

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

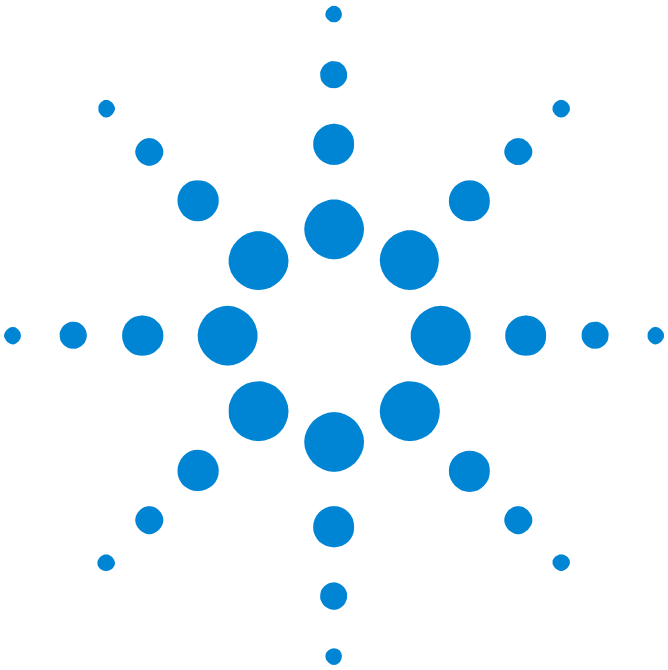
Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keysight / Agilent E2940A* at our website: ***Click HERE***



Agilent E2940A Opt. 300 PCI Exerciser

User's Guide

Agilent Technologies

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Important Notice

This document contains propriety information that is protected by copyright. All rights are reserved. Neither the documentation nor software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Agilent Technologies.

© Copyright 1999, 2000 by:
Agilent Technologies
Herrenberger Straße 130
D-71034 Böblingen
Germany

The information in this manual is subject to change without notice. Agilent Technologies makes no warranty of any kind with regard to this manual, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Agilent Technologies shall not be liable for errors contained herein or direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of this manual.

Brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Authors: Stephan Greisinger and Stefan Kunzi, t3 medien GmbH

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

PCI Exerciser Overview 7

Gaining an Active Part in Optimization and Validation 8
PCI Exerciser User Interface 10
PCI Exerciser Configurations 12
Setting Up a PCI Exerciser Test 14
What is a Transaction? 14

Running A Sample PCI Exerciser Session 17

Example Scenarios 17
Preparing for the Guided Tour 18
Guided Tour: Generating Master Transactions 20
- Loading the Setup Files 20
- Setting up a Master Transaction 20
- Specifying a Conditional Start 21
- Viewing the Results 22

Guided Tour: Specifying Custom Target Behavior 24
- Loading the Setup Files 24
- Setting up the Target Decoders 24
- Specifying the Target Protocol Behavior 26
- Viewing the Results 26

Guided Tour: Accessing VGA Frame Buffer Memory 27
- Starting the Command Line Interface 28
- Entering the Required CLI Commands 28

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The PCI Exerciser as a Master Device 31

Programming Master Transactions 32
Master Transactions Overview 33
Transaction Properties 36
Implementing Master Transaction Scripts 39
Controlling Master Attributes 40
Specifying Master Attributes 40
Master Address Phase Attributes 42
Master Data Phase Attributes 44
Master Control Attributes 46
Implementing Master Attribute Scripts 47
Running the Master 48
Preparing Test Execution 48
Starting the Master 51
Stopping the Master 52

The PCI Exerciser as a Target Device 53

Configuration Space and Target Decoders 54
Configuration Space Header 55
Target Decoder Properties 57
Data Resources 61
Target Decoder Setup 62
Programming the Decoders 63
Modifying the Configuration Space Header 65
Overwriting BIOS Settings 68
Standard and Power Up Databases 69
Controlling Target Attributes 71
Available Target Attributes 71
Implementing Target Attribute Scripts 75
Setting the Reset Mode for Target Attributes 77
Initializing the Target 78

Using the Data Memory 79

Organization of the Data Memory 80
Data Compare Unit 82
Using the Data Memory Editor 83

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Generating Interrupts 85
Interrupt Capabilities of the Testcard 86
Asserting and Deasserting Interrupts 87
Testing Hot Swap Systems 89
Hot Swap Capabilities of the Testcard 90
Asserting and Deasserting Hot Swap Signals 91
Using the Command Line Interface 93
Starting the CLI 94
Basic CLI Command Syntax 95
Using CLI Scripts 96
Bus Transaction Language Reference 97
BTL Commands 97
m_block() 98
m_xact() 99
m_data() 100
m_last() 101
m_xact64() 102
m_data64() 103
m_last64() 103
m_attr() 104
t_attr() 105
BTL Command Parameters 106
Block Transfer Parameters 107
Address Phase Attributes 108
Data Phase Attributes 109
BTL Syntax Diagrams 110
Bus Commands 110
Parameters 112
Identifiers 113

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PCI Exerciser Overview

Agilent's E2920 Verification Tools, PCI Series is your "window into the system" during product development, giving you access to almost all of the system components located on the PCI bus, as well as devices and adapters on secondary buses or within the system.

The PCI Exerciser (option #300) allows you to overcome the passive role in monitoring the PCI bus. With the PCI Exerciser, the testcard can be programmed to behave as a master and/or target device, and thus actively stimulate the bus.

For an overview of how you can use the Exerciser during the phases of the design cycle, refer to *"Gaining an Active Part in Optimization and Validation" on page 8*.

For an overview of the Exerciser's user interface, refer to *"PCI Exerciser User Interface" on page 10*.

"PCI Exerciser Configurations" on page 12 shows examples of possible configurations for PCI Exerciser tests.

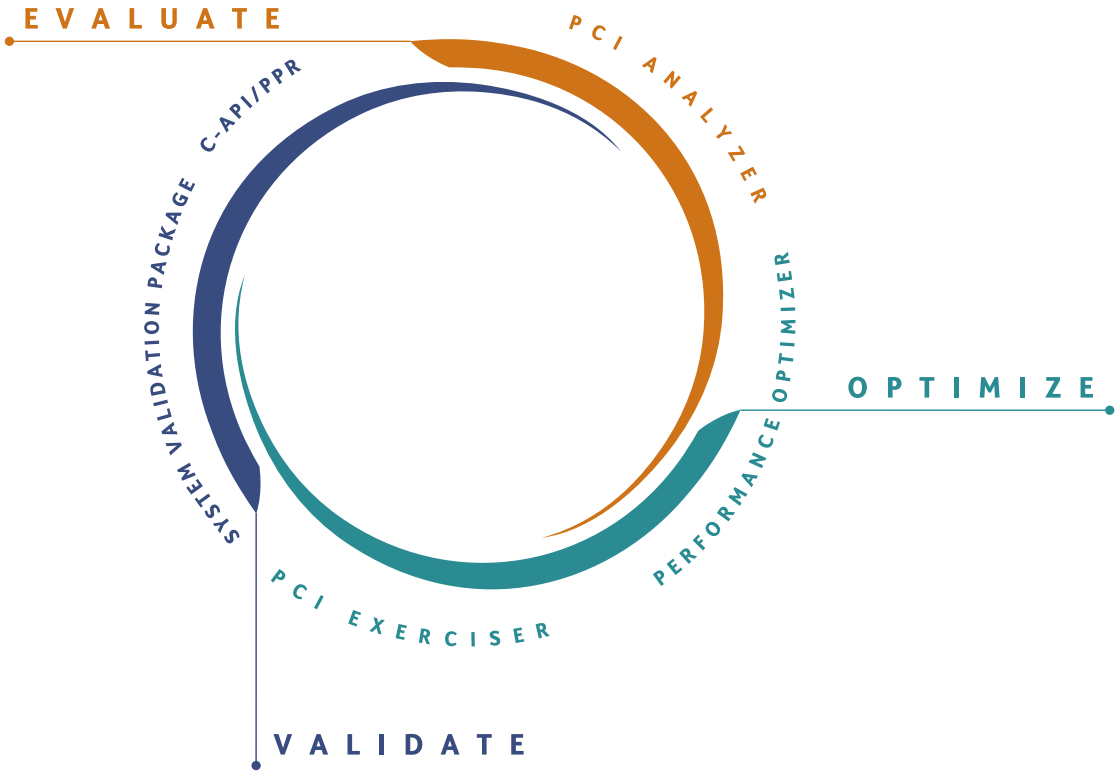
The major setup steps required are outlined in *"Setting Up a PCI Exerciser Test" on page 14*.

In most tests, the Exerciser will act as a master device and/or a target device on the PCI bus, initiating or serving transactions respectively. *"What is a Transaction?" on page 14* introduces this basic term.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Gaining an Active Part in Optimization and Validation

The Agilent PCI Exerciser's focus in the design cycle clearly lies on those phases, when you are optimizing and validating your PCI design for performance and reliability.



Optimize Phase

While the Performance Optimizer gives you in-depth performance analysis and hints for performance optimization, the Exerciser is designed to optimize your PCI design for reliability.

The Exerciser plays an active role in the analysis of complex PCI scenarios by letting you set up worst case test patterns quickly. It provides the full flexibility to set up all the test scenarios you need, with your own choice of protocol and traffic variations, wait states, retries and disconnects, latencies, and burst sizes. You can even control the use of memory and of read and write commands, which is especially important when testing the correct functioning of bridge devices.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This not only saves you the tedious chore of having to test your device with various other PCI components one after the other, but also allows you to test in a repeatable way, which means that you can reproduce any errors for deeper investigation.

You can also run functional tests, directing the Exerciser to generate and transmit large blocks of data in specified time intervals, thus testing how much PCI traffic your device can handle.

Validate Phase

Validating your PCI device means ensuring its reliability in the long run. This means ensuring that it is stable under any application conditions, with any combination of plug-ins and any kind of traffic on the bus.

Using the PCI Exerciser as a master or a target device, you can test your design, repeatedly and consistently from the earliest opportunity, even during early bring-up and integration before a complete system is available.

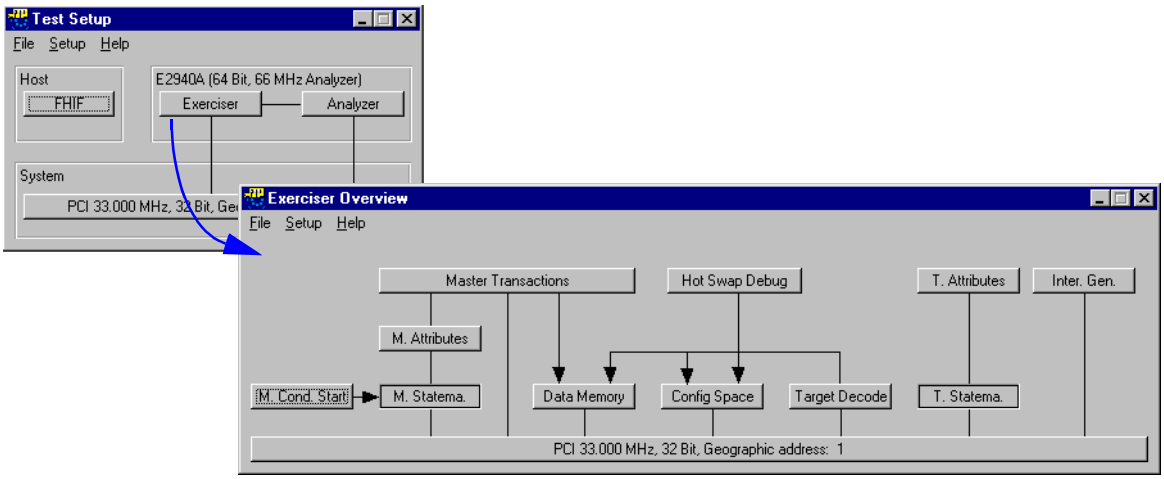
In combination with the System Validation Package and/or the C-API/PPR options, the Exerciser offers fully controllable system-test, with wide coverage, reproducibility, and root-cause-analysis capabilities that will reveal system critical problems faster than any hot mock-up testing could.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PCI Exerciser User Interface

With the Agilent PCI Exerciser Graphical User Interface you can access and control all features of the Agilent E2940A testcard's Exerciser.

Clicking the *Exerciser* button in the Test Setup window or selecting *Show Exerciser Overview* from the *Exerciser* menu in the main window shows the Exerciser Overview window. Button, menu and the overview window are only available if the PCI Exerciser option has been installed.



The Exerciser Overview window shows the individual components of the PCI Exerciser, how they interact, and how they act on the bus. Clicking the buttons brings up the respective setup windows.

The Agilent E2940A testcard can act as a master or a target device on the PCI bus. Thus, the Exerciser mainly consists of the two parts master and target. Both are implemented with state machines, that are generated when compiling the input scripts.

For the **master** you can specify

- the master transactions to be performed,
- the protocol attributes to be used with the transactions,
- the data to be used for the transactions,
- the start conditions that need to be fulfilled before the transactions are started,
- whether or not errors should be injected during the transaction.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For the **target** you can specify

- which addresses are to be decoded (target decoders and configuration space),
- how received data is to be handled,
- the data to be transferred on request,
- the protocol attributes to be used during transactions,
- the decode speed,
- whether or not errors should be injected during the transaction.

Furthermore, you have full control on the testcard's **configuration space**. This means, you can program every register of the configuration space header to adapt your testcard to any test requirement.

You can also control the testcard's **hot-swap** behavior, and monitor and modify the corresponding signals.

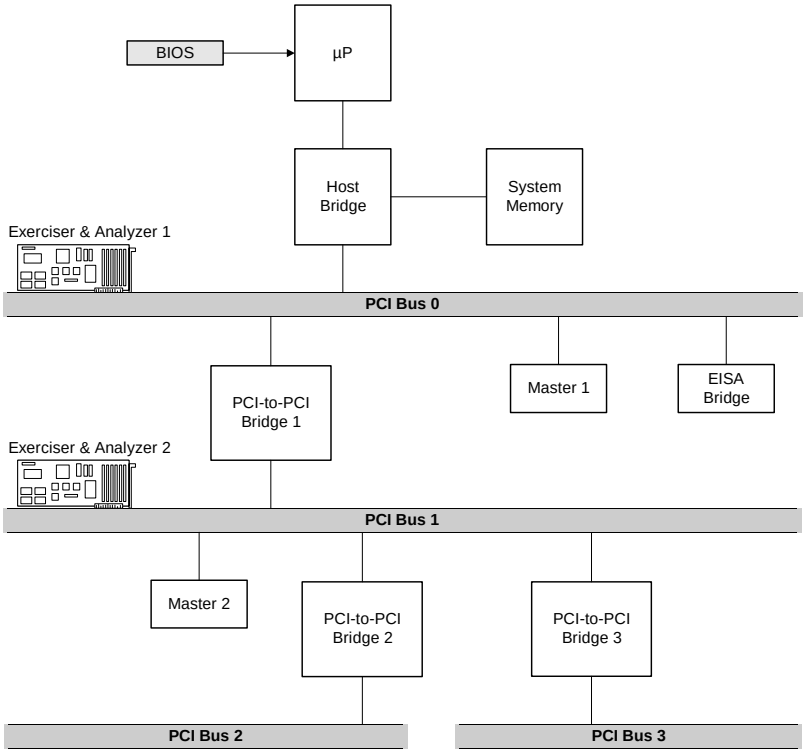
Finally, the Agilent E2940A testcard is able to generate the **PCI interrupts** INTA# ... INTD# and any combination of them.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PCI Exerciser Configurations

The configuration possibilities provided by the testcard and the GUI are basically the same as described for the Analyzer in the *Agilent E2940A PCI Analyzer User's Guide*.

When using the Exerciser, it depends on your testing requirements where you plug the testcard into the system under test. The following figure shows an overview of a typical computer system with two testcards plugged in.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You can use the Exerciser's master and target features to test each individual device—or the system as a whole:

- Testing devices

  The testcard's master can be used to initiate transactions to a target device under test. The testcard's target can be used to react to transactions initiated by a master device under test. Master devices can also be bridges, such as host bridges or PCI-to-PCI bridges.

  The testcard can also be used for functional tests. For example, when testing communication devices (LAN cards, ATM cards, telecom switches, ...), the testcard can be set up to generate and receive data blocks at specified data rates, thus allowing to test device and system behavior under full load.

- Testing a PCI-to-PCI bridge

  To test the interfaces of a PCI-to-PCI bridge, testcards can communicate with other devices (or testcards) over the bridge or within one bus. Thus, you can test whether the bridge forwards transactions to the correct bus.

  During start-up you can test whether configuration cycles are transferred correctly.

- Testing a system

  When testing a system as a whole, the Exerciser can be used to

  – set up any PCI traffic scenario quickly and in a repeatable way

  – simulate PCI devices that are not yet available

  – generate interrupts to test the system's interrupt processing capabilities

  – monitor and manipulate the testcard's hot-swap signals to test the system's hot-swap capabilities

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Setting Up a PCI Exerciser Test

Setting up a PCI Exerciser test includes the same steps as described for the PCI Analyzer:

- Inserting the testcard.
- Connecting to the testcard.

Additionally, you have to set up the Exerciser as required for your test. This is described in detail in

- *"The PCI Exerciser as a Master Device" on page 31*
- *"The PCI Exerciser as a Target Device" on page 53*
- *"Using the Data Memory" on page 79*
- *"Testing Hot Swap Systems" on page 89*

NOTE With the Agilent E2940A testcard, there is no need to shut down and reboot a CompactPCI system to perform tests. Its hot-swap capabilities allow you to plug the testcard directly into any hot-swap capable system to analyze individual modules or cards, or to perform post-mortem analysis on a crashed system. Engineers can swap the Agilent E2940A from one system to another as necessary.

What is a Transaction?

On every PCI system the different devices communicate via the PCI bus. This communication is controlled by the bus arbiter that determines which device may actively use the bus at any given time. This mechanism is used to avoid data collision and possible hardware damage. Basically, there are two different types of devices that operate on the bus: **masters** (actors) and **targets** (reactors).

The Agilent E2940A PCI Exerciser testcard can simulate either a master or a target device or both at the same time. This enables the testcard to emulate and/or test any device in your system under test.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The transfer of data between a master and a target device is performed in one or more data transactions. These transactions are initiated and controlled by the master, while the target reacts depending on the type of transaction.

Every successful transaction includes the following steps:

1. The master requests bus access from the bus arbiter.
2. The arbiter grants the master to use the bus for a transaction.
3. The master starts the transaction by driving the address of the target and the bus command on the bus.
4. The target that owns the corresponding address range responds by asserting the DEVSEL# signal.
5. The master drives one or more data phases on the bus. If the command was a write command, the master also sends the data. Otherwise, the data is sent by the target.
6. When reaching the last data phase of the transaction, the master deasserts the FRAME# signal to indicate that it is ready to complete the transaction.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Running A Sample PCI Exerciser Session

The following application examples explain how the testcard can be used in various tests. After introducing the major scenarios for the PCI Exerciser and showing how to prepare for the sample sessions, you will be guided through the following examples:

- *"Guided Tour: Generating Master Transactions" on page 20*
- *"Guided Tour: Specifying Custom Target Behavior" on page 24*
- *"Guided Tour: Accessing VGA Frame Buffer Memory" on page 27*

NOTE The examples given here are also part of the *Agilent E2920 Software Demo guide*. If you have already worked through this guide, you may skip them here.

Example Scenarios

The Agilent E2940A PCI Exerciser and Analyzer testcard can be used for basically any test that might be required, when developing devices, chipsets, or drivers for the PCI environment. Here you can find three examples that give a guided introduction to some of the main features of the testcard and its Graphical User Interface (GUI).

Testing a Target Device

Imagine that you are integrating a PCI chip into an adapter or system. Or that your validation team (or even customer) reports that your PCI chip or system is not working properly under certain circumstances. Then you probably need to generate or reproduce a given PCI scenario and find out how your chip behaves in the real environment on certain PCI commands, on certain protocol variations like master wait states, or on error conditions such as wrong parity.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In a case like this you can set up the Agilent PCI testcard as a master to access your chip or system under test with certain commands or attributes. This is explained in *"Guided Tour: Generating Master Transactions" on page 20*.

Testing a Master Device

If you are developing, debugging, validating, or characterizing a PCI bus mastering device, you need a programmable target that can react deterministically in the required fashion. The Agilent PCI Exerciser provides this functionality. It has a 512-KB data memory that can be accessed with either memory or I/O transactions. The target protocol behavior is fully programmable for every individual data phase. This includes the number of wait states, the termination type, error insertion, etc.

An example, in which the Exerciser is set up as a target, is found in *"Guided Tour: Specifying Custom Target Behavior" on page 24*.

Performing C Function Calls Directly

Besides the Graphical User Interface, the Agilent E2940A testcard also features a C Application Programming Interface (option #320), providing full access to all functions of the card.

If you do not have the API installed or do not want to write complete C programs, you can use the Command Line Interface (CLI) instead, which is part of the GUI. With the CLI you can run scripts written in a simple script language.

An example how this is done is presented in *"Guided Tour: Accessing VGA Frame Buffer Memory" on page 27*.

Preparing for the Guided Tour

The first two examples described in the guided tours are designed to be performed in Offline/Demo Mode—without any hardware required. For the third example the GUI actually needs to be connected to a hardware. Otherwise, communication with the functions on the testcard, of course, cannot work.

