcode> <id> ATI <report>

There may be literally dozens of these records, one for each response from the 256 ATI commands issued by OTTO:3800 to the modem. Only those responses contain intermediate responses are saved; ATI responses that consist of OK and ERROR are not recorded. The report is in the format “ATI#=text”.

9 Reference — NETSIM

9.1 Introduction

The NETSIM program controls the network simulator equipment attached to the OTTO:3800 computer via IEEE–488 connections. The purpose of this utility is to output setup commands on demand to configure the equipment for each test.

The settings are distributed into a number of files, each file named according to the particular set of impairments and configurations needed in testing.

The program is usually invoked from an OTTO:3800 test task definition file (ending with the .TBL extension) as the target of RUN and RERUN commands.

9.2 Invocation

The program is launched in the following manner:

NETSIM [-] file … [> logfile]

Where the lone hyphen indicates to reset the IEEE–488 bus (not functional in the TAS version of NETSIM) and the list of files indicate what command files to use to configure the network simulator.

MS–DOS imposes a limit of 128 characters on the invocation line. When invoked from OTTO:3800, the only limitation is the nine-parameter limit imposed by the RUN and RERUN commands in OTTO:3800.

The redirection of standard output to a file works only from the MS–DOS command line. When running from OTTO:3800, the standard output from OTTO:3800 is inherited by NETSIM. When such a redirection is in effect, output to the operator is echoed to the log as well, providing a complete document of any relevant information that affected the outcome of execution.

9.3 File Names

NETSIM will read the files listed in the command line from left to right. When all the files have been read NETSIM then reads and interprets the file “ZTAS.DEF” from the current directory as an IEEE–488 symbol dictionary.

There is no restriction to the names of the files used, other that the restrictions on file names placed on the program by MS–DOS. File names must consist of letters, digits, and the following graphics characters:

~ ! @ $ % ^ * ( ) - _ { } [ ] < >

NOTE: use of the special characters listed above is discouraged.

In general, the NETSIM program recognizes the standard file path convention, which consists of the following:

[<drive> :] [\] [<directory> [. <ext>] \] <name> [. <ext>]

where <drive> is a single alphabetic character, <directory> (and any associated <ext>) identifies the directory to use, and <name> (and any associated <ext>) identifies the file to use. The program recognizes the two special directories "." and ".." and will process them appropriately. The total length of a file name is restricted to 64 characters by MS-DOS.

9.4 File formats

Each file contains a series of ASCII text lines. The lines fall into three classifications: comments, IEEE–488 exchanges, and special operations. The rest of this section discusses each of these types of lines

9.4.1 Comments

A comment line consists of a semicolon as the first non–whitespace character on the line. Everything from the semicolon to the end of line is ignored. This is used to document information in a settings file.

A line with no non–whitespace characters (a blank line) is also treated as a comment.

9.4.2 IEEE–488 exchange

An IEEE–488 exchange consists of text to be sent to a specific device, and text that corresponds to the expected response from the device. The line consists of the following format:

<device> | <sendtext> | <responsetext> | ; <comment>

where <device> is a decimal number representing the IEEE–488 address to use *or* a symbolic address defined in ZTAS.DEF, <sendtext> is the text to be transmitted to the device described by <device>, and <responsetext> is the expected response from the device.

In many cases, the response from the device is informational and not a confirmation. Good examples are asking for the revision level of the device. By leaving <responsetext> blank and coding the two vertical–bar symbols together, this tells NETSIM to output the response to the operator and to the log (if there is one open). The output in that instance is the first 24 characters of the command, the text " -> ", and the first 24 characters from the device.

For examples of a standard NETSIM file, see section 9.6.2 below.

9.4.3 Special operations

The following are the possible special operations commands:

* <text>

Causes the entire line to be output to the operator and to the log (if there is one open). Used to record the name and version of settings files used in a particular test run.

#d <seconds>

Causes NETSIM to delay for the specified number of second.

#i <filename>

Causes NETSIM to search for the specified file and read it. This read occurs at the point where this command is encountered. Processing of the file with this command picks up with the line following the directive.

> NOTE: Watch for directory conflicts. The path for the file, if relative, starts with the currently selected directory and not the directory in which NETSIM or the invoking file is located. For example, if running NETSIM from OTTO:3800 the directive "#I channel" would include the file "channel" from the \OTTO directory, not the \OTTO\NETSIM directory as you might expect.

