Using the MPC800 Emulation Control Interface

The MPC800 Embedded PowerPC Emulation Control Interface works with an emulation module or emulation probe to give you an emulation interface for target systems which use an Motorola MPC800-family microprocessor. Also, with an analysis probe and a logic analyzer, you can make coordinated trace measurements.

Before you begin using the Emulation Control Interface, make sure you have set all of the necessary configuration options (see page 16).

• “At a Glance” on page 10
• “Connecting the Emulator to the Target System” on page 12
• “Configuring the Emulator” on page 16
• “Controlling Processor Execution” on page 23
• “Using Breakpoints” on page 24
• “Displaying and Modifying Registers” on page 28
• “Displaying and Modifying Memory” on page 30
• “Displaying and Modifying I/O” on page 34
• “Downloading an Executable to the Target System” on page 35
• “Coordinating Trace Measurements” on page 52

See Also

“Disconnecting from the Emulator” on page 71
“Error/Status Messages” on page 64
“To Use the Command Line Interface” on page 58
“Managing Run Control Tool Windows” on page 57
“Testing Target System Memory” on page 72
Using the MPC800 Emulation Control Interface

Main System Help (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume)

Glossary of Terms (see page 93)

Related Help

Setting Up and Starting the Emulation Control Interface (see the Emulation: Setting Up help volume)
Using the MPC800 Emulation Control Interface
Contents

Using the MPC800 Emulation Control Interface

1 Using the MPC800 Emulation Control Interface

At a Glance 10

Connecting the Emulator to the Target System 12
To Connect to a Target System (E5900A or E5901A) 12
To Connect to a Target System (E5900B) 13
To connect to an E2476A analysis probe 14
To connect to an E2476B analysis probe 14

Configuring the Emulator 16
Processor Clock Rate 17
"Trigger Out" Port 17
"Break In" Port 18
"Break In" Type 19
Level of Reset 20
Processor Type 20
Emulator Copies of Registers 20
To save configuration settings 22
To restore saved configuration settings 22

Controlling Processor Execution 23

Using Breakpoints 24

How hardware breakpoints work 26
How software breakpoints work 27

Displaying and Modifying Registers 28
To display registers 28
To modify register contents 29
Contents

Displaying and Modifying Memory 30
To display memory 30
To modify a memory location 31
To fill a range of memory locations 31
To display memory in mnemonic format 32

Displaying and Modifying I/O 34

Downloading an Executable to the Target System 35
To download into target RAM 36
To access a file from the logic analysis system 37
To choose a file format 37
To download into target flash ROM 38
To erase flash ROM 39
Error messages while downloading files 40
Details of the Motorola S-record format 42
Details of the Intel extended hex format 43
To choose a flash algorithm 44

Coordinating Trace Measurements 52
To break execution on a trigger 52
To trigger an analyzer on a break 53
To omit monitor cycles from the trace 54
To open windows 54
To close windows 56

Managing Run Control Tool Windows 57
To use the Status/Error Log window 57
To Use the Command Line Interface 58
Contents

Error/Status Messages  64
Address is a duplicate of Breakpoint #  65
Address is not on an instruction (4-byte) boundary  65
BDM Access Error  65
Debugger: memory update  65
Debugger: register update  65
Enable breakpoint failed  66
Hardware breakpoint  66
Memory access failed  66
MSR.RI bit not set - Break may not be recoverable  66
Software breakpoint:  66
Stepping failed  67
Target power is off  67
Unable to break  67
Unable to modify register: PC=value  67
Unknown break cause  67
Unimplemented opcode/register  67

To update firmware  69

Disconnecting from the Emulator  71
Using the MPC800 Emulation Control Interface
Chapter 1: Using the MPC800 Emulation Control Interface

At a Glance

Emulation using an E5901B emulation module and an E2476B analysis probe

Emulation using an E5901A emulation module and an E2476A analysis probe

- Logic analysis system Contains measurement modules such as the 16550A State/Timing Logic Analyzer Module. Contains the Emulation Control Interface that controls the emulation module or emulation probe.

- Analysis probe The E2476A or E2476B analysis probe provides state analysis for the PowerPC processor.
Chapter 1: Using the MPC800 Emulation Control Interface

At a Glance

- Emulation module or emulation probe  The emulator accesses the debugging facilities built into the microprocessor to give you control over processor execution, and easy access to processor registers, target system memory, and I/O.

  An E5900B emulation probe communicates with the logic analysis system via the LAN. An E5901B emulation module in the logic analysis system provides power, cross triggering, and limited communication for the E5900B emulation probe. The E5900A emulation probe connects to a BDM debug port designed into your target system or to the debug port on an E2476B analysis probe.

  An E5900A emulation probe or an E5901A emulation module requires a Target Interface Module (TIM) to connect the standard signal lines from the emulation module or emulation probe (emulator) to specific pins on the cable connected to the debug port. The E5900A and E5901A connect to an the 50-pin connector on the E2476A analysis probe or, via the TIM, to a BDM debug port designed into your target system.

- Target System  Your system using a Motorola MPC800-family processor.
Chapter 1: Using the MPC800 Emulation Control Interface

Connecting the Emulator to the Target System

You can connect the emulation module or emulation probe directly to a socket on the target system, or you can connect it using a socket built into the analysis probe.

The procedure for connecting to the target system depends on which kind of emulator you are using:

To connect directly to a debug port on your target system:

- “To Connect to a Target System (E5900A or E5901A)” on page 12 - If you have an E5900A emulation probe or an E5901A or 16610A emulation module.
- “To Connect to a Target System (E5900B)” on page 13 - If you have an E5900B emulation probe or an E5901B emulation module.

To connect directly to the debug port on your analysis probe:

- “To connect to an E2476A analysis probe” on page 14 - If you have an E5900A emulation probe or an E5901A or 16610A emulation module, plus an E2476A analysis probe.
- “To connect to an E2476B analysis probe” on page 14 - If you have an E5900B emulation probe or an E5901B emulation module, plus an E2476B analysis probe.

See Also

- “To decide if you have an E5900A or an E5900B” on page 90
- “To decide if you have an E5901A or an E5901B” on page 91

To Connect to a Target System (E5900A or E5901A)

In order to connect the emulation module or emulation probe to the microprocessor, a 10-pin debug port must be available on the target system.

To connect the probe to a target system:
Chapter 1: Using the MPC800 Emulation Control Interface

Connecting the Emulator to the Target System

1. Turn OFF power to the target system and the emulator.
2. Plug one end of the 50-pin cable into the emulator.
3. Plug the other end of the 50-pin cable into the target interface module.
4. Plug one end of the 10-pin cable into the target interface module. Be sure to align pin 1 of the cable to pin 1 on the target connector.
5. Plug the other end of the 10-pin cable into the debug port on the target system.
6. Turn on the power to the probe.
7. Turn on the power to the target system.

See Also
   The manual for your emulation probe or processor solution.

To Connect to a Target System (E5900B)

In order to connect the emulation module or emulation probe to the microprocessor, a 10-pin debug port must be available on the target system.

To connect the probe to a target system:
1. Turn OFF power to the target system.
2. Turn OFF power to the emulation probe.
3. Plug one end of the 10-pin cable into the emulation probe.
4. Plug the other end of the 10-pin cable into the debug port on the target system. Be sure to align pin 1 of the cable to pin 1 on the connector.
5. Turn on the power to the emulation probe.
6. Turn on the power to the target system.

See Also
   The manual for your emulation probe or processor solution.
Chapter 1: Using the MPC800 Emulation Control Interface

Connecting the Emulator to the Target System

To connect to an E2476A analysis probe

A 50-pin connector has been designed into the E2476A analysis probe. You can connect an E5900A emulation probe or E5901A emulation module to this port. A target interface module (TIM) is not needed.

The Setup Assistant can guide you through the process of connecting both the analysis probe and the emulator.

Here is a summary of the steps to connect the emulator to the analysis probe:

1. Turn OFF power to the target system, logic analysis system, and emulator.
2. Connect the analysis probe to the target system as described in the analysis probe or processor solution manual.
3. Plug one end of the 50-pin cable into the emulator.
4. Plug the other end of the cable into the analysis probe's connector.
5. Turn ON power to the emulator.
6. Turn ON power to the logic analysis system.
7. Turn ON power to the target system.

See Also

The manual for your emulation probe or processor solution.

To connect to an E2476B analysis probe

A BDM debug port has been designed into the E2476B analysis probe. To connect an E5900B emulator (standalone or interconnected to an E5901B emulation module) to this port:

1. Turn OFF power to the target system.
2. Turn OFF power to the emulator.
3. Connect the analysis probe to the target system as described in the processor solution manual.
4. Plug one end of the 10-pin cable into the emulator.
5. Plug the other end of the 10-pin cable into the analysis probe's connector.
6. Turn ON power to the emulator.
7. Turn ON power to the target system.

**See Also**

The *Solutions for the MPC8XX User's Guide*.

The *Setup Assistant* can guide you through connecting both the analysis probe and the emulator.
Chapter 1: Using the MPC800 Emulation Control Interface

Configuring the Emulator

Use the Setup window to configure the emulator for your target system.

1. Open the Setup window.

2. Make the appropriate selections for the probe configuration options you wish to change.

   The configuration options are:
   - “Break In” Type” on page 19
   - “Processor Clock Rate” on page 17
   - “Level of Reset” on page 20
   - “Processor Type” on page 20
   - “Emulator Copies of Registers” on page 20

   The following options apply only to emulation probes:
   - “Break In” Port” on page 18
   - “Trigger Out” Port” on page 17

   The configuration of the emulator is changed immediately when you make a selection. Select Load Processor to copy the “emulator copies of registers” to the processor.

3. When you have finished changing the configuration options, select File, then select Close.

See Also

- “To save configuration settings” on page 22
- “To restore saved configuration settings” on page 22
- The manual for your emulator or processor solution.
Chapter 1: Using the MPC800 Emulation Control Interface

Configuring the Emulator

Processor Clock Rate

This configuration option specifies the clock speed at which the emulator communicates with the debug port on the target system.

The BDM communication speed will be 1/3 of the configured processor clock speed. You may set the processor clock speed to a speed lower than the actual clock speed of your target system.

For the E5900B: The speeds you can specify are:

- 80 MHz
- 66 MHz
- 50 MHz
- 33 MHz
- 25 MHz
- 20 MHz
- 16 MHz
- 8 MHz
- 4 MHz
- 1 MHz
- 512 kHz
- 32 kHz

For the E5900A: The speeds you can specify are:

- 25 MHz
- 20 MHz
- 16 MHz
- 8 MHz
- 4 MHz
- 1 MHz
- 512 kHz
- 32 kHz

If your target system has additional loads on the lines used for the debug port, or if it does not meet the requirements described in the emulator User's Guide, using a slower clock speed may enable the probe to work.

"Trigger Out" Port

The Trigger Out BNC port can be used to tell devices external to the emulation probe when processor execution is in the monitor.

You can use this port to qualify a logic analyzer clock, so that monitor cycles are not captured.
Chapter 1: Using the MPC800 Emulation Control Interface

Configuring the Emulator

You can also use this port to signal the target system, for example, so that watchdog timers don't time-out when processor execution is in the monitor.

The options are:

**Always High**
Output is always high.

**Always Low**
Output is always low.

**High on Break**
Output is high when processor execution is in the monitor.

**Low on Break**
Output is low when processor execution is in the monitor.

"Break In" Port

The Break In port allows devices external to the emulator (for example, a logic analyzer's trigger output) to stop user program execution.

**Disabled**
Rising or falling edges on this port have no effect on the emulator.

**On Rising Edge**
Rising edges on this port will cause processor execution to break into the *monitor*.

This is the proper selection when the port is connected to the logic analysis system's trigger output.

**On Falling Edge**
Falling edges on this port will cause processor execution to break into the *monitor.*
"Break In" Type

The option controls what will happen when the emulator receives a "Break In". To generate a "Break In":

- For an emulation module, set up an intermodule measurement with the trigger coming from a logic analyzer. When the logic analyzer triggers, it will generate a "Break In".
- For an emulation probe, generate the appropriate input signal to the Break In BNC.

**Maskable, then if needed**

**NonMaskable**

A "Break In" will immediately cause a maskable break. If the maskable break fails, a non-maskable break will be attempted. The delay between an attempted maskable break and the non-maskable break will allow many instructions to be executed.

**NonMaskable Only (Possibly NonRecoverable)**

A "Break In" will immediately cause a non-maskable break. A non-maskable break can cause the processor to enter a non-recoverable state. Use this value if you are trying to halt the processor in an ISR.

**See Also**

- The Intermodule Window (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume)
- The processor solution manual for your processor (if available).
Chapter 1: Using the MPC800 Emulation Control Interface

Configuring the Emulator

Level of Reset

**Hard**  
A RESET command will cause a hard reset.

**Soft**  
A RESET command will cause a soft reset.

Processor Type

Select the processor to which the emulator is connected.

The Emulation Control Interface session will automatically be ended.

- **860**  
The emulator will be configured for an MPC860 target processor.

- **850**  
The emulator will be configured for an MPC850 target processor.

- **821**  
The emulator will be configured for an MPC821 target processor.

- **801**  
The emulator will be configured for an MPC801 target processor.

