

Fault Diagnosis Using μ Master - General Principles

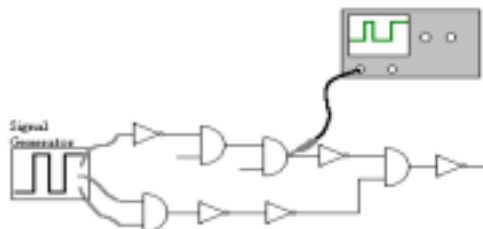
Pre-requisites

It is assumed that the reader is familiar with the basic operation of our μ Master product, and has a reasonable understanding of processor board architecture.

The μ Master Interactive mode is used to illustrate the basic techniques. Application Note #4 further illustrates the techniques using an example board.

Introduction

Conventional troubleshooting techniques involve a combination of stimulating the defective circuit using a signal generator, and probing using an oscilloscope. Figure 1 shows a typical set-up. The component at which the correct stimulus disappears is the defective component.



1. Conventional Circuit Troubleshooting.

With processor-based boards this technique is not possible, as there is no obvious place to connect a signal generator. In addition, with large scale integration and high density packaging (e.g. BGAs), probing is often not possible. The following alternative techniques vary in their effectiveness.

BIT (built-in test)

Test software is built into the board's boot ROM. Test results are output via an available output port (e.g. serial). Although an effective technique, BIT has a number of disadvantages: results are generally only pass/fail; the board kernel (CPU, memory, some I/O) must be

operational before BIT can be used; it must be built in at design time.

Logic Analyser

This is a useful method for monitoring board activity, however, it is difficult to connect, and to be used effectively a high degree of skill is needed. Essentially, an engineer's tool!

Emulator

An emulator, such as International Test Technologies' μ Master, combines functional testing, stimulus generation, and signal monitoring, making it a highly effective tool.

μ Master takes control of a processor-based board, and allows the user to fully control the test process. Even completely 'dead' boards can be controlled! It is the most flexible method for processor board troubleshooting, and little or no design-for-test is needed.

Overview of Troubleshooting Steps Using and Emulator

As with all engineering tasks, breaking the troubleshooting process into manageable steps is the best approach. The recommended approach consists of three top-level steps:

1. **Understanding the board structure.** That is, divide the board into functional blocks, and create an hierarchical block diagram of the board structure, similar to the one shown in Fig. 2. This example shows a familiar PC architecture, but the same principles apply to any board architecture.
2. **Fault Isolation.** Starting at the processor, and working through the board hierarchy, devise tests to functionally verify each block. This will isolate the fault area to a particular functional block.

available in the Interactive mode (fig. 4). This will verify most signals around the CPU area (e.g. reset, clock, etc.)



Fig. 4 – Pin Level test in the Interactive mode

The detected pin levels are verified against the currently loaded pin description file, which contains details of the pin name and its expected status (fig. 5).

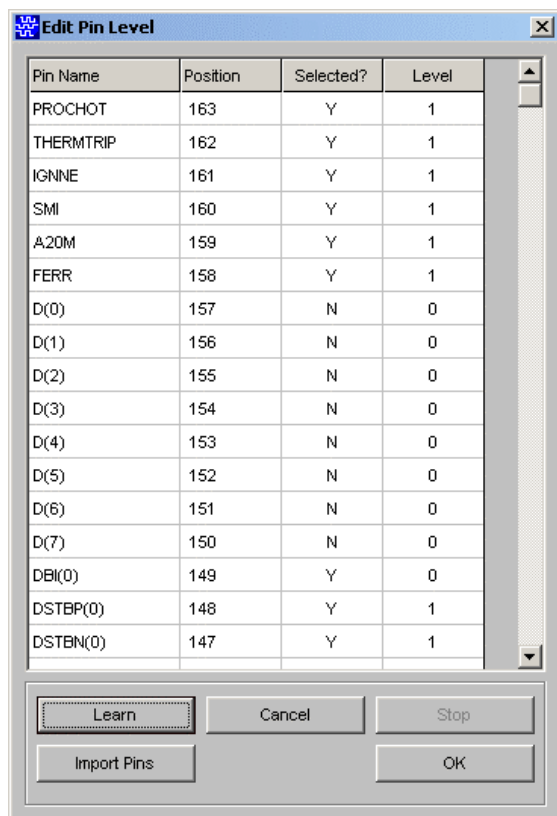


Fig. 5 – Pin File for a specific CPU, with expected pin levels defined

When the correct pin description file has been loaded for the CPU under test, expected pin levels can either be learnt from a known-good

board, or entered manually. When pin levels are learnt, they are sampled over time to establish whether they are constantly high (1), low (0), or changing over time (A – active).