9.5 Files provided with OTTO:3800

The NETSIM files provided with OTTO:3800 to support the TAS Series 2 Plus line of network simulators is as follows:

17C1 through 24C7 — TSB 37A test channel settings
EIA0 through EIA7 — TAS 240 settings for PN 3857 testing
I01A through I16D — PN 3857 trunk impairment combination settings
TLC0 through TLC7 — TAS 240 settings for TSB 37A test loop combinations
SWA-* and SWB-* — TAS 3508 switch settings
T1-D4 — Channel file for T1 channels using D4 framing
T1-ESF — Channel file for T1 channels using ESF framing
E1 — Channel file for E1 channels
BRI — Channel file for basic–rate ISDN channels
CHANNEL — Channel file used by all digital settings files
TEMPLATE — Channel file that the user can modify to taste
RESETD — Command file to fully reset the Series 2
RESET37A — Command file to fully reset the Series 2

For the TAS Series 2 Digital line of network simulators, the following files are also included:

U01A through U12D — PN 3857 settings for PN 3857 testing
PRI-D4 — Channel file for primary–rate ISDN channels using D4 framing
PRI-ESF — Channel file for primary–rate ISDN channels using ESF framing

9.6 Configuring NETSIM — General

Before NETSIM can be used, the file ZTAS.DEF must be set. See section 2.8.1 above for instructions.

9.6.1 Configuring NETSIM — TSB 37A testing

There is no special setup considerations when running TSB 37A testing.

9.6.2 Configuring NETSIM — PN 3857 testing

Before you can perform tests using PN 3857, you need to configure the file CHANNEL in the \OTTO\NETSIM directory to tell the TAS Series 2 exactly what kind of digital device you have. As a starting point, determine which of the standard files T1-D4, T1-ESF, E1, BRI, PRI-D4, or PRI-ESF most closely matches the configuration of your digitally–connected device. Issue the following command in an MS–DOS window while in the C:\OTTO\NETSIM directory:

COPY <selected-file> CHANNEL

From there, use the EDIT utility to change the configuation of the CHANNEL file to fit your needs. The following is an example of a CHANNEL file without any customization at all:

```
* channel   Digital Interface Channel Set   04/06/99
;
;>>TO DO:  Edit this file (using any ASCII editor such as the MS-DOS
;  EDIT editor) to indicate the test configuration you want to use.
;  This indication is made by removing the leading semicolon from the
;  line that contains the command that sets the TAS Series II into the
;  correct mode.
;
;  When the instructions say "removing the leading semicolon from the
;  line indicated", be sure to remove the semicolon at the left of
;  the line, and not the one in the middle of the line.
;
;<--remove this semicolon,        not this one -> ; uncomment for...
;
;  For examples of proper settings, see the files T1-D4, T1-ESF, E1,
;  BRI, PRI-D4, PRI-ESF, and PRI-E1
;
;  Purpose: This file is used to set a number of digital interface
;  parameters that are (arguably) improperly cleared by the
;  TAS Z1 reset command.  This file is read by every test
;  channel file after performing the IO:Z1 command and before
;  setting anything else.
;
;  Operation:  this file first sets a red-alarm condition in
;  the digitally-connected telephone device by purposely setting
;  parameters wrong and then waiting three seconds for the digital
;  circuitry to notice the distruption.  It then sets everything
;  right and waits eight seconds for the circuitry to see that
;  all is better.  In the process, the various modes are properly
;  selected.
;
;  SET RED ALARM:  uncomment (remove the leading semicolon) from the line
;  as indicated.
;
;NS|/DINT:FF=1/|/C/|        ; uncomment for T1 D4/SF
;NS|/DINT:FF=0/|/C/|        ; uncomment for T1 ESF
;NS|/DINT:FF=0/|/C/|        ; uncomment for E1
;NS|/DINT:FF=0/|/C/|        ; uncomment for TDM
;NS|/DINT:FF=0/|/C/|        ; uncomment for BRI
;NS|/DINT:FF=2/|/C/|        ; uncomment for PRI T1 D4/SF
;NS|/DINT:FF=2/|/C/|        ; uncomment for PRI T1 ESF
```

```
;NS|/DINT:FF=0/|/C/|        ; uncomment for PRI E1
;
#d 3
;
;  SET FRAME FORMAT:  uncomment (remove the leading semicolon) from the
;  line as indicated.
;
;NS|/DINT:FF=0/|/C/|        ; uncomment for T1 D4/SF
;NS|/DINT:FF=1/|/C/|        ; uncomment for T1 ESF
;NS|/DINT:FF=2/|/C/|        ; uncomment for E1
;NS|/DINT:FF=3/|/C/|        ; uncomment for TDM
;NS|/DINT:FF=4/|/C/|        ; uncomment for BRI
;NS|/DINT:FF=5/|/C/|        ; uncomment for PRI T1 D4/SF
;NS|/DINT:FF=6/|/C/|        ; uncomment for PRI T1 ESF
;NS|/DINT:FF=7/|/C/|        ; uncomment for PRI E1
;
;
;  SET T1 DIGITAL SIGNALING MODE:  uncomment (remove the
;  leading semicolon) from the line as indicated (default Loop Start):
;
;NS|/DCO:SM=0/|/C/|    ; uncomment for T1 Loop Start
;NS|/DCO:SM=2/|/C/|    ; uncomment for T1 E&M Immediate (FGB)
;NS|/DCO:SM=3/|/C/|    ; uncomment for T1 E&M Wink (FGD)
;
;  SET CHANNEL NUMBER:  uncomment (remove the leading semicolon
;  from the line as indicated and plug in the channel
;  number to use (default in all cases is channel 1):
;
;NS|/DINT:TCN=1/|/C/|  ; uncomment for T1 (1-24)
;NS|/DINT:ECN=1/|/C/|  ; uncomment for E1 (1-30)
;NS|/DINT:PTCN=1/|/C/| ; uncomment for PRI-T1 (1-23)
;NS|/DINT:PECN=1/|/C/| ; uncomment for PRI-E1 (1-30)
;
;  SET COMPANDING MODE:  Uncomment the following
;  line ONLY if you need A-law companding:
;
;NS|/DINT:PCMC=1/|/C/|  ; uncomment ONLY for A-law.
;
;  Wait for sync, check sync
#d 8
;
NS|/DINT:ALRM0/||
;
;  This completes the per-test-channel setup.
```