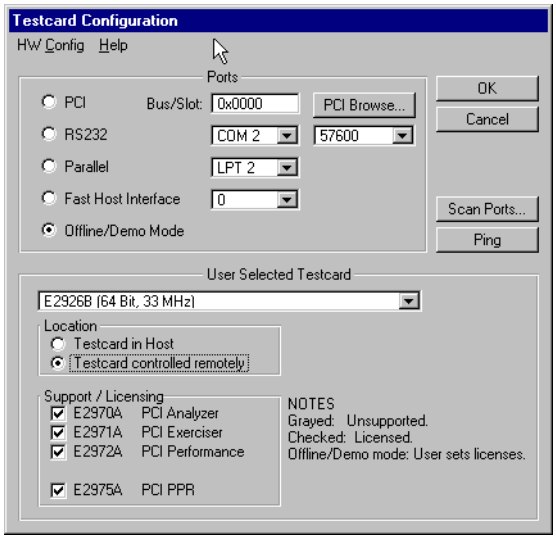
All the setup files (*.bst) and PCI signal waveform files (*.wfm) that are mentioned in the following text are found in <your_installation_directory>\samples\demo. If you did not change the default setting during installation, <your_installation_directory> will be C:\Program Files\Agilent\E2920 PCI Series <release_number>.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE The examples in the guided tours use an Agilent E2926B (64 bit, 33 MHz) testcard, but also apply to all other testcards of the Agilent E2920 series.

To prepare for the examples:

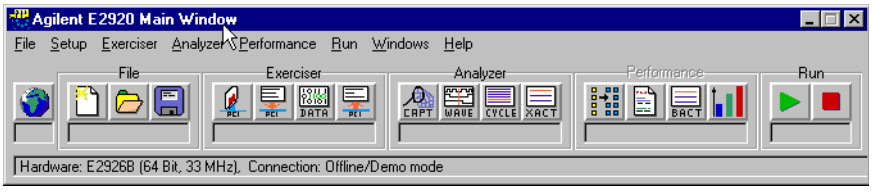
1. Launch the Agilent E2920 software.
2. From the *Setup* menu, choose *Testcard Configuration*.
3. In the Testcard Configuration dialog box, select the *Offline/Demo Mode* radio button.



4. Now choose *E2926B (64 bit, 33 MHz)* from the *User Selected Testcard* selection list, and check all license boxes in the *Support/Licensing* group.

   Your display should now look like the window shown above.

5. Click *OK* and the main window should look like this.



You are now ready to start the guided tours.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Guided Tour: Generating Master Transactions

This example shows how to set up the testcard as a master device that initiates data transfers on the PCI bus. More specifically, it generates burst transfers into the VGA frame buffer memory while varying certain protocol parameters.

Loading the Setup Files

If you actually were connected to a testcard and your system ran in DOS mode, the results of this test could be viewed on the screen. Because we are in offline mode, start by loading the required files:

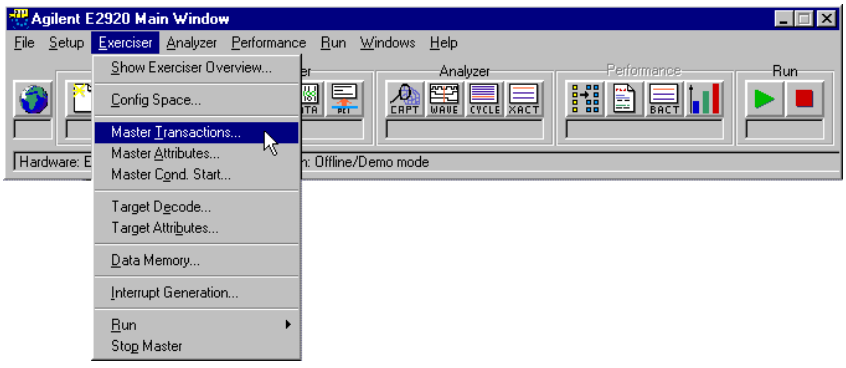
1 Load the setup file for this example (vga4.bst) by selecting *Load* from the *File* menu in the main window.

2 Load the PCI signal waveform file for this example (vga4.wfm) by selecting *Load from file* from the *File* menu in the Waveform Viewer window.

Setting up a Master Transaction

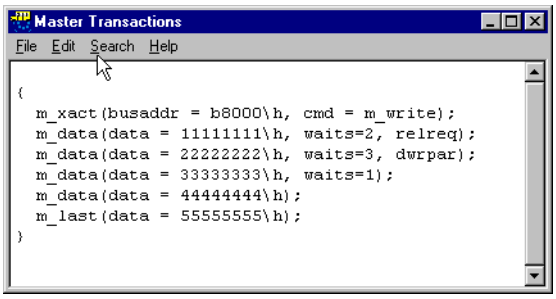
Once you have loaded the setup file and the waveform file, continue by setting up the master transaction for the test.

1 Open the Master Transactions window by clicking the Master Transactions button in the main window or by selecting the *Master Transactions* item in the *Exerciser* menu.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In the editor window you see that the master transactions for this test already are displayed. They are stored with all other settings in the setup file (vga4.bst).

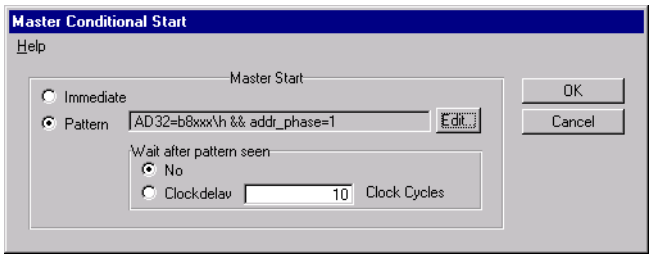


The transaction blocks can be identified by the curly braces at the beginning and the end of each block. There is one transaction block specified for this test, which generates a memory write transaction to address b8000\h. The transaction performs a continuous burst of 5 dwords (unless the target—in this case the graphics card—disconnects earlier). The m_data and m_last statements specify the data to be transferred, along with optional protocol attributes such as wait states, wrong parity, release REQ#.

Specifying a Conditional Start

When the Run button is clicked in the main window, the software writes the specified master transactions into the testcard's internal memory and starts execution either immediately or after a conditional start pattern has occurred on the bus. For this example, use a **conditional start pattern**.

1 Open the Master Conditional Start dialog box by selecting the *Master Cond. Start* item from the *Exerciser* menu.



In this dialog box the pattern is set to start the transactions after an access to the address range b8000\h...b8fff\h is done. This is coded in the boolean expression next to the *Pattern* radio button. To start the master transactions, both the value of the AD32 bit vector must be b8xxx\h and it must be an address phase at the same time.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The conditional start feature of the Agilent PCI Exerciser allows synchronization of traffic generation to start when the pattern appears either on the PCI bus or on any of the 12 external input ports of the testcard.

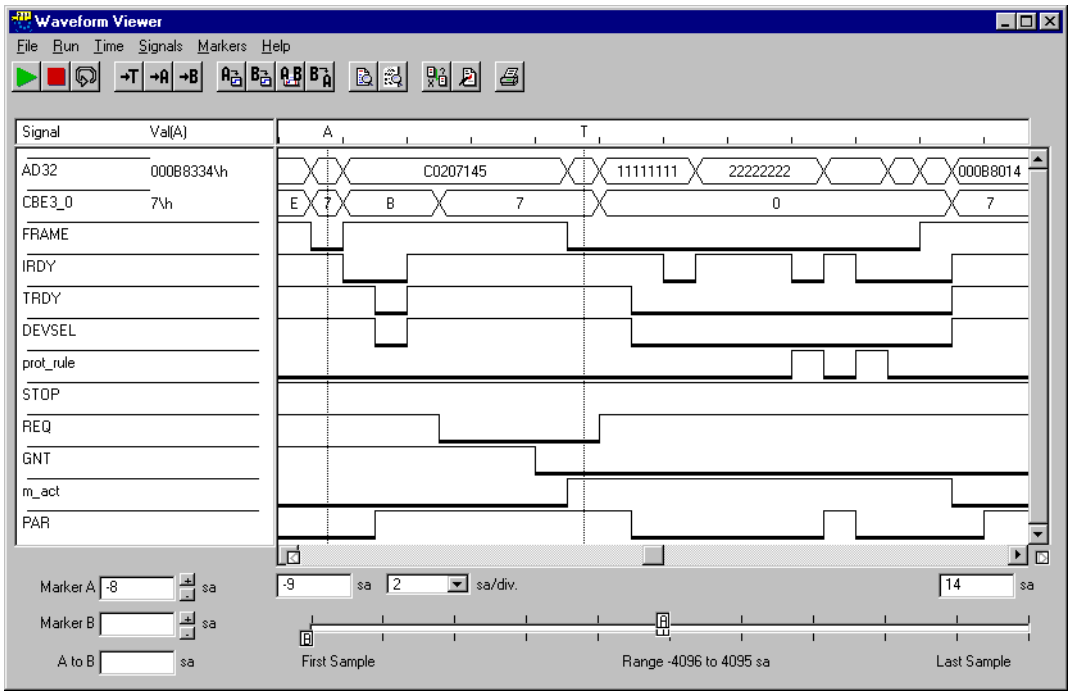
If you actually were connected to a testcard, you could start the test by clicking the Run button in the main window or by choosing *Run* from the *Exerciser* menu.

Viewing the Results

Use the Analyzer's Waveform Viewer to inspect the results of this test. The waveforms were loaded from the waveform file vga4.wfm.

1 Click the Waveform Viewer button  in the main window (or use the *Waveform Lister* item from the Analyzer menu) to open the waveform viewer.

In the following screenshot you see the section of the waveform diagram in which the transaction block was executed.

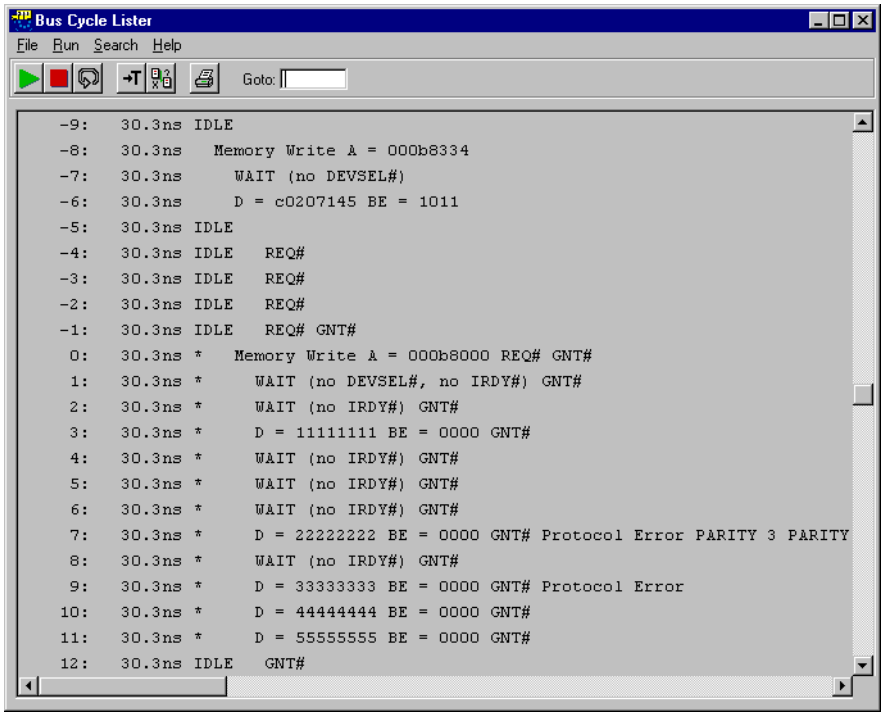


Note the transaction that triggers the start of the Exerciser (shown at marker A). It is also easy to identify the transaction block during which the master is active (m_act is high from the trigger point T on; it is always high while the testcard's master is active). During this time the AD32 signal first holds the target address (address phase) and then the transferred data in the five data phases.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To see the corresponding section of the Bus Cycle Lister window:

2 Click the Bus Cycle Lister button  in the main window (or use the *Bus Cycle Lister* item from the Analyzer menu) to open the bus cycle lister.



Note that two protocol violations are indicated. If the tool was connected to an actual testcard, examining the Protocol Check window would reveal that the second protocol violation indicates that a parity error occurred but was not signaled by the receiver of the data.

When larger amounts of data need to be transferred, an alternate syntax allows you to generate bursts of virtually any length (subject to the target's behavior). To see an example of the syntax select *New* from the *File* menu in the Master Transactions editor window.

There, the second transaction block, commented out though, shows how to specify the number of dwords to transfer (nod=2) within a block transfer. Additionally, you can assign an attribute page to each transaction block, which defines further protocol attributes for the master. These attribute pages are found in the Master Attributes editor window accessible via the *Master Attributes* item in the *Exerciser* menu of the main window.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Guided Tour: Specifying Custom Target Behavior

This example illustrates how to set up your Agilent PCI testcard as a target device. The behavior of the target is fully programmable. This includes the address ranges that are decoded, how received data is handled, which data is transferred on request, and the protocol attributes that are used during the transactions.

Loading the Setup Files

If you actually were connected to a testcard and your system ran in DOS mode, the results of this test could be viewed on the screen. Because we are in offline mode, start by loading the required files:

1 Load the setup file for this example (target1.bst) by selecting *Load* from the *File* menu in the main window.

2 Load the PCI signal waveform file for this example (target1.wfm) by selecting *Load from file* from the *File* menu in the Waveform Viewer window.

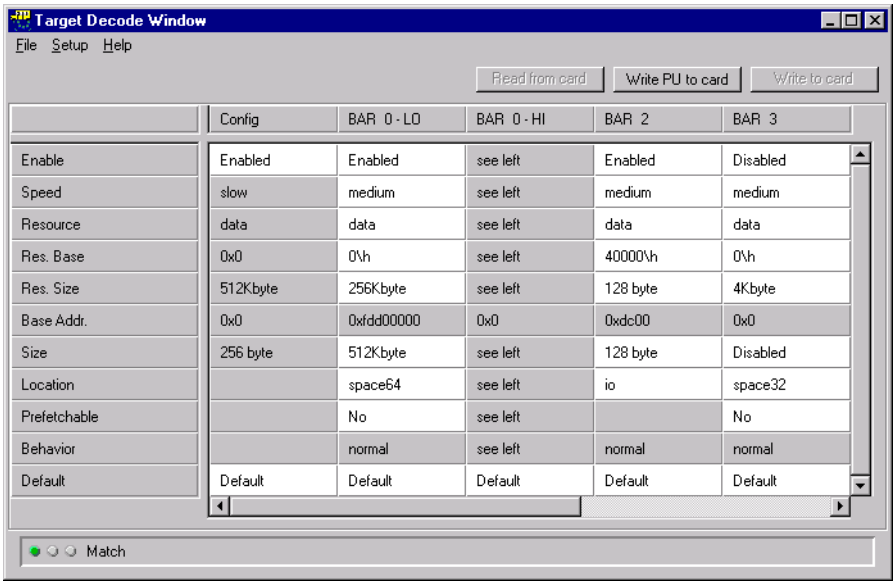
Setting up the Target Decoders

For this guided tour, we will only view the prepared target decoder setups.

1 To view the setup for the target decoders, choose the *Target Decode* item from the *Exerciser* menu.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In Offline/Demo Mode the Target Decode window will look slightly different than the screenshot below.



Base Address Registers

The content of the Target Decode window mainly is an interpretation of the configuration space settings. In the figure above you see that two standard decoders together are set up to decode 64-bit addresses. The two columns representing them are titled *BAR 0-LO* and *BAR 0-HI*, where the abbreviation BAR stands for base address register. The address space type and location are set in the location field of the respective decoder.

Internal Resources

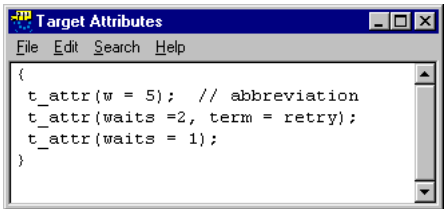
The three rows *Resource*, *Res. Base*, *Res. Size* determine to which internal data resource the decoders are connected to. These settings define how the received data is to be handled or which data is to be sent on request.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Specifying the Target Protocol Behavior

The target protocol behavior is defined in a similar way to that of the master behavior.

1 To inspect the protocol attributes, open the Target Attributes editor window by selecting the *Target Attributes* item in the *Exerciser* menu.

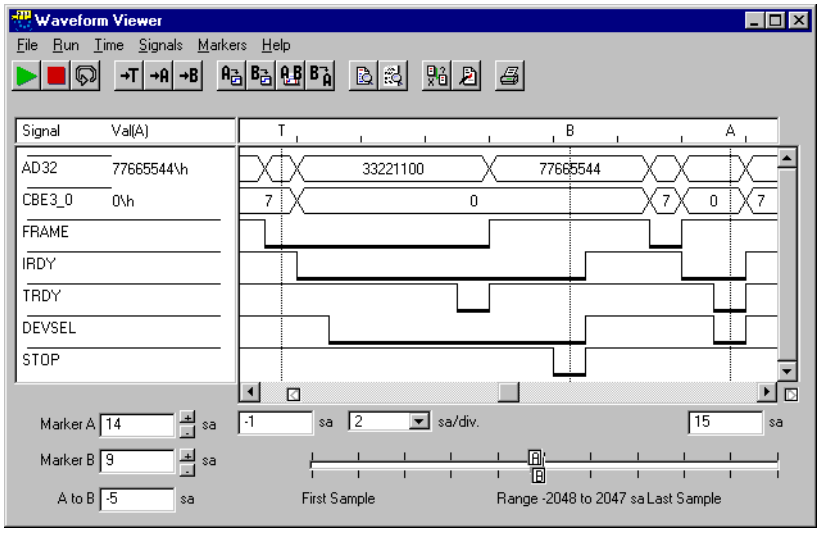


These attributes specify what happens when the testcard's target is accessed by a master.

Viewing the Results

Compare the settings made in the previous step with the traffic captured on the bus.

1 Click the Waveform Viewer button in the main window (or use the *Waveform Lister* item from the Analyzer menu) to open the waveform viewer.



Notice that the PCI Exerciser target behaves exactly as it was set up:

- The first data phase has 5 wait states (after the triggerpoint T).
- The second data phase does not complete until it is retried once.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Marker B shows the retry and Marker A shows the completed data transfer. This behavior can also very easily be observed in the transaction lister.

NOTE The t_act signal—not shown on the screenshot—is always high while the testcard's target is active.

Guided Tour: Accessing VGA Frame Buffer Memory

In this example, the Agilent PCI testcard is used to poke some data values directly into the video frame buffer memory of a VGA graphics adapter. If you are working interactively, the testcard's Command Line Interface is a very easy way to use these host access functions.

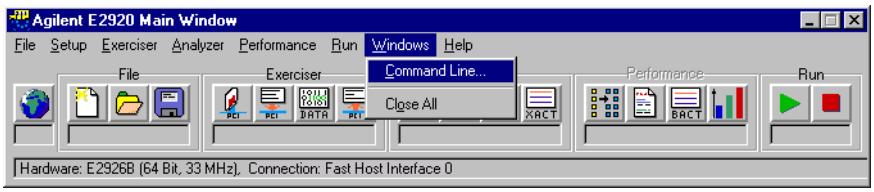
The Command Line Interface uses the CLI equivalents to the functions provided by the C-API (option #320) described in detail in the *Agilent E2940A C-API/PPR Reference*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

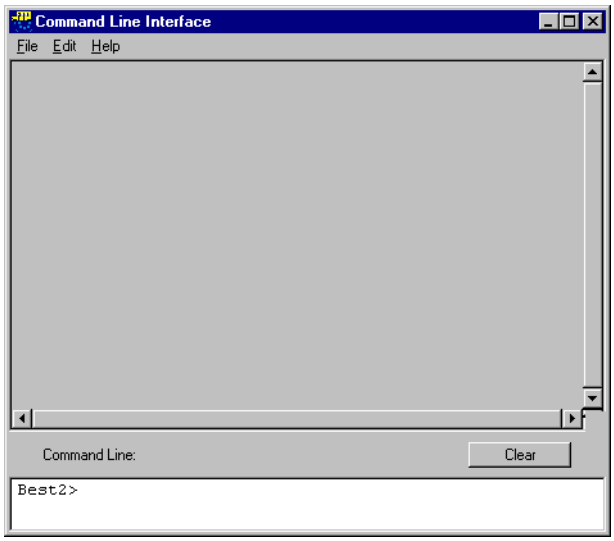
Starting the Command Line Interface

To open the command line interface:

1 Select *Command Line Interface* from the *Windows* menu.

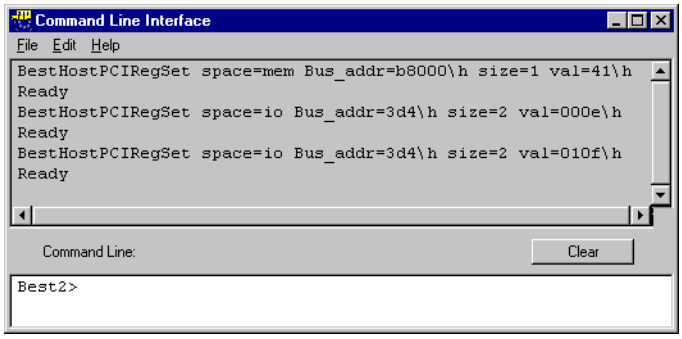


This opens the Command Line Interface window:



Entering the Required CLI Commands

With the use of the Command Line Interface a character will be written into the top left corner of the VGA text screen (visible in DOS mode only). This is done by writing a single byte to the physical memory address 0xb8000. Then the cursor is set next to this character by doing two I/O word writes to I/O address 0x3d4. See the following screenshot.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Call three commands were typed in at the *Best2>* prompt. The commands are displayed along with the results in the output area of this window. In this case no specific output was returned by the testcard. Hence, only *ready* is displayed to indicate successful completion of the command.

When the Agilent E2920 software is running in demo mode, no connection to a testcard is established. Thus, typing commands in the command line interface window has, of course, no effect, and generates a reply:

B_E_ERROR: Error transferring command (1).

With the command line interface you can also log and run CLI scripts. Hence, you do not need to type all commands by hand for every test. The option providing full flexibility and control on your system is available with the C Application Programmer's Interface (C-API) which is the Option #320 to your testcard. With the C-API you can implement your own C programs that may access all functions of the testcard.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The PCI Exerciser as a Master Device

The Exerciser of your Agilent E2940A testcard can simulate any device on the PCI bus under test. There are mainly two different types of devices—masters and targets. Other devices (for example, network interface cards) have both master and target functionality.