Emulator Copies of Registers

Internal registers are typically initialized at reset by the target system's initialization code. During development this code might not exist on the target system. To aid in development, the most important of these registers can be set directly by the emulator. This enables functions such as clocks and memory control to be established before executing any user code.

Enter values for the internal registers in the fields in the Setup window. These values will be copied from the emulator to the registers of the processor under test when you select *Load Processor*, or upon a transition from Reset to Background.

To disable this automatic copying of registers, set *Use Emulator*.
copies of Registers to No. To enable this automatic copying, set Use Emulator copies of Registers to Yes.

If you have initialization code that properly defines the registers in the processor under test, you can copy those values into the Setup window by selecting Read Processor Setup.

Three registers must be set up for the emulator to work properly with an MPC800-family target processor:

**CF IMMR**

The IMMR register in the processor under test is automatically programmed from the emulator's copy after a break from reset.

The IMMR register specifies the location of memory-mapped registers. Before any memory-mapped registers can be accessed by the processor probe, you must tell the emulator at which address the processor maps the registers.

Only the upper 16 bits are programmed into the IMMR; the lower 16 bits are read-only.

**CF_SYPCR**

The SYPCR register in the processor under test is automatically programmed from the emulator's copy after a break from reset.

The value you enter will be OR'ed with 0x00000080 to ensure that the watchdog timer is disabled in background.

**CF DER**

The DER register (debug enable register) controls which exceptions cause the processor under test to enter the debug mode. The DER register of the processor under test is automatically programmed from the emulator's copy after a break from reset.

The default value for the CF_DER register is 3082400f.

**Initializing the configuration registers after a reset-break**

By default, the emulation module's copies of the memory-mapped registers and the IMMR, SYPCR, and DER registers are automatically programmed into the processor after a break from reset.
Chapter 1: Using the MPC800 Emulation Control Interface

Configuring the Emulator

To save configuration settings

You can save emulation configuration settings as part of a logic analysis system configuration file. This saves all workspace configurations, including the emulation module and emulation probe configuration settings.

See Also

Saving a New Configuration File (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume)

To restore saved configuration settings

If you saved emulation configuration settings as part of a logic analysis system configuration file, you can restore them by loading the configuration file. This restores all workspace configurations, including the emulation module and emulation probe configuration settings.

See Also

Loading Configuration Files (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume)
Controlling Processor Execution

Processor execution is controlled using the Run Control window.

- To run the program, select Run.
- To stop program execution, select Break.
- To reset the processor, select Reset.
- To step a single instruction, select Step.

The status line, which appears under the run control buttons, shows the current status of the emulator. It shows whether:

- the processor is running the user program,
- the processor is in background debug mode;
- there is no power to the target system.
Chapter 1: Using the MPC800 Emulation Control Interface
Using Breakpoints

The Breakpoints window allows you to set breakpoints to stop processor execution when an instruction at a particular address is executed, or when a memory location is written or read.

You can enable up to two hardware breakpoints and up to two software instruction breakpoints.

To set a breakpoint

1. Decide whether you want to set a hardware breakpoint or a software breakpoint.

2. Enter an address for the breakpoint in the Memory Address field.
   Type the address in hexadecimal format. The address must be on an even boundary; that is, the address must end in 0, 2, 4, 6, 8, A, C, or E. Here are a few examples of valid breakpoint addresses:
   
   00000120
   00002ff0
   00003ff0
   00005100
   000052ff

3. If you are setting a hardware breakpoint, select whether to Break On Instruction Execution, Data Writes, Data Reads, or Data Reads or Writes but not instruction fetches.

4. Select the enable/disable toggle button to enable the breakpoint.
Chapter 1: Using the MPC800 Emulation Control Interface

Using Breakpoints

A breakpoint is enabled when the button is in.

**To clear a breakpoint**

- Select the enable/disable toggle button to disable the breakpoint.

A breakpoint is disabled when the button is out.

*Read Breakpoints* will display the breakpoints which are currently in effect in the target system. If you have used a debugger with the emulator to set breakpoints, you can use this button to read the breakpoints into the Breakpoints window.

Breakpoints are set or cleared in the target system when you select the enable/disable toggle button. If you change the address of an enabled breakpoint, the new breakpoint address is set in the target system when you move the mouse pointer from the address field. When a new breakpoint is being set in the target system, the mouse pointer will appear as an hourglass for a few moments.

**See Also**

“How hardware breakpoints work” on page 26

“How software breakpoints work” on page 27
How hardware breakpoints work

The PowerPC processor provides several built-in breakpoints. Because these breakpoints are implemented by the processor, you can use them to set breakpoints for addresses in ROM.

When a hardware breakpoint is enabled, the processor compares the address of each instruction to be executed with the values in the development mode comparators. If the processor finds a match, it immediately generates an exception.

If your target has an breakpoint exception routine, it will not be used. Instead, the emulator configures the processor to stop immediately when the exception occurs.

For more detailed information on Embedded PowerPC breakpoints, refer to the "Development Support" chapter in Motorola's User's Manual for your chip.
How software breakpoints work

Software breakpoints are implemented by replacing the instruction at the specified address with a trap instruction.

When you disable the software breakpoint, the original instruction is restored.

If you Step or Run from the breakpoint, the original instruction will be executed.

Limitations of software breakpoints

Because software breakpoints write to the breakpoint location, you cannot set software breakpoints in ROM or outside of physical memory. Software breakpoints cannot be used to debug exceptions.
Displaying and Modifying Registers

The Register window lets you display and modify the contents of processor registers.

- “To display registers” on page 28
- “To modify register contents” on page 29

The Emulation Control Interface displays groups (also called "classes") of related registers.

To display registers

- To add a group of registers to the display, select the group from the Groups menu.
- To remove a group of registers from the display, select the group from the Groups menu.
- To move a group to the bottom of the display, remove the group then add it again.
- To read the register values from the processor, select Read Registers.

The Registers window will be updated as the processor runs. If you want to disable this automatic updating, change Freeze Window (No)
to Freeze Window (Yes) in the Update menu.

The registers displayed vary according to the target processor.

---

**To modify register contents**

1. Display the group of registers that contains the register whose contents you wish to modify.

2. Select the value field you wish to modify. The underline cursor indicates the field that has been selected.

3. Type the new register value.

   The new value is actually written as soon as:

   - you press the Enter (or Return) key, or
   - the mouse pointer leaves the entry field.
Displaying and Modifying Memory

- “To display memory” on page 30
- “To modify a memory location” on page 31
- “To fill a range of memory locations” on page 31
- “To display memory in mnemonic format” on page 32
- “Addresses” on page 33

To display memory

1. Open the Memory window.
2. Specify the display format by selecting the appropriate Access Size and Base of Data options.
3. Enter the address (see page 33) you wish to read in the Read from field, and select Apply.
To modify a memory location

1. Open the Memory window.
2. Select Memory Write....
3. Enter the address (see page 33) of the location in the Start Address field.
4. Enter a length of 1. (Entering a larger value will fill multiple memory locations with the value.)
5. Enter the value that you want to write in the Data field.
6. Specify the size of the memory location and the format of the value you wish to enter by selecting the appropriate Access Size and Base of Data options.
7. Select Apply.

To fill a range of memory locations

1. Open the Memory window.
2. Select Memory Write....
3. Specify the size of the memory locations and the format of the value you wish to enter by selecting the appropriate Access Size and Base of Data options.
4. Enter the address (see page 33) of the first location in the Start Address field, enter the number of addresses to be written with the value in the Length field.
5. Enter the value that you want to write in the Data field.
6. Select Apply.
Chapter 1: Using the MPC800 Emulation Control Interface

Displaying and Modifying Memory

To display memory in mnemonic format

- Open the Memory Disassembly window.
  Memory is disassembled beginning at the current program counter value.
  The line corresponding to the program counter will be highlighted.

To display memory beginning at another address

1. Enter the first address (see page 33) to disassemble in the Starting Address field.
2. Select the appropriate Data Column Width.
3. Select Apply.
Scrolling

Page Forward and Page Back let you scroll the displayed memory locations. You cannot page back beyond the starting address.

Program counter tracking

If you step the processor or run and then stop, the highlighted line will track the current program counter.

If you do not want the highlighted line to track the program counter, select Freeze Window (No) in the Update menu.

Addresses

Enter all addresses in hexadecimal format.

Examples:

ABCDEF89
00000100
0040
Chapter 1: Using the MPC800 Emulation Control Interface

Displaying and Modifying I/O

Displaying and Modifying I/O

**To display I/O locations**

1. Open the I/O window.
2. Specify the display format by selecting the appropriate *Access Size* and *Base of Data* options.
3. Enter the address you wish to read in the *Read from* field, and select *Apply*.

*Clear Buffer* empties the contents of the I/O window.

**To modify an I/O location**

1. Open the I/O window.
2. Specify the size of the I/O location and the format of the value you wish to enter by selecting the appropriate *Access Size* and *Base of Data* options.
3. Enter the value that you want to write in the *Write* field, enter the address of the location in the *to* field, and select *Apply*. 
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

Use the Load Executable window to load executables (or other data) into your target system. You can load your data into RAM or into flash ROM.

- “To download into target RAM” on page 36
- “To download into target flash ROM” on page 38
- “To erase flash ROM” on page 39

**NOTE:** You must always erase flash ROM before you program it.

**NOTE:** Programming and erasing flash ROM may take a very long time on some processors. Additionally, Intel Quick-Pulse parts take longer to erase than others. The following examples show how long it took to perform various flash tasks on different processors. Your times will be different from the times in the examples. Use the Load Executable window to program or erase flash ROM only if these measurements are acceptable for your needs.

* To program a 1 megabyte file using AMD 5 Volt parts:
  * CPU32: 5 minutes * MPC860: 20 minutes * PowerPC 603e: 1 hours, 30 minutes

* To erase 1/2 megabyte Intel Quick-Pulse part:
  * CPU32: 2 minutes
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

* To erase 2 megabyte AMD 5 Volt parts:
  * MPC860: 10 seconds * PowerPC 603e: 5 seconds

Note that Intel Quick-Pulse parts always take longer to erase than other parts. PowerPC 60x always takes longer to program than Motorola MPC860 or CPU32.

---

To download into target RAM

1. Save the executable where the logic analysis system can read it (see page 37).
2. Open (see page 54) the Load Executable window.
3. Select the Load Executable operation. The parts of the window relating to flash ROM programming will be "grayed out".
4. Select the format (see page 37) of the file.
5. Change the Access Size option, if necessary.
6. If you chose the Motorola S-Records format or the ELF Object format, change the Set PC after load option, if necessary. If this option is selected (the button is in), the PC will be set to the execution start address, if it is specified in the file.
7. Enter the name of the file to load.
8. Select Apply.

When the file has been successfully loaded, "load completed" message will be displayed.

See Also

- “To access a file from the logic analysis system” on page 37
- “To choose a file format” on page 37
- “Error messages while downloading files” on page 40
To access a file from the logic analysis system

You need to save the executable file where the logic analysis system can read it.

If you will be downloading many files, you should NFS-mount the file system (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume) of the computer where you are compiling your program. If you do this, you can directly access your executable, regardless of how big it is, as soon as it is compiled or assembled. Mounting the file system may require the help of a system administrator.

You can also copy the file (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume) from a floppy disk to the logic analysis system’s hard disk. If the file is too big to fit on the floppy disk, use PKZIP to compress the file, then uncompress (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume) the file using the File Manager.

To choose a file format

You can choose one of the following file formats:

- **Motorola S-records** Most compilers for Motorola microprocessors can generate Motorola S-records. (see page 42) This file format represents the binary data as hexadecimal numbers in a ASCII file. Each record in the file includes a load address for the data in the record.

- **Intel Extended Hex** The Intel extended hex (see page 43) format represents binary data as hexadecimal numbers in a ASCII file. Each record in the file includes a load address for the data in the record.

- **Plain binary** If you choose Plain binary, the data in the file will be copied directly to the target system. Because plain binary files contain no information about where to load the file in memory, you must enter a start address.

- **ELF Object** Many compilers generate ELF Object files. This option will allow statically linked ELF images to be loaded into the target memory.
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

Dynamically linked ELF shared libraries cannot be loaded with this option. If your executable is not in one of these formats, recompile, reassemble, or relink your program with the appropriate options to generate one of these formats.

See Also

- “Details of the Motorola S-record format” on page 42
- “Details of the Intel extended hex format” on page 43

To download into target flash ROM

1. Save the executable where the logic analysis system can read it (see page 37).
2. Open (see page 54) the Load Executable window.
3. Select the Program Flash operation. The part of the window relating to “Erase Flash” will be "grayed out."
4. Select the Algorithm (see page 44) used by your flash ROM part.
5. Select the Bus Width of the data bus connected to your part.
6. Select the Device Width of your part.
   Notice that the Bus Width refers to your target system, but that Device Width refers to the flash ROM part.
7. Enter the ROM Start Address. This is the first address in memory which maps to your flash ROM part.
8. Enter the ROM Size.
   This is the total size (in bytes, hexadecimal) of the ROM address space. For example, if you have four flash parts which are each 1 MB in size, enter 400000.
   The emulator will not write to or erase addresses outside of the range [Start Address .. Start Address + ROM Size]. If your executable file contains data for both ROM and RAM addresses, only the data corresponding to the flash ROM addresses will be written.
9. If the flash ROM has not been erased, erase (see page 39) it.
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

10. Select the format (see page 37) of the file.

11. Change the Access Size option, if necessary.

12. If you chose the Motorola S-Records format or the ELF Object format, change the Set PC after load option, if necessary. If this option is selected (the button is in), the PC will be set to the execution start address, if it is specified in the file.