Note the file contains instructions as to how to make changes according to your needs. For example, if you need to use A–law companding, you would change the line

```
;NS|/DINT:PCMC=1/|/C/|  ; uncomment ONLY for A-law.
```

to

```
NS|/DINT:PCMC=1/|/C/|  ; uncomment ONLY for A-law.
```

Notice that the *only* change was to remove the first semicolon from the line. This changes the line from a comment line to an active line, so that NETSIM will send the command to the Series 2 network simulator.

10 Reference — OTTO:POST

10.1 General

OTTO:POST is a post–processor that reads each of the log files generated by OTTO:3800 standard testing and produces a summary file with data in standard places. The goal of this post–processor is to analyze the log files and place the data in such a way that a spreadsheet template can pick up the data by position instead of by analyzing tags.

10.2 Invocation

The program is invoked as follows:

OTTOPOST <pgm> <base>

where <pgm> is the name of the OTTO:POST script to use, and <base> is the base name of the test to be analyzed.

OTTO:POST makes the assumption that each log file contains a three–character test identification, a one– to five–character modem or job identifier, and the extension ".XLS". The standard distribution contains files that generate log files that fit this assumption.

If no extension is provided for <pgm>, OTTO:POST will automatically add the .PGM extension to the name before opening the file.

If <base> has an extension, that extension is stripped before processing begins. This is how the standard batch files can invoke OTTO:POST and pass the .MDM file name as the base name.

10.3 Operation of the program

The program searches the current directory using the pattern "???xxxx.xls", where "xxxxx" is the base name. For every file it finds, it begins reading lines from the file and passing the lines to a script execution engine. The engine uses the script (.PGM extension) to determine what lines should be processed, and how. Once all the files in the current directory matching the criteria above have been read and processed, the program then outputs the results to a summary file of the name "xxxxx.XLS".

In the process of reading the files, two files of the same name are created as working files, one created in the subdirectory GOOD and one created in the subdirectory BAD. The file created in the GOOD subdirectory contains only good test data (data without an ERR record); the file created in the BAD subdirectory contains only test data with errors (data with an ERR record).

The copy of the file in the BAD subdirectory can be used to perform detailed failure analysis of the test.

10.4 Language

OTTO:POST takes log files generated by OTTO:3800 and extracts information from those log files in a form suitable to process in Excel or other spreadsheet programs. Each OTTO:POST program consists of a series of rules, which are applied to each line from each log file processed. It is possible for a single line to satisfy multiple rules, a single rule, or no rule at all.