The Exerciser contains both a master and a target along with a few other components. All necessary information about the testcard's **master device** is provided here.

Programming Transactions

A device on a PCI bus is called a master if it requests to access the bus and performs data transfers when the bus is granted to it. A data transfer consists of one or more transactions. Information on the programming of **master transactions** and their **properties** is found in *"Programming Master Transactions" on page 32*.

Master Attributes

Additionally, the Agilent PCI Exerciser allows to control the protocol behavior of its master device. These protocol **attributes** normally do not have any effect on the result of a transaction. They only determine the master's behavior in terms of inserted delays of different types, transaction terminations, and more. How to program these attributes is described in *"Controlling Master Attributes" on page 40*.

Run Options

After the master device is set up completely, you can run it with several options. These are explained in *"Running the Master" on page 48*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Programming Master Transactions

In order to set up the Agilent PCI Exerciser as a master, you need to specify **master transactions**. After the master is started, it will perform these transactions on the bus. For this purpose it requests from the PCI bus arbiter to use the bus. When the bus is granted to the master, it initiates the data transfers on the bus. For more details, refer to *"Master Transactions Overview" on page 33*.

For each transaction, you can specify **transaction properties** like bus command or bus address (see *"Transaction Properties" on page 36*) valid for the whole transaction.

NOTE These transaction definitions basically define *what* is to be done by the master, that is, they specify the data that is to be transferred from one location to the other. On the other hand, the **master attributes** specify how the transaction is to be performed (see *"Controlling Master Attributes" on page 40*).

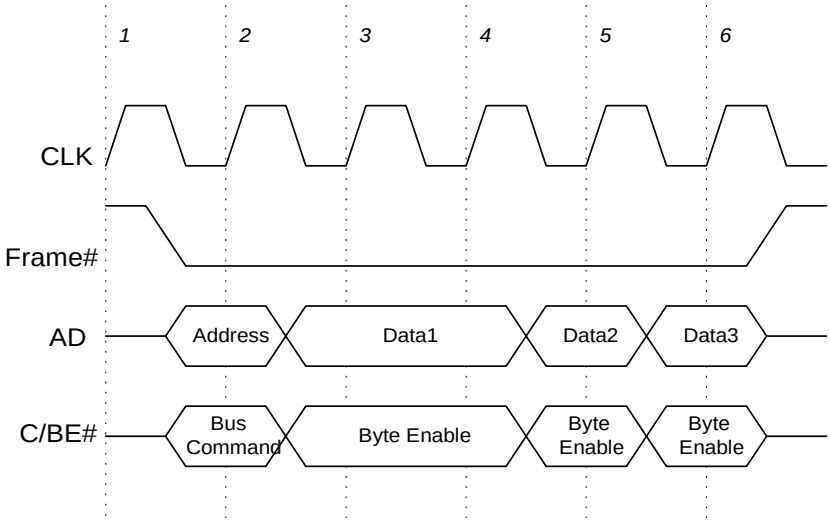
For specifying transactions and properties, the Agilent PCI Exerciser provides a master transaction editor. The editor uses the **Bus Transaction Language (BTL)** as input format for the transactions. BTL descriptions for transactions may, for example, look like this:

```
{
   m_xact(busaddr = b8000\h, cmd = mem_write);
   m_data(data = 8f458f42\h, waits = 3);
   m_last(data = 8f548f53\h);
}
...
{
   m_block(busaddr = b8008\h, cmd = mem_write, intaddr = 0\h,
   nofdwords = 20, attrpage = mypage);
}
```

A complete list of the commands and parameters of the Bus Transaction Language is found in *"Bus Transaction Language Reference" on page 97*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Master Transactions Overview



The figure shows a typical transaction, consisting of one address phase and one or more data phases.

According to the number of data phases, two transaction types can be distinguished: **single transactions** and **block transactions** (see *"Single Transactions and Block Transactions" on page 34*).

For your master transactions, you can specify any combination of single transactions and block transactions.

Address Phases and Data Phases

Address Phases

In the address phase the master asserts the FRAME# signal and drives the address of the target device on the bus along with the bus command. All target devices on the bus latch and decode the address. The target that finds this address in its own address range, replies with a device select DEVSEL# signal.

The bus commands typically are of one of the following types. The respective command in BTL syntax is given in parentheses:

- memory read or write (`mem_read`, `mem_write`),
- I/O read or write (`io_read`, `io_write`),
- configuration read or write (`config_read`, `config_write`),
- memory read line (`mem_readline`),
- memory read multiple (`mem_readmultiple`),
- memory write invalidate (`writeinvalidate`),

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- interrupt acknowledge (`int_ack`),
- special cycle (`special`),
- reserved cycle (`reserved_4`, `reserved_5`, `reserved_8`, `reserved_9`).

Data Phases

After the addressed target signaled that it is ready for the transaction (by asserting TRDY#), the first data phase takes place. In the data phases the data is transferred. If the bus command was a write command, the data is sent by the master. In case of a read command, the data is sent by the target.

There can be more than one data phase in a transaction. Transactions with several data phases are commonly referred to as bursts. The way in which a transaction is eventually performed depends also on the protocol and the device properties.

Single Transactions and Block Transactions

According to the number of data phases, there are two different types of transactions:

- Single transactions
- Block transactions

Single Transactions

This type of transaction uses one BTL command per transaction phase. Use this type if you want to specify the transfer data directly in your editor. Also, you can define your master attributes for every individual transaction phase within the BTL commands.

The available BTL commands are:

- `m_xact()`, `m_xact64()`. These two commands initiate a new transaction by driving an address phase onto the bus. `m_xact64()` is used for 64-bit data transfers and for mixed data transfers (32-bit and 64-bit).
- `m_data()`, `m_data64()`. These two commands follow `m_xact()` and `m_xact64()` respectively. They perform a data phase on the PCI bus that is not the last or only data phase of the burst. `m_data64()` is used for 64-bit data transfers and for mixed data transfers (32-bit and 64-bit).
- `m_last()`, `m_last64()`. These two commands perform a data phase that is either the last or the only data phase of a burst. `m_last64()` is used for 64-bit data transfers and for mixed data transfers (32-bit and 64-bit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In the following example, the first line starts the transaction and specifies the transaction properties (`busaddr`, `cmd`). The next two lines perform the data phases and specify some master attributes (`data`, `waits`) for each phase.

```
{
   m_xact(busaddr = b8000\h, cmd = mem_write);
   m_data(data = 8f458f42\h, waits = 3);
   m_last(data = 8f548f53\h);
}
```

Block Transactions

When using a block transaction with `m_block()`, you can define the transfer of a larger amount of data within one BTL command. This data must be stored in the internal data memory and you need to specify the internal start address of this data.

The master then performs one burst containing the complete data—unless master termination is specified by a master attribute, or the burst is terminated by a participating device or an error. In that case, the master initiates a new transaction and continues transferring the data from where it was stopped.