13. Enter the name of the file to load.

14. Select Apply.

When the file has been successfully loaded, "Load completed" message will be displayed.

See Also

- “To choose a flash algorithm” on page 44
- “To erase flash ROM” on page 39
- “To access a file from the logic analysis system” on page 37
- “To choose a file format” on page 37
- “Error messages while downloading files” on page 40

To erase flash ROM

1. Open (see page 54) the Load Executable window.

2. Select Flash Erase. The "Load Options" part of the window will be "grayed out."

3. Select the Algorithm (see page 44) used by your flash ROM part.

4. Select the Bus Width of the data bus connected to your part.

5. Select the Device Width of your part.

   Notice that the Bus Width refers to your target system, but that Device Width refers to the flash ROM part.

6. Enter the ROM Start Address. This is the first address in memory which maps to your flash ROM part.
7. Enter the ROM Size.

This is the total size (in bytes, hexadecimal) of the ROM address space. For example, if you have four flash parts which are each 1 MB in size, enter 400000.

The emulator will not erase addresses outside of the range [Start Address .. Start Address + ROM Size].

8. If the Erase chip button label is not "grayed out", select the button to erase the whole part.

The Intel Quick-Pulse and AMD 12 Volt Embedded algorithms only allow erasing of the who part. The Intel Auto algorithm requires one or more sectors of the part to be specified for erasing. The AMD 5 Volt Embedded algorithm allows you to choose whether to erase the whole part or just specific sectors.

9. If you did not select Erase chip, enter the address of each sector you want to erase.

You may erase up to four sectors at once. To erase more than four, simply enter the additional sectors after applying the erase function with the first four.

---

Error messages while downloading files

- AMD5 Volt algorithm requires either the "Erase chip" option or more sectors to be specified. If you do not select Erase chip, then you must select one or more sector addresses to erase. To select a sector to erase, select the box beside one of the sectors then enter a hex number for the address of the sector.

- Bad Load Address The start address which you entered for the plain binary file is invalid.

- Bad ROM Size. Specify a hex number. The number in the field is invalid. Re-enter a hexadecimal value.

- Bad ROM Start Address. Specify a hex number. The number in the field is invalid. Re-enter a hexadecimal value.

- Bad Sector Address. Specify a hex number. The number in the field is
invalid. Re-enter a hexadecimal value.

- Checksum errors found in non-data records. Checksum errors were found in one or more non-data records, but no errors were found in the data records. The data records were loaded.

- ELF file has no object image to load. Load not completed. This means we found a valid ELF Object file, but the target image was not included with the file. With no target image, we had nothing to load.

- File could not be opened. The executable file could not be opened. Check that the file exists, that the file permissions allow reading, and that the file is a data file and not a directory.

- File not in Intel Extended Hex file format. Load not completed, check format selection. Check that you have selected the correct file format button. If you still get this message, compare the file to the description in “Details of the Intel extended hex format” on page 43.

- File not in Motorola S-Record file format. Load not completed, check format selection. Check that you have selected the correct file format button. If you still get this message, compare the file to the description in “Details of the Motorola S-record format” on page 42.

- Flash ROM memory access failure. The emulator is unable to access ROM memory at the specified location. Check that the Start Address and ROM Size values are correct. Also, check that your target system has been set up correctly. For example, check that the chip select registers are set to allow access to the ROM memory.

- Intel Auto algorithm requires one or more sectors to erase. The Intel Auto algorithm does not support "bulk erasing" the entire part. Select one or more sector addresses to erase. To select a sector to erase, select the box beside one of the sectors then enter a hex number for the address of the sector.

- Load Failed: binary data load exceeds memory boundary. If the plain binary file were to be loaded at the the start address which you entered, some of the data would be written past the end of memory. Check the start address, then check that the file has a size that will fit in your target system's memory.

- Load failed: Checksum Errors found in data record(s). Checksum errors were found in one or more data records. The load was aborted. Generate a new executable file.
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

- Load Failed: ELF file has a format problem. ELF file has been altered in some way that we can no longer recognize it as an ELF file.

- Load Failed: Refer to Status/Error Log window. The Status/Error Log window will contain additional information about the errors that occurred.

- Non-ROM data encountered and ignored. This message means that a "Program Flash" operation was performed using a file which contained both data within the ROM address range and data outside the ROM address range. The ROM address range is determined by the values you entered: [Start Address .. Start Address + ROM Size]. Every byte in the file which is within the ROM address range was written to ROM; the data outside this range was ignored. If the data file should not have contained data for addresses outside of the ROM, check that the Start Address and ROM Size values are correct.

- Program AMD12V not erased. A message similar to this results from attempting to program ROM which has not been erased first. Erase the ROM before programming.

- Sector not erased - outside of ROM range. A sector address for the "Erase Flash" operation was outside the range [Start Address .. Start Address + ROM Size]. All sectors specified to be erased must reside within the ROM address range. Check the ROM address range and the sector address.

Details of the Motorola S-record format

An S-Record file has the following format:

```
[$$[module_record]
symbol_records
$$ [module_record]
symbol_records
$$]
header_record
data_records
record_count_record
terminator_record
```

Each record in the file consists of

- Record type (2 characters)
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

- S0 - Header record
- S1,S2,S3 - Data record
- S5 - Record count
- S7,S8,S9 - Terminator

- Record length, in bytes, not including the record type field (2 characters)
- Load address (2-8 characters)
- The data, represented in hexadecimal. (length determined by the record length field)
- Checksum, calculated as the 1's complement of all bytes in the record, not including the record type field (2 characters)

When downloading a file, anything before the header (such as any module or symbol records) is ignored. All optional records are ignored.

Example

Here is an example of an S-Record file:

S00600004844521B
S2070065400B24C40
S2050065F45F0
S21400600033FC000A000B24C202F00046C0622006D
...  
S2060066424E758E
S50300728A
S80400624455

Details of the Intel extended hex format

Each record in the file consists of

- A colon (:) (1 character)
- Number of bytes in the data field (2 characters)
- Load address (4 characters)
- Record type (2 characters)
- 00 Data
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

- 01 End of file
- 02 Extended segment address (bits 4-19 of a segment base address)
- 03 Start segment address
- 04 Extended linear address (the upper 2 bytes of a data load address)
- 05 Start linear address

The data, represented in hexadecimal. (length determined by the record length field) For the extended address record types, the data consists of 4 characters representing the appropriate bits of the address.

- Checksum, calculated as the 2’s complement of the sum of all of the bytes in the record after the colon (2 characters)

When downloading a file, only record types 00, 01, 02, and 04 are parsed. Any additional records (such as any module or symbol records) before the data records are ignored.

Example

Here is an example of a data record followed by an end of file record:

```
:20000000F00100230B340C4510561B6720782B89200BC40CD4CD
E50F60F06C01701290
:2002000AC23010112321453167418951AB61C0F1EF7121D001C01
F08F9090F18F7190660
:200400028F5290438F3390248F1490058FC59068FE690F78FA790
98B109812A834B865A7
:2006000C867D88E89A8180A1E0E0ABE1464B00AB00AB21AB32A4
3AB54AB65AB76AB874B
:20080000AB98AB299C309C819CF29CE39CC49CB59CD69CA79C18BBF
F9B38EC0CEC0DEC0E1
:200A000EC0FEC86EC8AE8EC8EC8EC8DEC87EC08EC04EC00EC01EC0
2EC03EC06EC07EC8F88
:0A01A000000D7200000D780030004
:00000001FF
```

To choose a flash algorithm

In the Flash Options section of the Load Executable window, you need to select the algorithm which will be used to program or erase your flash ROM part.
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

The supported algorithms are:

- AMD 5 Volt Embedded
- AMD 12 Volt Embedded
- Intel Auto
- Intel Quick-Pulse (AMD FlashRite)

If you do not know which algorithm your part uses:

- Look for your part in the “Table of Algorithms Used by Common Flash Parts” on page 45.

If your part is listed in the table, select the algorithm listed for your part. If your part is not listed, but you know that it is a new component belonging to one of the listed families, select the algorithm listed for the parts in the family.

- Check the manufacturer's literature to see if one of the supported algorithms is mentioned.

See Also

- “Table of Algorithms Used by Common Flash Parts” on page 45
- “Table of Unsupported Flash Parts” on page 51

Table of Algorithms Used by Common Flash Parts

Choose the manufacturer of your flash ROM part:

- “Table of Algorithms Used by AMD Flash Parts” on page 46
- “Table of Algorithms Used by Fujitsu Flash Parts” on page 50
- “Table of Algorithms Used by Hitachi Flash Parts” on page 49
- “Table of Algorithms Used by Intel Flash Parts” on page 47
- “Table of Algorithms Used by Micron Flash Parts” on page 50
- “Table of Algorithms Used by Mitsubishi Flash Parts” on page 49
- “Table of Algorithms Used by SGS-Thomson Flash Parts” on page 50
- “Table of Algorithms Used by Sharp Flash Parts” on page 51
- “Table of Algorithms Used by TI Flash Parts” on page 49
Information regarding the algorithm used by the listed parts is based on the manufacturer's specifications.

**See Also**

- “Table of Unsupported Flash Parts” on page 51

### Table of Algorithms Used by AMD Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD 12V Bulk Erase</td>
<td>Am28F256</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>Am28F512</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>Am28F010</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>Am28F020</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>Am28F256A</td>
<td>AMD 12V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am28F512A</td>
<td>AMD 12V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am28F010A</td>
<td>AMD 12V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am28F020A</td>
<td>AMD 12V Embedded</td>
</tr>
<tr>
<td>AMD 5V only Sector erase</td>
<td>Am29F010</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F100T</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F100B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F002T</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F002B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F002NT</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F002NB</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F200T</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F200B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F040</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F400AT</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F400AB</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F400BT</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F400BB</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F080</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F800T</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F800B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F800BT</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F800BB</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F016</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>Am29F017</td>
<td>AMD 5V Embedded</td>
</tr>
</tbody>
</table>
### Chapter 1: Using the MPC800 Emulation Control Interface

#### Downloading an Executable to the Target System

- **AMD 3V only Sector Flash**
  - Am29DL800T
  - Am29DL800B
  - Am29LV002T
  - Am29LV002B
  - Am29LV200T
  - Am29LV200B
  - Am29LV004T
  - Am29LV004B
  - Am29LV400T
  - Am29LV400B
  - Am29LV081
  - Am29LV008T
  - Am29LV008B
  - Am29LV800T
  - Am29LV800B
  - Am29LV800BT
  - Am29LV800BB
  - Am29LV160BT
  - Am29LV160BB

- **AMD 2.2V only Sector Flash**
  - Am29LL800T
  - Am29LL800B

### Table of Algorithms Used by Intel Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Performance Fast Flash</td>
<td>28F016XD</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F016XS</td>
<td>Intel Auto</td>
</tr>
<tr>
<td>Intel FlashFile</td>
<td>DD28F032SA</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F016SA</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F016SV</td>
<td>Intel Auto</td>
</tr>
</tbody>
</table>
## Chapter 1: Using the MPC800 Emulation Control Interface

### Downloading an Executable to the Target System

<table>
<thead>
<tr>
<th>Type</th>
<th>Part Number</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Boot Block</td>
<td>28F001BX-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F001BN-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F002BX-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F002BL-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F002BC</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F200BX-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F200BL-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F400BX-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F004BX-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F400BL-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F004BL-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td>Intel SmartVoltage Boot Block</td>
<td>28F800BV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F008BV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F800CV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F800CE-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F008BE-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F400BV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F004BV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F400CV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F400CE-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F004BE-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F200BV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F002BV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F200CV-T/B</td>
<td>Intel Auto</td>
</tr>
<tr>
<td>Intel Bulk erase</td>
<td>28F020</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>28F010</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>28F512</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>28F256A</td>
<td>Intel Quick-Pulse</td>
</tr>
</tbody>
</table>
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

### Table of Algorithms Used by Mitsubishi Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mitsubishi</td>
<td>M5M28F101</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>M5M28F102A</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>M5M28F800</td>
<td>Intel Auto</td>
</tr>
</tbody>
</table>

### Table of Algorithms Used by TI Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI</td>
<td>TMS28F010A</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>TMS28F010B</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>TMS28F512A</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>TMS28F210</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>TMS28F020</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>TMS28F040</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>TMS28F400BZT</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F400BZB</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F200BZT</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F200BZB</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F004Axy</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F400Axy</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F002Axy</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>TMS28F200Axy</td>
<td>Intel Auto</td>
</tr>
</tbody>
</table>

### Table of Algorithms Used by Hitachi Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hitachi</td>
<td>28F4001</td>
<td>Intel Quick-Pulse</td>
</tr>
<tr>
<td></td>
<td>28F101</td>
<td>Intel Quick-Pulse</td>
</tr>
</tbody>
</table>