The rules are specified as a network of statements. Outer test statement serve as filters for inner test statements, which then serve as additional filters for enclosed test statements, until a capture statement(s) indicate(s) what to do when each of the conditions are met.

Nesting test statements is identical to having an AND operator, while test statements at the same level of nesting operate as those they are connected by an OR operator. There is no NOT operator equivalent in this language.

To keep the parsing of the language simple, every operator is a single character. Operands are digit strings, symbol names, or character strings. The exception is the field-test statement, which consists of a character string and a digit string, with no operator.

Here is an index of the language elements:

10.4.1 TEST STATEMENTS

"<string>" @ { ... } Test the file name for <string>, and if a match execute the contents of the braces.

"<string>" <number> { ... } Test the field indexed by <number> against the text in "string", and if a match execute the contents of the braces.

! { ... } If the test result is good (no ERR record in the set) execute the contents of the braces.

? { ... } If the test result is bad (ERR record in the set) execute the contents of the braces.

10.4.2 DATA CAPTURE STATEMENTS

.set <number> Interpret the data in field indexed by <number> and save the LATEST occurrence.

.min <number> Interpret the data in field indexed by <number> and save the lowest value encountered.

.avg <number> Interpret the data in field indexed by <number> and save the arithmetic mean of all values encountered.

.max <number> Interpret the data in field indexed by <number> and save the highest value encountered.

.count Increment the counter for each occurence.

.list <number> Save all occurrences of text in field indexed by <number> and output them with tab delimiters. Each text element is limited to 40 characters.

.text <number> Save the LATEST occurrence of text in field indexed by <number> and output with a newline. Up to 2048 characters are saved. (Every occurence of the string is saved, but only the last one is output -- use this function with caution)

.line <number> Save all occurrences of text in field indexed by <number> and output them with line delimiters. Up to 2048 characters are saved.

10.4.3 ANNOTATION STATEMENTS

"<string>" > (Greater-than sign) Output <string> as text. Used to label information in the file. The following characters have special meaning: \n = newline; \r = carriage return; \t = tab.

10.4.4 COMMENT STATEMENTS

; <comment> <end-of-line> Everything from a semicolon to the end of line is ignored.

<comment> <end-of-line> Everything from a hash mark to the end of line is ignored.

10.5 LIMITATIONS

All the limits built into this program are designed to be as generous as possible.

There is a limit of 490 nested statements at any given time. Nested statements are statements included inside other statements. The nesting level can be predicted at any given point by counting the number of left curly braces ({) and subtracting the number of right curly braces (}) at each point in the program. If that number exceeds 490 the compilation will abort. (In real programs this nesting value rarely exceeds 10, so there is lots of room.)

There is a limit of 16000 program steps. That means that OTTO:POST will handle a minimum of 8000 statements. Each test statement requires two program steps. Each capture statement requires one program step. Each annotation statement requires one program step. Each comment statement requires no program steps.

There is a limit of 10000 numeric values that can be captured. A numeric block is allocated for each .set, .min, .avg, .max, and .count statement. This limit is almost impossible to reach. (Please don't take this as a challenge.)

There is a limit of 10000 list values that can be captured. A list block is allocated for each .text, .list, and .line statement. A list block is also allocated for each line captured in a .list or .line statement.

There is a limit of 64000 characters of text. Text is allocated for each annotation statement, and also for each .text, .list, and .line statement in the program.

To aid the program designer, OTTO:POST prints statistics at the end of each run. These statistics look something like this:

```
34 (0 dynamic) of 10002 list blocks used.
2564 (193 dynamic) of 64002 string bytes used.
```

The first number indicates the total amount of the resource used in that particular run of OTTO:POST. The second number, marked "dynamic", indicates the amount of the resource consumed after compilation during execution of the program.

10.6 SAMPLE PROGRAM

The following is a simple program taken from the OTTO:3800 package:

```
"ATI" @ {
  "\nModem Version Information\n" >
  "ATI" 2 { .line 3 }
  "-30-\n" >
  }
```

The output of OTTO:3800 (abridged) look like this:

```
col-0       col-1  col-2   col-3
-----       -----  -----   -----
(FILE ATI50-01.XLS)
50-01:MAX   iscan  TST     i_scan * 1200
50-01:MAX   iscan  PIP     19980915  180825
50-01:MAX   iscan  ATI     0=5601
50-01:MAX   iscan  ATI     1=C702
50-01:MAX   iscan  ATI     4=V.90/56K Data-Fax Modem Settings...
50-01:MAX   iscan  ATI     5=V.90/56K Data-Fax Fax Settings...
```

During pass one, OTTO:POST reads each line and performs the entire script on that line. After OTTO:POST has finished all files, it then runs through the script again in pass 2, outputting the results to the summary file.

During pass one:

- The first line of the file tests to see if the first three characters of the file name are "ATI"; if not, the entire block of statements is bypassed and processing continues with the next line of input.
- The second line is an annotation statement, which during pass one does nothing.
- The third line examines to see if the third column (column 2 with the first column being column zero) starts with the string "ATI". If it is, the .line statement saves the data in the fourth column (colum 3 with the first column being column zero) for later output.
- The fourth line is an annotation statement, which during pass one does nothing.

During pass two:

- The first line contains only a test statement; no output is generated from this line.
- The second line is an annotation statement; the text is output verbatim to the summary file. Note that the string contains line formatting commands (newline, or \n), because

the annotation statement doesn't output any characters not specified in the string. The programmer must provide the formatting characters.

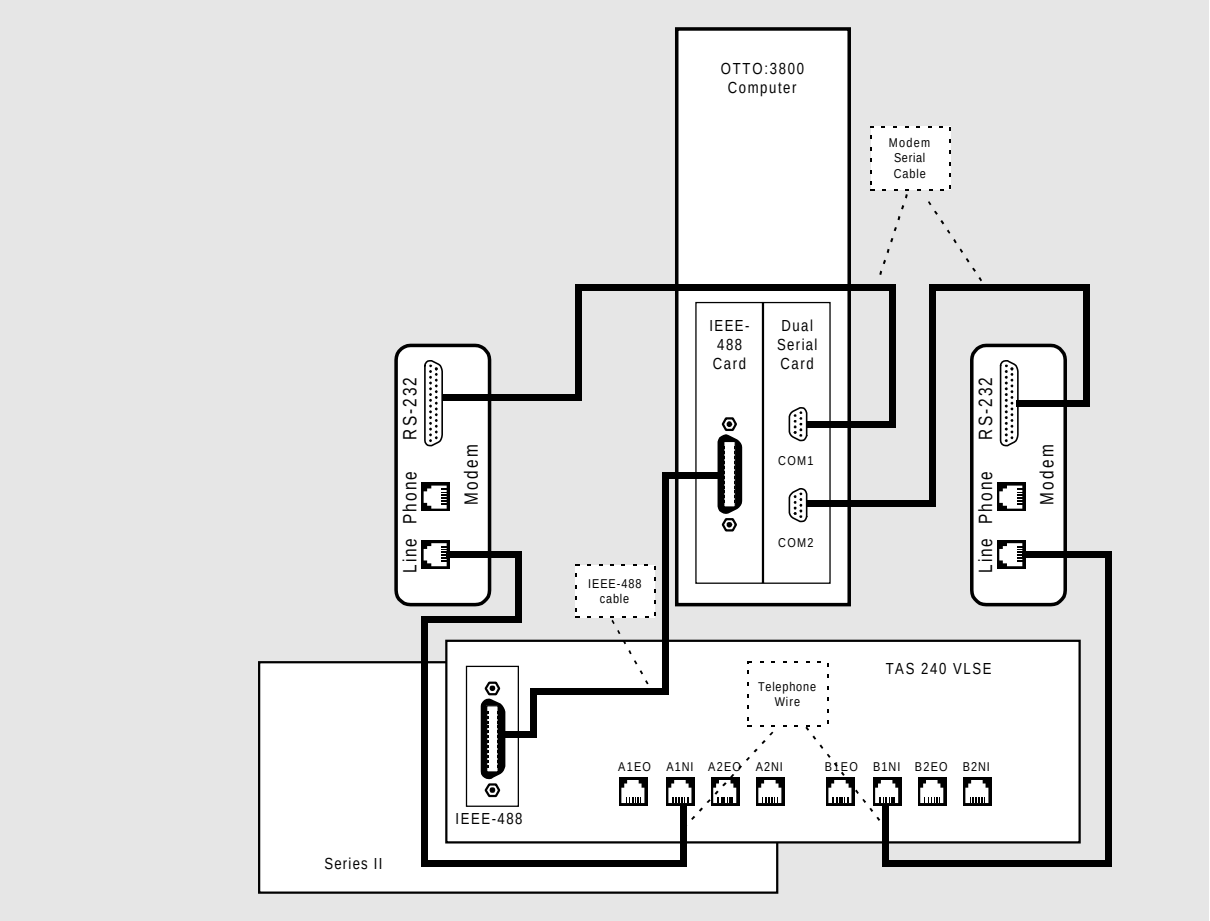
- In the third line, the test statement outputs nothing. The .line statement, though, outputs all lines captured during the execution of the program. In the example file, four lines would be output, each line terminated by newline characters. (In other circumstances, the .list command acts identically except that each captured block of text would be terminated with a tab instead of a newline.)
- The fourth line is an annotation statement, which is written verbatim. Remember that the .list command outputs a newline after each line, so placing a "\n" at the beginnning of the annotation string would have caused a blank line to be written to the summary file.

Appendix A Wiring for OTTO:3800

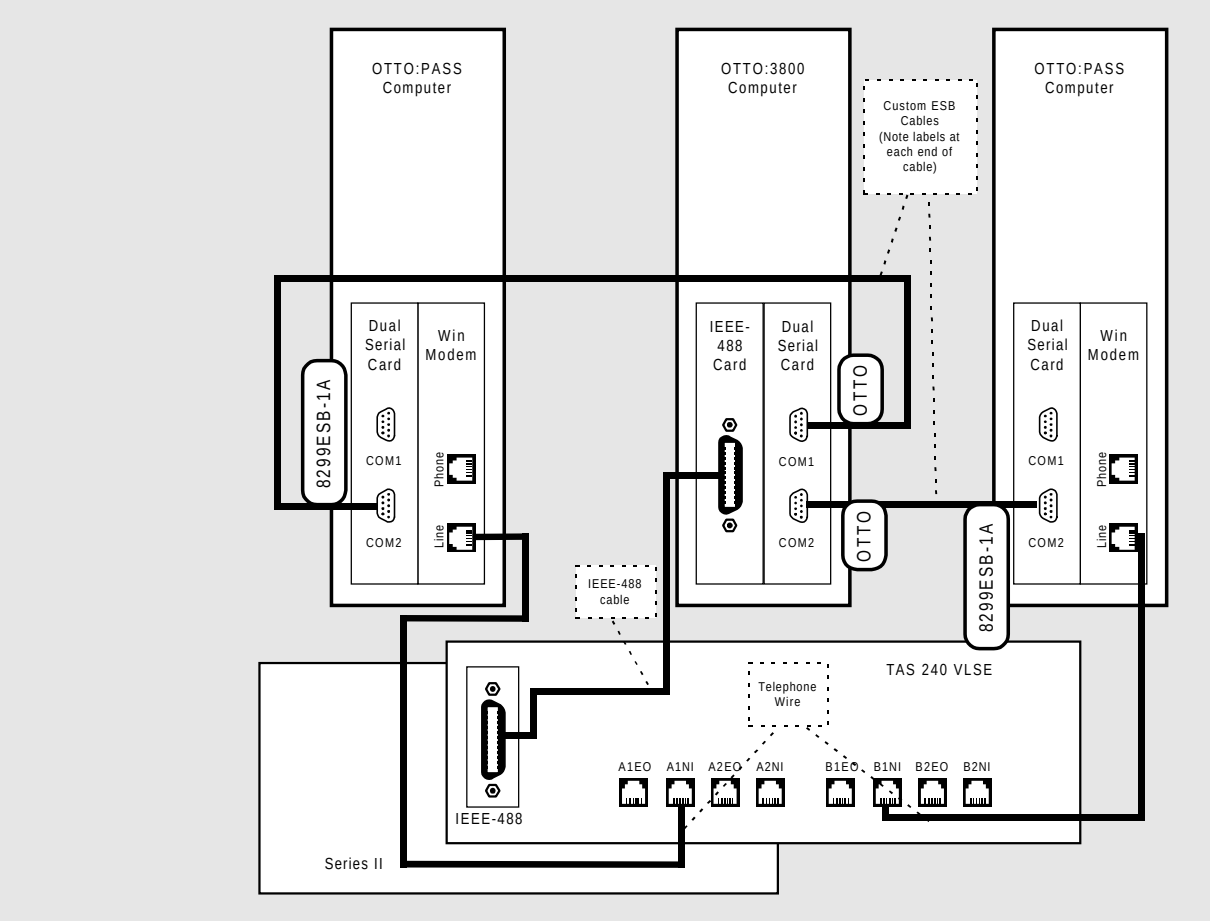
The following sections show how to wire the equipment and the test articles to perform testing. There are two options shown: how to test two external modems, and how to test two internal modems using OTTO:PASS.

> NOTE: For clarity in these drawings, they show only the IEEE-488 cable connected between the OTTO:3800 system and the TAS 240 VLSE. Don't forget that there also needs to be a cable from the TAS 240 VLSE to the TAS Series 2. If you have TAS 3508 modem switches, not shown in the diagrams, you should follow the additional wiring information contained in the TAS documentation.

A.1 Two External Modems



A.2 Two Internal Modems (OTTO:PASS)



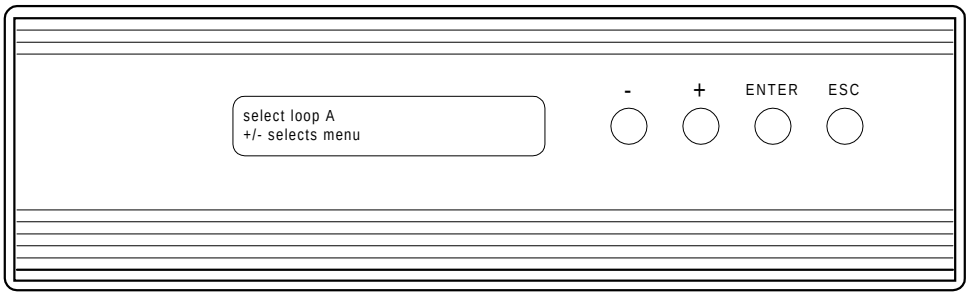
Appendix B Wiring the TAS hardware

This section describes how to wire and configure your TAS Network Simulator. **Follow these instructions only if your TAS 240 and TAS 1200 have not been wired and configured.**

If you are installing software to be used on an existing TAS setup, skip this section and go to the next .

Plug the female end of the equipment power cord into the back of the TAS 240. Plug the male end of the equipment power cord into any convenient outlet.

Turn on the TAS 240 unit. The switch is located on the rear panel near the equipment power cable. The TAS 240 will indicate that it is executing the self-test sequence by displaying the fact on the front–panel LCD screen and clicking relays for about 90 seconds. This is normal. After the TAS 240 becomes quiet, continue with the next step.



TAS 240

On the 240 front panel, press the ESC button five times. This ensures that the unit is at the top–most menu.

On the 240 front panel, press the plus (+) key until the following is displayed:

```
auxiliary parameters↵
+/- selects menu
```

Press the ENTER key. Then press the plus (+) key until the following is displayed:

```
configuration↵
+/- selects auxiliary parameter menu
```

Press the ENTER key. Then press the plus (+) key until the following is displayed:

```
REMOTE PROTOCOL: gpib↵
+/- selects remote protocol
```

Press the ENTER key. A display appears that is similar to the one below:

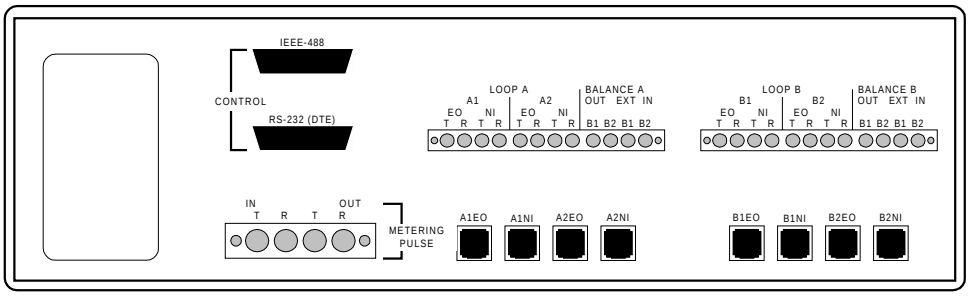
```
GPIB ADDRESS: 2
+/- selects GPIB address
```

Press the plus (+) and minus (–) keys until the display is exactly as shown above.

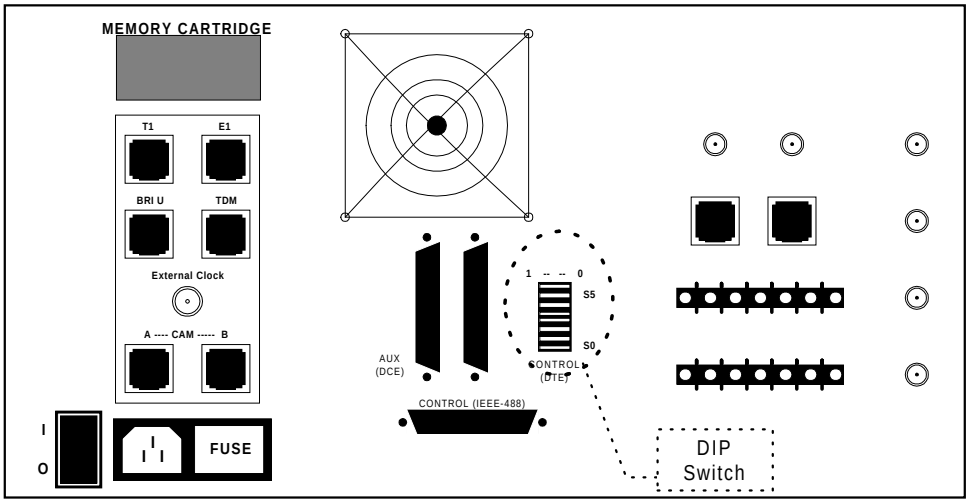
Turn off the TAS 240.

Remove the equipment power cord from the outlet.

Unplug the equipment power cord from the TAS 240.



TAS 240



TAS 1200

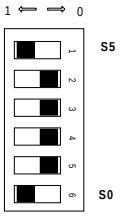
Place the TAS 1200 Series II unit on the table with the rear panel facing you.

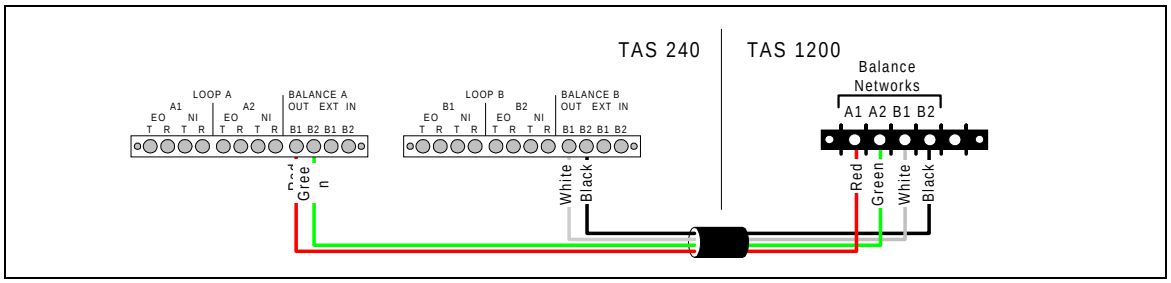
Place the TAS 240 VLSE on top of the TAS 1200 Series II unit with the rear panel facing you.

Install one end of the IEEE-488 cable (supplied by TAS) to the TAS 240 Control port. Tighten the lock screws finger tight.

Install the other end of the IEEE–488 cable to the TAS 1200 Control port. Tighten the lock screws finger tight.

Examine the DIP switch on the rear of the TAS 1200. The DIP switch is the small (0.7 x 0.3 inch) rectangular box, usually a red body color and with white slides. This switch is located just to the right of the "Control (DTE)" serial port connector (see illustration above for the exact location of the DIP switch on the rear of the TAS 1200). The illustration at right shows how the switches need to be set so that the TAS 1200 is configured to respond to IEEE-488 address 1. Set the switches to match this diagram.





Install the hybrid balance cable. The illustration above shows where each color lead needs to be fastened. For each lead, loosen the screw on the barrier strip; slide the lug under the screw head; then tighten the screw on the barrier strip until the screw resists a few pounds. Do not over tighten the screws.

Plug one end of the RJ-45 flat cable into the 240 jack labeled A1EO; plug the other end into the 1200 rear–panel jack labeled Station A.

Plug one end of the RJ-45 flat cable into the 240 jack labeled B1E0; plug the other end into the 1200 rear–panel jack labeled Station B.

NOTE: If the rear jacks on the TAS 1200 do not operate properly, move the RJ-45 cable connections from the rear panel to the front panel.

Plug one end of the RJ-45 flat cable into the 240 jack labeled A1NI; plug the other end into the RJ-45 coupler. This is where the A modem is connected.

Plug one end of the RJ-45 flat cable into the 240 jack labeled B1NI; plug the other end into the RJ-45 coupler. This is where the B modem is connected.

Plug the equipment power cords into the male power couplings on the back of both 240 and 1200; the other ends will connect to outlets.

This completes the cable interconnections for the TAS Network Simulator.

INDEX