The master attributes for the individual phases of the block transfers cannot be specified in the editor. But you can assign an attribute page to each block transfer, that is worked through line by line for the transaction phases. This is an example for a block transaction:

```
{
   m_block(busaddr = b8008\h, cmd = mem_write, intaddr = 0\h,
   nofdwords = 20, attrpage = mypage);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Transaction Properties

The commands of the Bus Transaction Language can be used with different properties. These transaction properties more specifically describe the master behavior. They always apply to all phases of a transaction. Hence, they are defined in the first command of the transaction, either `m_block()`, `m_xact()`, or `m_xact64()`.

The transaction properties are stored in the master block transfer memory on the testcard and are thus also referred to as the master block properties. They hold parameters like the bus command and the addresses from and to where data is to be sent. Some of these properties are strictly required, others use default values when omitted.

According to their contents, two types of properties can be distinguished:

- *"Properties Containing Values" on page 36*
- *"Properties Containing Pointers" on page 37*

  These properties hold pointers to other memories and thus control the co-operation between these memories on the testcard.

Properties Containing Values

The following list summarizes and briefly explains the available master block properties containing explicit values. The respective BTL parameter is put in parentheses.

- Bus Command (`cmd`).

  This property is always required and specifies the PCI bus command used for the transaction. The possible choices are memory read or write, I/O read or write, configuration read or write, memory read line, memory read multiple, and memory write invalidate. Beside of that, you can also drive interrupt acknowledge, reserved or special cycles on the bus.

- Dual Address Cycle, DAC (`busdac`).

  This flag is optional. It determines whether a 32-bit single address phase or a 64-bit dual address phase is performed.

- Bus Address (`busaddr`, `busaddr_hi`).

  The bus address also is required for every transaction. It can be 32 or 64 bits, depending on the dual address cycle flag and the bus command. To transfer 64-bit addresses, Dual Address Cycle (DAC) must be enabled. Certain bus commands require aligned PCI bus addresses, for example memory write invalidate.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Number of Dwords (`nofdwords`).

  This property is required for block transactions using `m_block()`. It specifies the amount of data that is transferred by the master. This means the number of data phases that are implemented, because you can send one dword (4 bytes) per data phase. Note, that the actual amount of transferred data might be less, due to byte enable settings or misaligned addresses.

- Compare Flag (`compflag`, `compoffs`).

  If the optional compare flag is set, data read from another device is not stored in the testcard's data memory but is compared to previously stored data. The parameter `compoffs` (compare offset) holds the address that determines to which reference data the comparison is done. If a data mismatch occurs, a compare error is reported.

- Continue With Attributes (`contattr`).

  This property determines whether the master attribute page is restarted at the beginning of a new transaction (default). Use longer attribute pages and set this flag to "1" to generate transactions with a greater variety of protocol attributes.

- Conditional Start (`condstart`).

  Setting this flag will cause the master to pause until a specified event occurs on the bus or on one of the trigger I/O ports.

  Note, that flag is only evaluated if the pattern that is used for detecting this event has been programmed with the Command Line Interface (CLI). This flag has no effect if the conditional start run mode is selected as described in *"Specifying a Start Condition" on page 49*.

Properties Containing Pointers

The following three properties are also stored in the master block transfer memory, but they merely hold pointers to other memories. They control the co-operation of the different types of memories during the master run.

These pointers always point to a start value in the respective destination memory. If enabled, an internal pointer moves on from this entry point one line per successfully completed data phase. Thus, the pointer values in the master block transfer memory remain unchanged.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Attribute Page (`attrpage`).

  This parameter is optionally available for `m_block()` transactions. It is stored as a pointer to an attribute page. An attribute page contains a sequence of attribute lines `m_attr()` in the attribute memory. With this property, arbitrary protocol attributes can be assigned to every phase of the transaction.

  See also *"Controlling Master Attributes" on page 40*.

- Byte Enable Control (`byten`, `byten_var`).

  With the parameter `byten`, you can define fixed byte enable values for a transaction (assuming `byten_var` has its default value 0). Using `byten_var=1` instead will cause the byte enable memory pointer to be incremented with every successfully completed data phase.

  Use the Command Line Interface (CLI) to store sequences of byte enable values in the byte enable memory. See *"Using the Command Line Interface" on page 93* for details.

- Internal Address (`intaddr`).

  This property is required for `m_block()`. This address points to the testcard's internal data memory. During write transactions, the data stored in this memory is sent to the target. In read transactions, the received data is stored here. When doing a data compare (compare flag set), this address does not have any effect. The address of the compare data is stored in `compoffs`.

  See also *"Using the Data Memory" on page 79*.

For more detailed information on the participating memories and registers on the testcard, please refer to the *Agilent E2940A Programmer's Guide*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Implementing Master Transaction Scripts

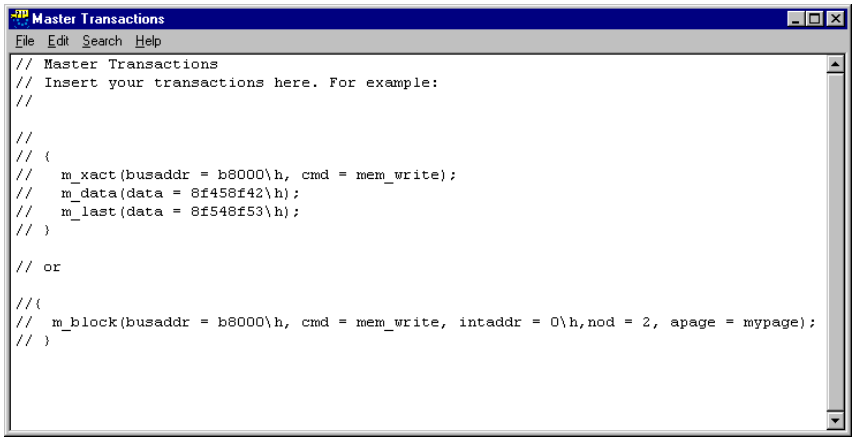
The setup of the master transactions is implemented in transaction scripts. These scripts use the Bus Transaction Language (BTL) and are entered into the master transaction editor. This editor is opened by clicking the Master Transactions button or by selecting the *Master Transactions* item from the *Exerciser* menu in the main window.

Bus Transaction Language

The Bus Transaction Language is a simple script language allowing you to easily set up PCI bus transactions in the Agilent Graphical User Interface (GUI). A BTL script consists of a sequence of commands and their parameters.

Example

The following screenshot shows a simple example script, containing two transactions, commented out though. To load this example into your editor, select the *New* item from the *File* menu.



Both transactions perform a memory write command to address `b8000\h` (VGA memory):

- The first transaction consists of three commands, one for each phase. Note that the data is specified within the data phase commands.
- The second transaction is implemented with the block transfer command `m_block()`. Here the data is taken from the data memory at internal address `0\h`. There is also an attribute page `mypage` assigned to the transaction.

Your scripts will be saved together with the setup in the .bst file. You can also save the scripts individually, using the *File* menu of the transaction editor.

The other menus contain a few more editor functions that may help you editing the scripts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Controlling Master Attributes

While the transaction script defines *what* the master does, the master attributes allow you to specify *how* this is done.

More specifically, the **master attributes** describe all properties of a bus phase during a data transfer. Examples are the number of wait states the master inserts into a data phase, or whether a parity error should be signaled during an address phase, etc.

The behavior specified in the master attributes should not affect the result of the data transfer. Thus, you can repeat a test with the same master transaction settings but varying attribute settings and then compare the results.

Depending on the type of transaction used, the master attributes can either be specified in the transaction BTL commands directly, or in so-called attribute pages (see *"Specifying Master Attributes" on page 40*).

For the latter case, the Agilent PCI Exerciser provides a **master attribute editor** to set up additional parameters that define the protocol behavior of the master device (see *"Implementing Master Attribute Scripts" on page 47*).

The different types of master attributes are:

- *"Master Address Phase Attributes" on page 42*
- *"Master Data Phase Attributes" on page 44*
- *"Master Control Attributes" on page 46*

Specifying Master Attributes

According to the type of transaction, the master attributes can be specified in one of two ways:

- For single transactions they are specified as part of the BTL command (see *"Attributes in BTL Commands" on page 41*).
- For block transactions they are specified in attribute pages, which will be processed line by line for the phases of the block transfer (see *"Attribute Pages" on page 41*).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Attributes in BTL Commands

If you specify single transactions in the transaction editor (one command per transaction phase), the attributes are set up along with the commands. Normally, you will specify address phase attributes in address phase commands and data phase attributes in data phase commands. But you can also mix them up.

Consider the following special cases:

- Defining data phase attributes in the address phase commands `m_xact()` and `m_xact64()` causes these attributes to be set as default values for the complete transaction. If this parameter then is set to another value in a data phase, all following data phases of this burst will still use the default value.
- On the other hand, address phase attributes that are specified in the data phase commands `m_data()`, `m_last()`, `m_data64()`, or `m_last64()` will only come into effect when the transaction is terminated at this point by some reason, and you want to use different attributes for the new transaction starting at this point.

Attribute Pages

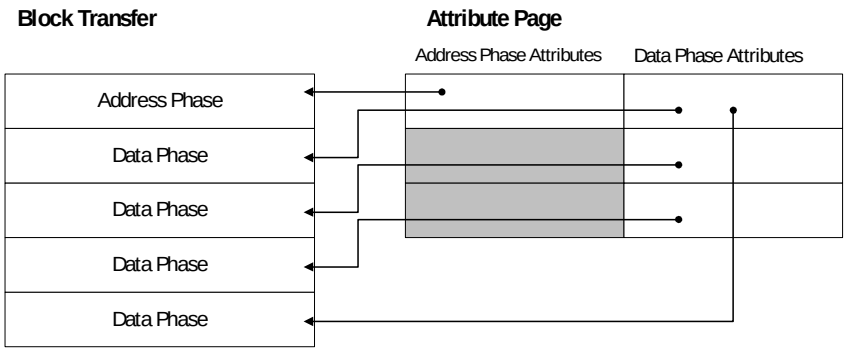
If you use block transactions, the attributes need to be defined in attribute pages in the master attributes editor window (see *"Implementing Master Attribute Scripts" on page 47*). And you can, optionally, assign one of these attribute pages to each block transaction.

When the master is running and performing a block transfer that has an attribute page assigned to it, the attribute lines (`m_attr()`) in this page are processed line by line for the phases of the block transfer.

Basically, any attribute line can happen to be used for both address phases and data phases. This depends on the number of lines in the page, the length of the corresponding bursts, and the restart and repeat settings.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following figure shows how the address phase and data phase attributes are used in an attribute page consisting of three lines.



During a **data phase**, only the specified data phase attributes in the current attribute line will come into effect. All address phase attributes are ignored. The next transaction phase will then use the next attribute line.

During an **address phase**, the address phase attributes of the current attribute line are used. Then, during the first data phase of this burst, the data phase attributes of still the *same* attribute line are used. After that, the next transaction phase (no matter of which type) will use the next attribute line.

When reaching the end of the attribute page before completing the block transfer, it is restarted with the first attribute line (unless this default behavior is disabled).

Master Address Phase Attributes

The following list gives an overview of the available address phase attributes. The respective BTL parameter is put in parentheses.

- Try Fast-Back-to-Back (`tryback`).

  Usually, there must be at least one idle cycle (turn-around cycle) between two transactions of the same master. However, if the master wants to perform two transactions, it does not need to deassert its REQ# signal, and if the arbiter leaves the bus to it, the master can skip the idle cycle(s) between the transactions. This is called “fast-back-to-back”.

  A number of conditions must be met for performing transactions fast-back-to-back. Therefore, the master can only be set up to *try* a fast-back-to-back transfer. Whether the fast-back-to-back transfer *really* can be performed depends on the conditions at runtime. For detailed information, please refer to the *Agilent E2940A C-API/PPR Reference*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Delay, Exerciser Idle (`delay`).

  If “Try Fast-Back-To-Back” is disabled, the Exerciser can stay idle after completion of a transaction before REQ# is asserted for the next transaction. Between blocks, the Exerciser always inserts a gap of at least 15 clocks.

  Note, that the arbiter also can insert idle cycles between transactions.

- 64-Bit Request (`req64`).

  With this attribute, REQ64# is asserted together with FRAME# to signal to the target that the master intends to use 64-bit width data transfers.

  The address must then be aligned to a qword (quad word) boundary, otherwise REQ64# will not be asserted. When an intended 64-bit data transfer has been denied once in a block, it is assumed that the address range cannot handle 64-bit data accesses and the master will not try to transfer 64-bit data in this block again.

- Lock (`lock`).

  The LOCK# signal is used by a master to lock a target. This signal can only be used by one master at a time, in this case the Agilent PCI testcard. This device then has exclusive ownership of the target device until it unlocks it (that is, release LOCK#).

  To test how the target behaves in this situation, the Exerciser can pretend to be two masters in one device. One master locks the target device, the “other” master tries to access the locked target.

- Wrong Parity Calculation (`awrpar`, `awrpar64`, `dacwrpar`, `dacwrpar64`).

  The Exerciser provides attributes to set the parity bit PAR to the wrong value in the following situations:

  – in the first (DAC) or only (32-bit address) address phase cycle of a 32-bit transaction (`awrpar`),

  – in the second address phase cycle of a dual address cycle transaction (`dacwrpar`).

  Furthermore, there are attributes to set the parity bit PAR64 to the wrong value:

  – in the first (DAC) or only (32-bit address) address phase cycle of a 64-bit transaction (`awrpar64`),

  – in the second address phase cycle of a dual address cycle transaction (`dacwrpar64`).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- System Error Signaling (`aperr`, `dacperr`).

  The Exerciser can signal a system error in the following situations:

  – for the first (DAC) or only (32-bit address) address phase cycle of a transaction (`aperr`),

  – for the second address phase cycle of a dual address cycle transaction (`dacperr`).

  In either case, the error is signaled with a two clock delay after the attribute is set. This is required by the PCI specification.

  Note, that the master is actually asserting the parity error signal PERR. But a parity error during a data phase is considered as a system error and will cause SERR to be asserted, too. As a result, all targets should draw from the bus immediately.

  System errors need to be enabled in configuration space. Otherwise, this attribute will not have any effect.

- Release Request (`relreq`).

  Usually, the request signal REQ# is released one clock after the bus enters the idle state after a transfer. The Exerciser, however, can release the request signal any number of clocks after the end of the transaction's address phase. Such behavior can place a load on the arbiter.

- Resume Delay (`resumedelay`).

  The Exerciser can insert a programmable number of clocks before it resumes after a target termination. This gives other masters a programmable chance to obtain the bus.

Master Data Phase Attributes

The following list gives an overview of the available data phase attributes.

- Waits (`waits`).

  The Exerciser's master can insert up to 30 wait cycles into a data phase. On the bus, wait cycles are inserted by deasserting the initiator ready signal (IRDY#).

- Data (`data`, `hi_data`).

  The data attribute is used for the data phase commands `m_data()`, `m_last()`, `m_data64()`, and `m_last64()`. Here the data is specified that is to be transferred. The hi_data attribute holds the upper 32 bit of a 64-bit data value and is, therefore, only used with `m_data64()` and `m_last64()`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Marker (`marker`).

  With this attribute you can set a marker to an integer value during a transaction phase. This number can then be used as input for pattern terms. Pattern terms are used for synchronization with other parts of the PCI testcard, for example, to trigger the trace memory of the Agilent PCI Analyzer.

- Wrong Parity Calculation (*dwrpar*, *dwrpar64*).

  In a data phase, the Exerciser can set the parity bit(s) to the wrong value only for write transfers, because in read transfers the parity is calculated by the target.

  The two different attributes used for this purpose are

  – *dwrpar* for the 32-bit parity bit,

  – *dwrpar64* for the 64-bit parity bit.

- Parity Error Signaling (`dperr`).

  The master can set a parity error (PERR#) that will then be asserted with a delay of two clocks. If actually a parity error should occur on the bus concurrently, it will be ignored. However, this error will be recognized by the testcard's protocol rule observer and can be used to trigger the trace memory.

  Parity errors need to be enabled in the configuration space. Otherwise, this attribute will not have any effect.

- System Error Signaling (`dserr`).

  The master can signal a system error during a data phase.

  System errors need to be enabled in the configuration space. Otherwise, this attribute will not have any effect.

- Release Request (`drelreq`).

  Usually, the request signal REQ# is released one clock after the bus enters the idle state after a transfer. The Exerciser, however, can release the request signal any number of clocks after the beginning of the transaction's data phase. Such behavior would stress the arbiter.

  The data phase release request instruction is ignored if the master has previously been instructed to release the request, for example, during the address phase or during a prior data phase of this transaction.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Master Control Attributes

The control attributes are used to determine how the master steps through the master attribute memory. The following list describes the attributes that control this behavior. Control attributes are valid both for address phase and data phase attributes.

- Repeat (`repeat`).

  Specifying the number of times a particular attribute line is repeated before switching to the next line. A line of attributes can be repeated up to 4 billion times (2^{32}).

- Burst Length.

  The burst length is controlled via the `last` attribute, which indicates the last data phase of a burst and therefore the end of the transaction.

  Regardless of this flag, a transaction will always be terminated if

  – the target signals “retry”, “disconnect”, or “abort”,

  – the specified number of dwords has been completely transferred (see *“Programming Master Transactions” on page 32*).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Implementing Master Attribute Scripts

The Agilent PCI Exerciser provides a master attributes editor, which works similar to the master transaction editor.

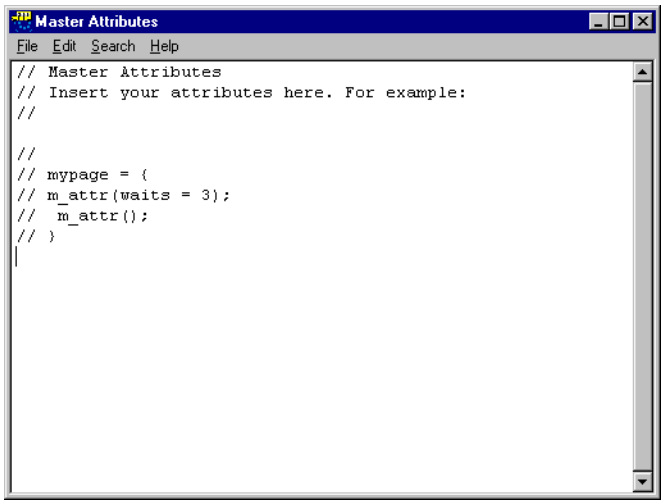
The setup of the master attributes is implemented in attribute scripts. These scripts use the Bus Transaction Language (BTL) and are entered into the master attributes editor. This editor is opened by selecting the *Master Attributes* item from the *Exerciser* menu in the main window.

Bus Transaction Language

The Bus Transaction Language is a simple script language allowing you to easily set up PCI bus transactions and attributes in the Agilent Graphical User Interface (GUI). A BTL script consists of a sequence of commands and their parameters.

Example

The following screenshot shows a simple example script, containing an attributes page `mypage`, commented out though. To load this example into your editor, select the *New* item from the *File* menu.



This attribute page contains two attribute lines that are defined with the command `m_attr()`. The first attribute line causes the master to insert three wait cycles into the first data phase. The second line will cause the master to use the default attributes for the following data phase. With no other settings the page will then be restarted, resulting in the master inserting three waits into every second data phase.

Your scripts will be saved together with the setup in the .bst file. You can also save the scripts individually, using the *File* menu of the transaction editor.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Running the Master

When you have completely set up the master, you can prepare to run your test:

- You can exactly specify the conditions for starting and running the master (see *"Preparing Test Execution" on page 48*).
- Depending on these conditions, there are several ways to start the master (see *"Starting the Master" on page 51*).
- If required, you can stop the master manually (see *"Stopping the Master" on page 52*).

Preparing Test Execution

You have several options how to run the test:

- You can start the test manually by means of the Run button or the *Start* item from the *Run* menu. You can use the Run Options to determine the testcard components to be started (see *"Setting up the Run Options" on page 48*).
- You can specify a start condition to start the master automatically after a certain event (see *"Specifying a Start Condition" on page 49*).

Setting up the Run Options

First of all, you need to decide whether you want to run your master alone or in combination with other parts of the Agilent E2940A testcard. These settings are made in the Run Options window:

1 In the main window select *Run Options* from the *Run* menu.



2 Select the testcard components to be started when using the Run button or the *Start* item from the *Run* menu.

Only components that are enabled with your testcard can be selected.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Specifying a Start Condition

You can specify the following options for the start of the master:

- Immediate Start. This is the default option. After the master is started, it will immediately perform the specified transactions.
- Conditional Start. Alternatively, you can define a start condition. This is done with a start pattern that needs to occur either on the bus or on one of the I/O trigger ports.
- Fixed Delay. The third option is to define a fixed delay for the start. After the master is started, it waits for the specified number of clocks before it asserts REQ# to request GNT to start a transaction.

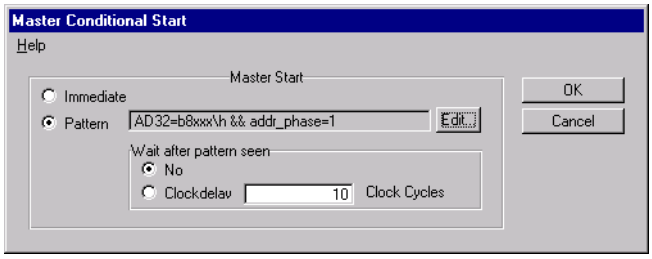
You can also set up a combination of the conditional start and the fixed delay. In this case the master first waits for the pattern event to occur. After this event has been detected, it waits the additional delay time before starting the transactions.

The detection of a pattern event is implemented with a boolean expression. If this expression turns true, the master starts running. The fixed delay is simply a number of clock cycles that you set in the GUI.

You can combine the two features or use only the conditional start or only the fixed delay start. For the latter case you set the boolean pattern expression to TRUE to make it switch immediately. Then the master only waits the fixed delay.

To set up a delayed start of either type, proceed as follows:

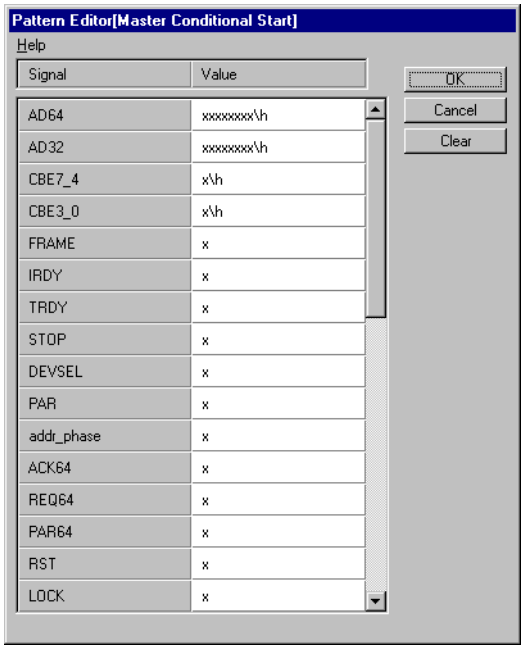
1 Select *Master Cond. Start* from the *Exerciser* menu in the main window.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 Select the *Pattern* radio button and then click the *Edit* button.

The pattern editor will be opened.



3 Assemble the pattern. For this purpose, click in the *Value* column pertaining to the required signals. If you keep the mouse button pressed, a box with suitable values appears (except for bus signals like *ADxx* or *CBEx_x*, which require hex value inputs).

The signals of the pattern are automatically combined with a logical AND. For example, setting the *IRDY* and *TRDY* signals to "1" and entering *FBFFxxxx* as the value for the *AD32* address, will result in the boolean expression:

IRDY=1 && TRDY=1 && AD32=FBFFxxxx

4 After you have included all required conditions for your signals, click *OK*.

The pattern editor will be closed, and the pattern term will appear next to the *Pattern* radio button.

5 Check the correctness of the pattern term.

6 If you wish to delay the start of the master transactions an additional number of clock cycles, click the *Clockdelay* radio button and enter the number of clocks in 32-bit range (up to 4096 M).

7 Click *OK* to finish your conditional start setup.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Starting the Master

After you finished programming the master transactions along with master attributes and a start condition, if required, you can start your test. The different ways to run the master are:

- To run the master alone, select *Run* > *Run Master* from the *Exerciser* menu.
- To start several devices simultaneously, select *Start* from the *Run* menu, or.
  click the Run button  in the main window.

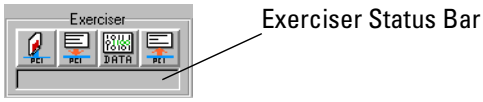
For the latter cases, the settings in the Run Options window define which devices are started.

Master Run Status

When the master is started, the entered transaction and attribute scripts are compiled. Internally, the master uses these settings to implement a state machine, which then performs all actions.

As a result, it requests bus access from the bus arbiter. When the bus is granted to the master, it drives the transactions on the bus as specified.

During the master run, the current status of the master is displayed in the status bar. Error status messages, such as *CE* for “Compare Error”, will help you to identify and solve possible problems in your system under test.



The following list explains every status that can be displayed in the Exerciser status bar:

- *Compiling*. The master is compiling the entries in the transaction editor and the attribute editor. No testcard programming is done yet.
- *Reinitializing*. The master and target state machines are programmed on the testcard.
- *Starting*. The master is going to be started.
- *Failed*. The start of the master or the initialization of the target has failed.
- *Running*. The master is running.
- *Stopped*. The master has stopped.
- *CE*. A data compare error occurred.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Stopping the Master

If the master is not set up in repetitive mode, it will stop automatically after it has successfully performed the specified transactions.

However, you can also stop the master by hand. The different ways to manually terminate the master run are:

- clicking the Stop button  in the main window,
- clicking the *Stop* button in the Run Options window,
- selecting *Stop Master* from the *Exerciser* menu,
- selecting *Stop* from the *Run* menu.

Manual Stop Required

There can be situations in which the master does not finish the transactions, and you are then required to stop the master manually.

- The conditional start pattern does not occur on the bus and, thus, the master does not transfer any data.
- The master runs in repetitive mode.
- The specified target only returns a retry on every initiated transaction.
- The bus hangs or other severe errors occurred.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The PCI Exerciser as a Target Device

The Agilent E2940A testcard can simulate any device on a PCI bus. As there are mainly two different types of devices—masters and targets—the Exerciser contains both a master and a target along with a few other components. All necessary information about the testcard's **target device** is found here.

If the Agilent E2940A testcard is set up as a PCI target device, it can be used to test the functionality of a master device on the bus. Another usage is to send data from the testcard's master to its own target to increase bus load.

Definition of a Target

A PCI device is called a target if it has one or more address spaces assigned to it, and if it reacts to PCI bus transactions that are sent to these addresses.

Configuration Space and Target Decoders

Like every PCI target device, the Agilent E2940A testcard has got a configuration space and target decoders. These features are described in detail in *"Configuration Space and Target Decoders" on page 54*.

For information on how to set up the target decoders and how to modify the configuration space header with the Agilent Graphical User Interface, see *"Target Decoder Setup" on page 62*.

When setting up the testcard you should consider that the testcard has two memories for storing information on the configuration space header and the target decoders (see *"Standard and Power Up Databases" on page 69*).

Target Attributes

Additionally, the Agilent E2940A testcard provides full control of the target's protocol behavior. This means, you can determine how the target is going to react to transactions in terms of inserted waits, retries, target aborts, etc. For more information on how to specify the target behavior, see *"Controlling Target Attributes" on page 71*.

Initializing the Target

Normally, after changes are done to the target setup, the target needs to be either initialized or rebooted, depending on the type of the changes. Information on how to properly initialize the target is found in *"Initializing the Target" on page 78*.

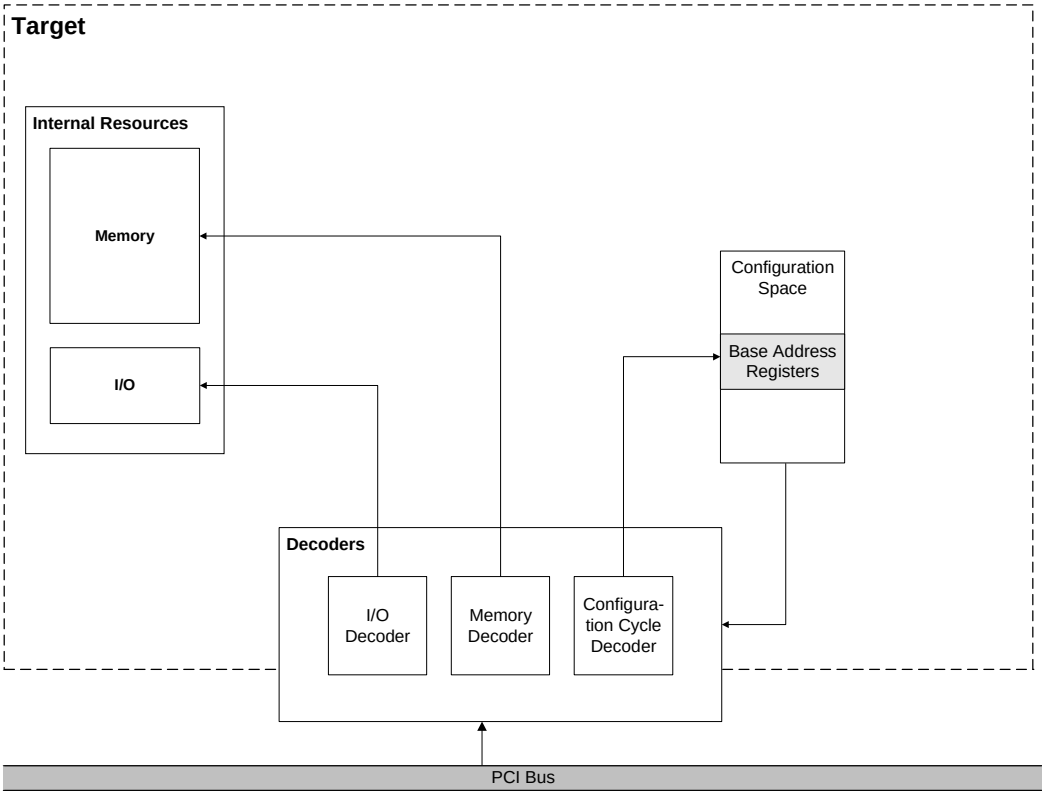
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Configuration Space and Target Decoders

In general, every PCI target device consists of the following components:

- a **configuration space** holding configuration data about the device (see *"Configuration Space Header" on page 55*),
- a set of **decoders** for different types of accesses (I/O, memory, configuration cycles, etc., see *"Target Decoder Properties" on page 57*),
- **data resources** where received data is stored or processed, or where data to be sent is taken from (see *"Data Resources" on page 61*).

The following figure gives an overview of the components of a target.



Different targets use different components to match their individual requirements. The Agilent E2940A testcard can be set up to simulate any type of target device.

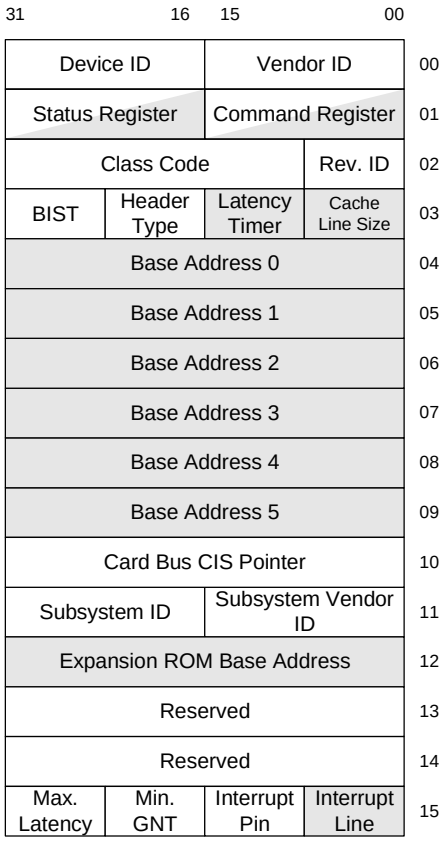
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Configuration Space Header

Every PCI device has a configuration space, which consists of 32-bit registers containing configuration data. Multi-function devices provide one configuration space per functional unit.

The configuration space of most PCI devices is divided into two sections, a public and a private section. The first 16 registers of the configuration space make up the public section. They are referred to as the **configuration space header**. It is accessible by the system BIOS and used for system management. The private section is optional and can be used for private, device specific purposes.