See Also

- “Table of Unsupported Flash Parts” on page 51
### Table of Algorithms Used by SGS-Thomson Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGS-Thomson</td>
<td>28F410</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>28F420</td>
<td>Intel Auto</td>
</tr>
</tbody>
</table>

### Table of Algorithms Used by Fujitsu Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fujitsu 5 Volt Operation</strong></td>
<td>MBM29F002T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F002ST</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F002SB</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F200TA</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F200BA</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F040A</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F400TA</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F400BA</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F080</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F800T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F016</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29F017</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td><strong>Fujitsu 3 Volt Operation</strong></td>
<td>MBM29LV002T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29LV200T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29LV004T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29LV400T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29LV008T/B</td>
<td>AMD 5V Embedded</td>
</tr>
<tr>
<td></td>
<td>MBM29LV800T/B</td>
<td>AMD 5V Embedded</td>
</tr>
</tbody>
</table>

### Table of Algorithms Used by Micron Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Micron Boot Block</strong></td>
<td>MT28F002</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>MT28SF002</td>
<td>Intel Auto</td>
</tr>
<tr>
<td></td>
<td>MT28F200</td>
<td>Intel Auto</td>
</tr>
</tbody>
</table>
Chapter 1: Using the MPC800 Emulation Control Interface

Downloading an Executable to the Target System

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT28SF200</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28F004</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28SF004</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28SF004ET</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28F400</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28SF400</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28SF400ET</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28(S)F008</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28F008</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28SF008</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>Micron Sectored Erase</td>
<td>MT28(S)F116</td>
<td>Intel Auto</td>
</tr>
<tr>
<td>MT28F161</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>MT28SF161</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>-----------</td>
</tr>
</tbody>
</table>

Table of Algorithms Used by Sharp Flash Parts

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharp</td>
<td>LH28F004SU-LC</td>
<td>Intel Auto</td>
</tr>
<tr>
<td>LH28F008SA</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>LH28F016SU</td>
<td>Intel Auto</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
</tbody>
</table>

Table of Unsupported Flash Parts

The following parts cannot be programmed by the emulator:

<table>
<thead>
<tr>
<th>FAMILY</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel</td>
<td>HN28F101</td>
</tr>
<tr>
<td>Hitachi</td>
<td></td>
</tr>
<tr>
<td>Toshiba</td>
<td></td>
</tr>
</tbody>
</table>
Coordinating Trace Measurements

If you are using a logic analyzer (and perhaps an analysis probe) along with an emulation probe to debug your target system, you can coordinate the logic analyzer trace measurements with the execution of your target processor.

- “To break execution on a trigger” on page 52
- “To trigger an analyzer on a break” on page 53
- “To omit monitor cycles from the trace” on page 54

To break execution on a trigger

**With an emulation module**

1. Create a logic analyzer trigger.

2. In the Intermodule Window (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume), select the emulation module icon, then select the analyzer which is intended to trigger it.

3. Set the “"Break In" Type” on page 19 for the emulator.

4. Select Run in the logic analyzer window to start the trace measurement.

5. Start processor execution, if necessary.

   When the trigger occurs, processor execution should break into the monitor.

**With an emulation probe**

1. Connect the trigger output to the emulation probe's "Break In" port.

2. Enable the "Break In" port configuration option.

3. Start the trace measurement. When the trigger occurs, processor execution should break into the monitor.

4. Start processor execution.
Chapter 1: Using the MPC800 Emulation Control Interface

Coordinating Trace Measurements

This type of coordination lets you break processor execution on more specific conditions than are provided by breakpoints. For example, you can trigger and break on the write of a particular value to a particular address.

See Also

- To enable or disable emulator break on trigger (see the Listing Display Tool help volume).
- To trigger after, about, or before a source line (see the Listing Display Tool help volume).
- The solution manual for your processor (if available).

To trigger an analyzer on a break

If you want to trace execution that occurs before a break:

With an emulation module

1. Set the logic analyzer to trigger on anystate.
2. Set the trigger point to center or end.
3. In the Intermodule Window (see the Agilent Technologies 16700A/B-Series Logic Analysis System help volume), select on the logic analyzer you want to trigger, then select the emulation module. The logic analyzer is now set to trigger on a processor halt.
4. Wait for the Run Control window to show that the processor has stopped.

NOTE: The logic analyzer will store states up until the processor stops, but will continue running. When the processor stops, you may see a "slow clock" message, or the logic analyzer display may not change at all.
5. In the logic analyzer window, select Stop to complete the measurement.

With an emulation probe

1. Connect the emulation probe’s "Trigger Out" port to the analyzer’s input port.
2. Configure the "Trigger Out" port to either be high or low when processor execution is in the monitor (see “Configuring the Emulator” on page 16).

3. Configure the logic analyzer's input port. For example, if "Trigger Out" is low when in the monitor, configure the analyzer to look for a falling edge.

4. Set up the analyzer to trigger on the input signal (storing states that occur before the trigger).

5. Start the trace measurement.

6. Start processor execution. When the break occurs, the analyzer should trigger.

To omit monitor cycles from the trace

If you have an emulation probe, you can omit monitor cycles from a logic analyzer trace by using the "Trigger Out" port signal as an analyzer clock qualifier. That is, the analyzer is only clocked when processor execution is outside the monitor.

1. Connect the emulation probe's "Trigger Out" port to the analyzer's clock qualifier input.

2. Configure the "Trigger Out" port to either be high or low when processor execution is in the monitor (see “Configuring the Emulator” on page 16).

3. Configure the logic analyzer's clock qualifier input. For example, if "Trigger Out" is low when in the monitor, configure the analyzer to clock only when the qualifier input is high.

4. Set up and start analyzer trace measurements normally. The processor must be executing user code (not executing in the monitor) in order for data to be clocked into the analyzer.

To open windows

The Emulation Control Interface gives you several ways to open new windows:
Opening windows from the Emulation Control Interface icon

1. Move the mouse cursor over the Emulation Control Interface icon in the system window or in the workspace window.

2. Press and hold the right mouse button.

3. Move the mouse cursor over the menu selection for the window you wish to open.

4. Release the right mouse button.

Opening windows from the Window menu

- In any emulation window menu bar, select Window.
- Select the emulator.
- Select the window you want to open.

Creating and naming new windows

You can open several copies of certain windows, such as the Memory window.

1. In the menu bar, select File then select New Window. The new window will be given a number, such as Memory <<3>>.

2. (Optional) In the new window, select File then select Rename. Enter a new name for the window and select OK.

The new window can be opened from the Emulation Control Interface icon or from the Window menu.
To close windows

- In the menu bar, select File then select Close.

Auto-close

You can turn on Auto-close so that when you select a new window from the Window menu, the current window will be closed. To turn on Auto-close, select Window in the menu bar, then select Auto-Close [ON].

Deleting windows

When you create a new window using File->New, the window will still be listed in the Window menu after you have closed it. To delete the window from the list, open the window then select File->Delete.
Managing Run Control Tool Windows

- “To open windows” on page 54
- “To close windows” on page 56
- “To use the Status/Error Log window” on page 57

To use the Status/Error Log window

The Status/Error Log window is the central repository for error and status messages.

To specify when the Status/Error Log window appears:

1. Open the Status/Error Log window.

2. If you want the Status/Error Log window to automatically appear every time an error or status message occurs, select Yes for Popup dialog upon receiving error/status message?. If you only want the Status/Error Log window to appear when you open it, select No.

3. Close the Status/Error Log window.

Clear Messages empty the contents of the Status/Error Log window.

See Also

“Error/Status Messages” on page 64
To Use the Command Line Interface

1. Open the Command Line window.
2. Enter commands in the Command Input field.

   The help command provides command syntax and other information. For example, for help on the m command, enter the help m command.

   Enter comments by beginning a line with the # character.

3. Each line begins with a status prompt, such as M> or U>.
   - M> indicates your target system is in the background debug mode, not running user code.
   - U> indicates your target system is running user program code.

   For definitions of all of the status prompts you might see in the Command Line window, type, help proc.

The command line interface is useful, for example, for creating scripts that initialize the target system to a known state. For example, scripts can set initial register or memory values or write an I/O sequence.

Playback Script will execute commands that have been saved in a script file.

Edit Script opens a simple text editor that lets you enter and save a sequence of commands in a file.

Clear Buffer clears the Command Output buffer.

Memory Test... displays the Memory Test window where you can determine the condition of your target system memory hardware.

Use the up and down arrow keys to scroll through a list of the last 20 commands which you have entered.

Use the log (see page 60) command to save command line output to a file.

Example

To create a script that reads a range of memory addresses, writes
22334455 to those addresses, and then reads the memory address range again:

1. Select Edit Script.

2. In the Text Editor window, enter the commands:

3. Select the File->Save command from the menu bar, enter the name of your file in the File Selection dialog, and select OK.

4. Close the Text Editor window.

To play back the script just created:

1. Select Playback Script.

2. Select the name of the probe configuration settings script file in the File Selection dialog, and select OK.

The output will look similar to the following:
Chapter 1: Using the MPC800 Emulation Control Interface

Managing Run Control Tool Windows

Limitations

- Some of the commands listed when you type `help` are not available. In particular, the `pv` and `init` commands cannot be used.

- In demo mode, commands entered in this window (including the help command) have no effect.

See Also

- “To save command line output in a log file” on page 60
- “To create a script from a log file” on page 61
- “To cancel a command” on page 61
- “Timeouts in the Command Line window” on page 62
- “Commands not available in the Command Line window” on page 62
- “Testing Target System Memory” on page 72

To save command line output in a log file

1. In the Command Line window, enter the `log filename` command.
   
   Enter the full path for the log file.

2. Enter commands or playback a script. The commands and their output will
be saved to the file.

3. To stop logging to the file, enter the \texttt{log off} command.

\textbf{Example}

To save the values of all of the registers to the file "reglog01.log" enter:

\begin{verbatim}
log /logic/mylogs/reglog01.log
reg
log off
\end{verbatim}

\textbf{To create a script from a log file}

1. Select \textit{Edit Script}.
2. Select \textit{File->Load Script}, then choose the log file.
3. Edit the text to remove:
   - the ">" prompt
   - the output of the command
   - the "Logging to" line
   - the "Logging turned off" line
4. Save the script.

\textbf{To cancel a command}

Select the \textit{Cancel} button in the Busy dialog to cancel the currently executing command.

Commands that create no output cannot be cancelled and will cause the session to timeout (see page 62).

If you use the \texttt{rep} command to write a loop, make sure that there is a command in the loop which will generate some output.

For example, \texttt{rep 0 \{m 0..100=0\}} will repetitively write 0 to the entire memory range 0..100. Because the \texttt{m address=value} command produces no output, the command cannot be cancelled. By changing the command to \texttt{rep 0 \{m 0..100=0;echo .\}} a dot is written after every complete write of the range 0..100 and you will be able to
cancel the command.

**Timeouts in the Command Line window**

Timeout is one minute.

Every command must generate output that occurs at a rate of at least once a minute. Faster rates of output are desirable. For example, the command `w 70` (wait 70 seconds) will cause the system to timeout and should not be used. In addition, commands like `m 0..10000=0` which fill large segments of memory may cause the system to timeout. Use *Memory Fill* in the Memory Window to fill memory.

**Commands not available in the Command Line window**

Some of the commands listed when you type `help` in the Command Line window are not available.

The unavailable commands are:

- `cf_var`
- `cfsave`
- `cl`
- `dump`
- `end`
- `init`
- `lan`
- `load`
- `mo`
- `po`
- `pv`
- `sio`
- `sioget`
- `sioput`
Chapter 1: Using the MPC800 Emulation Control Interface
Managing Run Control Tool Windows

- slist
- start
- stty
- ureg
Error/Status Messages

Select the messages below for additional information on some of the error/status messages that can occur when using the Emulation Control Interface.

The following messages can appear in either the Error/Status Log window or in the Command Line Interface window.

- BDM Access Error (see page 65)
- Debugger: memory update (see page 65)
- Debugger: register update (see page 65)
- Hardware breakpoint (see page 66)
- Memory access failed (see page 66)
- MSR.RI bit not set - Break may not be recoverable (see page 66)
- Software breakpoint (see page 66)
- Stepping failed (see page 67)
- Target power is off (see page 67)
- Unable to break (see page 67)
- Unable to modify register: PC=value (see page 67)
- Unknown break cause (see page 67)
- Unimplemented opcode/register (see page 67)

The following error messages appear in the Breakpoint window:

- Address is a duplicate of Breakpoint #_ (see page 65)
- Address is not on an instruction (4-byte) boundary (see page 65)
- Enable breakpoint failed (see page 66)

See Also

“To use the Status/Error Log window” on page 57
Error/Status Messages

---

**Address is a duplicate of Breakpoint #_**

This error occurs if you try to set a breakpoint at an address which already has a breakpoint. Only one breakpoint is allowed at a given address.

---

**Address is not on an instruction (4-byte) boundary**

This error occurs if the address ends in a hexadecimal digit other than 0, 4, 8, or C. Breakpoints can only be set on 4-byte instruction boundaries.

---

**BDM Access Error**

This error occurs if there has been a communication error between the emulator and the target system.

Try the operation again. If the error occurs again, check all connections. If the emulator is connected directly to the target board (through a target interface module), check that the target's debug port meets the guidelines in the emulator *Installation and Service Guide*.