The µMaster 3000 series includes a range of pin files for most Intel® processors. When the “Import Pins” function is used, the µMaster 3000 series automatically detects the type of CPU on the board under test and offers the appropriate pin file for import (fig. 6). If an appropriate pin file is not available, it is possible to import a standard-format BSDL file to obtain the CPU pin description. This is the only option in the µMaster 4000 series, which supports a vast range of CPU’s.

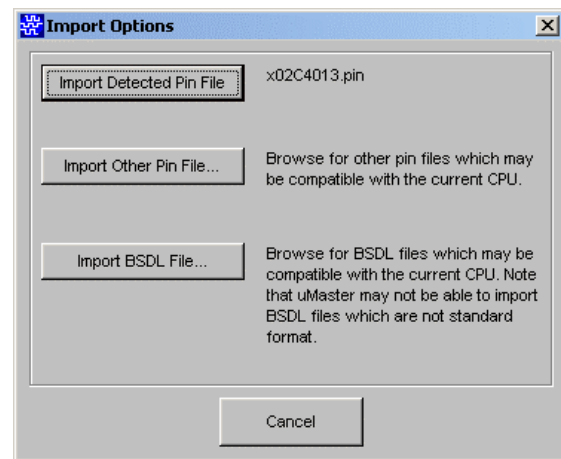


Fig. 6 – Import Pins menu in Interactive mode

Having run the “Pin Level” test, any defective signals can then be visually inspected, or traced using an oscilloscope. The following steps are recommended:

- 2.1.1. Verify power lines using an oscilloscope or DMM.
- 2.1.2. Verify the clock using an oscilloscope: the “Pin Level” test is only a presence detect, not a full check so the clock may still be defective.
- 2.1.3. Generate a repeated reset (using a simple µMaster test script), and check the processor’s main control signals. Lack of activity indicates a problem.
- 2.1.4. If all of the above steps pass, try replacing the CPU and/or main chipset (e.g. Northbridge).

Equivalent functions can be called from a test script. See the μ Master help file for information on:

[PinTestRun](#)
[PinTestCheck](#)

2.2. Buses

To diagnose bus defects (stucks, opens, shorts, etc.) μ Master provides a ROM bus test function. In the Interactive mode, this is called “Use Boot ROM” in the “Bus Tests” menu (fig. 7). To call this from a test script, the function name is **ROMBusTest**. See the μ Master help file for more details.

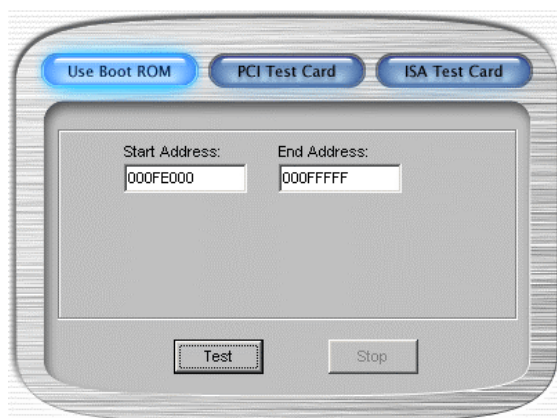


Fig. 7 – Bus Tests menu in Interactive mode

The ROM bus test verifies the operation of the buses from the processor to the boot ROM, and diagnoses most data and address bus problems (except shorts). In the “Use Boot ROM” menu, enter the address range of the boot ROM, and click the “Test” button. Any defective data or address bus lines found are reported by bus type and line.

An example error message is shown below. A “+” indicates a good bit, “0” is stuck low and “1” is stuck high.

**Bus Test- Data Test ROM Bus
Data Error: D07- D00: +++010++**

Other access tests will continue to verify the buses, for example, read/writes to Memory and I/O registers.

Note: Legacy devices from Intel® had their registers mapped in I/O space. For reasons of compatibility, registers are still mapped in I/O space but since this is restricted to 64KB, device registers are often now mapped in Memory space.

I/O tests can be found in μ Master’s Interactive mode in the “I/O Access” menu, called “R/W I/O Port” and “I/O Bus” (fig. 8).



Fig. 8 – I/O port and Bus tests in Interactive mode

Memory tests are in the “Memory Access” menu, called “R/W Memory” (fig. 9).

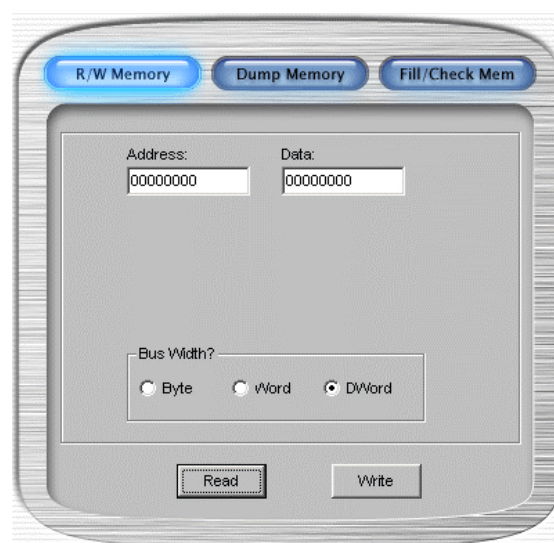


Fig. 9 – Memory tests in Interactive mode

Equivalent functions can be called from a test script. See the μ Master help file for information on:

[ReadIOByte](#)
[ReadIOWord](#)
[ReadIODWord](#)
[WriteIOByte](#)
[WriteIOWord](#)
[WriteIODWord](#)
[IOBusTest8](#)
[IOBusTest16](#)
[IOBusTest32](#)
[ReadMemory8](#)
[ReadMemory16](#)
[ReadMemory32](#)
[WriteMemory8](#)
[WriteMemory16](#)
[WriteMemory32](#)

2.3. Bus Bridges

A bus bridge converts one bus type to another, e.g. processor bus to PCI bus. To test and diagnose a bridge use the following methods:

- 2.3.1. All bridge chips contain programmable read/write registers. A read/write test to these registers will verify whether the chip is accessible to the processor. See the previous section for both μ Master's Interactive tests and scripting functions for I/O and Memory register access.
- 2.3.2. Step 2.3.1 doesn't fully verify a bridge. It is also necessary to ensure that devices can be accessed beyond the bridge to verify the bridge's "back-end". Successful writes and reads to/from the registers (Memory or I/O) of devices on the other side of the bridge verify that the bridge is performing its intended function.

When reading a device register (address XXXXXXXX) within a test script using [ReadMemory32](#), after having written a value ZZZZZZZZ into the register, an example error message would be:

```

Failure - Test Name
READ MEMORY FAILURE:
Address XXXXXXXX = YYYYYYYY
      not ZZZZZZZZ
    
```

2.4. Memory

μ Master is supplied with a number of pre-programmed RAM tests, all available in μ Master's Interactive mode (fig. 10).

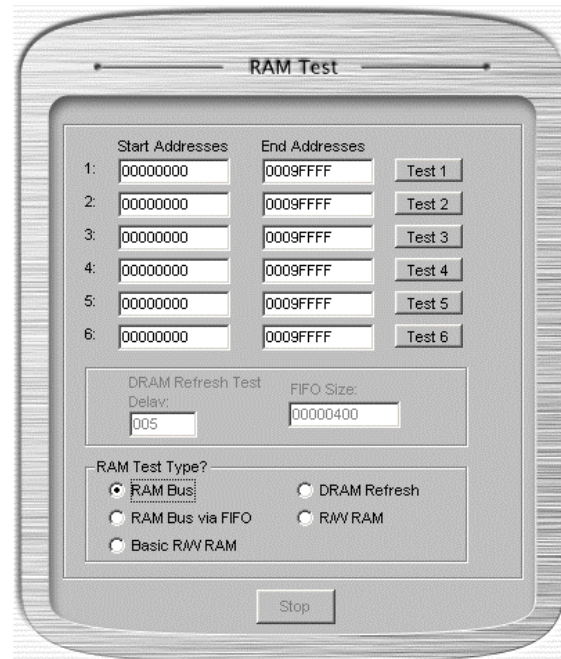


Fig. 10 – RAM tests in Interactive mode

- 2.4.1. The "RAM Bus" test verifies the buses up to the RAM chips. Bad data and address lines are automatically diagnosed and reported.
- 2.4.2. The "Basic R/W RAM" test (Intel® Pentium® processor family only) verifies the buses up to the RAM chips, and that all RAM cells are readable and writeable. Bad data and address lines, and RAM cells are automatically diagnosed and reported.
- 2.4.3. The "R/W RAM" test (Intel® Pentium® processor family only) performs a more pattern sensitive RAM test. Use this, if the other RAM tests pass and you still suspect the RAM as being the problem.
- 2.4.4. The "DRAM Refresh" test (Intel® Pentium® processor family only) verifies that all DRAM cells are refreshing correctly.

An example error message is shown below:

```
Basic R/W RAM- Data Hi/Lo Test
Defect at XXXXXXXX
D63-D48: ++++++
D47-D32: ++++++
D31-D16: ++++++
D15-D00: ++++++010+
```

“XXXXXXXX” is the first address of the RAM block, “+” means the data line is OK, “0” indicates a data line stuck low and “1” is a data line stuck high.

To test RAM in the Interactive mode menus, enter start and end addresses into one or more of the address range boxes and click the related test button(s).

Equivalent functions can be called from a test script. See the μ Master help file for information on:

- [RAMBusTest](#)
- [RAMBusTestviaFIFO*](#)
- [BasicRWRAMTest*](#)
- [DRAMRefreshTest*](#)
- [RWRAMTest*](#)

(* Four of these commands are only available in the μ Master 3000 series for Intel® Pentium® processors)

Often RAM is controlled by a memory controller. This device contains internal registers, which must be initialised before the RAM is visible to the processor. There are two ways to do this.

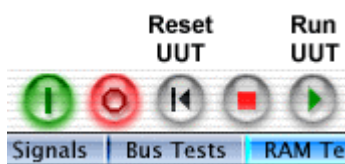


Fig. 11 – Reset and Run UUT toolbar buttons

2.4.5. By running the UUT’s boot ROM BIOS. Using the buttons on the μ Master toolbar (fig. 11), reset the UUT and then run (boot) it so that the BIOS initialises

the memory controller. Any of the RAM tests can then be used. There are two drawbacks with this method:

- On defective boards the BIOS may not be able to initialise the memory controller correctly, in which case the memory will fail the RAM tests.
- If the BIOS is not allowed to run far enough to initialise the memory controller, before the RAM tests are carried out, the memory will fail the tests. Only use this method, if you are confident that the BIOS has made it to the point where it has initialised the memory controller.

2.4.6. By manually initialising the memory controller, writing the appropriate data to its registers. See Application Note #5 for details on how to do this.

2.5. Input/Output

Input/Output ICs can be sub-divided into two parts, each part requiring its own type of test, as shown in Fig. 12.

- I/O Test Type 1 tests the part that interfaces to the processor and its buses. (2.5.1)
- I/O Test Type 2 tests the part that interfaces to the outside world. (2.5.2)

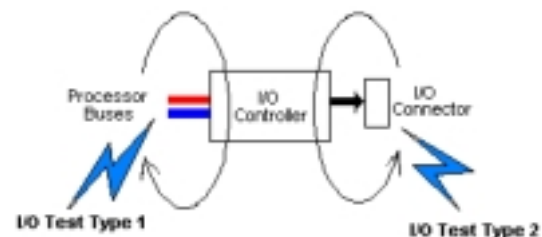


Fig. 12 – I/O devices require two types of test

All tests described below are available in μ Master’s Interactive mode and also as TSL/1 scripting commands. See the “Buses” section earlier in this application note for the I/O and Memory Read/Write commands. More details on I/O testing are presented in Applications Note #5.

2.5.1. I/O Test Type 1 – processor side

To verify the processor side of an I/O controller/device, there are normally internal registers that can be accessed by the

processor's read/write instructions. Test patterns can be written to these registers and then verified by reading them back.

For example, in the Interactive mode “**R/W I/O Port**” or “**R/W Memory**” menus, enter the address of a read/writeable register and then write and verify (read) a series patterns, such as AA and 55. Together, these two patterns verify the ability of each bit to go high and low. FF and 00 would do the same but AA and 55 will also highlight any shorts between adjacent lines – when no shorts are present, the correct pattern should be alternate high-low-high-low, etc.

Alternatively, if the I/O controller has an ID register, just verify the ID using a read I/O command. Such registers will verify that the processor and its buses can access the I/O controller.

More detailed diagnostic information can be derived using the “**I/O Bus**” test and the “**Find PCI devices**” search function (fig. 8).

2.5.2. I/O Test Type 2 – outside world

To verify the outside world side of an I/O controller, this test must successfully communicate with the external device (peripheral) under its control. The test consists of the following steps:

2.5.2.1. Initialise the I/O controller to enable it to communicate with the external device. This involves writing specific values to some of the I/O controller's internal registers. Details of the registers and values are available in device manufacturer's datasheets. However, μ Master is supplied with an extensive library of pre-written device descriptions that allow tests to be generated automatically (see the “ATG” section in the μ Master help file).

2.5.2.2. In the case of an output-only controller (e.g. video, audio loudspeaker), send data to the external device via the I/O controller's internal registers, and then verify it externally. Verification can be by:

- a. real peripheral with human intervention (even video and audio verification can be automated by our I/O Emulation Unit).
- b. 3rd party measurement equipment with human intervention (e.g. DMM or oscilloscope).
- c. μ Master's I/O Emulation Unit with appropriate measurement card. This is the ideal method because it is fully automated: verification is via closed loop feedback.

2.5.2.3. In the case of an input-only controller (e.g. keyboard, mouse, audio microphone), read the input generated by the external device, via the I/O controller's internal registers, and verify the data. Input can be generated by:

- a. real peripheral with human intervention.
- b. 3rd party signal generation equipment with human intervention.
- c. μ Master's I/O Emulation Unit with appropriate signal generation card. This is the ideal method because the signal generation is automatically triggered by μ Master test script commands.

2.5.2.4. In the case of an input-output controller (e.g. LAN, USB, SCSI), send data to the external device and then read back the data to verify it. These write-reads are made via the I/O controller's internal registers. Verification can be by:

- a. loop-back plug. This is a limited test that is only suitable for testing certain devices with simple Rx/Tx lines, e.g. LAN
- b. real peripheral device, e.g. USB Flash memory card
- c. μ Master's I/O Emulation Unit with appropriate measurement and signal generation card

(e.g. SCSI, FireWire). This is the ideal method because both the measurement and signal generation are automatically controlled by μ Master test script commands.

An example error message is shown below.

READ I/O FAILURE:
Port QQQQ.XXXXXXXXXX = YYYYYYYY
not ZZZZZZZZ

“QQQQ” is the Port Address, “XXXXXXXXXX” is the bit mask, “YYYYYYYYY” is the data read and “ZZZZZZZZ” is the expected data.

3. Fault Identification

The defective functional blocks isolated using the methods described above, will probably still contain a few components and numerous interconnects. To identify the actual defect within the failed functional block use the following methods.

3.1. Visual Inspection

Keep it simple! Before returning to the emulator, perform a detailed visual inspection of the isolated area. Look for shorts, bad solder joints, ‘tombstoned’ resistors, wrong ICs, etc.

3.2. Tactile Inspection

Again, keep it simple! Are there any overheating ICs? Now return to the emulator. Loop the failing test. Press down on all major ICs. Does the test pass now? This method can detect open circuits on devices such a BGAs etc.

3.3. Probing

Either loop a specific test in a μ Master test script from the Operator or Developer modes, or use the Interactive mode “**Stimulus**” function to inject known-good signals into the defective area (fig. 13). Probe (where possible) using an oscilloscope, or other instrument. Start by just looking for wrong levels, lack of activity, or tri-state conditions. If necessary, create the same stimulus on a good board and compare.

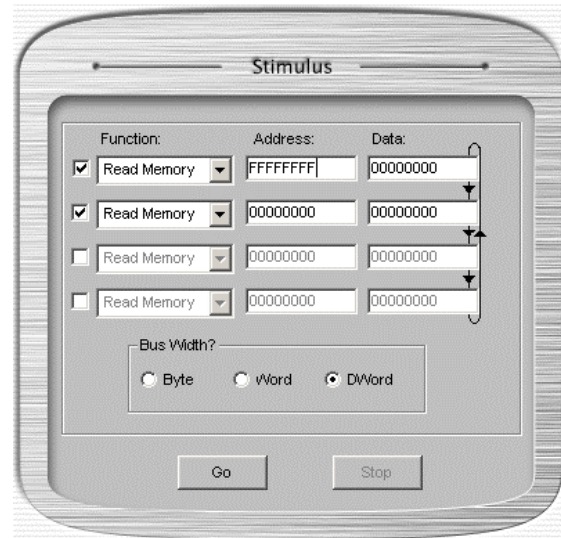


Fig. 13 – Stimulus menu in Interactive mode

In figure 13, all address lines are being repeatedly toggled between high and low states by two looping Read Memory commands, allowing any stuck high or stuck low lines to be located. This would not identify shorted lines, so it may be necessary to toggle each address line individually. For example, two looping Read commands set to 00000001 and 00000000 would allow any shorts between address line 0 and any other address line to be located (all other lines should remain low). A similar approach can be adopted for Data lines, but Write commands would be required instead of Reads.

3.4. Fault Histories

Refer to previously successful fixes, on the same board type, for the specific failure being investigated. μ Master has a built-in fault-logging database. Component failures can be automatically logged into the database. When fixes are found, the faulty component(s) can be added into the database. As the number of records in the database increases, fault identification using the fault history becomes more reliable.

The fault history database is accessed by clicking a button on the μ Master toolbar (fig. 14).

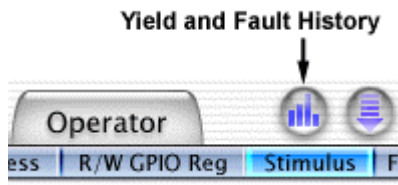


Fig. 14 – Yield and Fault History toolbar button

Features include Yield analysis by board type and test station number; frequency of specific component failures (%); frequency of specific fixes for specific components (%). See fig. 15.

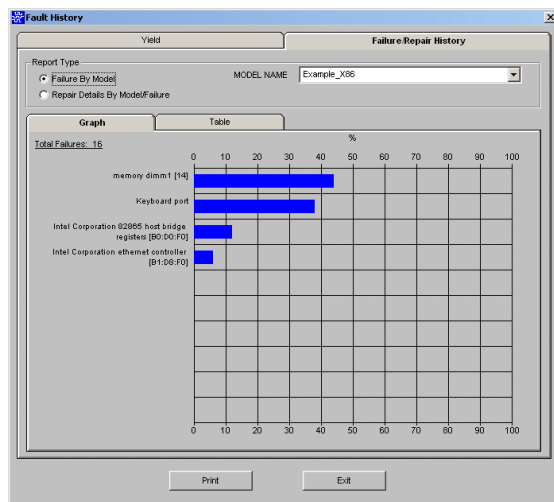


Fig. 15 –Fault History showing most likely causes of failures on a specific board

A stand-alone, test log database management program is also supplied with μ Master, called “uMaster Test Log”. This can be installed and run on any Microsoft® Windows®-based PC. It allows operators to add fix information into the fault logs, and also permits management personnel to analyse yield statistics, without requiring μ Master hardware on their PCs.

Summary:

An emulator provides the best technique for troubleshooting microprocessor-based boards. By following a simple 3-step procedure, defects can be rapidly isolated.

1. Understand board structure. Break it into functional blocks and create a troubleshooting tree.
2. Isolate the fault area by using simple test sequences to verify each functional block.
3. Finally identify the actual defect using a combination of inspection and probing.

Happy Troubleshooting!

Contacts for additional information

European Sales and Support:

International Test Technologies,
Larkin House, Oldtown Road,
Letterkenny, County Donegal, Ireland.
Tel: +353 (0)749 188 100
Fax: +353 (0)749 188 128
E-mail: sales@intertesttech.com
Web: www.intertesttech.com

N. American Sales and Support:

International Test Technologies,
2694 21st. Avenue,
San Francisco, CA 94116
Tel: 415 753 5376
Fax: 415 753 3635
E-mail: sales_usa@intertesttech.com

Asian Sales and Support:

International Test Technologies,
32 Maxwell Road,
#03-07 White House,
Singapore 069115.
Tel: +65 9642 3164
E-mail: sales_asia@intertesttech.com

For contact details of our worldwide reps. please see our website:

http://www.intertesttech.com/ate/company_locations.htm