The following figure shows the configuration space header of a PCI device according to the PCI specification.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The configuration space header consists of 16 32-bit registers containing configuration data. The information stored in these registers is needed by the system to provide proper communication between the different devices on the bus. In the figure above, the registers with white background are fixed—they cannot be modified by the system. These registers contain information like the device type, latency and interrupt information, and more. Status and command registers are partially fixed.

Base Address Registers

The gray registers can be modified by the system. They hold the base addresses of the address spaces assigned to the different target decoders. The BIOS assigns these address ranges to the PCI devices during power up to ensure non-conflicting address ranges for all devices.

The main two different types of base address registers are:

- Base Address Registers 0 to 5

  These registers also determine whether a decoder decodes memory or I/O transactions, the location of the address range (in 32 or 64-bit address space, or below 1 MB) and whether it is prefetchable.

- Expansion ROM Base Address Register

  A PCI device may provide a device ROM containing a power-on self-test, BIOS or interrupt service routines. This register contains information on whether an expansion ROM exists and where its address space is located.

The target needs the address range information to decide whether it has to react to transactions driven onto the bus. The component of the target decoding and claiming a transaction is called a decoder. There are different types of decoders for different types of accesses. The decoders are tightly connected to entries in configuration space.

The Agilent Testcard's Configuration Space Header

In order to simulate all different possible types of PCI devices, the configuration space header of the Agilent E2940A testcard is freely programmable.

The contents of every single register can be changed according to your test requirements. Furthermore, you can determine which bits of the registers are fixed and which can be changed by the BIOS during the configuration cycles.

The settings in the configuration space header of the testcard always determine the *current* behavior of the associated decoders. They can be reprogrammed after the BIOS configuration phase, so that the testcard can behave different than initially declared to the BIOS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE The PCI specification exactly defines the meaning of the various bits in the configuration space header. Thus, a fair amount of knowledge is needed to do proper settings and to avoid errors. The option to completely program the configuration space header is provided for very sophisticated tests, for example, to test systems without a BIOS.

All the settings regarding the base address registers and target decoders can be done in the GUI by specifying the respective properties in the Target Decode window (see *"Target Decoder Setup" on page 62*).

Target Decoder Properties

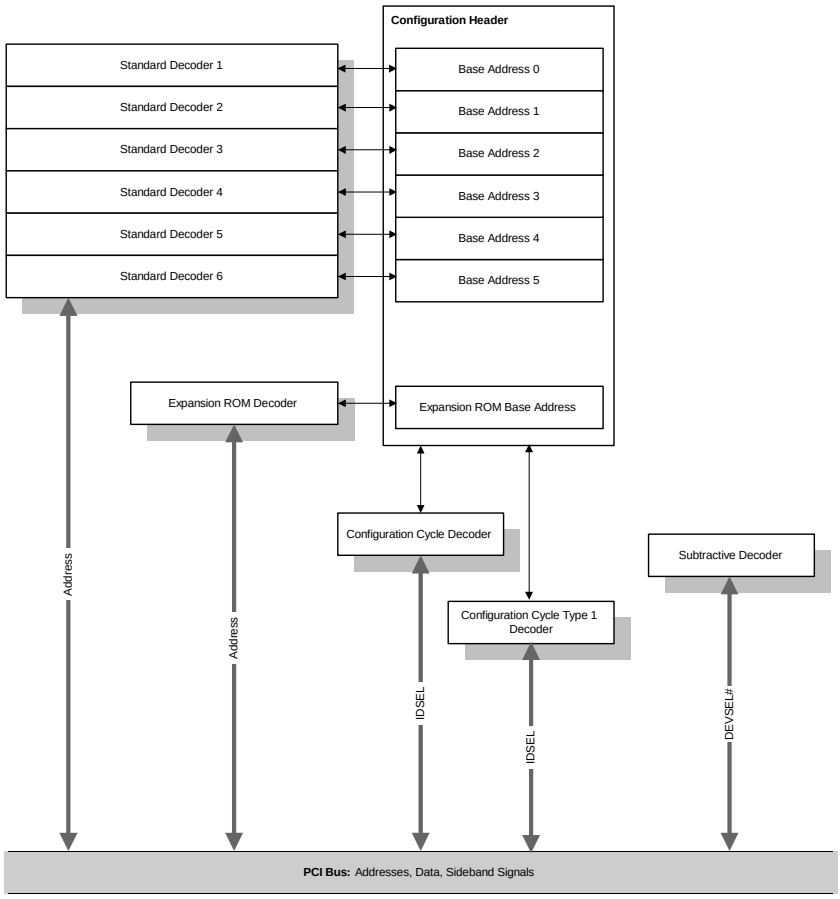
The Agilent E2940A testcard provides several decoders for different purposes. Depending on the contents of the base address registers in the configuration space header they claim transactions from the bus.

The following picture shows all decoders of the Agilent E2940A testcard, but only the following decoders are accessible from the GUI:

- the 6 standard decoders,
- the expansion ROM decoder,
- the configuration decoder.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The configuration cycle type 1 decoder and the subtractive decoder can only be controlled via the C-API. For details, see the *Agilent E2940A Programmer's Guide*.



The transactions to be claimed by each decoder are specified by the base addresses (see *"Base Address" on page 60*).

Additionally, each decoder is equipped with a programmable set of parameters that define its properties such as the decode speed, whether the decoded memory space is prefetchable, or which data resource it is connected to (see *"Data Resources" on page 61*).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Standard Decoders

The standard decoders are connected to the 6 base address registers in the configuration space header of the testcard. They claim transactions to memory and I/O address ranges as defined by the base address entry and the programmed size.

To decode 64-bit addresses (dual address cycles, DAC), two standard decoders are used. For example, the standard decoders *BAR 0* and *BAR 1* may be combined to build *BAR 0-LO* and *BAR 0-HI*.

The standard decoders can be programmed to “fast decode”, that is they reply to a master’s request without delay by asserting the DEVSEL# signal. However, fast decoding reduces the available address space and the number of bus commands that can be employed.

When programming the testcard, note that there must not be any unused decoders between used ones. In other words, if you need three standard decoders, use decoders 1, 2, and 3, and not 1, 3, and 4, for example. Otherwise, the setup is not PCI-compliant and the complete card may be disabled by the BIOS.

Expansion ROM Decoders

The Agilent E2940A testcard provides a programmable expansion ROM. Expansion ROMs typically are used as boot ROM. They can contain code as, for example, own power-on-self-tests, BIOS and interrupt service routines. These can be loaded and executed by the BIOS of the system under test by using the expansion ROM decoder.

The expansion ROM decoder behaves just like a standard decoder. It is, however, connected to the Expansion ROM Base Address Register in the configuration space.

Configuration Decoder

This decoder decodes configuration cycles. There is no entry in the configuration space needed, because configuration cycles aim to the configuration space itself. Whether or not the decoder claims a transaction depends on the bus commands (“config read” and “config write”), the IDSEL signal (Initialization Device Select), and the function, if it is a multi-function device.

This decoder can be used, for example, to test the start-up behavior of the system under test (BIOS tests, host bridge test, etc.).

The configuration decoders can be programmed to “fast decode”, that is without delay between decoding and asserting the DEVSEL# signal.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Base Address

The information stored in the base address registers determines which transactions the decoder claims. The following parameters are coded:

- Base Address and Size.

  The contents of the base address register always start with a bit sequence followed by a number of zeros. The base address is coded in the most significant bits of the registers. The number of following zeros defines the size of the decoded address range. A transaction to any address that starts with the same sequence as the base address will be decoded. As an example, if the base address is `3b4c0000`, every transaction between this start address and `3b4cffff` will be claimed.

- Location and Prefetchable.

  There is a minimum size for address ranges, because the least significant bits are needed to code the address range location and the prefetchable flag.

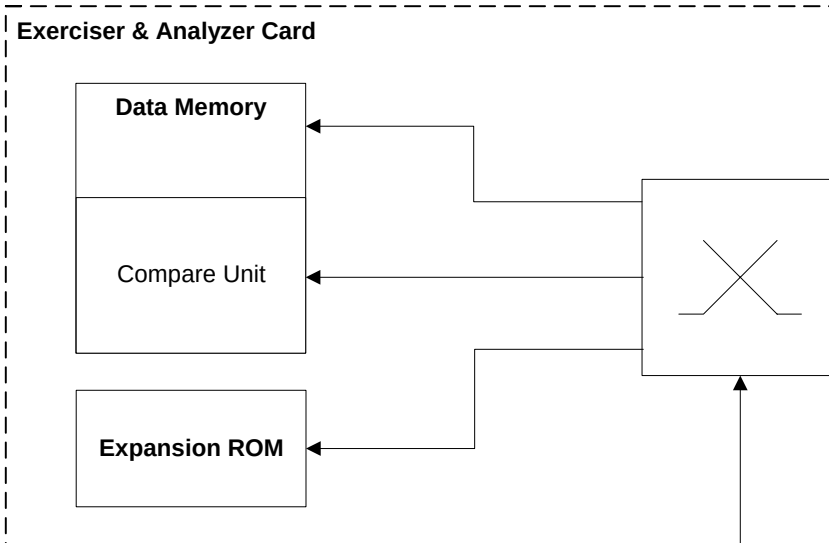
  The location must be considered by the BIOS when allocating the memory. The available options are in 32-bit range, in 64-bit range, below 1 MB, and in I/O range.

  The prefetchable flag specifies whether memory is prefetchable and, thus, whether a master can take advantage of optimized access to the target's memory. This also must be considered by the BIOS when allocating the memory. This property is not available in I/O range.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data Resources

After a decoder has claimed a transaction, data will be transferred in a certain direction (performing either a read or a write command). For this data, the testcard provides several memories as data resources. An internal switch allows connection of any resource to any decoder.



Along with the selected resource, you need to specify an internal start address and the size for this resource. This size does not need to be equal to the size of the address range that the decoder actually decodes.

The available resources of the Agilent E2940A testcard are:

- Data Memory and Compare Unit.

  You can specify several internal memory address ranges (with an internal base address and the size) as resources. Alternatively, instead of storing the data in the memory, it can be compared to reference data from the memory.

  For more information about the data memory and the compare unit, refer to *"Using the Data Memory" on page 79*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Configuration Space.

  The complete configuration space of the Agilent E2940A testcard (public and private section) can be employed as a resource. The private section contains the interrupt status register and some registers for internal use. This can be useful, for example, to read configuration space information or the interrupt status from I/O address space.

NOTE Overwriting the configuration space (especially the private section) can result in loosing the PCI connection to the testcard. But even in this case, you can still use the RS-232 cable to communicate with the testcard.

- Expansion ROM.

  This address range emulates a PCI device's expansion ROM, which usually contains the boot software. Its size is 64 kByte and it is available via the Expansion ROM Base Address Register.

Target Decoder Setup

In order to run your Agilent E2940A testcard as a target device on the PCI bus, you need to set it up according to your test requirements.

The Graphical User Interface of the Agilent E2940A testcard enables you to quickly view and edit the setup of the target decoders. In a system under test with a BIOS, this setup is mainly done automatically during power up. However, you might still be interested to view or change this setup.

Some settings, like the internal connections to data resources or the protocol behavior can be used directly after the target is initialized. Other settings concerning the BIOS and the configuration space require that the system is rebooted. For this purpose, your changes need to be stored in the power-up database which is used on restart.

Because the Agilent E2940A testcard is able to emulate any PCI device and to test any system, even without a BIOS, there is also the option provided to manually perform the necessary assignments (see *"Overwriting BIOS Settings" on page 68*).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The explanation of how to set up the target decoders is found in *"Programming the Decoders" on page 63*.

How to do changes to the configuration space header is described in *"Modifying the Configuration Space Header" on page 65*.

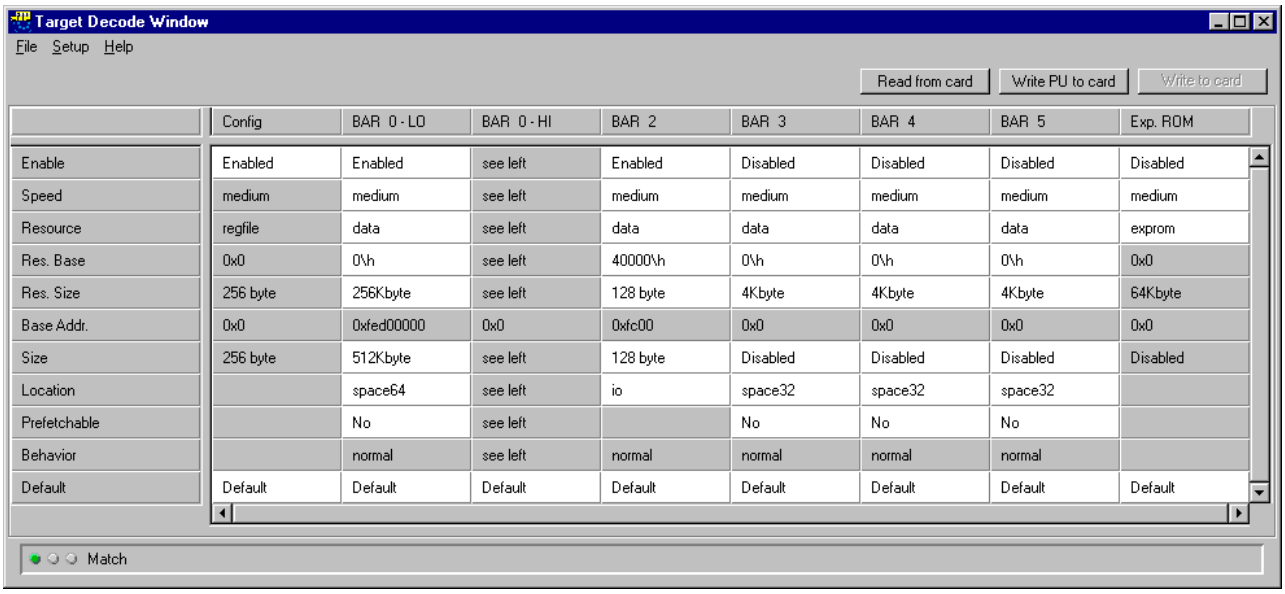
All settings that can be done with the Target Decode window of the GUI result in PCI specification compliant behavior of the target. Therefore, the behavior is described as *normal*. When doing changes to the configuration space that are not compliant with the PCI specification, the behavior becomes *custom*.

NOTE With the C Application Programming Interface (C-API), there are more options to program the decoders and their behavior. See the *C-API/PPR Reference*.

Programming the Decoders

To set up the target decoder settings, follow the steps below. If you only want to view the current settings, skip the respective steps.

1 To open the Target Decode window, select *Target Decode* from the *Exerciser* menu in the main window.



	Config	BAR 0 - LO	BAR 0 - HI	BAR 2	BAR 3	BAR 4	BAR 5	Exp. ROM
Enable	Enabled	Enabled	see left	Enabled	Disabled	Disabled	Disabled	Disabled
Speed	medium	medium	see left	medium	medium	medium	medium	medium
Resource	regfile	data	see left	data	data	data	data	exprom
Res. Base	0x0	0\h	see left	40000\h	0\h	0\h	0\h	0x0
Res. Size	256 byte	256Kbyte	see left	128 byte	4Kbyte	4Kbyte	4Kbyte	64Kbyte
Base Addr.	0x0	0xfed00000	0x0	0xfc00	0x0	0x0	0x0	0x0
Size	256 byte	512Kbyte	see left	128 byte	Disabled	Disabled	Disabled	Disabled
Location		space64	see left	io	space32	space32	space32	
Prefetchable		No	see left		No	No	No	
Behavior		normal	see left	normal	normal	normal	normal	
Default	Default	Default	Default	Default	Default	Default	Default	Default

The columns of this window contain the settings for the configuration decoder, the six standard decoders, and the expansion ROM decoder (from left to right).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note, that the default settings set by BIOS include the decoder settings *BAR 0-LO* and *BAR 0-HI*. The standard decoders *BAR 0* and *BAR 1* are set up together to decode 64-bit addresses.

2 To display the settings currently in use in the testcard's standard database, click the *Read from card* button.

Now the match indicator in the lower left corner of the window should show a green color.

3 For every decoder select *Enabled* or *Disabled*. If a decoder is disabled, it does not claim any transaction, no matter what the other settings for this decoder are.

4 For the enabled decoders, select the decode speed, either *slow*, *medium*, *fast*, *pfast* or *nodevsel*.

5 Select the *Resource*, the resource base address (*Res. Base*) and the resource size (*Res. Size*) for the decoders.

These parameters determine which data resource on the testcard the decoder is connected to internally. This resource is used to supply and/or receive data for the transactions. The available options are:

- Data memory and compare unit. The transactions to these resources can be performed with the protocol attributes specified in the attribute editor (resources *data* and *comp*) or with default attributes (resources *datadef* and *compdef*).
- Registerfile (*regfile*). This selection connects the decoder to the configuration space, both public and private section.
- CPU port 0 and 1.
- Expansion ROM.
- Static I/O.

6 Specify the address range to be covered by the decoders (parameters *Base Addr.*, *Size*, *Location*, and *Prefetchable*).

The settings for these parameters are stored in the configuration space header of the target. By default, they are protected and cannot be changed for the running system. However, you can store them in the power-up database by clicking the *Write PU to card* button. When restarting the system, they will be considered by the BIOS.

By default, the base address is not editable in this window. BIOS normally assigns the base addresses to the various targets. However, you can decide to overwrite the BIOS settings. See *"Overwriting BIOS Settings" on page 68* for details.

When selecting the base address manually, ensure that multiple targets do not decode overlapping address ranges.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CAUTION

Multiple devices decoding overlapping address ranges may result in hardware damage.

7 If you want to reset the settings of a decoder to the default values, tick the default check box.

8 Write your settings to the testcard.

- If you want to use the settings immediately, click the *Write to card* button. By default, BIOS-relevant settings will not be stored to the testcard (see *"Overwriting BIOS Settings" on page 68*).
- If you want to use the settings after the next power up, click the *Write PU to card* button. With this option also BIOS-relevant settings can be stored and employed.

9 Check the match indicator in the lower left corner of the window.

If the display matches the settings in the testcard's standard database, the indicator shows a green color. If the display matches the contents of the power-up database, it is yellow, and if it does not match at all, it is red. The latter can happen, because changes to BIOS relevant settings like the memory location cannot be written to the standard database of the testcard (see *"Overwriting BIOS Settings" on page 68*).

Modifying the Configuration Space Header

With the Graphical User Interface of the Agilent E2940A testcard you can freely program every single register of the testcard's configuration space header. Usually, all settings in the configuration space header that need to be done when starting the system are either done by the BIOS or from within the Target Decode window. The latter applies to settings concerning the base address registers.

The option to completely program the configuration space header is provided for very sophisticated tests, for example, to test systems without a BIOS or with a very rudimentary BIOS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For every bit of the various registers in the configuration space header, you can determine whether it is fixed or programmable from the outside (BIOS). For both types, values can be specified as required for BIOS configuration during system start up:

- Determine **fix** values, for example a “Vendor ID”, which is then read-only and can be evaluated by the BIOS.
- Determine **programmable** values, for example base address register entries. They can be used by the BIOS to determine the wanted size of the decoded address range and will then be overwritten with the actual base address.

NOTE The PCI specification exactly defines the meaning of the various bits in the configuration space header. Thus, a fair amount of knowledge is needed to do proper settings and to avoid errors.

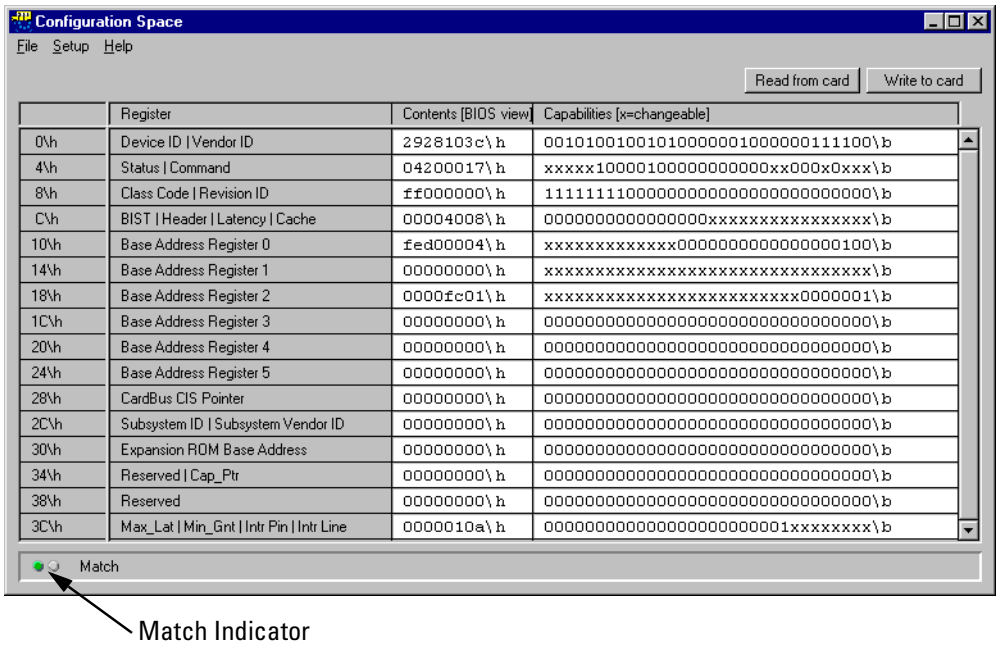
In some cases, manipulating the base address registers in the configuration space header can result in the testcard not being accessible via the PCI interface—neither by the controller nor by the software application—especially if the improper settings are stored as power-up defaults. In this case, you can still use the RS-232 interface cable to access the testcard.

CAUTION Setting up the PCI testcard to decode address ranges that are also decoded by other devices can result in hardware damage.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Edit the configuration space header of the Agilent E2940A testcard as follows:

1 Select *Config Space* from the *Exerciser* menu in the main window to open the Configuration Space window.



2 Regard the *Match* indicator in the status bar at the bottom of the window.

If the indicator shows a red color, the current settings on the PCI testcard and the values presented in this window do not match. To load the current settings from the card into this window, click the *Read from card* button.

3 Edit the fields of the table according to your test requirements.

The lines of the table represent the different registers of the configuration space header. Some lines contain several registers, indicated by a vertical line separator in the *Register* column. The columns of the table show from left to right:

- The hexadecimal byte address relative to the first byte of the configuration space.
- *Register*. The name(s) of the register(s) according to the PCI specification.
- *Contents [BIOS view]*. The current values in the register(s) as seen by the BIOS.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

– *Capabilities [x=changeable]*. These fields contain a representation of the registers' programming mask. 0 or 1 means: This bit is fixed (read-only) and its value cannot be changed by the BIOS or another routine from outside. x means: This bit is programmable by the BIOS as well as by other routines.

4 Click the *Write to card* button.

The current contents of this window will be written to the configuration space header and the mask memory of the Agilent E2940A testcard.

If a bit has a fixed value in the right column, a possibly differing value in the *Contents* column will be ignored.

5 Close the Configuration Space window.

A dialog box appears, informing you that you made changes to the settings in the configuration space header.

6 Click *Store*, if you want to make these changes persistent for the next time you power up the testcard.

Or click *Close*, if you want to use the new modifications only for the current test session.

Overwriting BIOS Settings

Usually, the system BIOS handles the assignment of the address spaces to the different target devices on the PCI bus. These address spaces are asserted when the system starts running and are then valid constantly.

Therefore, all settings that are relevant to this mechanism are protected from manual changes. This applies to the standard database where the settings are stored that are currently in use. However, you can write your modified settings to the power-up database which will be considered by the BIOS after restarting the system under test.

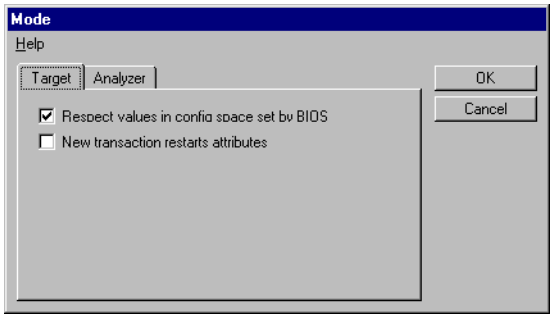
The exception to this behavior is the base address itself. For safety reasons, the fields holding the base addresses in the Target Decoder window are not editable by default. This is to avoid hardware damage possibly caused by overlapping address ranges for different decoders.

Because the Agilent E2940A testcard is able to emulate any PCI device and to test any system, even without a BIOS, there is also the option provided to manually perform the necessary assignments.

1 Open the Mode window by selecting the *Mode* item from the *Setup* menu in the main window.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 Select the *Target* tab.



3 Untick *Respect values in config space set by BIOS* to disable the protection. Or tick this property again to protect the BIOS settings.

4 Click *OK*.

Now you can do your changes to the target settings including the base addresses and store them in the power-up database as well as in the standard database.

CAUTION

Ensure that different targets do not decode overlapping address spaces. Decoding overlapping address spaces may result in hardware damage.

Standard and Power Up Databases

Some of the changes you do in the configuration space header and the target decoders can be used immediately in the running system. Others, however, do not come into effect until the system under test is rebooted. This applies to settings that are (at least partially) done by the BIOS during the configuration phase when starting the system.

For this purpose the Agilent E2940A testcard provides two databases to store all settings.

- Standard Database.

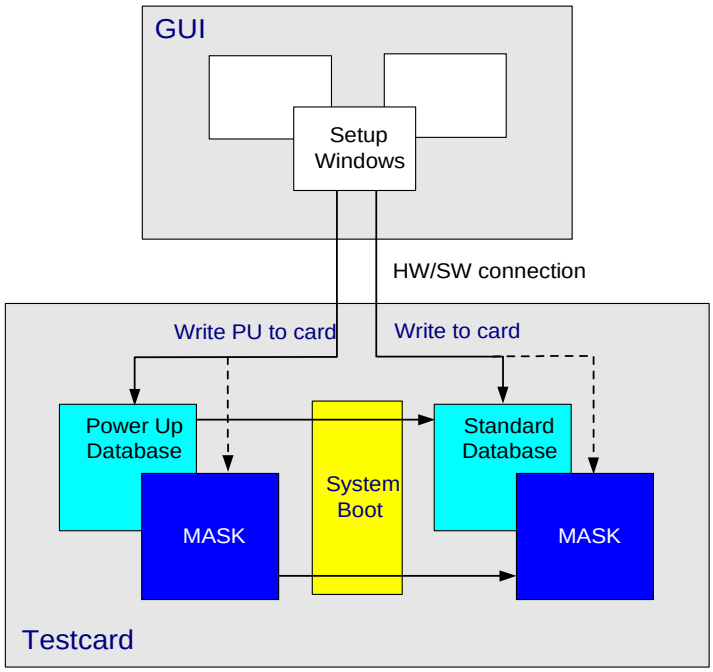
  This memory keeps the currently used settings for the configuration space header and the decoders.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Power Up Database.

  If you want to specify settings that shall be used when restarting the system under test, you have to store them in the power up database. When the system is booted, the contents of the power up database are loaded to the standard database. After that, the BIOS sets all bits that it has given access to.

The figure below illustrates the two databases.



For both databases you can determine freely, which bits are fixed and which can be modified by BIOS or other program routines. This information is stored in the masks.

This is done in the right column of the Configuration Space window of the GUI (see *"Modifying the Configuration Space Header" on page 65*). All bits that are marked with an x can be set by BIOS. All other bits are set fixed to their values 0 or 1. The values that BIOS has inserted in the programmable areas are shown in the middle column.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Controlling Target Attributes

In contrast to a master, a target cannot perform a “target run” as it has a passive behavior. After setting up and enabling the decoders and base address registers, the Agilent PCI Exerciser is able to react to accesses from master devices.

Besides these functional settings, the PCI Exerciser also allows to specify attributes that control the target behavior regarding the PCI protocol. This means, for instance, that you can define the number of waits that the target inserts in a data phase, or whether the target signals a retry or a parity error (see *“Available Target Attributes” on page 71*).

These attributes do not affect the result of a data transfer (unless you block the target completely). Hence, you can repeat a test with fixed target decoder settings but varying target attributes and then compare the results.

Target Attribute Memory

All these attributes are stored in the target attribute memory. In this memory the attributes are kept in memory lines, and every memory line is then assigned to a transaction phase.

Target Attribute Editor

To set up these target attributes, the PCI Exerciser provides the Target Attribute Editor window (see *“Implementing Target Attribute Scripts” on page 75*).

Available Target Attributes

The different types of target attributes are:

- *“Target Address Phase Attributes” on page 72*
- *“Target Data Phase Attributes” on page 72*
- *“Target Control Attributes” on page 75*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Target Address Phase Attributes

The PCI Exerciser uses the address phase attributes only during an address phase of a transaction. During data phases these attributes are ignored. The following list gives an overview of the available address phase attributes. The attributes are specified in Bus Transaction Language (BTL) scripts. The corresponding BTL parameter is added in parentheses.

For a complete list of the commands and parameters and their exact syntax, refer to the *"Bus Transaction Language Reference" on page 97*.

- 64-Bit Acknowledge (`ack64`).

  With this attribute, the target signals that it is capable of accepting 64-bit accesses.

- System Error Signaling (`aperr`, `dacperr`).

  As a target, the PCI Exerciser has two options to signal a system error for an address phase:

  – two clocks delayed after a (32-bit) address phase (`aperr`),

  – two clocks delayed after the second cycle of a dual address cycle (64-bit) access (`dacperr`).

  System errors for address phases always occur two clocks delayed, because the system needs this time to notice a parity error, which is the normal reason for system errors.

  System errors must be enabled in the configuration space header. Otherwise, this property will not have any effect.

Target Data Phase Attributes

The PCI Exerciser uses the data phase attributes only during a data phase of a transaction. During an address phase, these attributes are ignored. Note, that after the Exerciser has employed an attribute line for an address phase, it does not switch to the next line. Thus, the data phase attributes in this line will be used with the first data phase. Then, after the data phase is completed, the Exerciser switches to the next attribute line.

The following list gives an overview of the available data phase attributes. The corresponding BTL parameters are added in parentheses.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For a complete list of the commands and parameters and their exact syntax, refer to the *"Bus Transaction Language Reference" on page 97*.

- Waits (`waits`).

  The PCI Exerciser can insert up to 30 wait cycles into a data phase to create target latencies. The target latencies are defined as the number of waits between address and data phase (initial latency), and the number of wait states during data phases (subsequent latencies). Because master and target may use the same data resource internally, target latencies can be increased by master actions.

  The table below shows the minimum target latencies of the Agilent E2940A testcard in different situations. The first three columns—master status, direction, and successive linear address—describe the situation in which a transfer can occur:

  – The status "Idle" means that the master of the testcard has no pending transaction.

    "Transfer Intended" means that the master is preparing a transaction and, thus, internally hindering the target from using resources.

  – The "Direction" specifies the transfer direction determined by the bus command.

  – The "Successive Linear Address" tells whether the currently used bus address directly follows the address used in the previous transaction.

  The two columns to the right contain the minimum latencies achievable with the Agilent E2940A testcard.

Table 1

Exerciser Master Status	Direction	Successive Linear Address	Min. Initial Latency	Min. Subsequent Latency
Idle	Write	Don't care	0	0
	Read	Yes	1 (Turn around)	0
		No	7	0
	Write Compare	Yes	1	0
		No	7	0
Transfer Intended	Don't care		less than 16	0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Termination (`term`).

  With this attribute, the current transfer can be terminated after the specified number of waits have passed. The possible ways to terminate the transaction are as follows:

  – `retry` means that the Exerciser will not accept the data. This is implemented by signaling a retry or a disconnect-C, depending on the position in the burst.

  – `disconnect` means that the Exerciser will accept the data and then terminate with disconnect-A or disconnect-B, depending on the number of master waits.

  – `abort` means that the Exerciser signals a target abort.

- Marker (`marker`)

  With this attribute you can set the marker to an integer value during a transaction phase. This number can then be used as input for pattern terms. Pattern terms are used for synchronization with other parts of the PCI testcard, for example, to trigger the trace memory of the Agilent PCI Analyzer.

- Wrong Parity Calculation (`wrpar`, `wrpar64`).

  The PCI Exerciser can set the parity bits `wrpar` and `wrpar64` to the wrong value delayed by one clock after a read data transfer.

- Parity Error Signaling (`dperr`).

  The PCI Exerciser can assert the parity error signal (PERR#) two clocks after a write transfer, along with the target ready signal (TRDY#) or STOP#. During read transfers, this property is ignored. If a real parity error should occur concurrently, it will be ignored, but it will be recognized by the testcard's protocol rule observer and can be used to trigger the trace memory.

  Note, that parity errors must be enabled in the configuration space. Otherwise this property will not have any effect.

- System Error Signaling (`dserr`).

  The Exerciser can signal a system error two clocks delayed after a data phase.

  Note, that signal errors must be enabled in the configuration space. Otherwise this property will not have any effect.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Target Control Attributes

During the address phase of a transaction, the data phase attributes in the corresponding attribute line are ignored. They are used during the following data phase. Then the next attribute line is employed for the next transaction phase. In other words, the attribute memory switches to the next line after every data phase, but not after address phases. This process does not depend on the success of the data transfer during the data phases.

This general behavior can be overridden by so called control attributes. These attributes are used to determine how the Exerciser processes the lines in the target attribute memory. They are, therefore, valid for address phases as well as for data phases. From within the GUI only one control attribute is available:

- Repeat (`repeat`).

  With this attribute you can repeat one attribute line multiple times. The maximum number of repetitions is 4 billion (2^{32}).

Implementing Target Attribute Scripts

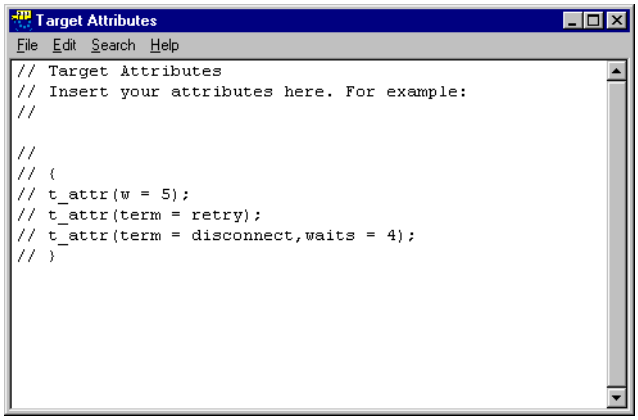
The target attributes determine the protocol behavior during the various phases of each transaction. They control, for example, whether errors are signaled during address phases or how many waits are to be inserted during a data phase. This should have no impact on the result of the transaction. The data must be transferred correctly regardless of how many waits or retries have occurred during the transaction.

Programming of target attributes is optional. If no attributes are specified, default attributes will be used. This results in a “friendly” target behavior.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Target Attribute Editor

The attributes are specified in a Bus Transaction Language (BTL) script, which is entered in the Target Attributes editor window. The target attributes are specified by the command `t_attr()`.



1. To open the target attributes editor, select *Target Attributes* from the *Exerciser* menu in the main window.
2. Enter your BTL script, for example, as shown in the figure above, without the comment slashes at the beginning of each row. Another example could be:

```
{
   t_attr(dperr);
   t_attr(wrpar);
   t_attr(waits = 10);
   t_attr(waits = 3, term = retry);
}
```

This example signals a data parity error in the first data phase of a burst and sets a wrong parity in the second data phase, if it is a write transaction. Into the next data phase it inserts 10 waits. The fourth data phase will then be answered with 3 waits and a following retry.

Your scripts will be saved together with the setup in the .bst file. You can also save the scripts individually, using the *File* menu of the transaction editor.

BTL Script Syntax

Basically, when writing a target attributes BTL script, the same rules apply as for writing a master attribute script. The only difference is that the `t_attr()` command is used instead of `m_attr()`. Also, the target attributes do not need to be set up in pages as the master attributes.

For an exact description of the BTL syntax, all commands and parameters, please refer to *"Bus Transaction Language Reference" on page 97*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

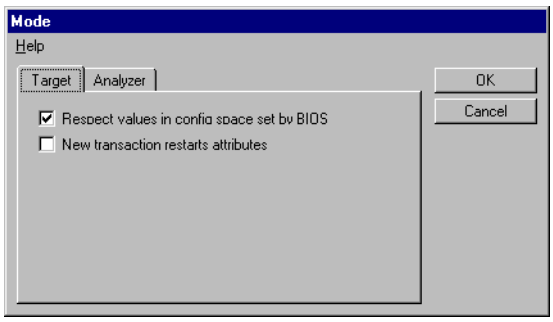
Setting the Reset Mode for Target Attributes

By default, the target attributes are restarted each time the testcard is accessed as a target, or when the end of the script is reached. You can, however, also set the reset mode for the target attributes. In this mode the target attributes are restarted already after completion of every transaction.

To set the reset mode for the target attributes, proceed as follows:

1 Select *Mode* from the *Exerciser* menu in the main window.

2 Select the *Target* tab of the *Mode* window.



3 Tick *New transaction restarts attributes*.

4 Click *OK*.

Now, after initializing the target, the attributes are restarted with every new transaction.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initializing the Target

Normally, the target of the Agilent E2940A testcard starts automatically after power up and runs without a break, employing the initial decoder and attribute page settings.

When programming new attribute settings, the target needs to be re-initialized. The process of initialization involves

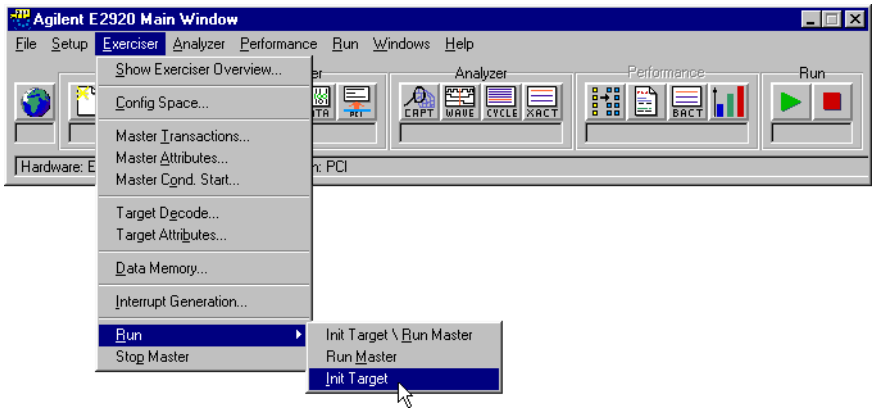
- compiling the target attribute script,
- writing the compiled script down to the testcard,
- setting the ready status again.

After that, the target is again ready to react on master transaction attempts.

Manual and Automatic Initialization

By default, the target is initialized automatically when running the Exerciser, for example by clicking the Run button in the main window.

If you want to initialize the target without affecting any other component of the Exerciser, select *Run* > *Init Target* from the *Exerciser* menu in the main window.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Data Memory

The Agilent E2940A testcard provides an internal data memory that can be used as a data resource by both the master and the target. One usage is to supply data that is sent onto the bus, another to store received data from the bus. Furthermore, received data can also be compared to reference data in the data memory (without being stored).

The data memory is optimized for high throughput and programmable protocol behavior. It is implemented as a wrap-around memory that can emulate a virtual data resource that is much larger than its actual size of 64K lines.

If you seek information on how the data memory is built up physically, see *"Organization of the Data Memory" on page 80*.

For information about the data compare unit, refer to *"Data Compare Unit" on page 82*.

Details about how to use the features of the data memory in the GUI is found in *"Using the Data Memory Editor" on page 83*.

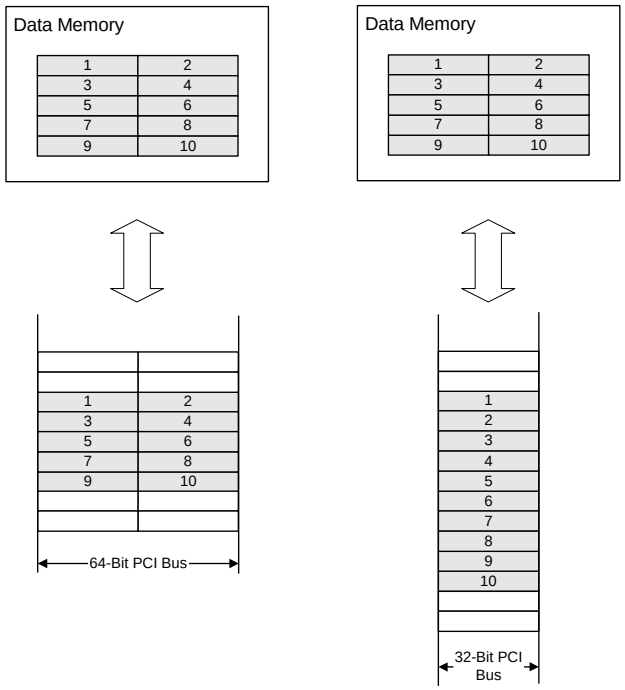
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Organization of the Data Memory

Data Alignment

The internal data memory provides 64K lines of each 64 bits memory space. For usability reasons, when transferring data between the PCI bus under test and the data memory, the data must be aligned with respect to the PCI bus width.

The following figure illustrates how data that is stored in the data memory is driven onto the bus—and how received data is stored—in 32 and 64-bit systems.



To align the data correctly for data transfers to or from the PCI bus, follow the rules below.

- For 64-bit transfers: The specified byte addresses must be 64-bit aligned. For both the internal address and the PCI bus addresses, the least significant **three bits** must be 0.
- For 32-bit transfers: The specified byte addresses must be 32-bit aligned. For both the internal address and the PCI bus addresses, the least significant **two bits** must be 0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Master and Target Partitions

Basically, you are free to use any part of the data memory both for the master and the target. However, if you need to use the data memory as a resource for both devices at the same time, it is recommended to define memory partitions.

This can be done by using different internal addresses for the master and the target. This applies to the following parameters:

- In the master: The block properties “internal address” and “number of dwords” must be set appropriately.
- In the target: When the “resource” property is set to “data memory” or “compare unit”, the properties “resource base address” and “resource size” must be set appropriately.

NOTE If the master of the Agilent E2940A testcard is communicating with its own target via the PCI bus, the target does not have any access to the data memory. When the master performs write commands, the target cannot store the received data in the data memory. Similarly, when performing read commands, the target will send dummy data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data Compare Unit

The compare unit on the Agilent E2940A testcard can be used by both the Exerciser's master and target. If enabled, the master uses it for read operations and the target decoders use it during write operations.

For the master, the usage of the compare unit can be enabled and disabled for every block transfer individually. For the target, single target decoders can be set up to do data comparisons, while others do not. In any case, all devices can employ the compare unit when they receive data. Only bytes are compared that are not disabled by byte enable settings.

Compare Unit Usage

The compare unit compares the incoming data with the reference data held in the data memory in real time. It supplies a signal for the trace memory in the case of a comparison error (miscompare).

If a comparison error occurred, the testcard can use this output signal to trigger the trace memory to capture the data traffic before and after the occurrence of the error. Detailed information on the circumstances of the error, such as used bus command, bus address, etc. can then be obtained from the trace memory. See "Capturing Data In The Trace Memory" in the *Agilent E2940A PCI Analyzer User's Guide*.

Reference Data

The reference data, that the incoming data is compared to, is found in the data memory. The internal data space assigned to the master is protected from target access and can therefore not be overwritten by the target. However, the master can use data that has previously been recorded by the target.

Compare Unit Features

The compare unit features the following:

- It can cover the complete data memory.
- It can handle the full PCI bandwidth of 264 Mbytes/s over a long time (64 bits, 33 MHz, long bursts, zero waits).
- It is selectively addressable by programmable multiple target address ranges.
- It can process 64-bit accesses with zero waits within a transaction.

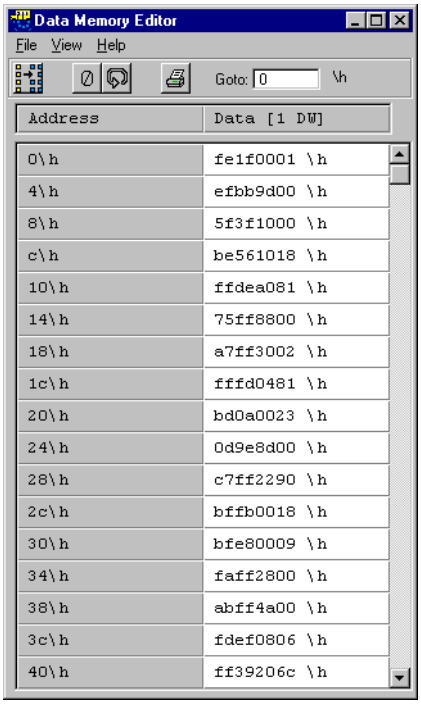
NOTE Transfers terminated by a master abort result in a data comparison error, too.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Data Memory Editor

To view and edit the contents of the data memory:

1 Open the Data Memory Editor window by selecting *Data Memory* from the *Exerciser* menu in the main window.



Features The data memory editor provides the following features:

- Setting the contents to zero.

  To set the complete data memory to zero, click the Zero button .

- Search address.

  The *Goto* textfield can be used to directly jump to a certain address in the data memory. Enter the address in hex format in the text field and press the Return key. Another way to locate a certain address is to use the scrollbar to the right.

- Modifying the contents.

  You can modify the contents of the data memory at a certain address by clicking into the respective data field (right column) and typing the new value. The changes are written to the testcard immediately after you press the Return key.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Reload the contents from the testcard.

  To reload the contents of the window from the testcard, click the Refresh button or select *Refresh Data* from the *View* menu.

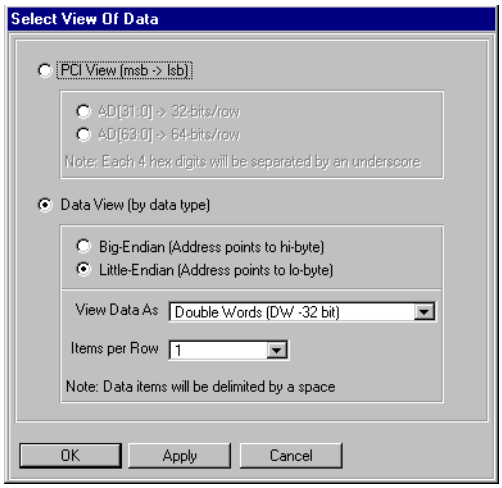
- File actions.

  You can also save the whole memory content into a memory file (extension .mem). Previously saved files can be loaded into the editor for examination or reuse. The functions for the file actions are found in the *File* menu.

Modifying the Data View

You can also modify the way the data is displayed in the data memory editor:

1 Click the Data View button or choose *Select View Of Data* from the *View* menu. This opens the Select View Of Data dialog box.



The two different views of the memory contents are:

- PCI View.

  The data memory contents are displayed as they are sent on the PCI bus. Thus, the data amount per row represents one bus cycle. Select AD[31:0] to display 32 bits/row for 32-bit buses. Or select AD[63:0] to display 64 bits/row for 64-bit buses.

  In this view always the most significant bit is displayed first in every row.

- Data View.

  This view displays the memory contents regarding the data type. The available options are:

  – The size of the data packages: bytes, words, dwords, or qwords.

  – The number of items displayed per row. The options here depend on the item size.

  – The byte weight within the item: little endian or big endian.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Generating Interrupts

Devices that do not have REQ# and GNT# lines connected to them cannot request to access the PCI bus as a master device. To request servicing from the system software they can generate interrupt signals.

PCI Interrupts

The PCI specification defines the following four interrupt signals:

- INTA#,
- INTB#,
- INTC#,
- INTD#.

Single-function devices with only one configuration space always have to use INTA#. Multi-function devices act as several independent devices on the PCI bus and have one configuration space each. They may use different interrupt signals as long as they meet the following preconditions:

- INTB# may only be used, if INTA# is covered by at least one function of the multi-function device.
- INTC# may only be used, if INTB# is covered by at least one function of the multi-function device.
- INTD# may only be used, if INTC# is covered by at least one function of the multi-function device.

Apart from these restrictions any combination of interrupts is permitted by the specification. The interrupts can also be shared by several devices.

The PCI specification permits a wide range of scenarios how the four PCI interrupts, the ISA master interrupt controller, the ISA slave interrupt controller, and optionally, a programmable interrupt router may be interconnected. See the PCI specification for details.

The interrupt features provided by the Agilent E2940A testcard are described in *"Interrupt Capabilities of the Testcard" on page 86*. *"Asserting and Deasserting Interrupts" on page 87* shows how to use these features.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupt Capabilities of the Testcard

The Agilent E2940A testcard can generate any of the PCI interrupts INTA#, INTB#, INTC#, and INTD#. This functionality is essential when developing interrupt drivers for PCI devices. Interrupts can be asserted and deasserted by means of the Graphical User Interface.

Interrupt Status Register

The current status of the interrupts—asserted or deasserted—is stored in the interrupt status register. This register is located in the private section of the configuration space header. It can be read, for example, by interrupt drivers to determine whether the testcard has generated an interrupt.

With the use of this register, any interrupt and any combination of interrupts can be generated by the testcard. Thus, you can simulate both single-function devices and multi-function devices.

Furthermore, the testcard can act as several devices asserting interrupts at the same time. It can also behave as a multi-function device that generates an illegal combination of interrupts which is not PCI specification compliant.

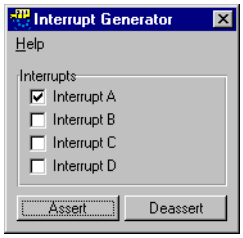
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Asserting and Deasserting Interrupts

By asserting and deasserting interrupts, you can examine the behavior of the system under test in response to the various interrupts and combinations of them.

To generate the interrupts, proceed as follows:

1 Select *Interrupt Generation* from the *Exerciser* menu in the main window.



2 Tick the interrupts that you want to assert or deassert.

3 Click on the *Assert* or *Deassert* button.

The referring interrupts will be asserted or deasserted immediately.

NOTE Asserting interrupts can cause the system under test to hang. If you are running the control software (GUI) on the system under test, unsaved data will be lost. Either run the GUI on a remote PC or save all your settings and test data before asserting interrupts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Testing Hot Swap Systems

The CompactPCI specification allows PCI cards to be inserted into and extracted from running systems. This mechanism is called “Hot Swapping”.

The Agilent E2940A CompactPCI testcard is capable of hot swapping. Even more, the testcard can emulate all types of hot-swap behavior that plug-in devices may have, thus allowing to test the hot-swap capabilities of CompactPCI systems.

To interactively monitor and debug the hardware and software connection performed by the hot-swap mechanism, the Agilent E2940A testcard’s GUI provides the Hot Swap Debug window. With the use of this window the contents of the hot-swap control and status register can be controlled as well as certain signal pins.

The testcard’s features for hot swap testing are described in *“Hot Swap Capabilities of the Testcard” on page 90*. *“Asserting and Deasserting Hot Swap Signals” on page 91* describes how to use these features.

For more information on hot swapping, please refer to the *CompactPCI Hot Swap Specification*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

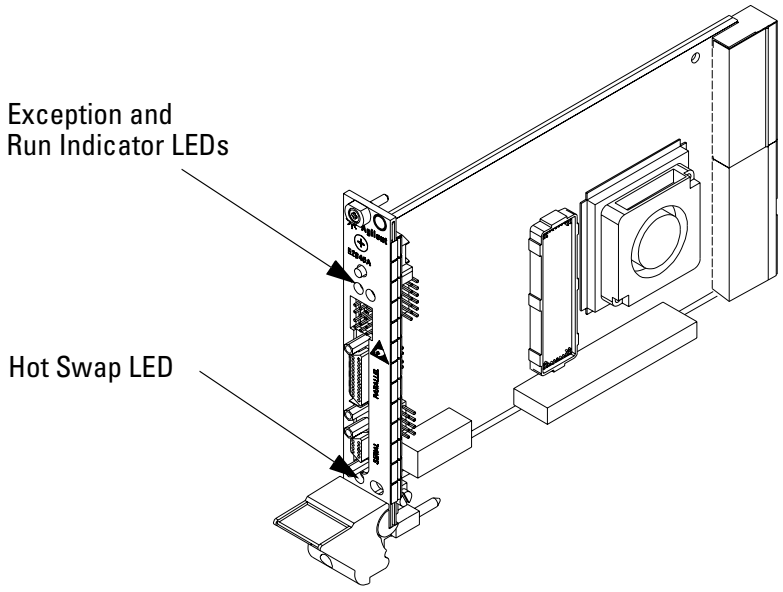
Hot Swap Capabilities of the Testcard

The Agilent E2940A CompactPCI testcard allows interactive debugging of the hardware and software connections for hot swapping from within the Graphical User Interface.

The Hot Swap Debug window displays status information and allows user interaction in the hot-swap mechanism via several signal pins.

Hot Swap LED

On the Agilent E2940A testcard a blue LED indicates the hardware and software connection status.



When inserting the testcard, the blue LED will light up as soon as the testcard is powered. After successful hardware and software layer connection, the LED will turn off.

When opening the ejector handle to extract the testcard from the CompactPCI bus, the blue LED indicates that the software layer has disconnected and the hardware layer is ready for disconnection. Only when the LED is lit, the board may be extracted.

For information on the hot swap modes, please refer to the *"Agilent E2940A PCI Analyzer User's Guide"*.

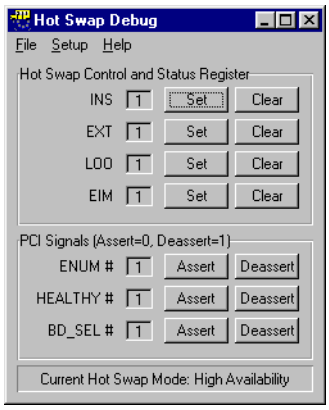
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Asserting and Deasserting Hot Swap Signals

The Agilent E2940A testcard provides the Hot Swap Debug window to view and control the status of the hot-swap mechanism.

Access the hot-swap status register and the signal pins as described below:

1 Select *Hot Swap Debug* from the *Exerciser* menu in the main window to open the Hot Swap Debug window.



The currently selected hot-swap mode of the testcard is displayed in the status bar at the bottom of the window.

2 View the current contents of the *Hot Swap Control and Status Register*.

The individual bits in this register are used to control the connection of the software layer:

– *INS* (Inserted). This bit is set to 1 by the system upon card insertion. It will be reset by the system after the software connection has been established.

– *EXT* (Extract). This bit is set to 1 by the system when the red handle has been pulled out. It will be reset by the system before the blue LED is turned on to indicate the release of the software connection.

– *LOO* (LED on/off). This bit controls the blue hot swap LED.

– *EIM* (ENUM# Interrupt Mask). This bit is set by the system to mask the assertion of the ENUM# pin.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3 Use the *Set* and *Clear* buttons to set or reset the individual bits as required.

4 View the current contents of the signal pins in the *PCI Signals* group. The individual pins are used for control of the hardware connection:

- *BD_SEL#* (Board Select). A shorter pin that signals to the system that the card is completely and correctly inserted in the backplane system. If *BD_SEL#* is deasserted, the card is not fully powered.
- *ENUM#* (Enumeration). In order to initiate the enumeration process, this signal informs the system that the card has just been inserted or is about to be extracted.
- *HEALTHY#*. This signal indicates whether the card has been powered up and booted properly without any faults or errors being detected.

5 Use the *Assert* and *Deassert* buttons to set or reset the individual signals as required.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Command Line Interface

The Command Line Interface (CLI) provides a very convenient way to directly call the C functions that control the Agilent E2940A testcard. You simply enter the commands with their parameters into the CLI window, and by that you can communicate interactively with the testcard.

Basically, you can call any of the testcard's C functions via the CLI. However, for full flexibility and control of the testcard's features, there is also an Application Programming Interface (C-API) available as option #320 for your Agilent E2940A testcard.

There is one CLI command for every C function, but within the GUI only those commands are supported that are part of your licensed software components. For example, if you do not have the Performance Optimizer option enabled, you cannot call its C functions via the CLI.

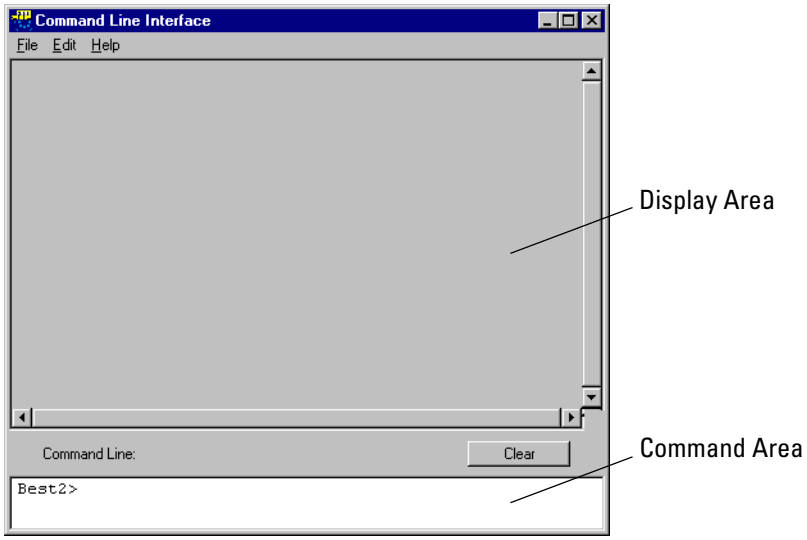
A description to all CLI commands and their CLI abbreviations is found along with the respective C-API function descriptions in the *Agilent E2940A C-API/PPR Reference*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Starting the CLI

To open the Command Line Interface window:

1 Select *Command Line* from the *Windows* menu in the main window.



The Command Line Interface window is divided into two areas: The display area and the command area. The CLI commands (or their abbreviations) are entered in the command area and executed with the return key. The commands and the system responses are then listed in the display area.

TIP Use the cursor-up and cursor-down keys in the command area to scroll through the last 50 entered command lines.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Basic CLI Command Syntax

Because calls to C functions in C programs do not look very comprehensive, a more convenient syntax is used for the CLI commands and the CLI abbreviations. The software interprets the command lines and calls the respective functions with the specified parameters, if applicable.

The syntax for the three types of CLI commands is:

- Commands without parameters.

  Commands that do not require any parameters are entered as they are. The following example simply checks the connection to the testcard. As a result the LEDs on the testcard flash five times.

  Example: `BestPing`

  Or the abbreviation: `ping`

- Commands that pass parameters.

  Use the “equals” mark to pass a value to a command’s parameter. The example shows how to lock the onboard resource “CPU port”.

  Example: `BestResourceLock resource=cpuport`

  Or the abbreviation: `reslock res=cpuport`

- Commands that request parameters.

  The syntax for commands that request parameters is the same as when passing parameters. For example, check whether the command above did lock the CPU port successfully.

  Example: `BestResourceIsLocked resource=cpuport`

  Or the abbreviation: `resislock res=cpuport`

  Assuming that a connection is established to the testcard and no error occurred, the system will report in the display area that the CPU is locked.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using CLI Scripts

A CLI script is a sequence of CLI commands and can be stored in a text file. Such script files can later be loaded and executed in the Command Line Interface window. This reduces typing effort, when you want to use the same sequence of CLI command lines multiple times.

Generating Scripts

To generate and save a CLI script, you can

- use any text editor. Enter the sequence of CLI commands in the text editor and save it as a text file.
- copy and paste command lines from either CLI area into a text editor and save the file.
- use command logging. While you are performing CLI commands, you can log this input into a script file. To start the logging, select *Start Logging* from the *File* menu, and specify a file name in the file dialog box. Now, all the command lines that you type are simultaneously written to the script file until you select *Stop Logging* from the *File* menu.

Executing CLI Scripts

There are two different ways to run a CLI script from a file:

- Select *Run Script* from the *File* menu of the CLI window and pick the script file from the file dialog box.
- In the CLI command area, use the command `do scriptfile`, where `scriptfile` is the absolute or relative name and path of the file.

NOTE Command logging only stores the sequence of command lines. If you want to store the complete contents of the display area for later information or comparison with other reports, select *Save As* from the *File* menu.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bus Transaction Language Reference

The different commands of the Bus Transaction Language (BTL) are used in the various transaction and attribute scripts to specify the behavior of the PCI testcard's master and target. Descriptions to these commands are found in the section *"BTL Commands" on page 97*.

Every command within these scripts uses a set of parameters for defining the behavior more closely. Some of these parameters are mandatory, others are optional. The definitions of all parameters are located in *"BTL Command Parameters" on page 106*.

The *"BTL Syntax Diagrams" on page 110* will help you to correctly set up commands, parameters, identifiers, etc.

BTL Commands

The available bus commands are:

- *"m_block()" on page 98*
- *"m_xact()" on page 99*
- *"m_data()" on page 100*
- *"m_last()" on page 101*
- *"m_xact64()" on page 102*
- *"m_data64()" on page 103*
- *"m_last64()" on page 103*
- *"m_attr()" on page 104*
- *"t_attr()" on page 105*

The parameters used by the commands are described in *"BTL Command Parameters" on page 106*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_block()

Description This transaction command transfers a complete data block. The data is sent from or to an internal data resource.

The protocol behavior during the transaction is set up with the Master Attribute Editor. The optional parameter “attrpage” refers to the protocol attribute page.

Required Parameters busaddr, cmd, intaddr, nofdwords

Optional Parameters attrpage, busaddr_hi, busdac, byten, byten_var, compflag, compoffs, condstart, contattr

Example The following example sets up a block data write transaction for 1000 dwords, starting at the internal address 100\h, using the protocol behavior as specified in the attribute page “MyPage_1”:

```
{
...
   m_block(cmd=mem_write,
      busaddr=b9000\h,
      intaddr=100\h,
      attrpage=Mypage_1,
      nofdwords=1000);
...
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_xact()

Description This command starts a transaction by driving an address phase onto the bus, using the specified bus address and command. The transaction will be aborted if no target responds within 5 clock cycles.

The data phases of the transaction must be specified by the commands *"m_data()" on page 100* and/or *"m_last()" on page 101*.

Required Parameters `busaddr`, `cmd`

Optional Parameters

- **Block transfer parameters** are valid during the complete transfer:

  `busdac`, `byten`, `byten_var`, `compflag`, `compoffs`, `condstart`, `contattr`, `intaddr`

- **Address and data phase parameters** (can be altered a multiple number of times during a transfer):
  - Address phase parameters:

    `aperr`, `awrpar`, `dacperr`, `dacwrpar`, `delay`, `lock`, `relreq`, `resumedelay`, `tryback`

  - Data phase parameters:

    `dperr`, `dserr`, `dwrpar`, `waits`, `marker`, `repeat`

Examples The following two examples show how to specify single transfers and a burst using the `m_xact()` command. They write the word "BEST" and "AGILENTBVS" on the screen in DOS mode.

```
{ /* two single transactions */
   m_xact(busaddr=b8004\h, cmd = mem_write);
   m_last(data=86458642\h); //BE
   m_xact(busaddr=b8008\h, cmd = mem_write);
   m_last(data=86008600\h | 'T'<<16 | 'S'); //ST
}
```

```
{ /* a burst * /
   m_xact(busaddr=b8004\h, cmd = mem_write);
   m_data(data=86008600\h | 'G'<<16 | 'A'); //AG
   m_data(data=864C8649\h); //IL
   m_data(data=86008600\h | 'N'<<16 | 'E'); //EN
   m_data(data=86008600\h | 'B'<<16 | 'T'); //TB
   m_last(data=86008600\h | 'S'<<16 | 'V'); //VS
}
```

NOTE If data phase attributes are specified with an `m_xact()` command, they will be used as the default values for all data phases of this transaction. They can be overwritten in a later data phase. Such changes, however, do only apply for this particular data phase and not the following ones.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_data()

Description This command generates a data phase that is not the last or the only data phase of a burst.

For the data phase, the attributes are used as specified by the parameters. Parameters that are not specified with this command are used as specified by the previous `m_xact()` command, or, if they are not specified there either, their default values are used.

You can also specify address phase attributes here. If the burst has been interrupted at this point and therefore a new transaction has to be started—beginning with the data phase specified by `m_data()`—it first generates an address phase with these address phase attributes.

If `m_data()` follows after an `m_last()` command, a new transaction will be started with `m_data()`, using the same bus command and the address that follows the address of the previous transfer.

Parameters

- **Address phase parameters**:

  `aperr`, `awrpar`, `dacperr`, `delay`, `lock`, `relreq`, `resumedelay`, `tryback`

- **Data phase parameters**:

  `data`, `dperr`, `drelreq`, `dserr`, `dwrpar`, `waits`, `marker`, `repeat`

Example Refer to the examples given for *"m_xact()" on page 99*. The `m_data()` command is used in combination with `m_xact()` only.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_last()

Description This command must be used for the last or the only data phase of a burst (instead of `m_data()`). It completes the transaction.

For the data phase, the attributes are used as specified by the parameters. Parameters that are not specified with this command are used as specified by the previous `m_xact()` command, or, if they are not specified there either, their default values are used.

You can also specify address phase attributes here. If the burst has been interrupted at this point and therefore a new transaction has to be started—beginning with the data phase specified by `m_last()`—it first generates an address phase with these address phase attributes.

Parameters

- **Address phase parameters**:

  `aperr`, `awrpar`, `dacperr`, `delay`, `lock`, `relreq`, `resumedelay`, `tryback`

- **Data phase parameters**:

  `data`, `dperr`, `drelreq`, `dserr`, `dwrpar`, `waits`, `marker`, `repeat`

Example Refer to the examples given for *"m_xact()" on page 99*. The `m_last()` command is used in combination with `m_xact()` only.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_xact64()

Description This is the same command as *"m_xact()" on page 99*.

The data phases of the transaction must be specified by the commands *"m_data64()" on page 103* and/or *"m_last64()" on page 103*.

Required Parameters `busaddr, cmd`

Optional Parameters

- **Block Transfer parameters** are valid during the complete transfer:

  `busaddr_hi, busdac, byten, byten_var, compflag, compoffs, condstart, contattr, intaddr`

- **Address and data phase parameters** can be altered a multiple number of times during a transfer:
  - Address phase parameters:

    `aperr, awrpar, awrpar64, dacperr, dacwrpar, dacwrpar64, delay, lock, relreq, req64, resumedelay, tryback`

  - Data phase parameters:

    `dperr, dserr, dwrpar, dwrpar64, waits, marker, repeat`

Examples The examples show how to specify two single 64-bit transfers and a 64-bit burst using the `m_xact64()` command.

```
{ /* two single 64-bit transactions */
   m_xact64(busaddr=b8004\h, cmd = mem_write);
   m_last64(hi_data=75315982\h, data=86458642\h);
   m_xact64(busaddr=b8008\h, cmd = mem_write)
   m_last64(hi_data=75315982\h, data=86008600\h);
}
```

```
{ /* a 64-bit burst * /
   m_xact64(busaddr=b8004\h, cmd = mem_write);
   m_data64(hi_data=75315982\h, data=88600860\h);
   m_data64(hi_data=75315982\h, data=86458642\h);
   m_last64(hi_data=75315982\h, data=88600860\h);
}
```

NOTE If data phase attributes are specified with an `m_xact64()` command, they will be used as the default values for all data phases of this transaction. They can be overwritten in a later data phase. Such changes, however, do only apply for this particular data phase and not the following ones.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_data64()

Description This is the same command as *"m_data()" on page 100*, but requires a 64-bit transfer.

Parameters

- **Address phase parameters**:

  `aperr, awrpar, awrpar64, dacperr, delay, lock, relreq, req64, resumedelay, tryback`

- **Data phase parameters**:

  `data, dperr, drelreq, dserr, dwrpar, dwrpar64, hi_data, waits, marker, repeat`

Example Refer to the example given for *"m_xact64()" on page 102*. The `m_data64()` command is used in combination with `m_xact64()` only.

m_last64()

Description This is the same command as *"m_last()" on page 101*, but requires a 64-bit transfer.

Parameters

- **Address phase parameters**:

  `aperr, awrpar, awrpar64, dacperr, delay, lock, relreq, req64, resumedelay, tryback`

- **Data phase parameters**:

  `data, dperr, drelreq, dserr, dwrpar, dwrpar64, hi_data, waits, marker, repeat`

Example Refer to the example given for *"m_xact64()" on page 102*. The `m_last64()` command is used in combination with `m_xact64()` only.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

m_attr()

Description

This command defines the protocol behavior during a master block transfer. When setting up the block transfer with the command `m_block()`, an attribute page can be assigned to it. Attribute pages are specified with unique names and contain sequences of `m_attr()` commands that are worked through successively when the page is called.

If the number of required data phases for the block transfer exceeds the number of programmed attribute lines, the attribute page is restarted with the first attribute line of the page.

Parameters

- **Address phase parameters**:

  `aperr`, `awrpar`, `awp64`, `dacperr`, `dacwrpar`, `dacwrpar64`, `delay`, `lock`, `relreq`, `req64`, `resumedelay`, `tryback`

- **Data phase parameters**:

  `dperr`, `drelreq`, `dserr`, `dwrpar`, `last`, `waits`, `marker`, `repeat`

Example

The following example sets up a master attribute page “Mypage_1” with master protocol behavior specified for two data phases. In the first data phase a 64-bit request is issued and 5 wait cycles are inserted. The second data phase is specified as the last data phase (that means the maximum burstlength of a transaction calling this page is two), including 2 wait cycles:

```
M_ATTRIBUTES Mypage_1=
{
   m_attr(req64=1, waits=5);
   m_attr(waits=2, last=1);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_attr()

Description This command sets the protocol behavior during target transactions.

If the number of data phases exceeds the number of programmed attribute lines, the attribute page is restarted with the first attribute line.

You can set the attribute page to automatically be restarted with each new transaction. Otherwise it continues where it stopped before. Refer to *"Setting the Reset Mode for Target Attributes" on page 77*.

Parameters

- **Address phase parameters**:

  `aperr`, `dacperr`, `ack64`

- **Data phase parameters**:

  `dperr`, `dserr`, `waits`, `term`, `wrpar`, `wrpar64`, `marker`, `repeat`

Example The following example illustrates a target programmed to respond to mixed master read and write commands.