---

**Debugger: memory update**

This message occurs when target system memory has been modified by another external connection (presumed to be a debugger).

---

**Debugger: register update**

This message occurs when a processor register has been modified by
another external connection (presumed to be a debugger).

---

**Enable breakpoint failed**

This error occurs if an attempt to write a software breakpoint at the indicated memory address failed. If you are setting a software breakpoint, check that the address is in RAM, not ROM.

---

**Hardware breakpoint**

This message occurs when the processor has stopped at a hardware breakpoint at the specified address.

---

**Memory access failed**

This error occurs if the processor was unable to access memory (processor-generated exception).

---

**MSR.RI bit not set - Break may not be recoverable**

The emulator performed a non-maskable break. The MSR.RI bit was not set, so you may not be able to recover and continue program execution without resetting the target system.

---

**Software breakpoint:**

This message occurs when the processor has stopped at a software breakpoint at the specified address.
Chapter 1: Using the MPC800 Emulation Control Interface

Error/Status Messages

---

**Stepping failed**

This error occurs if the step command was unsuccessful because attempt to write to the program counter failed.

---

**Target power is off**

This error occurs if the emulator detects that power is off on the target processor.

---

**Unable to break**

This error occurs if the processor does not return a status message to the emulator indicating that it has stopped.

---

**Unable to modify register: PC=value**

This error occurs if a register command was unsuccessful because attempt to write to the program counter failed.

---

**Unknown break cause**

Something other than the emulator caused a break. Possible causes include a "break" button on the target system, a trap instruction, or noise in the target system circuitry.

---

**Unimplemented opcode/register**

This error occurs if the processor tried to execute an illegal instruction
Chapter 1: Using the MPC800 Emulation Control Interface

Error/Status Messages

(processor-generated exception).
To update firmware

Update the firmware if:

- The emulation module or emulation probe is being connected to a new analysis probe or TIM, or
- The emulation module was not shipped already installed in the logic analysis system, or
- You have an updated version of the firmware.

To install firmware from a CD-ROM to the hard disk

- Follow the instructions printed on the CD-ROM jacket.

  This will install the firmware onto the logic analysis system hard disk. Continue with either the "To update emulation module firmware" or the "To update emulation probe firmware" instructions.

To update emulation module firmware

1. End any emulation sessions that may be running. Remove any emulation module icons from the workspace in the Workspace window.

2. Install the firmware onto the logic analysis system's hard disk, if necessary.

3. In the system window, select the emulation module and select Update Firmware.

4. In the Update Firmware window, select the firmware version to load.

  Select the Additional Information button to display additional information about the firmware you selected.
To update firmware

5. Select Update Firmware.

To update emulation probe firmware

1. End any emulation sessions that may be running.
2. Install the firmware onto the logic analysis system's hard disk, if necessary.
3. In the workspace window, drag the emulation probe icon onto the workspace.
4. Select the emulation probe icon and select Update Firmware....
5. In the Update Firmware window, enter the LAN name or address of the emulation probe.
6. In the Update Firmware window, select the firmware version to load.
   Select the Additional Information button to display additional information about the firmware you selected.
7. Select Update Firmware.
8. Cycle power on the emulation probe.

To display the current firmware version

- Select Display Current Version to see what firmware version is already installed in your emulation module or emulation probe.

To install firmware from another source

If you obtained firmware from another source, such as from an ftp server or a World Wide Web site: //www.agilent.com/find/sw-updates

- Follow the instructions provided with the firmware, OR
- Copy the firmware files into /logic/run_cntrl/firmware on the logic analysis system hard disk. Be sure to copy all of the files which begin with the product number of the firmware for your microprocessor.
Chapter 1: Using the MPC800 Emulation Control Interface

Disconnecting from the Emulator

1. Select the Emulation Module icon or Emulation Control Interface icon and then select *Disconnect from Emulator*.

2. In the Connect - Emulator window, select *Disconnect from Emulator*. 
Testing Target System Memory

Many times when a system under test fails to operate as expected, you will need to determine whether the failure is in the hardware or the software. These tests verify operation of the memory hardware in the system under test.

To Test Target System Memory

1. Open the Memory Test window (see page 87).
2. Specify the Memory Test (see page 74) to be performed.
3. Specify the Memory Range (see page 86) to be tested.
4. Specify the Data Value (see page 86) to be used when performing the test.
5. Specify the Options (see page 86), number of times the test is to be repeated and type of error message information to be presented during the test.
6. Open the Command Line window if it’s not already open.
7. Select the Execute Test button.
8. View test results in the Command Line window.

The Command Input line in the Command Line window will show the equivalent terminal interface command during each test.

The Busy dialog box appears while the test executes. It also shows the equivalent terminal interface command during each test. A Cancel button in the Busy dialog box can be used to stop a test in progress.

To check untested memory hardware, follow the “Recommended Test Procedure” on page 87.

Example

The following example writes the value "55555555" to addresses 1000 through 1fff (hexadecimal). Then it compares the values in memory with the values that were written. Next, the test writes the compliment "aaaaaaaa" to the same range of addresses and compares the new
values in memory with the values that were written.

1. Open the Command Line window and select the Memory Test button.

2. In the Memory Test window, specify the memory test shown below.

3. Select the Execute Test button.

4. View the test results in the Command Line window.

See Also

- “Recommended Test Procedure” on page 87
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

Memory Test:

The Memory Test feature of the Emulation Control Interface can perform seven different types of tests. Use these tests to find problems in address lines, data lines, and data storage. Use these tests in combination because no single test can perform a complete evaluation of the target system memory.

Basic Pattern

- Use this test to validate data read-write lines.
- “To perform the Basic Pattern test” on page 75
- “How the Basic Pattern test works” on page 76

Address Pattern

- Use this test to validate address read-write lines.
- “To perform the Address Pattern test” on page 77
- “How the Address Pattern test works” on page 78

Rotate Pattern

- Use this test to validate data read-write lines, and test voltage and ground bounce.
- “To perform the Rotate Pattern test” on page 78
- “How the Rotate Pattern test works” on page 80

Walking Ones

- Use this test to validate individual storage bits in memory.
- “To perform the Walking Ones test” on page 81
- “How the Walking Ones test works” on page 81

Walking Zeros

- Use this test to validate individual storage bits in memory.
- “To perform the Walking Zeros test” on page 82
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

- “How the Walking Zeros test works” on page 83

Oscilloscope Read
- Use this test to generate the signals associated with reading from memory so they can be viewed on an oscilloscope.
- “To perform the Oscilloscope Read test” on page 83
- “How the Oscilloscope Read test works” on page 84

Oscilloscope Write
- Use this test to generate the signals associated with writing to memory so they can be viewed on an oscilloscope.
- “To perform the Oscilloscope Write test” on page 84
- “How the Oscilloscope Write test works” on page 85

---

**To perform the Basic Pattern test**

1. Open the *Memory Test* dialog box from either the Memory window or the Command Line window.
2. In the *Memory Test* dialog box, select the *Basic Pattern* Memory Test.
3. Type in the *Memory Range* to be tested.
4. Select the *Access Size* to be written, and type in the *Pattern* to be written.
5. Set *Repetitions* to 1. This causes the test to be performed one time.
6. Set *Verbosity* to *Ten Errors*. This is the recommended verbosity for the first test in a series of tests.
7. Select the *Execute Test* button.
8. When the test is complete, the Command Line window will show the test results. Error messages will be shown if any errors were found during the test.

This test verifies that the data lines of the selected memory range are without error. This test also checks the individual memory locations in the memory range specified.
Chapter 1: Using the MPC800 Emulation Control Interface
Testing Target System Memory

Consistent errors such as a particular bit incorrect every four bytes typically indicate a problem with the data lines. Random or sparse errors may indicate hardware data memory errors—check individual locations with the Walking Ones and Walking Zeros tests.

This test will halt and generate an error message if your Memory Range specification causes this test to be performed outside the range of valid memory in your target system.

This test will not halt but it will generate an error message if it is run on ROM or on locations with data line or location errors.

You can open a memory window on your logic analyzer to view the memory content. Expect to see the pattern and the compliment of the pattern that was specified.

NOTE:
This test will not always detect errors in the address lines. For example, if a bit in the address lines is stuck high or low, the Pattern Test will write to a different location in memory. Then the read from memory for comparison will also be made from that different location so the data will be correct. Use the Address Pattern test with this test to completely evaluate the memory range.

See Also:
• “How the Basic Pattern test works” on page 76
• “If problems were found by the Basic Pattern test” on page 88

How the Basic Pattern test works

This test writes the Pattern and the compliment of the Pattern to the Memory Range, and then compares the values in memory with what was written. The compliment of the Pattern and then the Pattern are then written, read, and compared.

Example:

<table>
<thead>
<tr>
<th>First Write/Read</th>
<th>Second Write/Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000 5555</td>
<td>AAAA</td>
</tr>
<tr>
<td>00000002 AAAA</td>
<td>5555</td>
</tr>
<tr>
<td>00000004 5555</td>
<td>AAAA</td>
</tr>
<tr>
<td>00000008 AAAA</td>
<td>5555</td>
</tr>
</tbody>
</table>
The Basic Pattern test finds data bits in memory that are stuck high or low. It also detects data lines that may be tied to power ground, or not connected at all.

To perform the Address Pattern test

1. Open the Memory Test dialog box from either the Memory window or the Command Line window.
2. In the Memory Test dialog box, select the Address Pattern Memory Test.
3. Type in the Memory Range to be tested.
4. Select the Access Size to be tested.
5. Set Repetitions to 1. This causes the test to be performed one time.
6. Set Verbosity to Ten Errors. This is the recommended verbosity for the first test in a series of tests.
7. Select the Execute Test button.
8. When the test is complete, the Command Line window will show the test results. Error messages will be shown if any errors were found during the test.

This test verifies that the address lines of the selected memory range are without error.

This test does not ensure that the data lines or individual data locations are without error. If a bit is stuck in a memory location, but is stuck in the written value, the stuck bit will not be detected.

You can view the memory in an analyzer memory window. You should see direct correlation between each address and the data stored at that address.

Consistent errors typically indicate problems in the address lines. This is especially likely if the results of the Basic Pattern test were without errors.

Errors in specific memory locations may indicate errors in the memory hardware instead of the address lines.
Testing Target System Memory

How the Address Pattern test works

This test writes the address of each memory location as data to each location. The data is then read back to see if it matches the addresses.

The pattern written to the memory is generated at the start of the test and is dependent upon the start address, access size, and the number of bytes in the memory range.

Depending on the last Access Size selected, subsets of the addresses may be written to memory. For example, if the last access size was 1 byte, address 00000001 will have 01 written to it, and address 00000002 will have 02 written to it.

For example, the data written in address 00001000 will look like this, depending on the last Access Size.

1 Byte = 00001000 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
2 Byte = 00001000 1000 1002 1004 1006 1008 100a 100c 100e
4 Byte = 00001000 00001000 00001004 00001008 0000100c
8 Byte = 00001000 00000000 0000001000 0000000001000

The upper four bytes of an 8 Byte access size are not tested for a 4 Byte address. The upper four bytes will always be zeros. Use a smaller access size to test these locations with the Address Pattern test.

Unless the access size is 1 Byte, the odd bits of the memory locations will not be tested. Use the Basic Pattern test to check the odd bits.

To perform the Rotate Pattern test

1. Open the Memory Test dialog box from either the Memory window or the Command Line window.
2. In the Memory Test dialog box, select the Rotate Pattern Memory Test.
3. Type in the Memory Range to be tested.

4. Select the Access Size to be written, and type in the Pattern to be written. A good pattern to use is 01, 0001, or 00000001.

5. Set Repetitions to 1. This causes the test to be performed one time.

6. Set Verbosity to Ten Errors. This is the recommended verbosity for the first test in a series of tests.

7. Select the Execute Test button.

8. When the test is complete, the Command Line window will show the test results. Error messages will be shown if any errors were found during the test.

This test can be used to test voltage and ground bounce problems associated with the selected memory range. This test verifies that the data lines of the selected memory range are without error. This test also checks the individual memory locations in the memory range specified.

Consistent errors such as a particular bit incorrect every four bytes typically indicate a problem with the data lines. Random or sparse errors may indicate hardware data memory errors—check individual locations with the Walking Ones and Walking Zeros tests.

This test will halt and generate an error message if your Memory Range specification causes this test to be performed outside the range of valid memory in your target system.

This test will not halt but it will generate an error message if it is run on ROM or on locations with data line or location errors.

You can open a memory window on your logic analyzer to view the memory content. Expect to see the pattern and the compliment of the pattern that was specified.

See Also:

- “How the Rotate Pattern test works” on page 80
How the Rotate Pattern test works

This test writes the Pattern and the compliment of the Pattern to the Memory Range, and then compares the values in memory with what was written. Next, the rotated Pattern and the rotated compliment of the Pattern are written, read, and compared. Now the Pattern is rotated again, and again it is written, read, and compared. This continues until the rotations of the pattern return it to its original arrangement. That constitutes one Repetition of the Rotate Pattern test.

Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>First Write/Read</th>
<th>Second Write/Read</th>
<th>Third Write/Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>01</td>
<td>FE</td>
<td>02</td>
</tr>
<tr>
<td>00000001</td>
<td>FE</td>
<td>02</td>
<td>FD</td>
</tr>
<tr>
<td>00000002</td>
<td>02</td>
<td>FD</td>
<td>04</td>
</tr>
<tr>
<td>00000003</td>
<td>FD</td>
<td>04</td>
<td>FB</td>
</tr>
<tr>
<td>00000004</td>
<td>04</td>
<td>FB</td>
<td>08</td>
</tr>
<tr>
<td>00000005</td>
<td>FB</td>
<td>08</td>
<td>F7</td>
</tr>
<tr>
<td>00000006</td>
<td>08</td>
<td>F7</td>
<td>10</td>
</tr>
<tr>
<td>00000007</td>
<td>F7</td>
<td>10</td>
<td>EF</td>
</tr>
<tr>
<td>00000008</td>
<td>10</td>
<td>EF</td>
<td>20</td>
</tr>
</tbody>
</table>

Larger Access Size selections take more time because they require more patterns to be written to all locations (2-byte Access Size requires writing 32 patterns, and 4-byte Access Size requires writing 64 patterns).

The Access Size you select will affect the appearance of memory when you view memory after a test. When a test is complete, memory contains the last set of patterns that was written to it. For example, consider the following listing from a Rotate Pattern test that was performed one time with an Access Size of 2 bytes, and an initial pattern of 0001.

What you see below is the 32nd set of patterns written to memory during the test.

```
00000000 7ffe 0001 fffe 0002 fffd 0004 fffb 0008
00000010 ffff 0010 fffe 0020 fffd 0040 fffb 0080
00000020 ffff 0100 fffe 0200 fffd 0400 fffb 0800
00000030 ffff 1000 fffe 2000 fffd 4000 fffb 8000
00000040 7fff 0001 ffe0 0002 ffd0 0004 ffbf 0008
00000050 ffff 0010 ffe0 0020 ffd0 0040 ffbf 0080
00000060 ffff 0100 ffe0 0200 ffd0 0400 ffbf 0800
```
The Rotate Pattern test finds data bits in memory that are stuck high or low. It also detects data lines that may be tied to power ground, or not connected at all. This test more than any other in this set of tests will detect problems with voltage and ground bounce associated with the selected memory range.

To perform the Walking Ones test

1. Open the Memory Test dialog box from either the Memory window or the Command Line window.
2. In the Memory Test dialog box, select the Walking Ones Memory Test.
3. Type in the Memory Range to be tested.
4. Select the Access Size to be tested.
5. Set Repetitions to 1. This causes the test to be performed one time.
6. Set Verbosity to Ten Errors. This is the recommended verbosity for the first test in a series of tests.
7. Select the Execute Test button.
8. When the test is complete, the Command Line window will show the test results. Error messages will be shown if any errors were found during the test. You can also select the Memory window to see the content of memory.

The Walking Ones test finds data bits stuck in logical "0".

See Also:
- “How the Walking Ones test works” on page 81

How the Walking Ones test works

This test cycles "1" through each bit position in memory, and checks results. It does this by writing and then reading a pattern sequence of ones and zeros from all memory locations in the range.

For example, the hexadecimal values 01, 02, 04, ... are written to each
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

location in the \textit{Memory Range}.

\begin{tabular}{lcccccccc}
\hline
Address & 1st & 2nd & 3rd & 4th & 5th & 6th & 7th & 8th \\
\hline
00000000 & 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 \\
00000001 & 02 & 04 & 08 & 10 & 20 & 40 & 80 & 01 \\
00000002 & 04 & 08 & 10 & 20 & 40 & 80 & 01 & 02 \\
00000003 & 08 & 10 & 20 & 40 & 80 & 01 & 02 & 04 \\
00000004 & 10 & 20 & 40 & 80 & 01 & 02 & 04 & 08 \\
\hline
\end{tabular}

\textbf{NOTE:}

1st, 2nd, 3rd, etc. are the first, second, third, etc. complete passes through the memory.

Larger \textit{Access Size} selections take more time because they require more patterns to be written to all locations (2-byte \textit{Access Size} requires writing 16 patterns, and 4-byte \textit{Access Size} requires writing 32 patterns).

\textbf{Example of 2-byte Access Size writing 16 patterns:}

\begin{tabular}{lcccccccccccccccc}
\hline
\hline
00000000 & 0001 & 0002 & 0004 & 0008 & 0010 & 0020 & 0040 & 0080 & 0100 & 0200 & 0400 & 0800 & 1000 & 2000 & 4000 & 8000 \\
00000001 & 0002 & 0004 & 0008 & 0010 & 0020 & 0040 & 0080 & 0100 & 0200 & 0400 & 0800 & 1000 & 2000 & 4000 & 8000 & 0001 \\
00000002 & 0004 & 0008 & 0010 & 0020 & 0040 & 0080 & 0100 & 0200 & 0400 & 0800 & 1000 & 2000 & 4000 & 8000 & 0002 & 0002 \\
00000003 & 0008 & 0010 & 0020 & 0040 & 0080 & 0100 & 0200 & 0400 & 0800 & 1000 & 2000 & 4000 & 8000 & 0004 & 0004 & 0004 \\
\hline
\end{tabular}

This test finds data bits stuck in logical "0".

---

To perform the Walking Zeros test

1. Open the \textit{Memory Test} dialog box from either the Memory window or the Command Line window.

2. In the \textit{Memory Test} dialog box, select the Walking Zeros Memory Test.

3. Type in the \textit{Memory Range} to be tested.

4. Select the \textit{Access Size} to be tested.

5. Set \textit{Repetitions} to 1. This causes the test to be performed one time.

6. Set \textit{Verbosity} to \textit{Ten Errors}. This is the recommended verbosity for the first test in a series of tests.

7. Select the \textit{Execute Test} button.

8. When the test is complete, the Command Line window will show the test...
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

results. Error messages will be shown if any errors were found during the test. You can also select the Memory window to see the content of memory.

The Walking Zeros test finds data bits stuck in logical "1".

See Also:

- “How the Walking Zeros test works” on page 83

---

How the Walking Zeros test works

This test cycles "0" through each bit position in memory, and checks results.

For example, the hex values FE, FD, FB, ... are written to each location in the Memory Range.

```
Address    1st 2nd 3rd 4th 5th 6th 7th 8th
00000000    FE  FD  FB  F7  EF  DF  BF  7F
00000001    FD  FB  F7  EF  DF  BF  7F  FE
00000002    FB  F7  EF  DF  BF  7F  FE  FD
00000003    F7  EF  DF  BF  7F  FE  FD  FB
00000004    EF  DF  BF  7F  FE  FD  FB  F7
```

NOTE:
1st, 2nd, 3rd, etc. are the first, second, third complete pass through the memory.

Larger Access Size selections take more time because they require more patterns to be written to all locations (2-byte Access Size requires writing 16 patterns, and 4-byte Access Size requires writing 32 patterns).

```
Example of 2-byte Access Size writing 16 patterns:
Address     1st   2nd   3rd   4th   5th   6th   7th   8th   9th   ...   16th
00000000    FFFE  FFFD  FFFB  FFF7  FFEF  FFD7  FFBF  FF7F  FFF7  ...  FFFE
00000001    FFFD  FFFB  FFF7  FFEF  FFD7  FFBF  FF7F  FFF7  FFFE  ...  FFFD
00000002    FFFB  FFF7  FFEF  FFD7  FFBF  FF7F  FFF7  FFFE  FFFD  ...  FFFB
00000003    FFF7  FFEF  FFD7  FFBF  FF7F  FFF7  FFFE  FFFD  FFFB  ...  FFFE
00000004    FFEF  FFD7  FFBF  FF7F  FFF7  FFFE  FFFD  FFFB  FFFE  ...  FFFE
```

This test finds data bits stuck in logical "1".

---

To perform the Oscilloscope Read test

1. Connect your oscilloscope to view signals on the lines to be tested. These
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

will be the signals generated to perform read transactions from the memory in your target system.

2. Open the Memory Test dialog box from either the Memory window or the Command Line window.

3. In the Memory Test dialog box, select the Oscilloscope Read Memory Test.

4. Type in the Memory Range to be read.

5. Select the Access Size to be read.

6. Set Repetitions to 0. This repeats the test continuously until you cancel the test.

7. Set Verbosity to Summary Only.

8. Select the Execute Test button.

9. When you have finished using your oscilloscope to view the read-from-memory signals, select the Cancel button in the Busy dialog box that appears on screen.

You will see an error message if your test attempts to read memory addresses outside the range of available memory.

See Also:

• “How the Oscilloscope Read test works” on page 84

How the Oscilloscope Read test works

This test repetitively reads the present content from the Memory Range for the number of Repetitions specified, typically reads continuously until cancelled.

The Oscilloscope Read test does not print or store the data it has read.

To perform the Oscilloscope Write test

1. Connect your oscilloscope to view signals on the lines to be tested. These will be the signals generated to perform write transactions to the memory

84
in your target system.

2. Open the Memory Test dialog box from either the Memory window or the Command Line window.

3. In the Memory Test dialog box, select the Oscilloscope Write Memory Test.

4. Type in the Memory Range to be written.

5. Select the Access Size to be written, and type in the Pattern to be written.

6. Set Repetitions to 0. This repeats the test continuously until you cancel the test.

7. Set Verbosity to Summary Only.

8. Select the Execute Test button.

9. When you have finished using your oscilloscope to view the write-to-memory signals, select the Cancel button in the Busy dialog box that appears on screen.

You will see an error message if your test attempts to write to memory addresses outside the range of available memory.

See Also:

- “How the Oscilloscope Write test works” on page 85

---

How the Oscilloscope Write test works

- This test repetitively writes your selected Pattern to the Memory Range for the number of Repetitions specified, typically continuously until cancelled.

- If your pattern is larger than the access size, it will be truncated to fit. If your pattern is smaller than the access size, it will be zero-padded to fit.

- This test does not generate error messages for unsuccessful write transactions, such as writes to ROM.

- If desired, you can open a memory window in the logic analyzer and view the memory where the pattern was written. If the memory is ROM or if it contains errors, it may not contain the pattern that was written.
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

Memory Range

A memory test can be performed in one range of memory addresses in your system under test.

1. Type the lowest address value of the range to be tested in the Starting Address text box.

2. Type the highest address value of the range in the Ending Address text box.

**NOTE:** If you wish to test only one location, enter the address in both the Starting Address and Ending Address text boxes.

Data Value

- *Access Size:* Specify the desired memory access size. If you are performing the Address Pattern test, the Access Size you choose will affect the value that is written to each memory location.

  If you are performing the Walking Ones, Walking Zeros, or Rotate Pattern test, larger access sizes will take more time because more patterns will be written to the memory locations.

- *Pattern:* Specify the pattern (in hexadecimal) to be used when performing the Oscilloscope Write, Basic Pattern, or Rotate Pattern test.

Options

- *Repetitions:* Enter the number of times you want the memory test to be repeated in the Repetitions text box. The maximum number of Repetitions is 10,000.

  If you enter Repetitions: 0, the test activity will run continuously until you select the Cancel button in the Busy dialog box. This is the normal selection when performing the Oscilloscope Read or Oscilloscope Write test.
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

- **Verbosity**: Select the type of error message presentation desired in the Verbosity selection box. The available options are:

  **Summary Only**
  
  Show no error messages during the tests. At the end of the last repetition, show a summary of the errors that were found during the tests.

  **Repetition Summary**
  
  Show no error messages during the tests. At the end of each repetition, show a summary of the errors that were found.

  **Ten Errors**
  
  Show error messages for the first ten errors found during each repetition. Then complete the repetition, but show no more error messages. At the end of the tests, show a summary of the errors that were found.

  **All Errors**
  
  Show a complete error message each time an error condition is found. At the end of the tests, show a summary of the errors that were found.

---

**To open the Memory Test window**

There are two ways to open the Memory Test window.

- Open the Command Line window and select the *Memory Test* button.
- Open the Memory window and select the *Memory Test* button.

---

**Recommended Test Procedure**

Two types of tests are offered for testing target memory: oscilloscope tests, and memory functionality tests.
Oscilloscope Tests

1. Connect the oscilloscope to view activity on the bits of interest.

2. Start an Oscilloscope Read (see page 83) or Oscilloscope Write (see page 84) test, as desired.

The test activity will be written onto the bits you specified continuously until you cancel the test.

Use both the Oscilloscope Read test and the Oscilloscope Write test to thoroughly check the connections of interest.

Memory Functionality Tests

1. Run the Basic Pattern (see page 75) test on the entire Memory Range.

   Result:
   - No Problems. Perform the Address Pattern (see page 77) test next.
   - Problems found. Refer to “If problems were found by the Basic Pattern test” on page 88.

2. Run the Address Pattern (see page 77) test on the entire Memory Range.

   Result:
   - No Problems.
   - Problems found. Refer to “If problems were found by the Address Pattern test” on page 89.

If no problems were found by the Basic Pattern test and the Address Pattern test above, you can ignore the rest of the tests. The memory in your system has been tested thoroughly and it is good.

If problems were found by the Basic Pattern test

Below are two examples of problems found by the Basic Pattern test:

- If there is a consistent error, such as:

```
Starting: Basic Pattern Test
Error: 1 at address 00000200:
Read 5557  (0101 0101 0101 0111)
Expected 5555  (0101 0101 0101 0101)
```
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

Error: 2 at address 00000204:
Read  5557  (0101 0101 0101 0111)
Expected  5555  (0101 0101 0101 0101)
Error: 3 at address 00000208:
Read  5557  (0101 0101 0101 0111)
Expected  5555  (0101 0101 0101 0101)
Error: ...
Read ...
Expected ...

Assume the data line bit associated with the error is stuck high. This could happen if the suspected data line bit were soldered to power.

NOTE:

For an additional test of suspected memory, perform the Walking Ones (see page 81) and Walking Zeros (see page 82) tests on the problem memory range.

• If there are random errors, such as indicated by the following output:

Starting: Basic Pattern Test
Error: 1 at address 00000200:
Read  8000  (1000 0000 0000 0000)
Expected  0000  (0000 0000 0000 0000)
Error: 2 at address 000004a2:
Read  efff  (1110 1111 1111 1111)
Expected  ffff  (1111 1111 1111 1111)
Repetition: 1 - FAILED found 2 errors
Completed: Basic Pattern Test
Summary: 1 of 1 - FAILED (2 errors total)

From the above listing, we assume there are two location errors in memory. At location 200, there is a bit stuck high. At location 4a0, there is bit stuck low. Use the Walking Ones and Walking Zeros tests to verify the errors.

There is one bit stuck high at location 200 so the Walking Zeros test will print one error message when it tests this location. Use the Walking Ones test to isolate the bit that is stuck low at location 4a0. Again, this will print only one error message.

See Also:

• “If problems were found by the Address Pattern test” on page 89

If problems were found by the Address Pattern test

You may see no errors in the Basic Pattern test, but errors in the Address Pattern test. For example, you might see the following result in the Address Pattern test:
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory

Error: 1 at address 00000000:
Read 0020 (0000 0000 0010 0000)
Expected 0000 (0000 0000 0000 0000)

Error: 2 at address 00000002:
Read 0022 (0000 0000 0010 0010)
Expected 0002 (0000 0000 0000 0010)

You would see that the data stored at locations 00 through 0f is the data that should be at locations 20 through 2f. This indicates an address line problem. Address bit 5 must be stuck low because the addresses that should have been written to the range 20 through 2f were written instead to 00 through 0f.

NOTE:
Random errors typically do not indicate address line errors. Use the Walking Ones (see page 81) and Walking Zeros (see page 82) tests to check the locations of random errors.

See Also:
• “If problems were found by the Basic Pattern test” on page 88

To decide if you have an E5900A or an E5900B

There are two kinds of emulation probes: the E5900A and the E5900B. Use the following information to decide which kind you have.

E5900A
• Label: E5900A or no product number
• Target connection: through target interface module (TIM) or analysis probe

E5900B
• Label: E5900B
• Target connection: directly to background debug port on the target system or on an analysis probe
• Host connection: Often connects to logic analysis system through a LAN connection and through a module/probe interconnect cable attached to an E5901B emulation module
To decide if you have an E5901A or an E5901B

There are two kinds of emulation modules: the E5901A and the E5901B. Use the following information to decide which kind you have.

**E5901A**

- Label: E5901A or HP 16610A
- Connects to: target interface module (TIM) or analysis probe

**E5901B**

- Label: E5901B
- Connects to: E5900B emulation probe
Chapter 1: Using the MPC800 Emulation Control Interface

Testing Target System Memory
**Glossary**

**absolute** Denotes the time period or count of states between a captured state and the trigger state. An absolute count of -10 indicates the state was captured ten states before the trigger state was captured.

**acquisition** Denotes one complete cycle of data gathering by a measurement module. For example, if you are using an analyzer with 128K memory depth, one complete acquisition will capture and store 128K states in acquisition memory.

**analysis probe** A probe connected to a microprocessor or standard bus in the device under test. An analysis probe provides an interface between the signals of the microprocessor or standard bus and the inputs of the logic analyzer. Also called a preprocessor.

**analyzer 1** In a logic analyzer with two machines, refers to the machine that is on by default. The default name is Analyzer<N>, where N is the slot letter.

**analyzer 2** In a logic analyzer with two machines, refers to the machine that is off by default. The default name is Analyzer<N2>, where N is the slot letter.

**arming** An instrument tool must be armed before it can search for its trigger condition. Typically, instruments are armed immediately when Run or Group Run is selected. You can set up one instrument to arm another using the Intermodule Window. In these setups, the second instrument cannot search for its trigger condition until it receives the arming signal from the first instrument. In some analyzer instruments, you can set up one analyzer machine to arm the other analyzer machine in the Trigger Window.

**asterisk (*)** See edge terms, glitch, and labels.

**bits** Bits represent the physical logic analyzer channels. A bit is a channel that has or can be assigned to a label. A bit is also a position in a label.

**card** This refers to a single instrument intended for use in the Agilent Technologies 16700A/B-series mainframes. One card fills one slot in the mainframe. A module may comprise a single card or multiple cards cabled together.

**channel** The entire signal path from the probe tip, through the cable and module, up to the label grouping.

**click** When using a mouse as the
pointing device, to click an item, position the cursor over the item. Then quickly press and release the left mouse button.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.

clock channel A logic analyzer channel that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a data channel, except in the Agilent Technologies 16517A.
Glossary

instrument tool. Multiple data sets can be displayed together when sourced into a single display tool. The Filter tool is used to pass on partial data sets to analysis or display tools.

debug mode  See monitor.

delay  The delay function sets the horizontal position of the waveform on the screen for the oscilloscope and timing analyzer. Delay time is measured from the trigger point in seconds or states.

demo mode  An emulation control session which is not connected to a real target system. All windows can be viewed, but the data displayed is simulated. To start demo mode, select Start User Session from the Emulation Control Interface and enter the demo name in the Processor Probe LAN Name field. Select the Help button in the Start User Session window for details.

deskewing  To cancel or nullify the effects of differences between two different internal delay paths for a signal. Deskewing is normally done by routing a single test signal to the inputs of two different modules, then adjusting the Intermodule Skew so that both modules recognize the signal at the same time.

device under test  The system under test, which contains the circuitry you are probing. Also known as a target system.

don't care  For terms, a "don't care" means that the state of the signal (high or low) is not relevant to the measurement. The analyzer ignores the state of this signal when determining whether a match occurs on an input label. "Don't care" signals are still sampled and their values can be displayed with the rest of the data. Don't cares are represented by the X character in numeric values and the dot (.) in timing edge specifications.

dot (.)  See edge terms, glitch, labels, and don't care.

double-click  When using a mouse as the pointing device, to double-click an item, position the cursor over the item, and then quickly press and release the left mouse button twice.

drag and drop  Using a Mouse: Position the cursor over the item, and then press and hold the left mouse button. While holding the left mouse button down, move the mouse to drag the item to a new location. When the item is positioned where you want it, release the mouse button.
Glossary

Using the Touchscreen:
Position your finger over the item, then press and hold finger to the screen. While holding the finger down, slide the finger along the screen dragging the item to a new location. When the item is positioned where you want it, release your finger.

**edge mode**  In an oscilloscope, this is the trigger mode that causes a trigger based on a single channel edge, either rising or falling.

**edge terms**  Logic analyzer trigger resources that allow detection of transitions on a signal. An edge term can be set to detect a rising edge, falling edge, or either edge. Some logic analyzers can also detect no edge or a glitch on an input signal. Edges are specified by selecting arrows. The dot (.) ignores the bit. The asterisk (*) specifies a glitch on the bit.

**emulation module**  A module within the logic analysis system mainframe that provides an emulation connection to the debug port of a microprocessor. An E5901A emulation module is used with a target interface module (TIM) or an analysis probe. An E5901B emulation module is used with an E5900A emulation probe.

**emulation probe**  The stand-alone equivalent of an *emulation module*. Most of the tasks which can be performed using an emulation module can also be performed using an emulation probe connected to your logic analysis system via a LAN.

**emulator**  An *emulation module* or an *emulation probe*.

**Ethernet address**  See *link-level address*.

**events**  Events are the things you are looking for in your target system. In the logic analyzer interface, they take a single line. Examples of events are \( \text{Label1} = XX \) and \( \text{Timer 1} > 400 \text{ ns} \).

**filter expression**  The filter expression is the logical OR combination of all of the filter terms. States in your data that match the filter expression can be filtered out or passed through the Pattern Filter.

**filter term**  A variable that you define in order to specify which states to filter out or pass through. Filter terms are logically OR’ed together to create the filter expression.

**Format**  The selections under the logic analyzer *Format* tab tell the
logic analyzer what data you want to collect, such as which channels represent buses (labels) and what logic threshold your signals use.

frame The Agilent Technologies or 16700A/B-series logic analysis system mainframe. See also logic analysis system.

gateway address An IP address entered in integer dot notation. The default gateway address is 0.0.0.0, which allows all connections on the local network or subnet. If connections are to be made across networks or subnets, this address must be set to the address of the gateway machine.

glitch A glitch occurs when two or more transitions cross the logic threshold between consecutive timing analyzer samples. You can specify glitch detection by choosing the asterisk (*) for edge terms under the timing analyzer Trigger tab.

grouped event A grouped event is a list of events that you have grouped, and optionally named. It can be reused in other trigger sequence levels. Only available in Agilent Technologies 16715A or higher logic analyzers.

held value A value that is held until the next sample. A held value can exist in multiple data sets.

immediate mode In an oscilloscope, the trigger mode that does not require a specific trigger condition such as an edge or a pattern. Use immediate mode when the oscilloscope is armed by another instrument.

interconnect cable Short name for module/probe interconnect cable.

intermodule bus The intermodule bus (IMB) is a bus in the frame that allows the measurement modules to communicate with each other. Using the IMB, you can set up one instrument to arm another. Data acquired by instruments using the IMB is time-correlated.

intermodule Intermodule is a term used when multiple instrument tools are connected together for the purpose of one instrument arming another. In such a configuration, an arming tree is developed and the group run function is designated to start all instrument tools. Multiple instrument configurations are done in the Intermodule window.

internet address Also called Internet Protocol address or IP address. A 32-bit network address. It
is usually represented as decimal numbers separated by periods; for example, 192.35.12.6. Ask your LAN administrator if you need an internet address.

**labels**  Labels are used to group and identify logic analyzer channels. A label consists of a name and an associated bit or group of bits. Labels are created in the Format tab.

**line numbers**  A line number (Line #s) is a special use of *symbols*. Line numbers represent lines in your source file, typically lines that have no unique symbols defined to represent them.

**link-level address**  Also referred to as the Ethernet address, this is the unique address of the LAN interface. This value is set at the factory and cannot be changed. The link-level address of a particular piece of equipment is often printed on a label above the LAN connector. An example of a link-level address in hexadecimal: 0800090012AB.

**local session**  A local session is when you run the logic analysis system using the local display connected to the product hardware.

**logic analysis system**  The Agilent Technologies 16700A/B-series mainframes, and all tools designed to work with it. Usually used to mean the specific system and tools you are working with right now.

**machine**  Some logic analyzers allow you to set up two measurements at the same time. Each measurement is handled by a different machine. This is represented in the Workspace window by two icons, differentiated by a 1 and a 2 in the upper right-hand corner of the icon. Logic analyzer resources such as pods and trigger terms cannot be shared by the machines.

**markers**  Markers are the green and yellow lines in the display that are labeled *x*, *o*, *G1*, and *G2*. Use them to measure time intervals or sample intervals. Markers are assigned to patterns in order to find patterns or track sequences of states in the data. The *x* and *o* markers are local to the immediate display, while *G1* and *G2* are global between time correlated displays.

**master card**  In a module, the master card controls the data acquisition or output. The logic analysis system references the module by the slot in which the master card is plugged. For example, a 5-card Agilent Technologies 16555D would be referred to as Slot C.
**Glossary**

*machine* because the master card is in slot C of the mainframe. The other cards of the module are called *expansion cards*.

**menu bar** The menu bar is located at the top of all windows. Use it to select *File* operations, tool or system *Options*, and tool or system level *Help*.

**message bar** The message bar displays mouse button functions for the window area or field directly beneath the mouse cursor. Use the mouse and message bar together to prompt yourself to functions and shortcuts.

**module/probe interconnect cable**

The module/probe interconnect cable connects an E5901B emulation module to an E5900B emulation probe. It provides power and a serial connection. A LAN connection is also required to use the emulation probe.

**module** An instrument that uses a single timebase in its operation. Modules can have from one to five cards functioning as a single instrument. When a module has more than one card, system window will show the instrument icon in the slot of the *master card*.

**monitor** When using the Emulation Control Interface, running the monitor means the processor is in debug mode (that is, executing the debug exception) instead of executing the user program.

**panning** The action of moving the waveform along the timebase by varying the delay value in the Delay field. This action allows you to control the portion of acquisition memory that will be displayed on the screen.

**pattern mode** In an oscilloscope, the trigger mode that allows you to set the oscilloscope to trigger on a specified combination of input signal levels.

**pattern terms** Logic analyzer resources that represent single states to be found on labeled sets of bits; for example, an address on the address bus or a status on the status lines.

**period (.)** See *edge terms*, *glitch*, *labels*, and *don’t care*.

**pod pair** A group of two pods containing 16 channels each, used to physically connect data and clock signals from the unit under test to the analyzer. Pods are assigned by pairs in the analyzer interface. The number of pod pairs available is determined
Glossary

by the channel width of the instrument.

**pod** See **pod pair**

**point** To point to an item, move the mouse cursor over the item, or position your finger over the item.

**preprocessor** See **analysis probe**.

**primary branch** The primary branch is indicated in the **Trigger sequence step** dialog box as either the **Then find** or **Trigger on** selection. The destination of the primary branch is always the next state in the sequence, except for the Agilent Technologies 16517A. The primary branch has an optional occurrence count field that can be used to count a number of occurrences of the branch condition. See also **secondary branch**.

**probe** A device to connect the various instruments of the logic analysis system to the target system. There are many types of probes and the one you should use depends on the instrument and your data requirements. As a verb, "to probe" means to attach a probe to the target system.

**processor probe** See **emulation probe**.

**range terms** Logic analyzer resources that represent ranges of values to be found on labeled sets of bits. For example, range terms could identify a range of addresses to be found on the address bus or a range of data values to be found on the data bus. In the trigger sequence, range terms are considered to be true when any value within the range occurs.

**relative** Denotes time period or count of states between the current state and the previous state.

**remote display** A remote display is a display other than the one connected to the product hardware. Remote displays must be identified to the network through an address location.

**remote session** A remote session is when you run the logic analyzer using a display that is located away from the product hardware.

**right-click** When using a mouse for a pointing device, to right-click an item, position the cursor over the item, and then quickly press and release the **right mouse button**.

**sample** A data sample is a portion of a **data set**, sometimes just one point. When an instrument samples the target system, it is taking a single
measurement as part of its data acquisition cycle.

**Sampling** Use the selections under the logic analyzer Sampling tab to tell the logic analyzer how you want to make measurements, such as State vs. Timing.

**secondary branch** The secondary branch is indicated in the Trigger sequence step dialog box as the Else on selection. The destination of the secondary branch can be specified as any other active sequence state. See also primary branch.

**session** A session begins when you start a local session or remote session from the session manager, and ends when you select Exit from the main window. Exiting a session returns all tools to their initial configurations.

**skew** Skew is the difference in channel delays between measurement channels. Typically, skew between modules is caused by differences in designs of measurement channels, and differences in characteristics of the electronic components within those channels. You should adjust measurement modules to eliminate as much skew as possible so that it does not affect the accuracy of your measurements.

**state measurement** In a state measurement, the logic analyzer is clocked by a signal from the system under test. Each time the clock signal becomes valid, the analyzer samples data from the system under test. Since the analyzer is clocked by the system, state measurements are synchronous with the test system.

**store qualification** Store qualification is only available in a state measurement, not timing measurements. Store qualification allows you to specify the type of information (all samples, no samples, or selected states) to be stored in memory. Use store qualification to prevent memory from being filled with unwanted activity such as no-ops or wait-loops. To set up store qualification, use the While storing field in a logic analyzer trigger sequence dialog.

**subnet mask** A subnet mask blocks out part of an IP address so that the networking software can determine whether the destination host is on a local or remote network. It is usually represented as decimal numbers separated by periods; for example, 255.255.255.0. Ask your LAN administrator if you need a the subnet mask for your network.
symbols  Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:

- Object file symbols - Symbols from your source code, and symbols generated by your compiler. Object file symbols may represent global variables, functions, labels, and source line numbers.

- User-defined symbols - Symbols you create.

Symbols can be used as pattern and range terms for:

- Searches in the listing display.
- Triggering in logic analyzers and in the source correlation trigger setup.
- Qualifying data in the filter tool and system performance analysis tool set.

system administrator  The system administrator is a person who manages your system, taking care of such tasks as adding peripheral devices, adding new users, and doing system backup. In general, the system administrator is the person you go to with questions about implementing your software.

target system  The system under test, which contains the microprocessor you are probing.

terms  Terms are variables that can be used in trigger sequences. A term can be a single value on a label or set of labels, any value within a range of values on a label or set of labels, or a glitch or edge transition on bits within a label or set of labels.

TIM  A TIM (Target Interface Module) makes connections between the cable from the emulation module or emulation probe and the cable to the debug port on the system under test.

time-correlated  Time correlated measurements are measurements involving more than one instrument in which all instruments have a common time or trigger reference.

timer terms  Logic analyzer resources that are used to measure the time the trigger sequence remains within one sequence step, or a set of sequence steps. Timers can be used to detect when a condition lasts too long or not long enough. They can be used to measure pulse duration, or duration of a wait loop. A single timer term can be used to delay trigger until a period of time after detection of a significant event.
Glossary

**timing measurement**  In a timing measurement, the logic analyzer samples data at regular intervals according to a clock signal internal to the timing analyzer. Since the analyzer is clocked by a signal that is not related to the system under test, timing measurements capture traces of electrical activity over time. These measurements are asynchronous with the test system.

**tool icon**  Tool icons that appear in the workspace are representations of the hardware and software tools selected from the toolbox. If they are placed directly over a current measurement, the tools automatically connect to that measurement. If they are placed on an open area of the main window, you must connect them to a measurement using the mouse.

**toolbox**  The Toolbox is located on the left side of the main window. It is used to display the available hardware and software tools. As you add new tools to your system, their icons will appear in the Toolbox.

**tools**  A tool is a stand-alone piece of functionality. A tool can be an instrument that acquires data, a display for viewing data, or a post-processing analysis helper. Tools are represented as icons in the main window of the interface.

**trace**  See acquisition.

**trigger sequence**  A trigger sequence is a sequence of events that you specify. The logic analyzer compares this sequence with the samples it is collecting to determine when to trigger.

**trigger specification**  A trigger specification is a set of conditions that must be true before the instrument triggers.

**trigger**  Trigger is an event that occurs immediately after the instrument recognizes a match between the incoming data and the trigger specification. Once trigger occurs, the instrument completes its acquisition, including any store qualification that may be specified.

**workspace**  The workspace is the large area under the message bar and to the right of the toolbox. The workspace is where you place the different instrument, display, and analysis tools. Once in the workspace, the tool icons graphically represent a complete picture of the measurements.

**zooming**  In the oscilloscope or timing analyzer, to expand and contract the waveform along the time base by varying the value in the s/Div
Glossary

field. This action allows you to select specific portions of a particular waveform in acquisition memory that will be displayed on the screen. You can view any portion of the waveform record in acquisition memory.
**Index**

A
- aborting a command, 61
- access size, I/O window, 34
- access size, memory, 30
- access space, I/O window, 34
- Address is a duplicate of Breakpoint error, 65
- Address is not on a 4-byte instruction boundary error, 65
- addresses, in emulation control interface, 33
- addressing modes, 33
- AMD flash parts, 46
- analysis probe, connecting to emulator to, 14

B
- background monitor, breaking to, 23
- base of data, emulation, 30
- base of data, I/O window, 34
- basic pattern test, how it works, 78
- BNC, Break In, emulation probe, 18
- BNC, Trigger Out, emulation probe, 17
- break from reset, automatic register initialization, 20
- Break In port, emulation probe, 18
- break on external trigger, 52
- break to background, 23
- break type, emulator, 19
- break, triggering an analyzer on a, 53
- breakpoints, debugger, 24
- breakpoints, emulation, 24
- breakpoints, hardware, 26
- breakpoints, restoration on RESET, 20
- breaks, more specific than debug registers, 52

C
- Clear Buffer button in I/O window, 34
- clock speed, specifying, 17
- command line interface, cancelling a command, 61
- command line interface, creating script from log file, 61
- command line interface, emulation commands, 62
- command line interface, log file, 60
- command line interface, loops, 61
- command line interface, time out, 62
- command line interface, wait command, 62
- common tasks in the Run Control Tool interface, 57
- coordinated trace measurements, making, 52

D
- Debugger: memory update message, 65
- Debugger: register update message, 65
- DER register, initialization by emulator, 20
- disassembled mnemonic memory display, 32
- download executable to target, 35

E
- E2476A, 10
- E2476A analysis probe, connecting to emulator, 14
- E2476B analysis probe, connecting to emulator, 14
- E3497A, 10
- E5900A, identifying, 90
- E5901A, identifying, 91
- Edit Script button, emulation, 58
- editing registers, 29
- emulation module, at a glance, 10
- emulation module, connecting to analysis probe, 14
- emulation module, connecting to target, 12, 13
- emulation module, identifying, 91
- emulation module, processor personality, 69
- emulation probe, at a glance, 10
- emulation probe, connecting to analysis probe, 14
- emulation probe, connecting to target, 12, 13
- emulation probe, identifying, 90
- emulation, configuring, 16
- emulator command line interface, using the, 58
- emulator copies of registers, 20
- emulator setup, 16
- emulator, connecting emulator to target, 12
- emulator, disconnecting from, 71
- Enable breakpoint failed message, 66
- error log, emulation, 57
- error messages, emulation, 64
- errors, file download, 40
- example memory test, 72

F
- file format, for downloading executable, 37
- files, accessing for download to target, 37
- files, downloading to target, 35
- filling memory, 31
- firmware, updating emulation module or probe, 69
- flash ROM, erasing with emulator, 39
flash ROM, programming
algorithms, 44
flash ROM, programming with
emulator, 38
format, addresses, 33
Fujitsu flash parts, 50
G
groups of registers, 28
H
Hardware breakpoint message, 66
hardware breakpoints, 24
Hitachi flash parts, 49
I
I/O space, memory-mapped in I/O
window, 34
I/O, displaying and modifying with
emulator, 34
if problems were found by the
address pattern test, 89
if problems were found by the basic
pattern test, 88
IMMR register, initialization by
emulator, 20
Intel extended hex file format, 43
Intel flash parts, 47
intermodule measurements,
emulator break, 19
L
Load Processor button, 20
logic analysis, 52
M
memory parts, programming with
emulator, 35
memory test runs longer than
expected, 86
memory test, memory view
unexpected, 86
Memory Write window, 31
memory, displaying, 30
memory, displaying and modifying,
30
memory, displaying in mnemonic
format, 32
memory, filling a range of locations,
31
memory, modifying, 31
memory-mapped I/O space, 34
messages, emulation error/status,
57
Micron flash parts, 50
Mitsubishi flash parts, 49
mnemonic memory display, 32
monitor cycles, omitting from
trace, 54
monitor, break into on Break In
port signal, 18
monitor, break into on external
trigger, 52
monitor, RESET and breaks into,
20
monitor, trigger analyzer on break
into, 53
monitor, Trigger Out port when in,
17
Motorola S-record file format, 42
MPC 8xx emulation, at a glance, 10
N
number bases, I/O window, 34
O
oscilloscope read test, how it
works, 84
oscilloscope write test, how it
works, 85
P
Playback Script button, emulation,
58
power, target system status, 23
probe configuration options,
changing, 16
processor execution, controlling,
23
Q
qualified clock (logic analyzer), and
emulator, 17
R
RAM, downloading to, 36
read breakpoints button, emulation
Breakpoint window, 24
Read Processor Setup button, 20
Read Registers button, Register
window, 28
refresh button, emulation
Breakpoint window, 24
register classes, 28
register groups, 28
register initialization, 20
registers, configuration, 20
registers, displaying, 28
registers, displaying and
modifying, 28
registers, modifying contents, 29
registers, refreshing the display, 28
reset processor, 23
RESET, specifying action on, 20
rotate pattern test, how it works,
80
Run Control Tool interface,
windows, 57
Run Control Tool windows,
opening, 54
run control window, 23
Index

S
- scripts, emulation command line interface, 58
- selecting error message style in memory tests, 86
- selecting memory range to be tested, 86
- selecting the data value used in a memory test, 86
- session, ending emulation, 71
- SGS-Thomson flash parts, 50
- Sharp flash parts, 51
- single-step through instruction execution, 23
- slow clock message, after emulator break, 53
- Software breakpoint, 66
- software breakpoints, 24, 27
- specifying number of memory tests to be performed, 86
- S-record file format, 42
- status line, run control, 23
- status messages, emulation, 57, 64
- step through instruction execution, 23
- Stepping failed, 67
- SYPCR register, initialization by emulator, 20

T
- table, flash algorithms, 45
- Target power is off, 67
- target system, connecting emulator to, 12, 13
- testing target memory, recommended procedure, 87
- testing target system memory, 72
- text editor, emulation command line, 58
- TI flash parts, 49
- to open the Memory Test window, 87
- to perform the address pattern test, 77
- to perform the basic pattern memory test, 75
- to perform the oscilloscope read test, 83
- to perform the oscilloscope write test, 84
- to perform the rotate pattern test, 78
- to perform the walking ones test, 81
- to perform the walking zeros test, 82
- trace measurements, coordinating, 52
- trace, omitting monitor cycles, 54
- Trigger Out port, emulation probe, 17
- trigger output, to emulation probe, 18
- trigger, on emulation break, 53
- types of memory tests, 74

U
- Unable to break, 67
- Unable to modify register, 67
- unavailable commands, emulation command line, 62
- unsupported flash parts, 51

W
- walking ones test, how it works, 81
- walking zeros test, how it works, 83
- watchdog timers in target systems, 17
- windows (in the Run Control Tool), opening, 54
- windows, cloning, 55
- windows, closing emulation, 56
- windows, opening and closing, 57
Index