```
{
   /* burst of 2 transfers */
   t_attr(dperr);
   t_attr(wrpar);

   /* 64-bit request with 10 wait cycles */
   t_attr(req64=1, waits=10);

   /* data write, target terminates with retry after 3 wait
cycles*/
   t_attr(waits=3, term=retry);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BTL Command Parameters

The following tables list all parameters available for the BTL bus commands. The following types of parameters are listed:

- *"Block Transfer Parameters" on page 107*
- *"Address Phase Attributes" on page 108*
- *"Data Phase Attributes" on page 109*

Each table contains:

- the parameter and its abbreviation. For convenience, you can use the abbreviation instead of typing the complete parameter name.
- a short description,
- the commands that use the parameter either optionally or as a required parameter,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Block Transfer Parameters

The following parameters are used when setting up master block transfers. Descriptions on the parameters in the table can be found in *"Transaction Properties" on page 36*.

More details and value ranges can be found under "b_blkproptype" in the *Agilent E2940A C-API/PPR Reference*.

Parameter	Abbrev.	Short Description	Used by		
			m_block()	m_xact()	m_xact64()
attrpage	apage	Points to a page with master protocol attributes for each phase of the transfer.	o		
busaddr	bad	Bus address in the address phase (32-bit address).	r	r	r
busaddr_hi	badhi	Upper 32 bits of a 64-bit bus address. Parameter "busdac" must be enabled.	o		o
buscmd	cmd	Bus command.	r	r	r
busdac	dac	Enables 64-bit address width. Used with parameter "busaddr_hi".	o	o	o
byten	ben	Byte enable settings during data transfers.	o	o	o
byten_var	benv	Variable (1) or fixed (0) byte enables. (Use the CLI to edit the byte enable memory.)	o	o	o
compflag	cflag	Determines whether data comparison is performed (1) or not (0).	o	o	o
compoffs	coffs	Internal address pointing to reference data for comparison.	o	o	o
condstart	cond	Delays the block run until a start condition occurred.	o	o	o
contattr	contattr	Determines whether the attribute page is continued (1) or restarted (0) after a block transfer.	o	o	o
intaddr	iad	Internal address pointing to data for transfer (write) or received data (read).	r	o	o
nofdwords	nod	Number of dwords to transfer in a block.	r		
			r = required o = optional		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Address Phase Attributes

The following parameters are used to control the protocol attributes during address phases. Descriptions of the attributes in this table can be found in *"Controlling Master Attributes" on page 40* or *"Controlling Target Attributes" on page 71*, depending on the type of device.

More details and value ranges can be found under "b_mattrproptype" for master attributes and "b_tattrproptype" for target attributes in the *Agilent E2940A C-API/PPR Reference*.

Parameter	Abbrev.	Short Description	Used by					
			m_xact()	m_data() m_last()	m_xact64()	m_data64() m_last64()	m_attr()	t_attr()
aperr	aperr	Asserts a system error (a parity error during address phase is a system error).	×	×	×	×	×	×
awrpar	awp	Sets a wrong parity (PAR).	×	×	×	×	×	
awrpar64	awp64	Sets a wrong parity (PAR64).			×	×	×	
dacperr	dacperr	Asserts a system error for the second cycle of a 64-bit address phase.	×	×	×	×	×	×
dacwrpar	dacwp	Signals a wrong parity (PAR) in the second cycle of a dual address cycle.	×		×		×	
dacwrpar64	dacwp64	Signals a wrong parity (PAR64) in the second cycle of a dual address cycle.			×		×	
delay	delay	Idle time before the master turns to the next transaction.	×	×	×	×	×	
lock	lock	Locks the target for exclusive access.	×	×	×	×	×	
relreq	rreq	Controls the REQ# release behavior.	×	×	×	×	×	
req64	req64	Issues a 64-bit transfer request.			×	×	×	
resumedelay	resume	Resume after a target termination.	×	×	×	×	×	
tryback	tryback	Master tries fast back-to-back.	×	×	×	×	×	
ack64	ack64	Target acknowledges 64-bit request.						×

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data Phase Attributes

The following parameters are command parameters to control the protocol attributes during data phases. Descriptions of the attributes in this table can be found in *"Controlling Master Attributes" on page 40* or *"Controlling Target Attributes" on page 71*, depending on the type of device.

More details and value ranges can be found under "b_mattrproptype" for master attributes and "b_tattrproptype" for target attributes in the *Agilent E2940A C-API/PPR Reference*.

Parameter	Abbrev.	Short Description	Used by					
			m_xact()	m_data() m_last()	m_xact64()	m_data64() m_last64()	m_attr()	t_attr()
data	data	Specifies the data value to transfer.		×		×		
dperr	dperr	Asserts a parity error.	×	×	×	×	×	×
drelreq	drreq	Controls the REQ# release behavior.		×		×	×	
dserr	dserr	Asserts a system error.	×	×	×	×	×	×
dwrpar	dwp	Sets a wrong parity after a write transfer (PAR).	×	×	×	×	×	
dwrpar64	dwp64	Sets a wrong parity after a write transfer (PAR64).			×	×	×	
hi_data	hi_data	Specifies the upper 32 bits of a 64-bit data value.				×		
last	last	Indicates the phase as the last of a burst.					×	
waits	w	Inserts the specified number of wait cycles.	×	×	×	×	×	×
term	term	Terminates the current transaction.						×
wrpar	wp	Signals a wrong parity (PAR) during read transfers.						×
wrpar64	wp64	Signals a wrong parity (PAR64) during read transfers.						×
marker	marker	Sets a marker to an integer value that can be used to trigger the PCI Analyzer	×	×	×	×	×	×
repeat	repeat	Number of times a particular attribute line is repeated	×	×	×	×	×	×

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BTL Syntax Diagrams

This section contains all the syntax diagrams for the bus transaction language BTL:

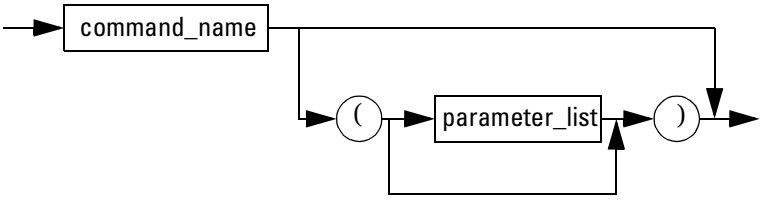
- *"Bus Commands" on page 110*
- *"Parameters" on page 112*
- *"Identifiers" on page 113*

Bus Commands

Find the following syntax diagrams:

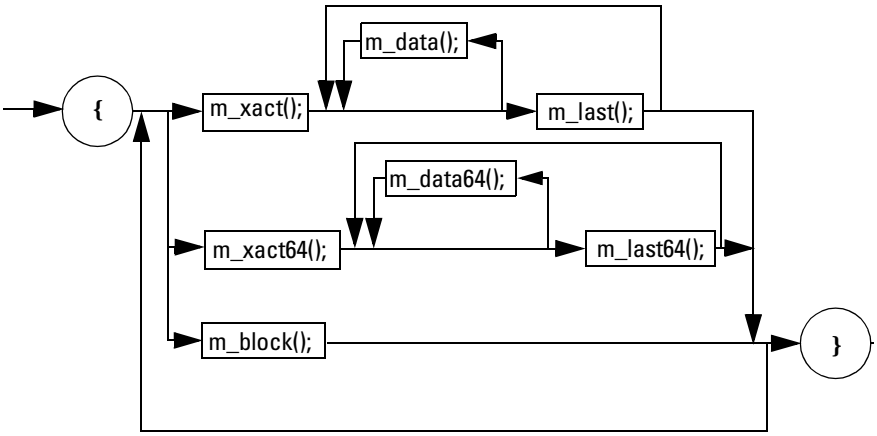
- *"Syntax Diagram: Bus Commands" on page 110*
- *"Syntax Diagram: Master Transaction Commands" on page 111*
- *"Syntax Diagram: Master Attribute Command" on page 111*
- *"Syntax Diagram: Target Attribute Command" on page 111*
- *"Syntax Diagram: Bus Command Parameters" on page 112*

Syntax Diagram: Bus Commands

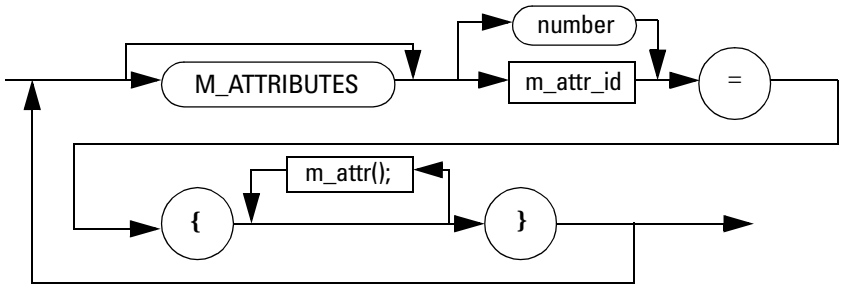


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

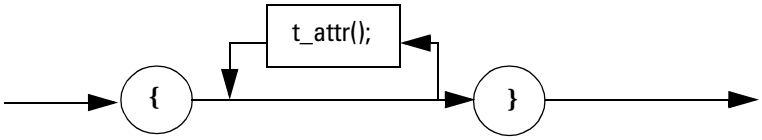
Syntax Diagram: Master Transaction Commands



Syntax Diagram: Master Attribute Command



Syntax Diagram: Target Attribute Command



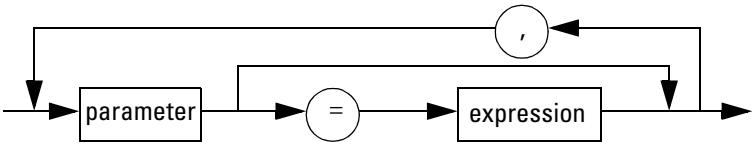
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters

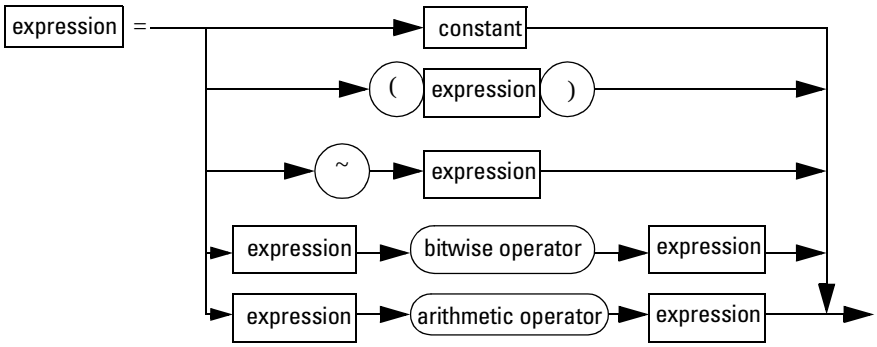
Find the following syntax diagrams:

- *"Syntax Diagram: Bus Command Parameters" on page 112*
- *"Syntax Diagram: Expression" on page 112*
- *"Syntax Diagram: Constant" on page 113*

Syntax Diagram: Bus Command Parameters



Syntax Diagram: Expression



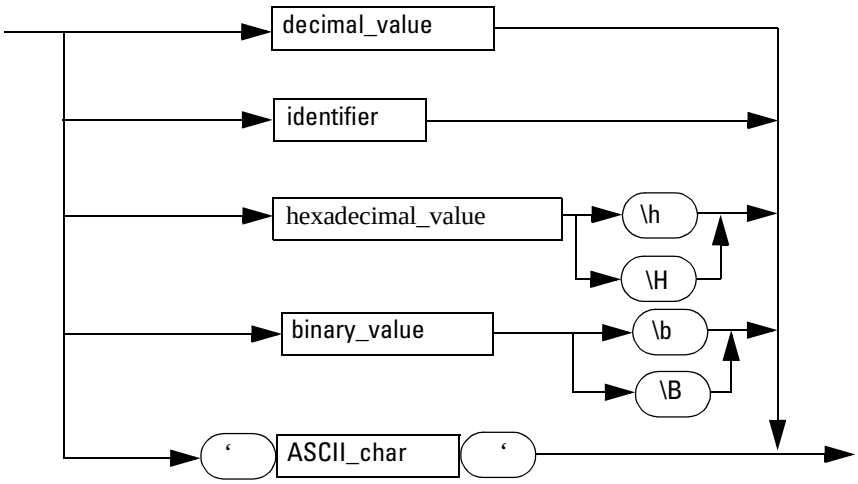
Example

Bitwise Operator	Meaning	Arithmetic Operator	Meaning
~	NOT	!	NOT
&	AND	&&	AND
|	OR	||	OR
^	XOR	+	Plus
<<	Shift left	–	Minus
>>	Shift right	*	Multiply
		/	Divide

```
data = 01101111\h | 'B'<<16 | 'E'
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Syntax Diagram: Constant



Examples

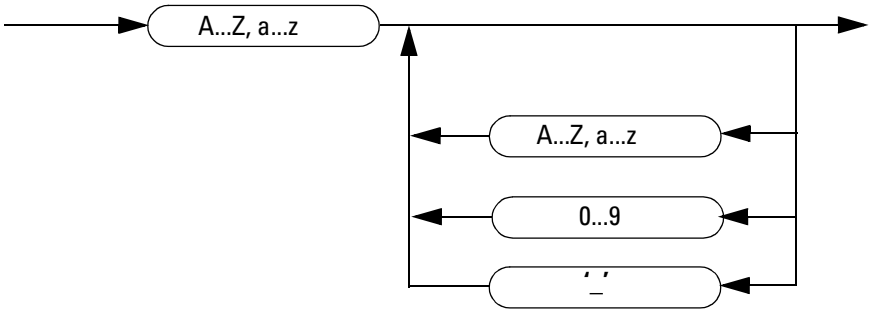
- decimal value: `data=1024`
- identifier: `retry`
- hexadecimal value: `busaddr=B8000\h`
- Binary value: `data=01101111\b, data=01101111\B`
- ASCII character: `data=86008600\h`

Identifiers

Find the following syntax diagrams:

- *“Syntax Diagram: Identifier” on page 113*

Syntax Diagram: Identifier



Examples

- `Mypage_1`
- `attr1`
- `PCI_attributes`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Index

#

64-Bit Request (req64) 43

A

Address 108
Address Phase Attributes 108
  Master Device 42
  Target Device 72
Address Phases 33
Asserting and Deasserting
  Hot Swap Signals 91
  Interrupts 87
Attribute Page 41
  (attrpage) 38
Attributes in BTL Commands 41

B

Base Address 60
  Registers 56
Basic CLI Command Syntax 95
BIOS Settings, Overwriting 68
Block transactions with m_block 35
Block Transfer Parameters 107
BTL Commands 97
  Attributes 41
  Parameters 106
BTL Script Syntax 76
Burst Length 46
Bus Address (busaddr, busaddr_hi) 36
Bus Command 110
  (buscmd) 36
  Parameters, Syntax Diagram 112
  Syntax Diagram 110
Bus Transaction Language
  Master Attributes 47
  Master Transactions 39
  Reference 97
Byte Enable Control (byten, byten_var) 38

C

Command Line Interface 93
  Example 28
  Scripts 96
  Starting 94
  Window 94
Commands
  m_attr() 104
  m_block() 98
  m_data() 100
  m_data64() 103
  m_last() 101
  m_last64() 103
  m_xact() 99
  m_xact64() 102
  t_attr() 105
Compare Flag (compflag, compoffs) 37
Conditional Start
  (condstart) 37
  Master Transactions 21
Configuration Decoder 59
Configuration Space 54
  Window 67
Configuration Space Header 55
  Modify 65
Configurations 12
Constant (Syntax Diagram) 113
Continue With Attributes (contattr) 37
Control Attributes
  Master 46
  Target 75
Controlling
  Master Attributes 40
  Target Attributes 71
CPU Port 62
Custom Target Behavior 24

D

Data (data, hi_data) 44
Data Alignment 80
Data Compare Unit 82
Data Memory
  Editor 83
  Organization 80
  Use 79
Data Phase Attributes
  Master Device 44
  Overview 109
  Target Device 72
Data Phases 34
Data Resources 61
Data resources
  Configuration space 62
  CPU port 62
  Data memory 61
  Expansion ROM 62
  Static I/O 62
Databases (Standard and Power Up) 69
Delay, Exerciser Idle (delay) 43
Dual Address Cycle, DAC (busdac) 36

E

Example Scenarios 17
Exerciser Status Bar 51
Expansion ROM Decoders 59
Expression (Syntax Diagram) 112

G

Generating Interrupts 85

H

Hot Swap
  (De)asserting Signals 91
  Capabilities of the Testcard 90
  LED 90
  Systems 89
Hot Swap Debug Window 91

I

Identifier (Syntax Diagram) 113
Implementing
  Master Attribute Scripts 47
  Master Transaction Scripts 39
  Target Attribute Scripts 75
Initializing the Target 78
Interfaces, User Interface 10
Internal Address (intaddr) 38
Interrupt
  (De)asserting 87
  Capabilities of the Testcard 86
  Generating 85
  Generation (dialog box) 87
  PCI 85
  Status Register 86

L

Loading the Setup Files 20
  for Custom Target Behavior 24
Lock (lock) 43

M

m_attr() 104
m_block() 98
m_data() 100
m_data64() 103
m_last() 101
m_last64() 103
m_xact() 99
m_xact64() 102

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Marker (marker)
Master 45
Target 74
Master Attribute Command (Syntax Diagram) 111
Master Attributes 31
Controlling 40
Scripts 47
Specification 40
Master Conditional Start (dialog box) 49
Master Device
Starting 51
Stopping 52
Master Transaction Commands (Syntax Diagram) 111
Master Transactions
Conditional Start 21
Dialog box 21
Overview 33
Programming 32
Scripts 39
Match Indicator 67
Mode (dialog box)
BIOS settings 68
Target reset 77
Modifying the Configuration Space Header 65

N

Number of Dwords (nofdwords) 37

O

Optimization of PCI Design 8
Organization of the Data Memory 80
Overview
Master Transactions 33
PCI Exerciser 7
Overwriting BIOS Settings 68

P

Parameters
ack64 108
aperr 108
attrpage 107
awrpar 108
awrpar64 108
busaddr 107
busaddr_hi 107
buscmd 107
busdac 107
byten 107
byten_var 107
compflag 107
compoffs 107
condstart 107
contattr 107
dacperr 108
dacwrpar 108
dacwrpar64 108
data 109
delay 108
dperr 109
drelreq 109
dserr 109
dwrpar 109
dwrpar64 109
hi_data 109
intaddr 107
last 109
lock 108
nofdwords 107
relreq 108
req64 108
resumedelay 108
term 109
tryback 108
waits 109
wrpar 109
wrpar64 109
Parity Error Signaling (dperr)
Master 45
Target 74
Pattern Editor (Window) 50
PCI Exerciser
as a Master Device 31
as a Target Device 53
Configurations 12
Fields of Application 8
Overview 7
Sample Sesison 17
User Interface 10
PCI Interrupts 85
Power Up Database 69
Preparing for the Examples 18
Preparing Test Execution 48
Programming
Decoders 63
Master Transactions 32
Transactions for Master Device 31
Properties
Containing Pointers 37
Containing Values 36

R

Release Request
(drelreq) 45
(relreq) 44
Repeat (repeat) 46
Resume Delay (resumedelay 44
Run Options 48
Running a Sample PCI Exerciser Session 17
Running the Master 48

S

Sample PCI Exerciser Session 17
Select View Of Data (dialog box) 84
Setting the Reset Mode for Target Attributes 77
Setting Up
a Master Transaction 20
PCI Exerciser Tests 14
Run Options 48
Target Decoders 62
Single transactions 34
Specification of Master Attributes 40
Specifying a Conditional Start 21
Specifying the Target Protocol Behavior 26
Standard and Power Up Databases 69
Standard Decoders 59
Start Condition (Master Device) 49
Starting the Master 51
Static I/O 62
Stopping the Master 52
Syntax Diagram
Bus Command Parameters 112
Bus Commands 110
Constant 113
Expression 112
Identifier 113
Master Attribute Command 111
Master Transaction Commands 111
Target Attribute Command 111
System Error Signaling (aperr, dacperr) 44
System Error Signaling (dserr)
Master 45
Target 74

T

t_attr() 105
Target Attribute Command (Syntax Diagram) 111
Target Attribute Editor 76
Target Attribute Memory 71
Target Attributes
(Window) 26
Controlling 71
Reset Mode 77
Scripts 75
Target Decode (Window) 25
Target Decoders 54
Configuration Decoder 59
Expansion ROM Decoders 59
Programming 63
Properties 57
Setup 62
Standard Decoders 59
Target Device, Initializing 78
Target Latencies 73
Termination (term) 74
Testcard
Configuration (dialog box) 19
Hot Swap Capabilities 90
Interrupt Capabilities 86

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Testing Hot Swap Systems 89
Transactions
  Definition 14
  Properties 36
  Types 34
Try Fast-Back-to-Back (tryback) 42

U

User Interface 10
Using CLI Scripts 96
Using the Command Line Interface 93
Using the Data Memory 79
Using the Data Memory Editor 83

V

Validation of PCI design 8
VGA Frame Buffer Memory 27

W

Waits (waits)
  Master 44
  Target 73
Waveform Viewer 22
Wrong Parity Calculation
  Master (awrpar, awrpar64, dacwrpar, dacwrpar64) 43
  Master (dwrpar, dwrpar64) 45
  Target (wrpar, wrpar64) 74

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Publication Number: 5988-4881EN



Agilent Technologies

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP