



PROTOCOL SOLUTIONS GROUP
3385 SCOTT BLVD
SANTA CLARA, CA 95054

Conquest

USB 2.0/1.x

Analyzer/Exerciser/DC Verification

User Manual

Version 7.41



For Software Version 7.40

January 2009

Trademarks and Servicemarks

LeCroy and Conquest are trademarks of LeCroy Corporation.

Universal Serial Bus and On-The-Go are registered trademarks of USB-IF.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Intel and Pentium are registered trademarks of Intel Corporation.

AMD Duron and AMD Athlon are trademarks of Advanced Micro Devices Inc.

All other trademarks and registered trademarks are property of their respective owners.

Disclaimer

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL INFORMATION, EXAMPLES AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE REPRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS ARE FULLY RESPONSIBLE FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN INFORMATION THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT LeCroy FOR A COPY.

Copyright

Copyright © 2009, LeCroy Corporation; All rights reserved.

This document may be printed and reproduced without additional permission, but all copies should contain this copyright notice.

WEEE Program



This electronic product is subject to disposal and recycling regulations that vary by country and region. Many countries prohibit the disposal of waste electronic equipment in standard waste receptacles. For more information about proper disposal and recycling of your Catalyst product, please visit www.getcatalyst.com/recycle.

Table of Contents

Conquest Versions	1
Conquest M2 versus Conquest	1
Common Software	1
Conquest M2 (SBAE30) and Conquest Supported Features	2
Introduction	3
Overview	3
Analyzing USB Designs With the Analyzer	4
Conquest M2 Options	5
Host Exerciser	5
Device Emulation	5
OTG Exerciser	5
DC Compliance measurement	5
Timing Analyzer	5
Conquest M2 Interface	6
Status LED Function Description	6
Manual Trigger	6
External Signals (Ext. Out, Ext. Trig In, and Ext. Clk)	6
Ports	7
Power In	7
Analyzer/Exerciser Connection Identification	8
Receiving the Analyzer	9
Conquest M2	9
Conquest	9
Unpacking the Analyzer	9
Installing the Analyzer	10
Hardware Setup	10
Exerciser Configuration Connections	11
OTG Analysis Connections	11
OTG Exerciser as DRD A-Device Connections	12
OTG Exerciser as DRD or Peripheral Only B-Device Connections	12

Contents

Device Emulation Connections	13
Conquest Analyzer Connections	14
Conquest Exerciser Connections	14
Conquest M2 External I/O Connector Pin Assignment	15
Conquest External I/O Connector Pin Assignment	15
Software Installation	16
Manual USB Driver Installation	17
Updating the Conquest USB Driver Manually	17
Directory Structure on Windows XP and Vista	18
Windows XP	18
Windows Vista	19
Launching the USB Analyzer	20
Connecting via Ethernet	21
Connecting to a Network	22
Connecting via Hub, Switch or Similar device	23
Operating in Simulation Mode	23
Authorization	24
Example Projects	25
Project File Types	26
Run an Example Project	28
Default Capture	29
Creating Projects	30
Capture and Timing Analysis Projects	30
Performance Analysis Projects	31
Device Emulation & OTG Advanced Exerciser	31
Current and Voltage Measurements	31
Protocol Analysis	37
Easy Mode (Pre-Defined Setups)	37
Easy Data Capture	38
Trigger on Data	40
Data Capture Options	47
Custom Transaction	49
Exercise and Capture (Optional)	50

Contents

Programming the Exerciser in Easy Mode	51
Settings	54
Auxiliary Port	57
Advanced Mode (User-Defined)	59
Capture Data Project	59
Defining Packets	60
Packet Payload Data	62
The Sequencer	64
Sequencer Operation Overview	64
Programming the Sequencer	65
Protocol Errors	70
Setting a Protocol Error Mask	70
Host Exerciser (Optional)	71
Programming the Exerciser In Advanced Mode	71
Programming with Transfers	72
Specifying a New Class	77
Specifying Data for Transfer	78
Reduced Bit Width	83
Exerciser Programming Shortcuts	84
Creating an Exerciser Program by Importing	84
Advanced Refresh	86
Creating an Exerciser Program by Copying and Pasting	87
Creating a Data Block	90
Naming a Data Block	91
Creating and Editing Data Blocks as Text	95
Settings	96
Running the Project	97
Set Data Capture Options	97
Run Project	97
Reports	98
View Statistical Report	98
View Data Report	99
View Data Report Statistics	101

Display Histogram - - - - -	102
Search a Data Report for a Pattern - - - - -	103
High Level Interpretation Assignment - - - - -	105
User-Defined Decodes - - - - -	106
Protocol Errors - - - - -	108
Protocol Errors Detected While Capturing Data - - - - -	108
Protocol Errors Detected Post-Process - - - - -	109
Pre-Trigger - - - - -	111
Performance Analysis - - - - -	113
Performance Analysis (Easy Mode) - - - - -	113
Performance Analysis (Advanced Mode) - - - - -	116
Real Time Analysis - - - - -	116
Perform a Pre-defined Analysis - - - - -	117
Create a New Analysis - - - - -	117
Define Packets - - - - -	118
Program The Exerciser - - - - -	118
Creating Analysis Expressions - - - - -	119
Saved Performance Analysis Review - - - - -	125
Timing Analysis (Optional) - - - - -	127
Easy Mode Timing Analysis - - - - -	127
Advanced Mode Timing Analysis - - - - -	129
Device Emulation (Optional) - - - - -	133
Device Emulation (Easy Mode) - - - - -	134
Programming the Device - - - - -	134
Endpoint Errors Generated by OTG DRD & Device Emulation - - - - -	141
Device Emulation (Advanced Mode) - - - - -	142
Programming the Device - - - - -	142
OTG Exerciser (Optional)- - - - -	147
OTG Exerciser as an A-Device - - - - -	147
Programming the Device - - - - -	148
Create Exerciser Program - - - - -	149

Contents

Define OTG Device Configuration - - - - -	151
Starting a Session - - - - -	151
Dropping a Session - - - - -	152
OTG Exerciser as a Peripheral B-Device - - - - -	152
Define OTG Device Configuration - - - - -	154
Create Exerciser Program - - - - -	154
Requesting a Session - - - - -	155
Terminating the Exerciser Program - - - - -	155
Host Negotiation Protocol (HNP) - - - - -	155
Session Request Protocol (SRP) - - - - -	156
How to Turn Off the VBUS (in an OTG Script) - - - - -	156
How Long Does Conquest Take to Drop D+ Pull-up? - - - - -	156
Current Measurement (Optional) - - - - -	157
Unconfigured Current Measurement- - - - -	158
Operating Current Measurement - - - - -	160
Making the VBus Measurement - - - - -	163
VBus Droop Measurement - - - - -	165
Inrush Current Measurement - - - - -	167
Inrush Current Display Features- - - - -	169
Suspend Current Measurement - - - - -	172
Current Measurement Calibration Board - - - - -	174
Display Manipulation- - - - -	177
Results Display Viewing Preferences - - - - -	178
Compact View - - - - -	179
Display Idle Time - - - - -	180
Waveform Display - - - - -	181
Filter - - - - -	182
Smart on Screen Filtering - - - - -	183
Save Display Settings - - - - -	184
Timing Analysis Display- - - - -	185
Set Timing Display Viewing Options - - - - -	186
View Timing Details - - - - -	186

Using the Cursors and Bookmarks - - - - -	188
Search- - - - -	191
Display Configuration- - - - -	196
Mnemonics - - - - -	199
Utilities - - - - -	200
Self Test - - - - -	200
View Scan Descriptors - - - - -	201
Attach/Detach Device - - - - -	202
Capture Screen - - - - -	203
Appendix A - - - - -	205
Advanced Script Language (ASL) - - - - -	205
Document Conventions - - - - -	205
Language Elements - - - - -	206
ASL Script Structure - - - - -	208
Protocol Extraction Section - - - - -	210
Protocol Decoding Section - - - - -	216
DefineOptions Block - - - - -	217
ValidRanges Block - - - - -	218
Main Block - - - - -	219
USB Descriptor Section - - - - -	229
Descriptor Block - - - - -	230
Functions - - - - -	231
Samples - - - - -	234
Script Editor - - - - -	242
Creating the Script - - - - -	243
China Restriction of Hazardous Substances Table - - - - -	246
Index - - - - -	247

Conquest Versions

Conquest M2 versus Conquest

Conquest M2 supersedes all SBAE-30 versions.

Conquest supports a subset of features of Conquest M2.

Conquest does not support the following:

- Device Emulation
- OTG Analysis and Exercising
- Timing Analysis
- VBus Measurement
- Automatic Bus Speed Detection
- Performance Analysis

Common Software

This manual describes the use and operation of all available Conquest options. In the product that you receive, access to options that were not included in your purchase are disabled. For feature upgrades and any other questions, please contact psgsupport@lecroy.com.

The following table summarizes the supported features of Conquest M2 and Conquest.

Conquest M2 (SBAE30) and Conquest Supported Features

Feature	Conquest		M2	
	Standard	Advanced	Pro	Pro Exerciser
Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)	+	+	+	+
High Speed (480 Mb/s)	+	+	+	+
Trace Memory Size	128 MB	128 MB	394 MB	394 MB
Upload to Host via USB 2.0 or LAN	+	+	+	+
Global & Raw Bit View	+	+	+	+
View Hex, Decimal or Binary	+	+	+	+
Search on Errors	+	+	+	+
Search within Data Payload	+	+	+	+
Export Text / ASCII / Binary	+	+	+	+
Snapshot Capture	+	+	+	+
Data Report	+	+	+	+
Upper-level USB Device Class Decodes	+	+	+	+
User-defined Decodes	+	+	+	+
Single-level Event Triggering (Easy mode)	+	+	+	+
Trigger on Protocol Errors	+	+	+	+
Trigger on Setup / In / Out / Data / Ping	+	+	+	+
Statistical Reports		+	+	+
Multi-level Event Triggering (Advanced mode)		+	+	+
Trigger on Split Setup Transaction		+	+	+
Trigger on Split Bulk In/Out Transaction		+	+	+
Trigger on Split Interrupt In/Out Transaction		+	+	+
Trigger on Split Isoch In/Out Transaction		+	+	+
Filter In / Out specific Address / Endpoints		+	+	+
Trigger on Data Pattern and Length			+	+
Trigger on Vbus & Operating Current			+	+
Event Counters			+	+
Event Timers			+	+
Auto Run (multiple trace capture)			+	+
Auto detect Speed			+	+
Slow-clock Capability			+	+
Performance Analyzer (Easy \ Adv. Option)			+	+
OTG Analysis			+	+
Timing Analysis Display			+	+
DC Compliance measurement			+	+
Find Device (View Scan descriptor)				+
Exerciser Graphical User Interface (GUI)				+
Traffic Generation / Host emulation				+
Simultaneous Transmit and Record				+
Device emulation				+
OTG Device Emulation				+

Introduction

This manual describes the installation and operation of the **LeCroy Conquest™ and Conquest M2 USB Analyzer/Exerciser**. Conquest has a subset of features of Conquest M2 (see “Conquest Versions” on page 1).

Conquest – The **Conquest** is LeCroy's entry-level USB analyzer, designed for easy setup with time-saving features like predefined trigger settings. The **Conquest Advanced** adds multi-state sequential triggering to provide sophisticated triggering debugging intermittent problems.

Conquest M2 - The **Conquest M2 Pro** combines full function analysis capabilities with an integrated exerciser option. The Pro model includes the same easy-to-understand display of bus traffic and adds advanced multi-state triggering, OTG support, real-time performance analysis, special timing and DC Compliance measurements. The **exerciser** can be enabled via software key to allow host, device, and OTG emulation.

Overview

Conquest M2 is a serial bus Analyzer/Exerciser that is capable of analyzing and exercising data transfers for USB 1.x and USB 2.0 Low speed, Full speed, and High speed protocols. One upstream and one downstream auxiliary connector allows convenient testing of Hub input and output I/O ports. The following is a description of Conquest M2 capabilities if all options are installed.

Conquest M2 incorporates two modes of operation, **Easy** and **Advanced**. You can operate each of the installed options, except the OTG Exerciser, in any mode combination. As an example, you can perform a Capture and Trigger with either the Easy Mode Exerciser or the Advanced Mode Exerciser.

The convenient, easy-to-use **Easy Mode** allows you to perform 95% of USB data capture and triggering with a minimum of programming. In the **Easy Mode**, you can quickly:

- Capture and Trigger on USB packets.
- Generate Host traffic with the **Host Exerciser** while monitoring and analyzing the result.
- Analyze and Exercise **OTG** Devices.
- Perform a real time **Performance Analysis**.
- Perform simultaneous **Protocol** and **Timing Analysis**.

The comprehensive **Advanced Mode** incorporates a powerful easy to use programming capability that allows you to design projects with sophisticated data capture and triggering. In the **Advanced Mode**, you can program the Analyzer/Exerciser to:

- Capture all bus traffic or specific packets
- Trigger on specific packets or events as programmed in a 32 state sequencer.
- Generate more flexible host traffic with the **Host Exerciser**.
- Emulate more complicated devices while monitoring and analyzing the result.
- Analyze and Exercise **OTG** Devices.
- Perform a detailed **Timing Analysis** to trigger as defined in the Sequencer or on a user-specified bus signal pattern.
- Program Conquest M2 as a **Device Emulation** for testing the USB development software.
- Perform **Unconfigured, Suspend, Operating, and Inrush** current measurements.
- Perform **VBus** and **VBus Droop** measurements.
- Trigger protocol analyzer based on VBus and Operating current measurements.

Analyzing USB Designs With the Analyzer

The Analyzer allows you to perform comprehensive USB analysis by capturing data and triggering on events and packets, conduct **Performance Analysis** in real-time, capture and trigger on USB errors and also to allow verification and debug of Hub input and output ports.

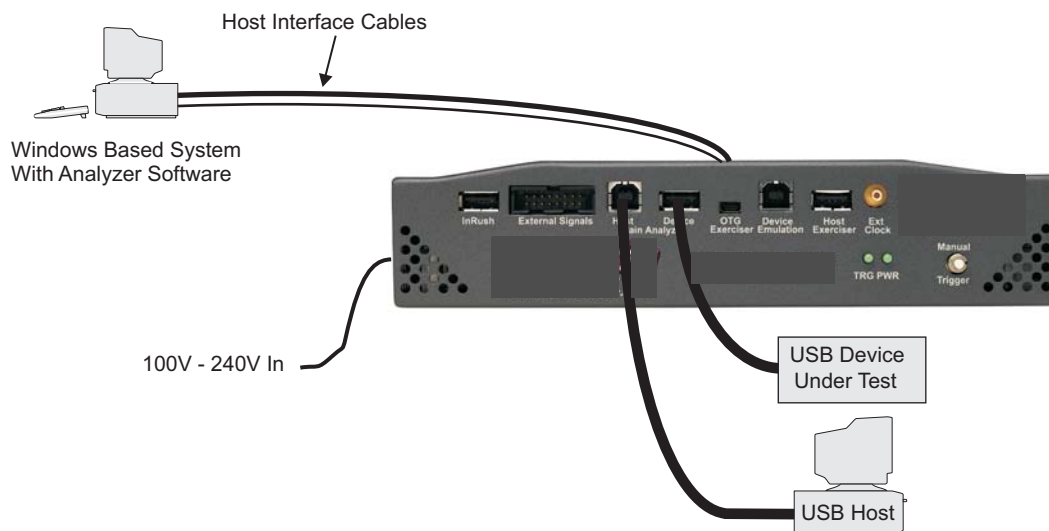


Figure 1 Typical USB Test Setup of Basic Analyzer

Introduction

Conquest M2 Options

You can expand Analyzer capability with the following options:

Host Exerciser

The Host Exerciser option allows you to generate bus traffic as a host for analysis.

Device Emulation

This option allows you to operate Conquest M2 as a Device for testing the USB development system. You can define each Device to include up to 3 configurations with 2 interfaces each and assign up to 7 endpoints to these interfaces. The Device can be a Low, Full, High, or a High/Full speed device.

OTG Exerciser

This option allows you to operate Conquest M2 as an OTG Dual-Role Device. In this mode, Conquest M2 is a combination of Device Emulation and Host Exerciser. A DRD emulation can be either a host or a peripheral.

DC Compliance measurement

This option allows you to measure operational current and inrush current through a special port over the first 10 milliseconds of Device activation and to display a pass/fail result with respect to the USB compliance specifications. This option also allows the measurement of VBus voltage through the Analyzer port.

Timing Analyzer

This option offers precision timing analysis. The Conquest M2 Timing Analyzer samples bus signals at different sampling rates depending on the bus speed selected.

Conquest M2 Interface

Status LED Function Description

PWR When illuminated indicates that Analyzer is powered up.

TRG When illuminated indicates that the Analyzer has triggered.

Manual Trigger

Manual trigger is activated by a push button switch on the front panel.

External Signals (Ext. Out, Ext. Trig In, and Ext. Clk)

The Conquest M2 Analyzer can output a unique 8-bit, user-defined CMOS level when triggered. You can use these outputs for activating external devices, such as other types of test equipment, when certain events are detected in USB data capture.

Additionally, the Conquest M2 Analyzer provides you with an External Trigger Input that allows you to trigger the Analyzer from externally generated sources. Trigger input can be masked or programmed to respond to a 1 or 0 level or a rising or falling edge. For setting the external trigger input options, see page 55.

Conquest M2 offers a number of internally generated pre-defined low frequency clock settings, however you can apply an external clock to the External Clock Input. Supported frequencies are in ranges of 400KHz - 8MHz for High Speed and 1KHz-100KHz, 400KHz - 8MHz for Full/Low speed.

Introduction

Ports

Main Analyzer

This port supports Analyzer using 8 packets and a 32 state sequencer that are used in Easy Mode or Advanced Mode for data capturing and triggering, operating and unconfigured current measurement and timing analysis.

InRush

A dedicated connector to allow the measurement of USB Device Inrush current.

Host Exerciser

A dedicated connector to exercise a USB Device and capture and trigger data in conjunction with the Analyzer port.

OTG Exerciser

A dedicated connector to exercise an OTG Dual-Role or SRP- capable Peripheral Device.

Device Emulation

A dedicated connector used when Conquest M2 is operated as a device.

Paral. Host Interface

Bi-directional Parallel Port interface located on the back of the unit for connection between the Analyzer and the system hosting the GUI software.

USB Host Interface

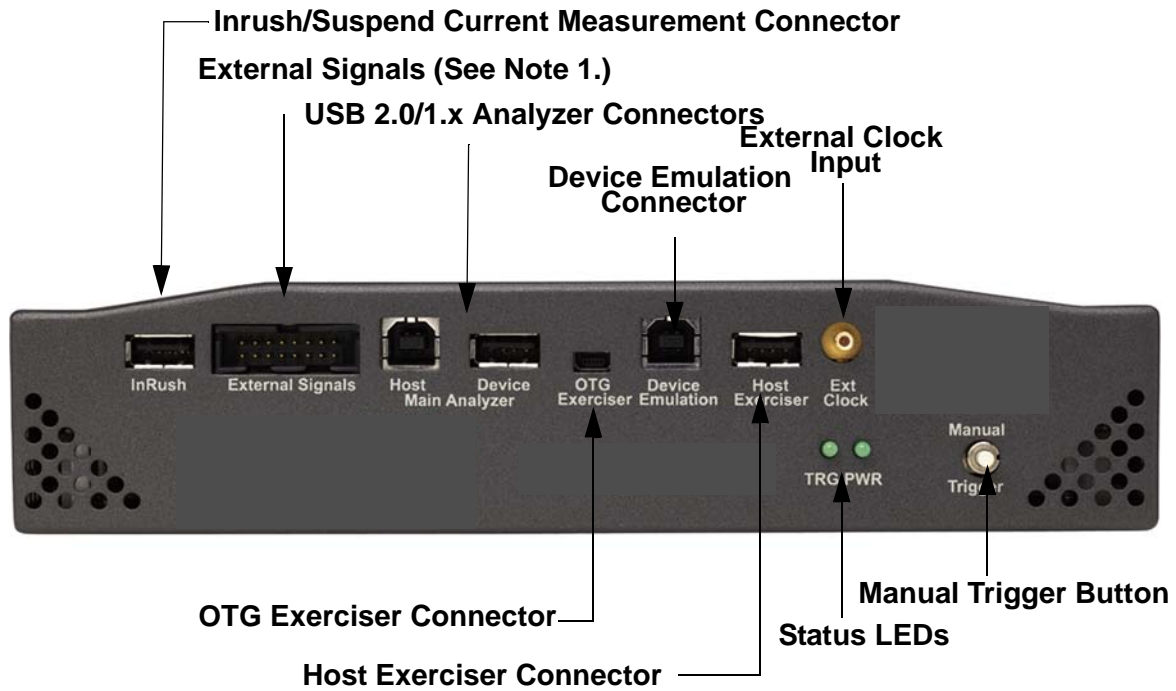
Interface connection, located on the back of the unit, for connection between the Analyzer and the system hosting the GUI software.

Note: Use only one, either Parallel Port or USB connection, for connecting the Analyzer and the system software.

Power In

Conquest and Conquest M2 operate on 100V - 240V 50/60Hz power that is connected on the back.

Analyzer/Exerciser Connection Identification



Note 1 For External I/O connector pin assignment, see **page 15**.

Receiving the Analyzer

Conquest M2

The Conquest M2 Analyzer includes the following components:

- Carrying Case
- Conquest M2 identified in the packing list
- Software on the CD Rom
- DB-25 -> SCSI II 26, Parallel Port connection
- Two USB 2.0 Cables, 1.8 meter
- USB 2.0 Cable, 4" (10 cm)
- Two USB 2.0 Cables, 2 meter Standard - A to Mini - B
- One Mini - A to Standard - B.
- 3M Ethernet Cable
- External I/O Ribbon and External Clock SMB to BNC Cables
- Current Measurements Calibration Board
- Power cable
- User Manual (PDF version on CD only) for both Conquest and Conquest M2

Conquest

The Conquest Analyzer includes the following components

- Carrying Case
- Conquest identified in the packing list
- Software on the CD Rom
- Two USB host interface cables, 1.8 M A to Mini - B
- USB, A to B cable, 4" (10cm)
- 3M Ethernet Cable 10ft (3M)
- External trigger cable (SMB2BNC)
- External signals cable
- Power cable, 6ft (1.8M)
- User Manual

Unpacking the Analyzer

Inspect the received shipping container for any damage. Unpack the container and account for each of the system components listed on the accompanying packing list. Visually inspect each component for absence of damage. In the event of damage notify the shipper and LeCroy. Retain all shipping materials for shippers inspection.

Installing the Analyzer

This section covers hardware and software setup.

Hardware Setup

You can set up the Analyzer/Exerciser in any of the following configurations:

- As an Analyzer to analyze USB Host and Device transactions.
- As an Exerciser to exercise a USB Device.
- As an Analyzer/Exerciser for OTG Device testing.
- As an Analyzer/Exerciser in Hub testing.
- To measure Inrush current, Configured current, Unconfigured current, Suspended current and VBus measurement
- As a Device for Host software testing.

Separate Systems

It is highly recommended that you employ separate systems for the Analyzer software and the USB Host as shown in Figure 2 (the preferred configuration) to analyze USB Host and Device transactions. Using just one system for both functions can lead to confusing results because of shared bus traffic.

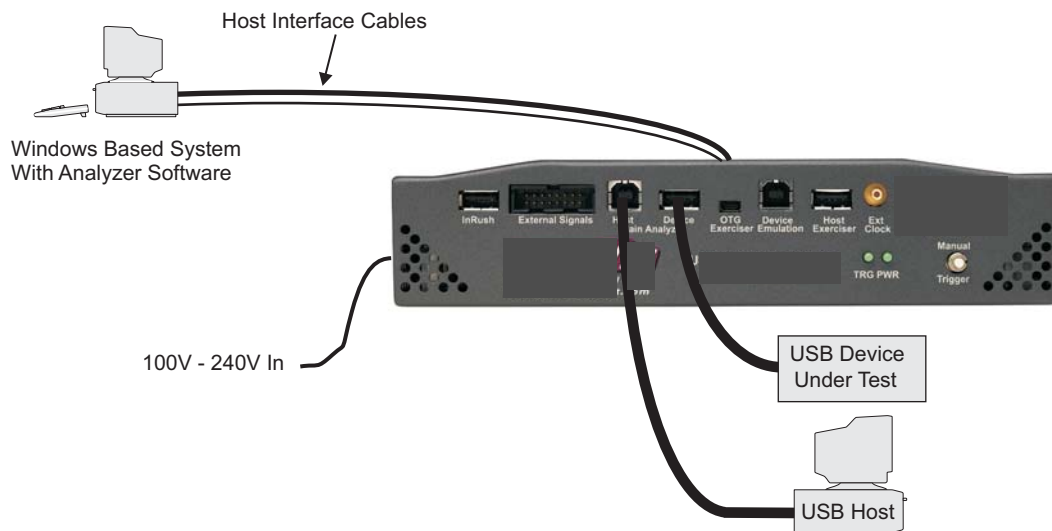


Figure 2 Conquest M2 Configured as an Analyzer

1. Connect Conquest M2 to a Parallel, USB, or Ethernet Port on a system running the Conquest Protocol Suite software, with the appropriate cable.
2. Connect the USB Device to the Main Analyzer Port on Conquest M2.
3. Connect the USB Host to the Main Analyzer Port on Conquest M2, using a USB cable.

Installing the Analyzer

Exerciser Configuration Connections

Figure 3 shows Conquest M2 used to exercise a USB Device. In this configuration, Conquest M2 acts as the USB Host.

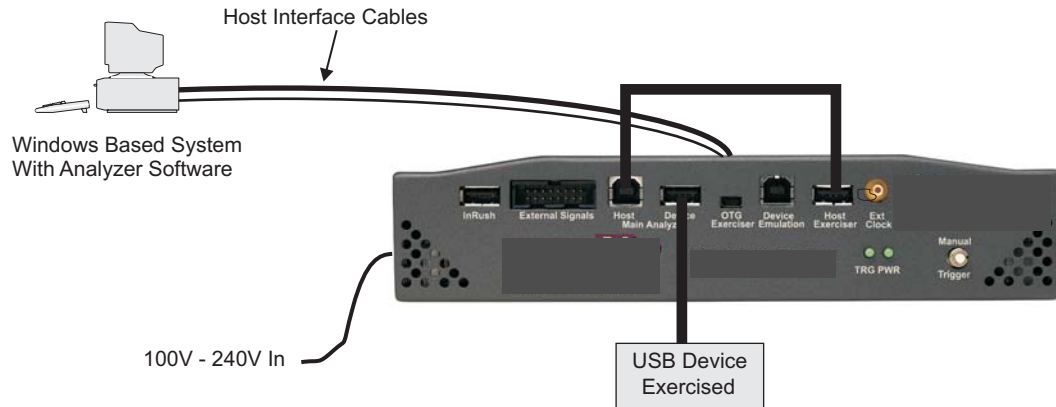


Figure 3 Conquest M2 Configured as an Exerciser

1. Connect Conquest M2 to a Parallel, USB, or Ethernet Port on a system running the Conquest Protocol Suite analysis software, with the appropriate cable.
2. Connect the USB Device to the Main Analyzer Port on Conquest M2.
3. Connect the Main Analyzer Port to the Host Exerciser Port using a USB cable.

OTG Analysis Connections

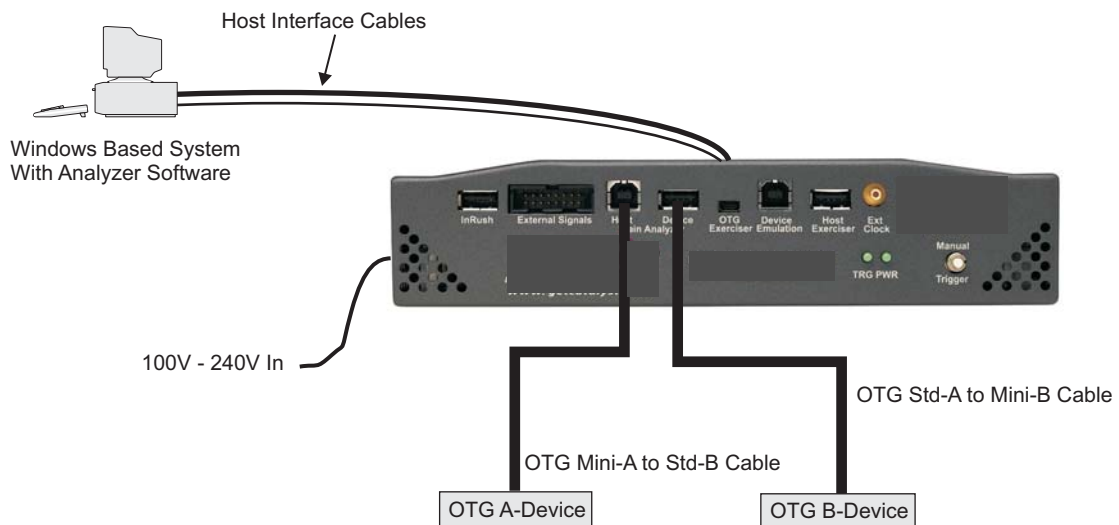


Figure 4 Conquest M2 Connections for OTG Analysis

OTG Exerciser as DRD A-Device Connections

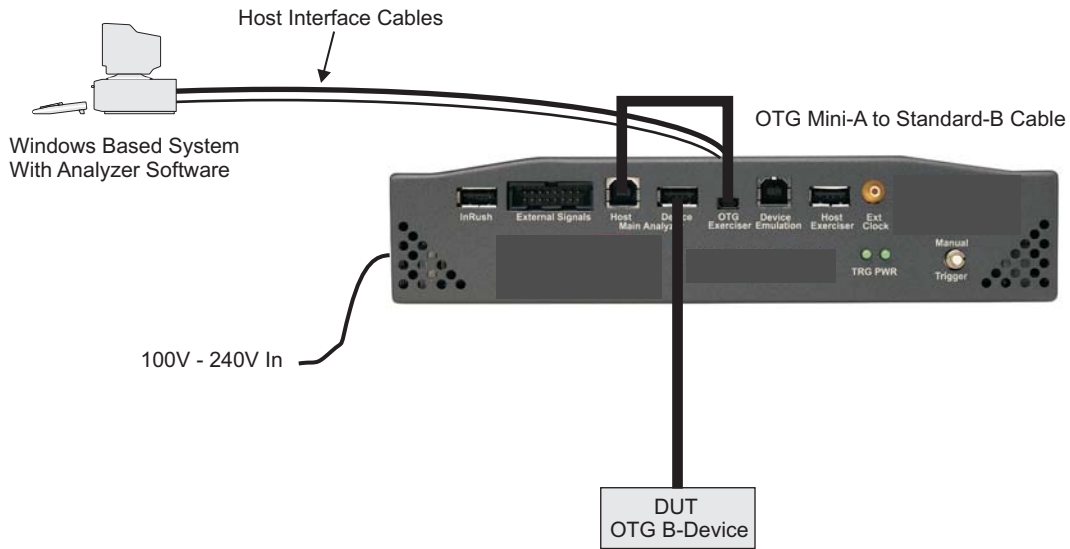


Figure 5 OTG Exerciser as DRD A-Device

OTG Exerciser as DRD or Peripheral Only B-Device Connections

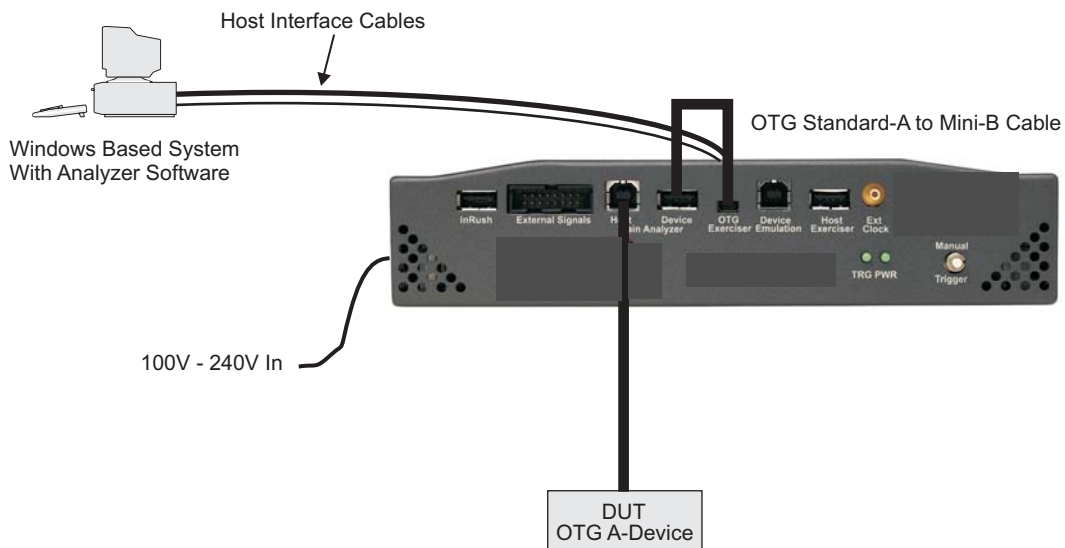


Figure 6 OTG Exerciser as DRD or Peripheral Only B-Device

Installing the Analyzer

Device Emulation Connections

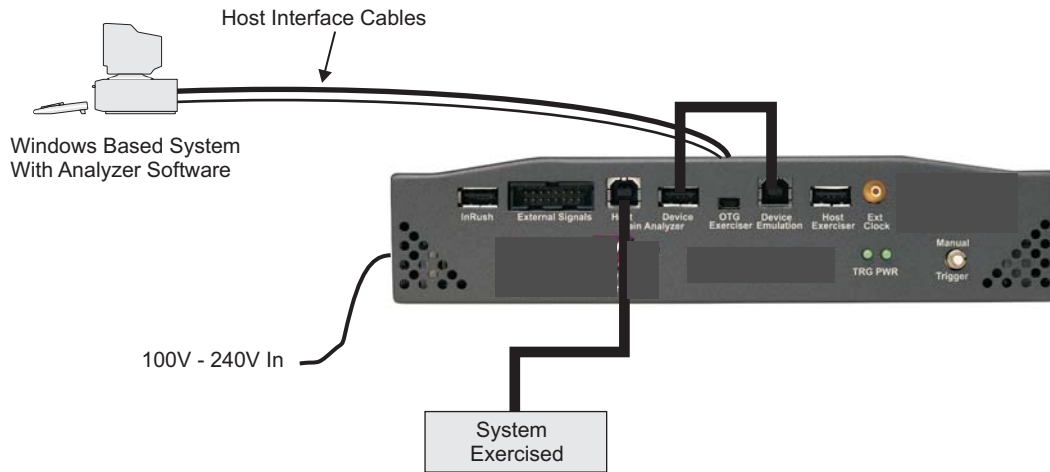


Figure 7 Conquest M2 Connections for Device Emulation

Conquest Analyzer Connections

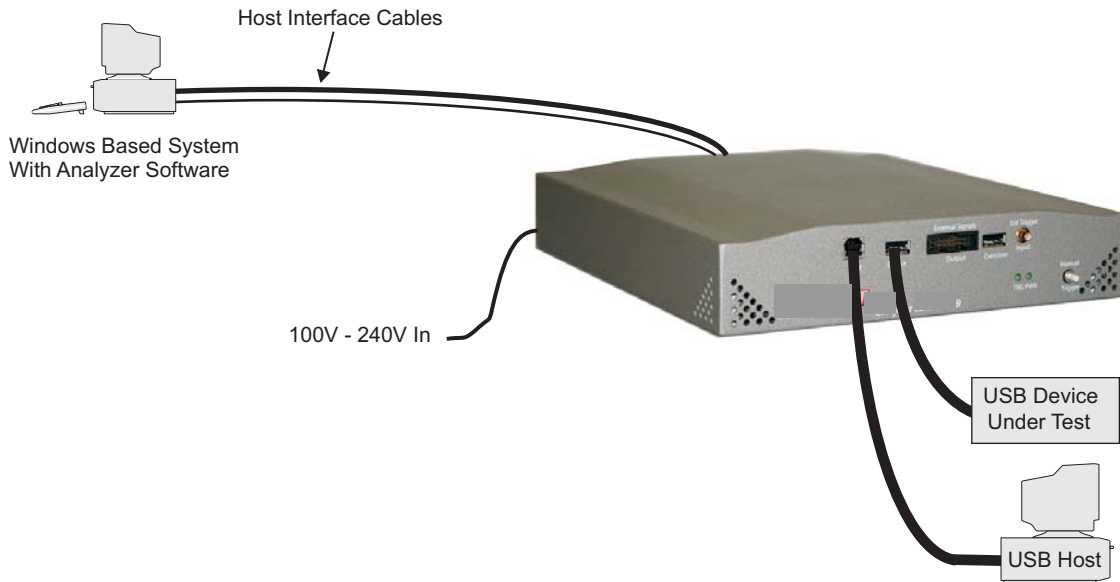


Figure 8 Conquest Connected as an Analyzer

Conquest Exerciser Connections

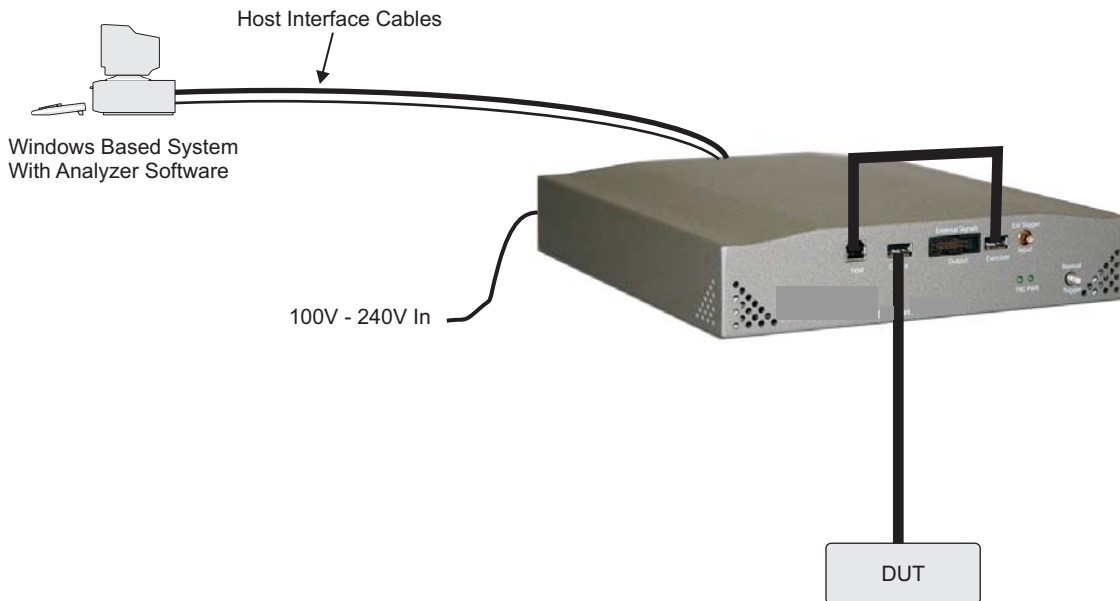


Figure 9 Conquest Connected as an Exerciser

Note: The Exerciser option is no longer available on the Conquest platform. Exerciser is offered on the Conquest M2 system only.

Conquest M2 External I/O Connector Pin Assignment

The external I/O connector provides the means for accepting a CMOS level input trigger and 8 CMOS level outputs that represent an 8-bit value.

Table 1 External I/O Connector Pin Assignment

Signal	Pin	Color	Signal	Pin	Color
External Trigger In	1	Brown	External Out 7	2	Red
GND	3	Orange	External Out 6	4	Yellow
GND	5	Green	External Out 5	6	Blue
GND	7	Violet	External Out 4	8	Grey
GND	9	White	External Out 3	10	Black
GND	11	Grey	External Out 2	12	Red
GND	13	Orange	External Out 1	14	Yellow
GND	15	Green	External Out 0	16	Blue

Conquest External I/O Connector Pin Assignment

Table 2 External I/O Connector Pin Assignment

Signal	Pin	Color	Signal	Pin	Color
No connect	1	Brown	No connect	2	Red
GND	3	Orange	External out 7	4	Yellow
GND	5	Green	External out 6	6	Blue
GND 2	7	Violet	External out 5	8	Gray
GND	9	White	External out 4	10	Black
GND	11	Brown	External out 3	12	Red
GND	13	Orange	External out 2	14	Yellow
GND	15	Green	External out 1	16	Blue
			External out 0	N/A	

Note 1 Pin 1 is the right-most in the cable (Brown) and pin 16 is the left-most in the cable (Blue).

Note 2 The signal “External out 0” (LSB) is not externally available.

Software Installation

On systems operating under Windows® 2000, Windows XP, and Windows Vista.

Note: Do not connect the Analyzer to the host system until software installation is complete!

1. Insert the CD Rom in the CD Rom drive.
2. The installation automatically starts the setup, unless the Auto Run is turned off. In this case, select the CD Rom from “My Computer” and click **setup**.
3. After the warning to close all other programs and before starting the installation, the Install component selection opens.
4. Select components for installation.
5. Click **Next** to complete the installation.

System Restart

You must restart the computer before you can use the Analyzer software.

Error Message

If you get an error message during installation of the drivers for Windows 2000, XP, or Vista, consult your system administrator. Your system may allow only an administrator-level user to copy driver files.

Manual USB Driver Installation

To manually install the USB driver, perform the following:

1. Power up Conquest.
2. Connect the USB cable between the Conquest USB Port interface and the Host Computer USB port.
3. After Windows detects the new Device, select the **Search For Suitable Driver for My Device** option button.
4. Click **Next**.
5. Choose the **Specific location** option button and deselect all other option buttons.
 - Click **Next**.
 - Go to the **C:\program files\LeCroy\Conquest\usb driver\win2k** folder.
 - Choose **ibusb.inf**.
 - Click **OK**.
6. Click **Next** and then **Finish** to complete the Conquest USB port driver installation.

Updating the Conquest USB Driver Manually

1. Click **Start > Settings > Control Panel > System** and click the **Device Manager** tab.
2. Find the **Test Tools** entry.
3. Select and right-click the **Conquest** board.
4. Select Properties and click the drivers tab.
5. Click **Update Driver**.
6. Select recommended method
7. Choose the **Specific location** option button and deselect all other option buttons.
 - Click **Next**.
 - Go to the **C:\program files\LeCroy\Conquest\usb driver\win2k** folder.
 - Choose **ibusb.inf**
 - Click **OK**.
8. Click **Next** and then **Finish** to complete the USB driver update.

Directory Structure on Windows XP and Vista

The Conquest Protocol Suite application stores files in a specific directory structure under the **C:\Program Files\LeCroy\Conquest** directory. It also stores files in Windows-specific locations that differ for Windows[®] XP and Windows Vista (which uses UAC).

In particular, Windows XP has a **Documents and Settings\All Users** folder, and Windows Vista has a **Users\Public** folder, which are not equivalent.

The Windows XP **All Users** folder is not readily available to users and is for storing shared files, to which normal user folders point using shortcuts.

The Windows Vista **Users\Public** folder and its subdirectories:

- Are available to all users logged onto a system (not just the person who installed the application).
- Are not hidden.
- Are always present, because it is provided by the Windows Vista OS installation.
- Are readable and writable by all users.
- Do not require Administrator privileges.

Methods of using shared folders differ for the two operating systems, so the Conquest Protocol Suite application stores files on them differently, as described in the next two sections.

Windows XP

For Conquest Protocol Suite on Windows XP, all application and user files (such as sample files, user files, default files, and scripts) are in the **<drive>\Program Files\LeCroy\Conquest** directory folder and its subdirectories:

- Doc
- CAPI
- SystemData
- UserData
- Examples

The installation process creates the folder **<drive>\Program Files\Lecroy\Conquest**.

Running the application the first time creates the folder **<drive>\Users\Public\Public Documents\LeCroy\Conquest 7.40.xx**

Installing the Analyzer

Windows Vista

User-modifiable Files

For Conquest Protocol Suite on Windows Vista, user-modifiable files are in the Windows Explorer

<drive>\Users\Public\Public Documents\LeCroy\Conquest 7.40.xx folder, which is the same as the Command Tool

<drive>\Users\Public\Public Documents\LeCroy\Conquest 7.40.xx folder and **Desktop\Public Documents\LeCroy\Conquest 7.40.xx** folder.

The Command Tool name is the correct folder name when using scripts.

The **Users\Public\Public Documents\LeCroy\Conquest 7.40.xx\UserData** folder contains files that the user can modify, such as **new.cpr** and **Out.smp**. It also has the directories:

- SystemData
- Temp
- UserData

Application Files

For Conquest Protocol Suite on Windows Vista, read-only application files (such as the application, bus engine, firmware, user manual, help, and read-me files) are in the **<drive>\Program Files\LeCroy\Conquest** folder, which has the directories:

- CAPI
- Doc
- Examples
- SystemData
- usb driver
- UserData

Note On Windows Vista systems, the Conquest Protocol Suite application creates an image of its directories in **Public\Public Documents\LeCroy\Conquest 7.40.xx**.

Note 2 For Windows Vista systems, the Conquest Protocol Suite application creates directories for Conquest when the application is first run, not when it is installed.

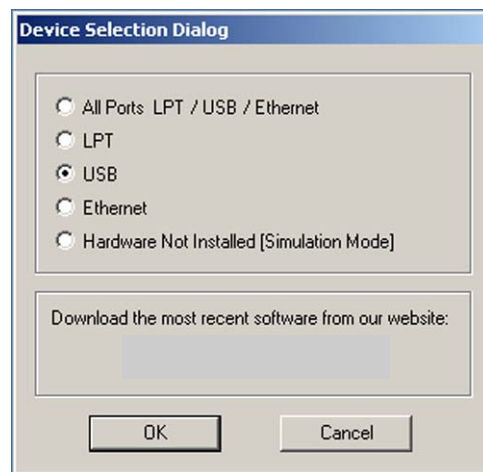
Launching the USB Analyzer

To launch the software, select LeCroy > Conquest in the Start menu or double-click the **Conquest Suite** button in the Program Manager Window.

Each time you run the software, it searches for a default host interface and, if it is found, the software launches. If no interface is found, the software launches in simulation mode.

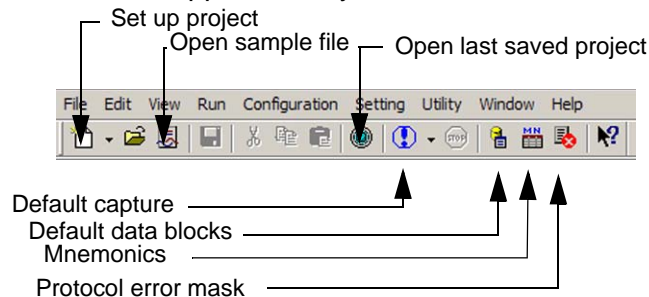
Establish Interface

If no interface is detected initially, then establish an available interface and relaunch the software.



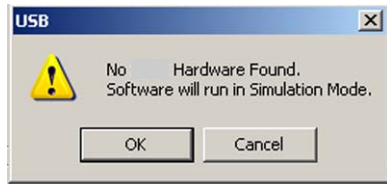
Click **OK**. The Analyzer launches and displays the Analyzer toolbar as shown below.

Note: The very first time that you run the software, the USB Class Decodes loads. This process takes approximately one minute.



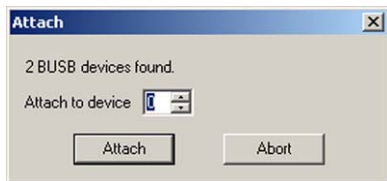
Launching the USB Analyzer

If Conquest is not detected, the software displays the **Hardware Not Detected** message.



Multiple Analyzers

If you have multiple Analyzers installed, you get the following message. In such cases choose the Analyzer and click **Attach**.



Connecting via Ethernet

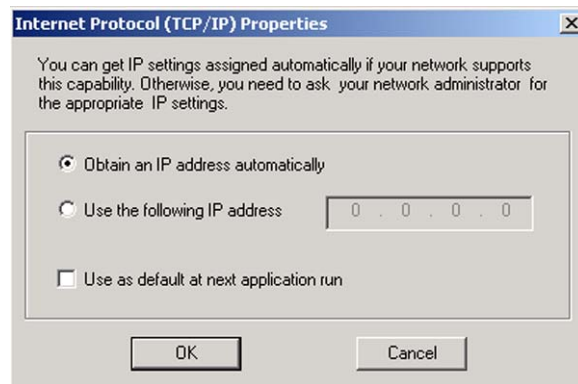
You can use the ethernet connection using any one of the following three supported configurations:

1. Conquest connected to a network via a hub, switch, or similar device.
2. Conquest connected to the host computer (machine running the application software), via a hub, switch or similar device.
3. Conquest connected directly to the host computer using a crossover cable.

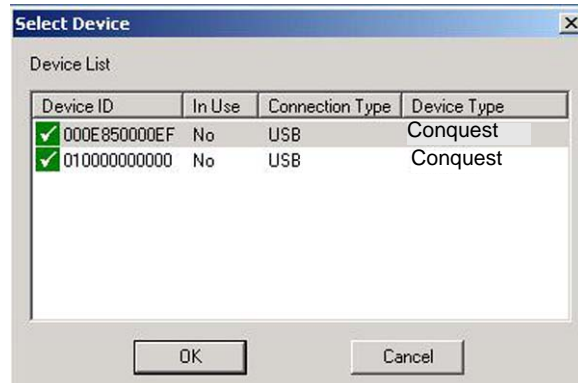
Connecting to a Network

When connected to a network, Conquest must communicate with the DHCP server to establish a connection. The DHCP server continually sends the next available IP address to Conquest until the Conquest Protocol Suite software starts.

When you start the software, the system may prompt to automatically use the offered IP address or use an assigned IP address. (The assigned IP address must be on the same network segment as the host computer.) The menu also allows you to save the selected option (automatic or specific address). If the assigned IP address is not available, the OS notifies you of an IP address conflict.



After you click **OK**, the software searches for all Conquest units connected to the network and displays a list of available Conquest units. After you select a Conquest unit, the software assigns the IP address to the selected unit, completing the connection, and launches the software.



Connecting via Hub, Switch or Similar device

When connected to the host machine via a hub, switch, or other similar device, or directly using a crossover cable, the board must communicate with the host computer to establish a connection. The host computer continually broadcasts the next available IP address to the board until the Conquest Protocol Suite software starts.

When the software starts, the system may prompt if you want the software to automatically use the offered IP address or use an assigned IP address. (The assigned IP address must be on the same network segment as the host computer.) The menu also allows you to save the selected option (automatic or specific address). If the assigned IP address is not available, the OS notifies you of an IP address conflict.

After you click **OK**, the software searches for all boards connected to the network and displays a list of available boards. After you select a board, the software assigns the IP address to the selected board, completing the connection, and launches the software.

Operating in Simulation Mode

The system operates in Simulation Mode as default if the hardware is not detected. However, you can operate in Simulation Mode directly without installing the Analyzer hardware. To operate without hardware, select **Hardware Not Installed (Simulation Mode)**.

The Analyzer software launches and displays the tool bar, but with the limitation that the Analyzer operates on static, previously captured bus data.

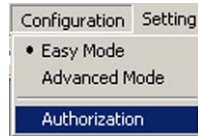
Limitations

The Simulation mode lets you try all of the available functions, but keep in mind that **the system is not capturing any real data and is displaying pre-captured results**.

Authorization

An authorization code is required to activate features not originally included and purchased at a later time.

To enter the authorization code, click **Configuration** on the main toolbar and then select **Authorization**.



The Authorization Code dialog opens.

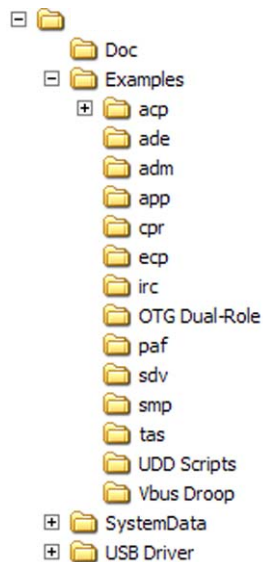


Enter the authorization code for the product(s) purchased and click **OK**.

Example Projects

Conquest includes a number of pre-defined example projects that you can use to perform an immediate analysis without any setup.

The Conquest Protocol Suite software comes with a pre-defined folder (directory) structure for storing all files. All example files are stored in the Examples folder.



It is strongly recommended that you open some of these files to get an introduction to project types.

Project File Types

The various Conquest configurations may have the following file types (see the next page for which configurations support which file types):

*.acp	Advanced capture project file
*.ade	Advanced emulation example
*.adm	Device Emulation project file
*.aoe	OTG Exerciser file
*.app	Advanced performance analysis project file
*.asl	Advanced Script language file
*.bin	Binary file
*.cfg	Viewer Display Configuration settings
*.cpr	Custom project file
*.dat	Data block file
*.ecp	Easy capture project file
*.epp	Easy performance analysis file
*.irc	Previously performed inrush current measurement file
*.paf	Easy performance analysis project file
*.rpt	Viewer report file
*.sdv	Scan descriptor file
*.smp	Previously performed project results file
*.tas	Previously performed timing analysis file
*.txt	Text file
*.vbd	VBus Droop file
*.vdr	Viewer data report file

Launching the USB Analyzer

Conquest Standard and Advanced configurations do not support all project types. The table below outlines which project and report types can be created, opened, executed by each of the Conquest configurations.

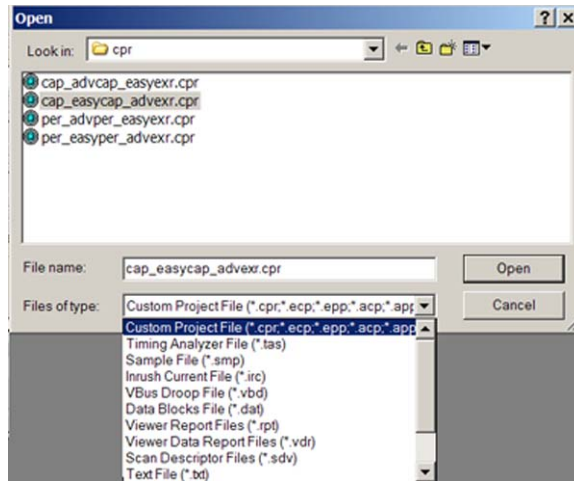
Table 3 File Types in Conquest Models

File Types	Conquest Standard	Conquest Advanced	Conquest M2 Pro	Conquest M2 Pro with Exerciser
*.ecp Easy capture project file	X	X	X	X
*.cfg Viewer Display Configuration settings	X	X	X	X
*.asl Advanced Script language file	X	X	X	X
*.vdr Viewer data report file	X	X	X	X
*.rpt Viewer report file	X	X	X	X
*.txt Text file	X	X	X	X
*.smp Previously performed project results file	X	X	X	X
*.acp Advanced capture project file		X	X	X
*.bin Binary file				X
*.cpr Custom project file			X	X
*.dat Data block file				X
*.epp Easy performance analysis file			X	X
*.irc Previously performed inrush current measurement file			X	X
*.paf Easy performance analysis project file			X	X
*.app Advanced performance analysis project file			X	X
*.sdv Scan descriptor file				X
*.tas Previously performed timing analysis file			X	X
*.vbd VBus Droop file			X	X
*.ade Advanced emulation example				X
*.adm Device Emulation project file				X
*.aoe OTG Exerciser file				X

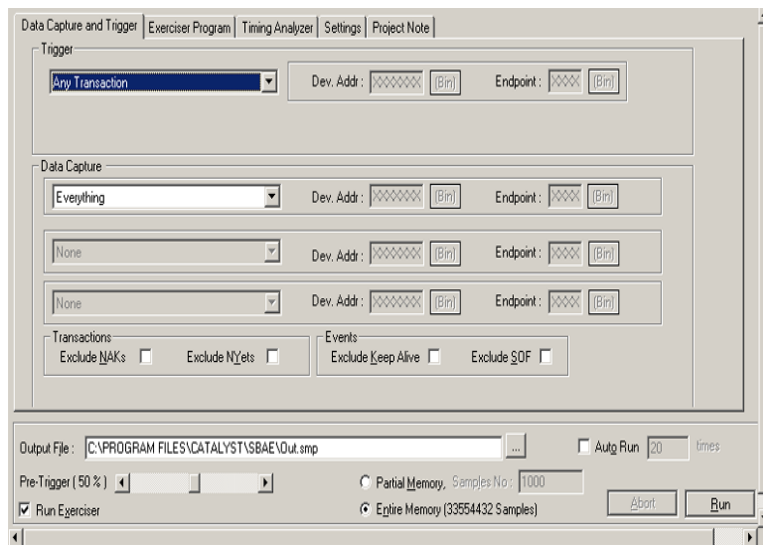
Run an Example Project

To run a project:

1. On the menu bar, select **File > Open**.



2. Choose the example type from the Files of Type Combination box, choose an example, and click **Open**.



3. Click **Run** to execute the example.

Default Capture

The default capture offers you one button activation of a default project.

The software includes a pre-defined default capture. However, you can define your own default capture by creating a new capture project and setting it to be the Default Project for immediate capture.



Click the **Default Capture** button on the main toolbar to perform an immediate data capture and trigger.

After a short time, current bus activity displays.

P	T	Control Transfer	S	Dev. Addr(Hex)	Interpreted Audio Class Request Data			
1	65	Class Type Request	F	02	GET_MN			
1	65	Class Type Request	F	02	GET_CUR			
1	67	Set Interface	F	02	1	1		
1	68	Class Type Request	F	02	SET_CUR			
1	69	Class Type Request	F	02	SET_CUR			
1	462	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	462	IN	F	013.296.379.63	02	0010	5	No Handshake
1	463	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	463	OUT	F	013.297.379.35	02	0001	176	No Handshake
1	464	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	464	OUT	F	013.297.504.23	00	0001	0	No Handshake
1	465	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	465	OUT	F	013.298.379.08	02	0001	176	No Handshake
1	466	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	466	OUT	F	013.299.378.82	02	0001	176	No Handshake
1	467	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	467	OUT	F	013.300.378.53	02	0001	176	No Handshake
1	468	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	468	OUT	F	013.301.378.27	02	0001	176	No Handshake
1	469	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	469	OUT	F	013.302.378.00	02	0001	176	No Handshake
1	470	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	470	OUT	F	013.303.377.72	02	0001	176	No Handshake
1	471	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	471	OUT	F	013.304.377.45	02	0001	176	No Handshake
1	472	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	472	OUT	F	013.305.377.18	02	0001	180	No Handshake
1	473	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	473	OUT	F	013.306.376.90	02	0001	176	No Handshake
1	474	Transaction Type	S	Start Time	Dev. Addr(Hex)	Data Length	Handshake	
1	474	OUT	F	013.307.376.63	02	0001	176	No Handshake

Note: After completion of data capture, the system uploads a 64-kilobyte page for each screen view. The system saves uploaded 64-kilobyte pages in the \Temp folder as temporary files with extension **.smp_**. If you **Save** the data capture, the system saves the whole recorded capture on a hard disk. Saving data replaces the temporary extension **.smp_** with extension **.smp**.

Creating Projects

The Conquest M2 Analyzer allows you to create projects using Easy and Advanced mode capabilities in any combination.



Click the **New Project** button on the main toolbar to display the Project Setup dialog.

Click the project type and configure the project to define Easy and/or Advanced mode operation as applicable.



Capture and Timing Analysis Projects

See page 37 for setup of an Easy Mode Capture.

See page 59 for setup of an Advanced Mode Capture.

See page 50 for setup of an Easy Mode Exerciser.

See page 71 for setup of an Advanced Mode Exerciser.

See page 127 for setup of Easy and Advanced Mode Timing Analysis.

Performance Analysis Projects

Check a **Performance Analyzer, Exerciser** option and click the **Create Project** button to display the corresponding project setup dialog.

See page 113 for setup of an Easy Mode Performance Analysis.

See page 116 for setup of an Advanced Mode Performance Analysis.

Device Emulation & OTG Advanced Exerciser

Check the **Emulation** to use to open the corresponding project setup dialog.

See page 134 for setup of an Easy Mode Device Emulation.

See page 142 for setup of an Advanced Mode Device Emulation.

See page 147 for setup of the OTG Advanced Exerciser.

Current and Voltage Measurements

Check a **measurement type** to start the measurement.

See page 167 to perform an Inrush Current measurement.

See page 158 to perform an Unconfigured Current measurement.

See page 172 to perform an Suspend Current measurement.

See page 160 to perform an Operating Current measurement.

See page 163 to perform a VBus measurement.

See page 165 to perform a VBus droop measurement.

Protocol Analysis

Easy Mode (Pre-Defined Setups)

This mode allows you to operate Conquest with a minimum of setup. You can perform a Trigger and Data capture only, or program an Exerciser to generate bus traffic for triggering and data capture. Figure 10 shows a functional Capture and Trigger setup.

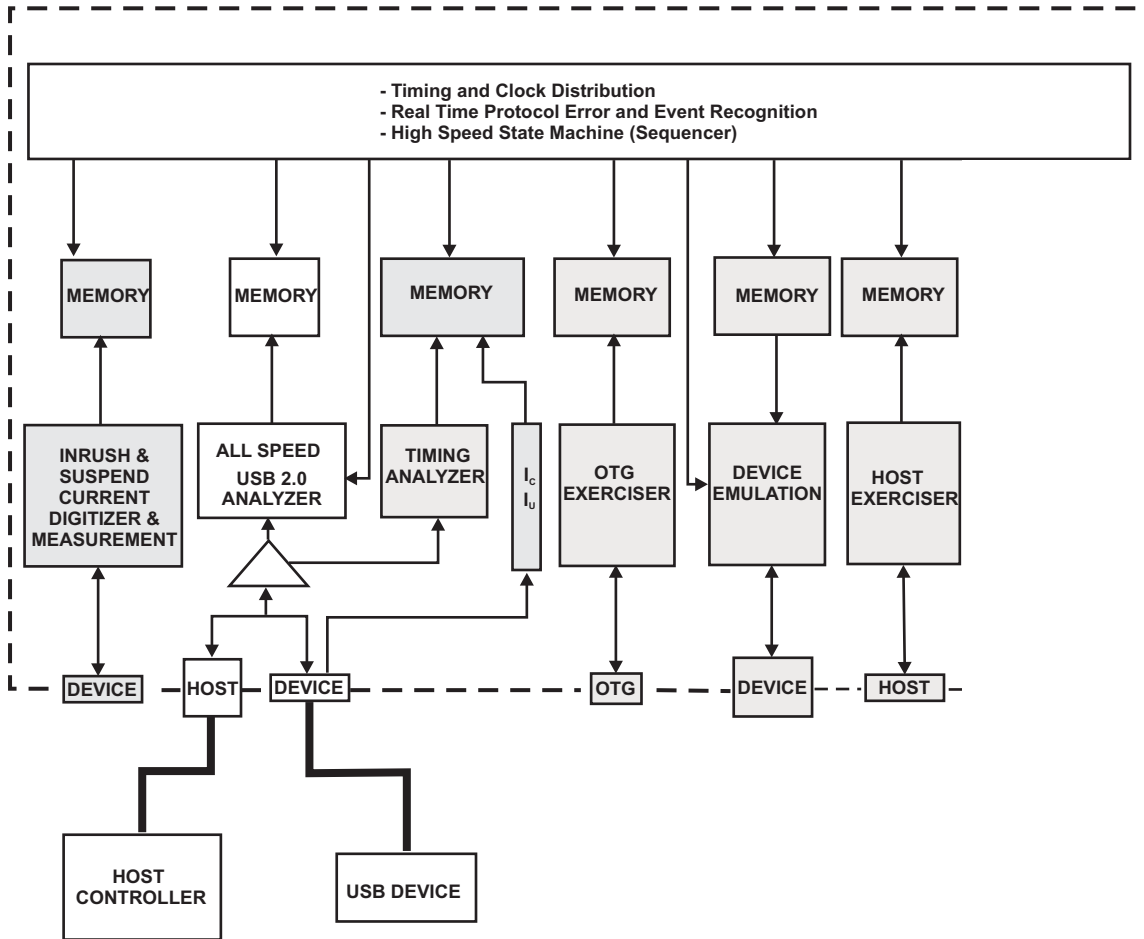


Figure 10 Capture and Trigger Functional Connection

Easy Data Capture

Make sure that the configuration is set to use Easy Analyzer. See “Creating Projects” on page 30.



Click the **Green** button on the main toolbar to open the last saved project.

New Project

To start a **New** project:

1. Select **File > New** and choose **Analyzer Host Exerciser Project**.

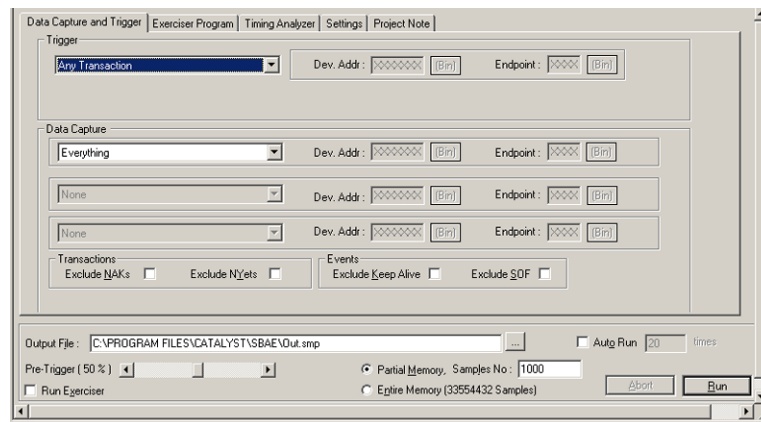


Figure 11 Trigger and Capture Data Dialog Box

2. Click the drop-down arrow next to the Trigger drop-down list box and choose a trigger. Table 4 on page 46 lists the available trigger selections. The available triggers are different for High speed and Full/Low speed operation. All triggering selections, however, are available when the Analyzer is set to operate in the Auto Speed mode.

Protocol Analysis

3. If known, you can specify a Device address (in hex) and an endpoint address (in binary) by making the appropriate entries in the **Dev Addr** and **Endpoint** text box.

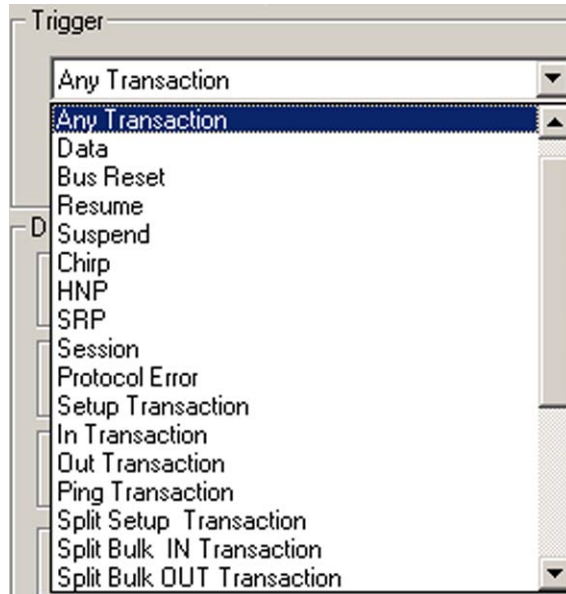


Figure 12 Trigger Choices in High Speed Mode

4. Choose up to three Data Capture selections from the three available Data Capture drop-down list boxes. Table 5 on page 48 lists the available selections. The capture selections are different for High speed and Full/Low speed operation. All capture selections, however, are available when the Analyzer is set to operate in the Auto Speed mode (only in Conquest M2).

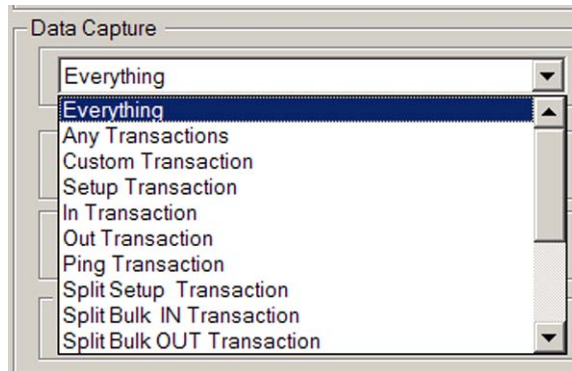


Figure 13 Data Capture Choices in High Speed Mode

To exclude **NAK**, **NYET** Transactions and/or **SOF**, **Keep Alive** Events, check the corresponding check box. These are filtered in real time by the hardware without using memory resources.

5. Click **Run** to capture and trigger immediately and wait for the results to display, as shown in Figure 14.

For ways to enhance results analysis in the results display, see “Display Manipulation” on page 177.

Note: After completion of data capture, the system uploads a 64-kilobyte page for each screen view. The system saves uploaded 64-kilobyte pages in the \Temp folder as temporary files with extension **.smp_**. If you **Save** the data capture, the system saves the whole recorded capture on a hard disk. Saving data replaces the temporary extension **.smp_** with extension **.smp**.

Data Capture Options

The capture and triggering is performed with default Data Capture Options. For explanation on setting the Data Capture Options see page 47.

Transaction ID	Direction	Transaction Type	Start Time	Dev. Addr	Data Length	Handshake	
462	IN	Transaction Type	013.296.379.63	02	0010	5	No Handshake
463	OUT	Transaction Type	013.297.379.35	02	0001	176	No Handshake
464	OUT	Transaction Type	013.297.504.23	00	0001	0	No Handshake
465	OUT	Transaction Type	013.299.379.08	02	0001	176	No Handshake
466	OUT	Transaction Type	013.299.378.82	02	0001	176	No Handshake
467	OUT	Transaction Type	013.300.378.53	02	0001	176	No Handshake
468	OUT	Transaction Type	013.301.378.27	02	0001	176	No Handshake
469	OUT	Transaction Type	013.302.378.00	02	0001	176	No Handshake
470	OUT	Transaction Type	013.303.377.72	02	0001	176	No Handshake
471	OUT	Transaction Type	013.304.377.45	02	0001	176	No Handshake
472	OUT	Transaction Type	013.305.377.18	02	0001	180	No Handshake
473	OUT	Transaction Type	013.306.376.90	02	0001	176	No Handshake
474	OUT	Transaction Type	013.307.376.63	02	0001	176	No Handshake

Figure 14 Captured Results Display

Trigger on Data

Trigger on Data is a useful triggering option when looking for a data stream of up to 8 bytes within one or two consecutive transactions. Selecting Data as a triggering choice offers the following Data triggering choices:

- Packet
- Pattern
- Length

Protocol Analysis

Trigger on a Data Packet

This is the default setting when trigger on Data is selected.

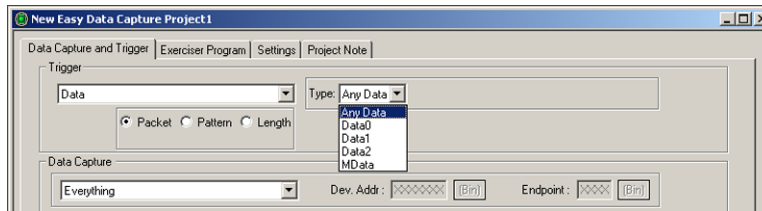


Figure 15 Trigger on a Data Packet Dialog

Choose Packet Type

Click the down arrow next to the Type list box and select a packet type on which to trigger.

Trigger on a Data Pattern

To trigger on a data pattern, check the Pattern button. A pattern can be across up to eight similar transaction types by checking the **Multiple Transaction** check box.

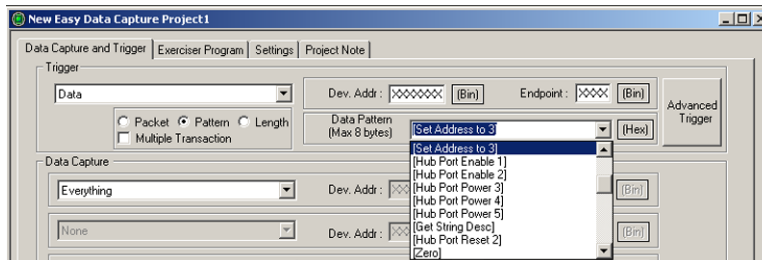


Figure 16 Trigger on Data Pattern Dialog

Click the down arrow next to the **Data Pattern** list box, then select a pre-defined pattern or enter a pattern directly in the list box (eight bytes maximum).

Multiple Transactions

To trigger on a pattern over multiple transactions, check the **Multiple Transaction** and define the transaction types for triggering in the Advanced Trigger dialog.

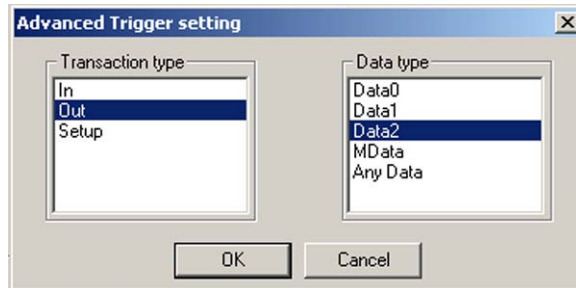


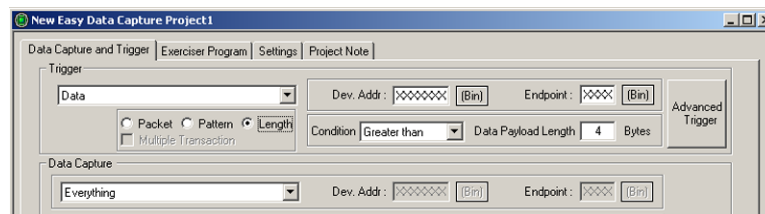
Figure 17 Advanced Trigger Setting Dialog

Set Address

Specify a Device Address and an Endpoint Address to prevent mis-triggering by different devices.

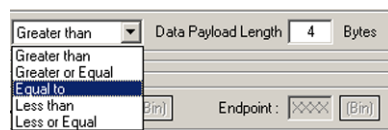
Trigger on Data Length

To trigger on Data Length, check the **Length** button.



Define Pattern Length

Choose a length condition by clicking the down arrow next to the Condition list box. Make a selection and enter a Payload Length value.



Manual Trigger

To perform a manual trigger, select **Manual** from the Trigger drop-down list and choose a Data Capture transaction.

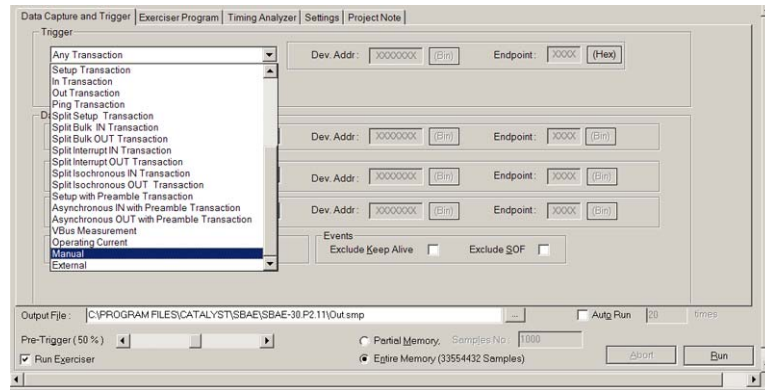
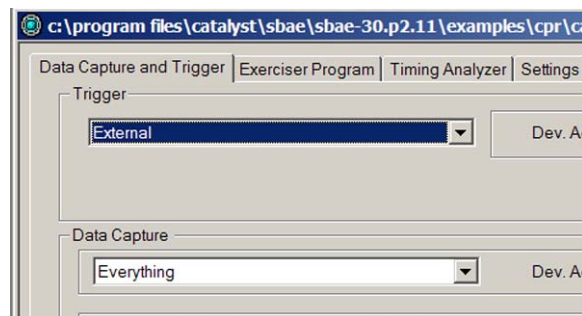


Figure 20 Enabling Manual Trigger

Push the Manual Trigger button on the front panel to trigger the Analyzer.

External Trigger

To trigger on an external signal, select **External** from the Trigger drop-down list box.



Protocol Analysis

To trigger on Protocol Error

Select **Protocol Error** from the Trigger drop-down list. For a definition of protocol errors see page 108.

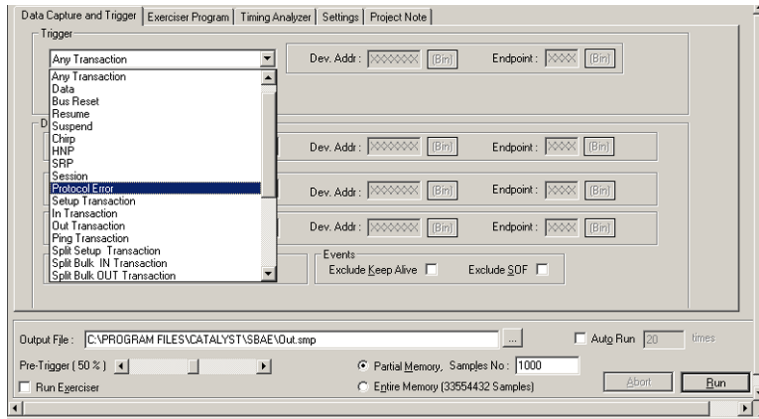


Figure 21 Triggering on Protocol Errors

Exclude Protocol Errors

To exclude specific protocol errors from triggering, define a Protocol Error mask.



To mask protocol errors, click the **Protocol Errors** button on the toolbar to open the Protocol Errors Dialog box.

Uncheck the protocol errors to exclude and click **OK**.

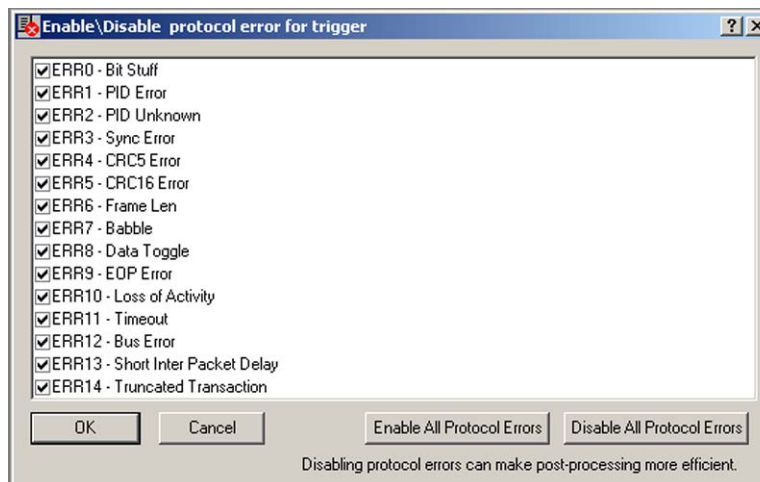


Figure 22 Protocol Error Mask

Table 4 Pre-Defined Trigger Selections

1	Snap Shot (Note 1)	16	Split Setup Transaction*
2	Any Transaction	17	Split Bulk IN Transaction*
3	Data	18	Split Bulk OUT Transaction*
4	Bus Reset	19	Split Interrupt IN Transaction*
5	Resume	20	Split Interrupt OUT Transaction*
6	Suspend	21	Split Isochronous IN Transaction*
7	Chirp*	23	Split Isochronous OUT Transaction*
8	HNP (OTG event)	24	Setup With Preamble Transaction [#]
9	SRP (OTG event)	25	Asynchronous IN With Preamble Transaction [#]
10	Session (OTG event)	26	Asynchronous OUT With Preamble Transaction [#]
11	Protocol Error	27	Timing Patterns (Note 4)
12	Setup Transaction	28	VBus Measurement (Note 2)
13	In Transaction	28	Operating Current (Note 3)
14	Out Transaction	29	Manual
15	Ping Transaction*	30	External
		31	Bus Error (SE1)

Note 1 Snapshot triggers at state 0 allowing you to get an overview of bus activity independently of the occurrence of an event, packet or transaction.

Note 2 To trigger on VBus, the level has to remain at the specified value for 2 µsec with a tolerance of ±10 mV.

Note 3 To trigger on Operating Current, the Operating Current has to remain at the specified value for at least 0.2 µsec with a tolerance of ±1.5 mA.

Note 4 Triggering on Timing Patterns is available only with the Timing Analysis option.

* Not available in Full/Low speed.

Full/Low speed only.

Data Capture Options

Memory Display

You can limit the captured data display to a specific number of samples by checking Partial Memory and entering the number of Samples to capture or, you can check entire memory to allow the capture for the entire memory.

Pre-Trigger

Pre-Trigger is set by default at 50%, which defines the percentage of data to capture before and after the triggering event. You can change this percentage by dragging the slider to a value.

Pre-Trigger Data

The capture of the specified percentage of the data prior to the triggering event cannot be guaranteed and may in some cases be 0. This can occur in cases where the triggering event occurs before the required amount of pre-trigger event data can be stored. In these cases, the data display shows fewer than the specified data points prior to the triggering event. For more detail see **Pre-Trigger** on page 111.

Manual Trigger

Select this option to manually interrupt the data capture based on some external event. Select this option from the Trigger drop down list (see page 44) and start the capture. Data is continually captured to memory and overwritten, until you push the Manual Trigger button on the front panel.

Auto Run

To repeat the current capture and trigger setup automatically, check the **Auto Run** checkbox and enter the number of times in the **Times** text box. The capture and trigger repeat automatically for the specified number of times and the results saved in consecutively numbered **Out.smp** files.

Table 5 Pre-Defined Data Capture Selections

1	Everything	10	Split Bulk OUT Transaction
2	Any Transaction	11	Split Interrupt IN Transaction
3	Custom Transaction	12	Split Interrupt OUT Transaction
4	Setup Transaction	13	Split Isochronous IN Transaction
5	In Transaction	14	Split Isochronous OUT Transaction
6	Out Transaction	15	Setup with Preamble Transaction
7	Ping Transaction	16	Asynchronous IN with Preamble Transaction
8	Split Setup Transaction	17	Asynchronous OUT with Preamble Transaction
9	Split Bulk IN Transaction		

Custom Transaction

This mode allows a quick setup to perform a capture that includes or excludes a specific transaction type.

To set up a custom capture:

1. Select **Custom Transaction** from the **Data Capture** dropdown list.

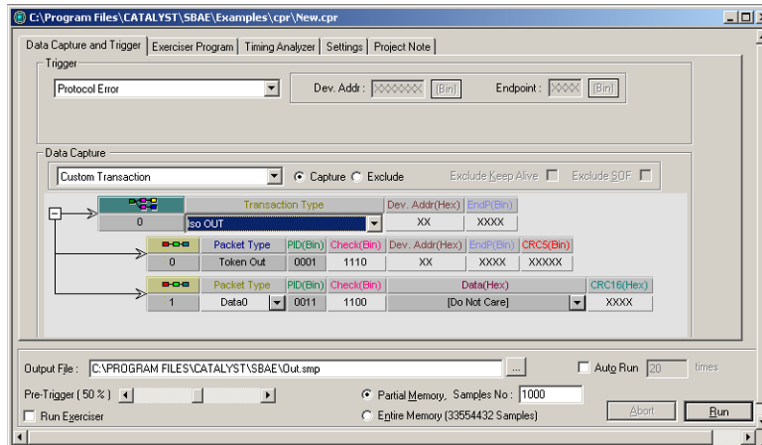
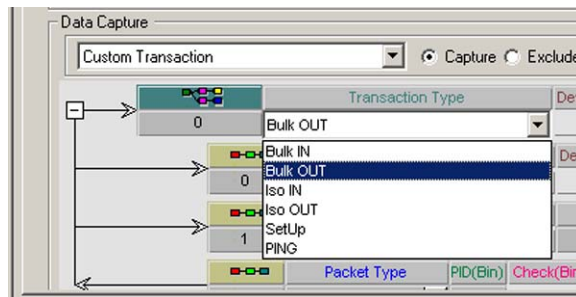


Figure 23 Custom Transaction Dialog

2. Click the down arrow next to the **Transaction Type** drop down list box and choose a transaction.



3. Click either the **Capture** or **Exclude** button to define the capture.
4. You can additionally refine the capture by specifying a **Device** and **Endpoint** address, a packet type and Data.

Exercise and Capture (Optional)

To perform Exercise and Capture you must connect the Exerciser to the port that is used for analysis with the external cable provided. Figure 24 shows a functional Exercise and Capture setup for a High Speed Device.

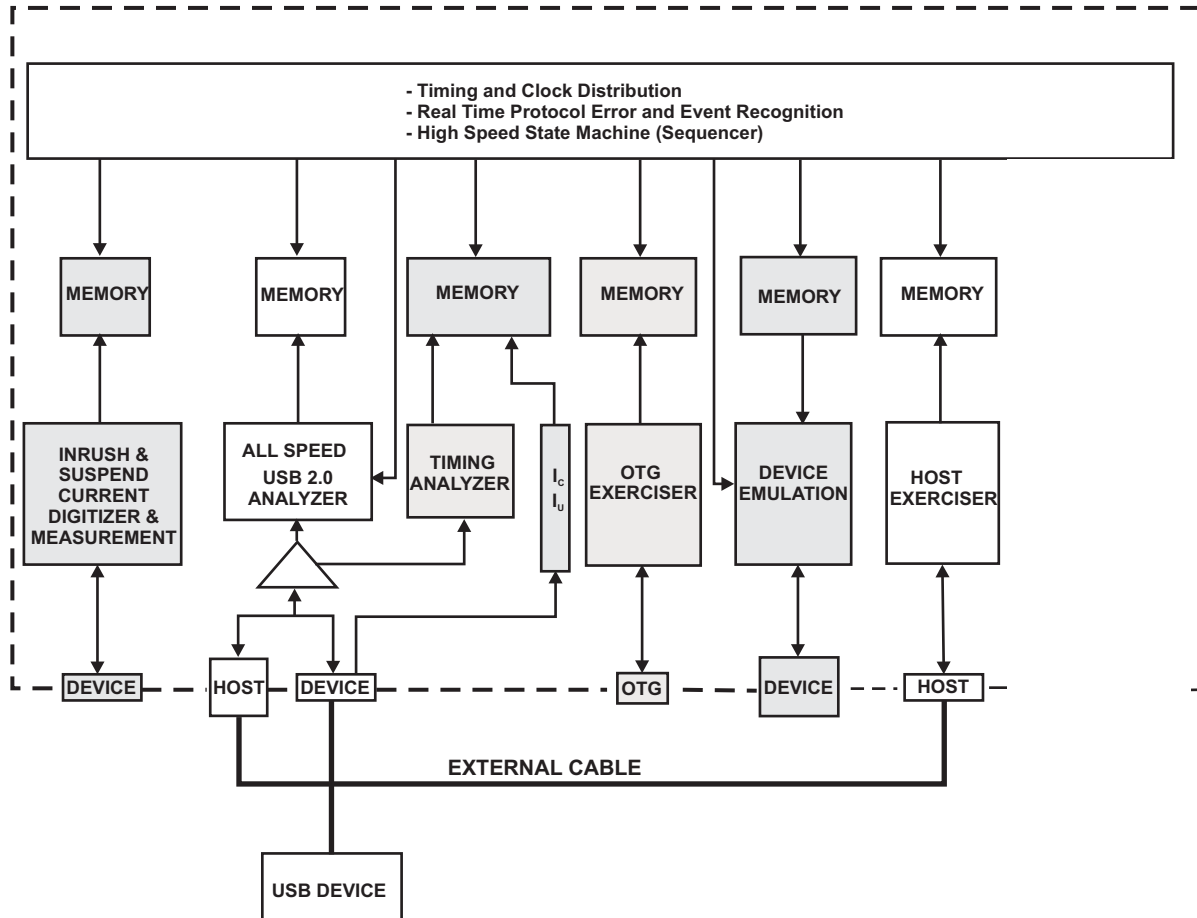


Figure 24 Exercise and Capture Functional Connection

Programming the Exerciser in Easy Mode

Click the **Exerciser Program** tab to open the Exerciser Programming dialog box as shown in Figure 25.

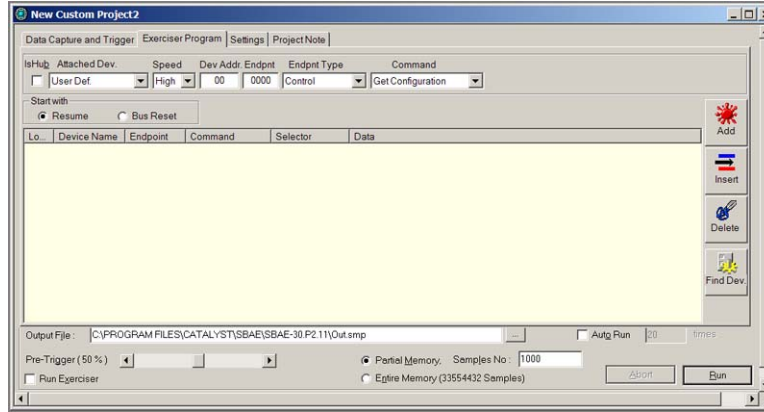


Figure 25 Exerciser Easy Mode Programming Dialog Box

Set up the Device

You must use **Find Device** first to enumerate the device automatically. After enumeration is complete, the system sends SOF or Keep Alive packets to prevent the device for entering a suspended state.



Click **Find Dev.** to command the system to identify the USB Device connected automatically.

Verify that the USB Device is powered up and operational by noting the display of **Detecting First Device ...**, followed by Devices that Conquest detects connected to the Main Analyzer USB port.

After a short time the connected USB Device identification appears in the **Attached Device** list box as shown in Figure 26.

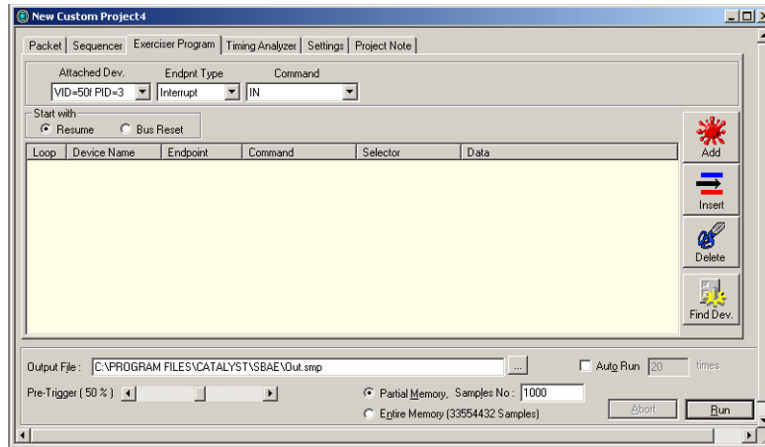


Figure 26 Device Identified

At this point you are ready to create an exerciser program for the identified Device. As an example, choose **Control** as the Endpoint Type, **Get Descriptor** as the command and **Device** as the Descriptor Type from the corresponding list boxes and click **Add**. Repeat for additional program steps.

Data

For commands requiring data blocks, click the down arrow next to the **Data** drop down list box and choose from a set of pre-defined data blocks. If you need a new data block, click the **Settings** tab and then the **Data Blocks** button. See page 54.

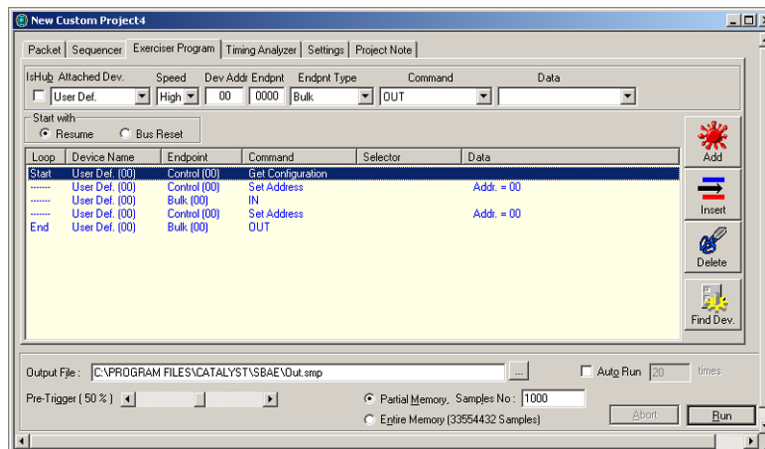


Figure 27 Sample Exerciser Program

Insert Line

You can insert an additional program line anywhere in the program by highlighting an existing program line above which to insert the new line, define the command, and click the **Insert** button.

Protocol Analysis

Define Loops

To define program loops, click under **Loops** in the program line in which to start the loop and click **Start**. Similarly click under **Loops** in the program line in which to end the loop and click **End**. See page 54 for setting the number of times to run the loop.

Enable Exerciser

Make sure that the **Run Exerciser** check box is checked.

Choose Trigger and Capture Options

Click the **Data Capture and Trigger** tab, then choose a Trigger and Data Capture option or options to analyze the traffic generated by the exerciser.

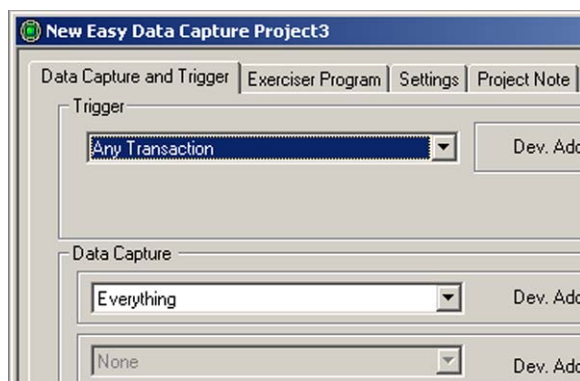


Figure 28 Sample Trigger and Data Capture Selection

Click **Run** to exercise the Device and wait for the captured results to display, as shown in Figure 29.

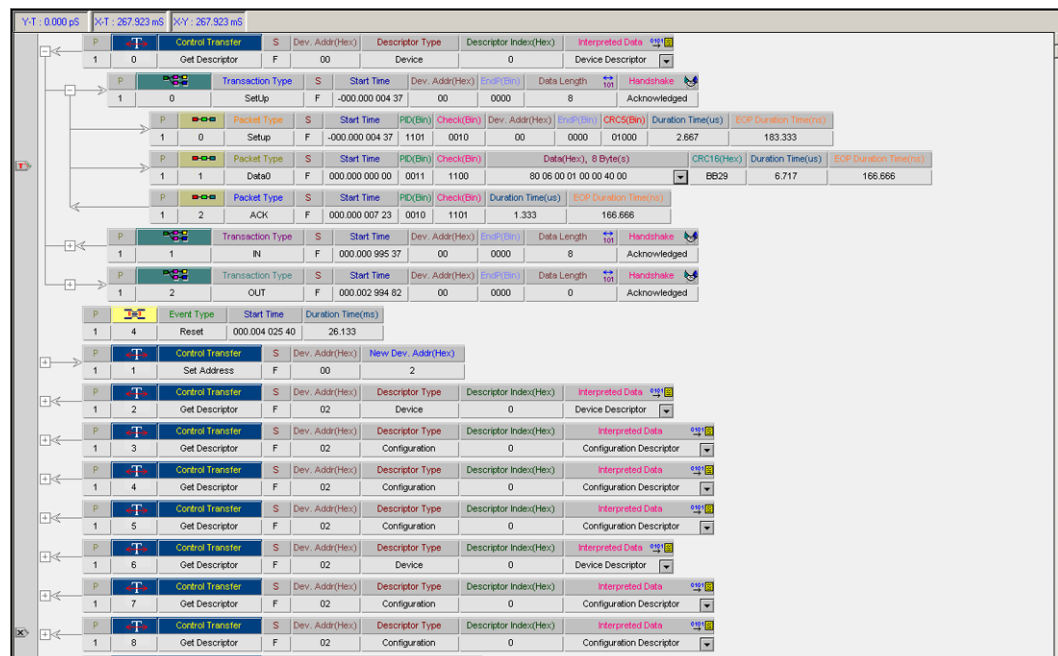


Figure 29 Captured Data Display

Settings

Click the **Settings** tab to open the Project settings dialog box as shown in Figure 30.

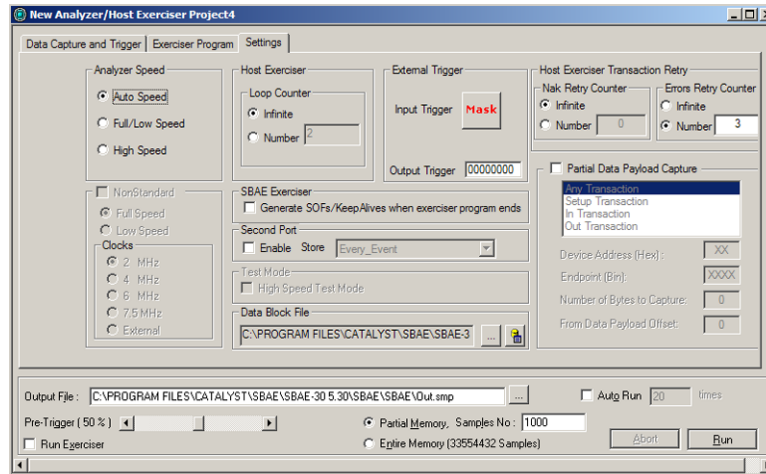


Figure 30 Project Settings Dialog Box

Analyzer Speed

The default is Auto Speed for automatic speed detection, however, you can force the speed by choosing Full/Low or High for special testing needs.

Looping

You can specify the number of times to loop the command line(s) or you can run them continually. You can set the number of times from 2 to a maximum of 1,048,575 times.

Clock


To specify a clock speed or an external clock, check the **Full/Low Speed** or **High Speed** option buttons and the **NonStandard** check box. Then make a **Clocks** choice. The available choices are four pre-set clock speeds and an External clock, which must be connected to the Analyzer as described on page 8.

Generate SOFs/Keep Alives when exerciser program ends

Check this box to keep device from suspending.

Data Blocks



If you require a new data block, click the  button next to the Data Block File dialog to open the data block definition dialog. Define a new data block as described on page 90.

Protocol Analysis

High Speed Test Mode

When a High Speed USB Device is set into a **Test_Packet** mode it must respond with a standard **Data0** packet with pre-defined data. Selecting this option allows capture of this packet.

High Speed Test Mode Usage

1. Set **Test Mode** on the Device (the exerciser can do this).
2. On the Settings tab, set Analyzer Speed to **High Speed** instead of Autospeed and select the **High Speed Test Mode**.
3. Press **Run** to capture the test packet.

Trigger Mask

If you selected an **External Trigger** as described on page 44, click the Trigger Mask button in the Project Settings dialog repeatedly to set a trigger option.



To trigger on a “0” level.



To trigger on a “1” level



To trigger on a rising edge.



To trigger on a falling edge.

External Bits

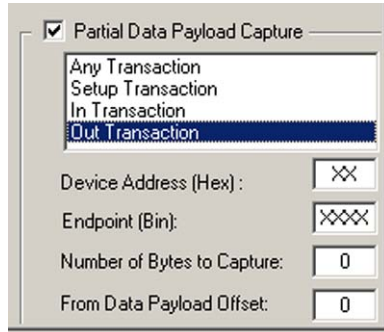
To output a data pattern whenever a trigger occurs, enter a pattern in the **External_Bits** text box in Easy Mode and in the Sequencer when operating in the Advanced Mode. To access the data pattern as external signals see “Conquest M2 External I/O Connector Pin Assignment” on page 15.

Transaction Retry

The settings dialog default is per USB specification, however, you can enter non standard conditions for the application.

Partial Data Payload Capture

To refine the capture check the **Partial Data Payload Capture** check box, choose a transaction type.



You can further enter a Device and Endpoint address, set a limit of the number of bytes to capture or to capture from an offset.

Protocol Analysis

Auxiliary Port

To perform Hub testing using the Auxiliary port, connect as shown in Figure 31.

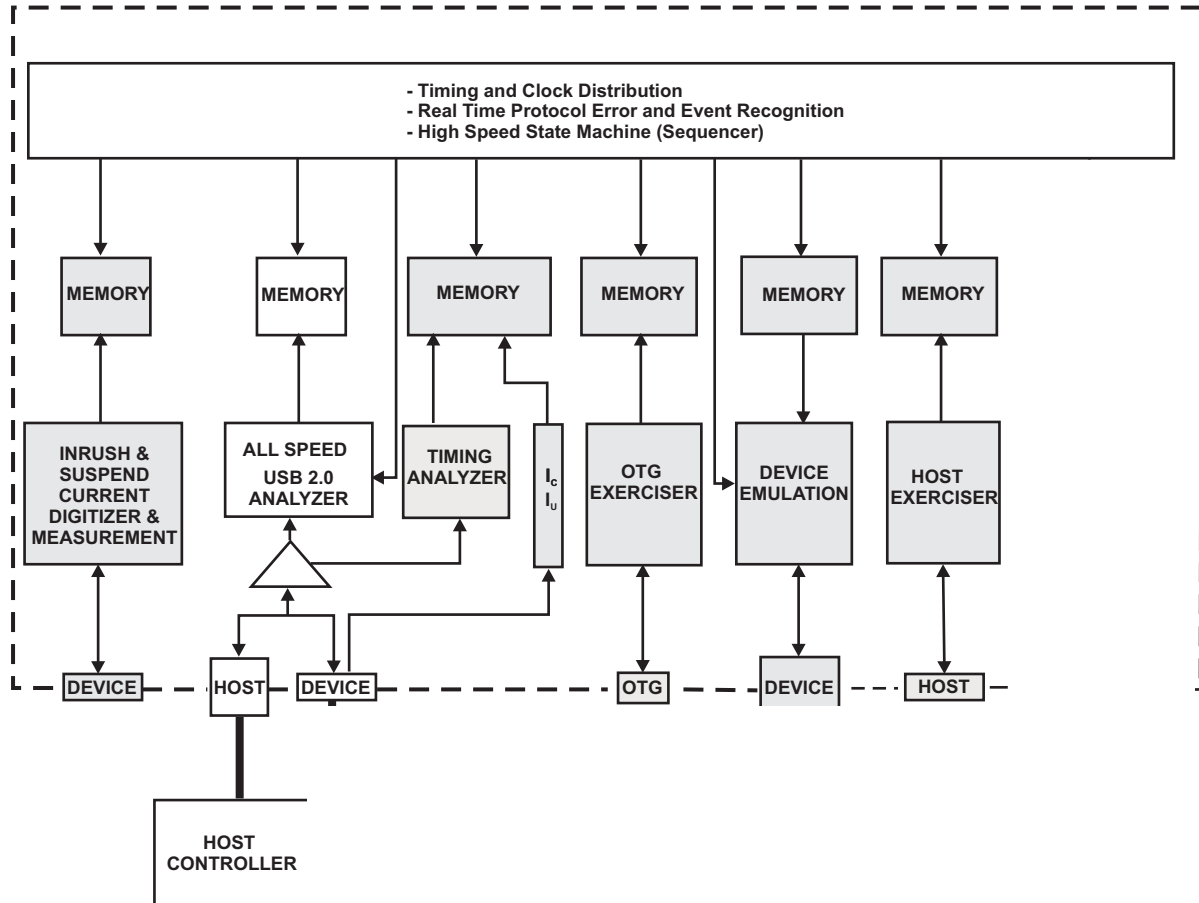


Figure 31 Auxiliary Connector in Hub Testing

Auxiliary Port Data Capture

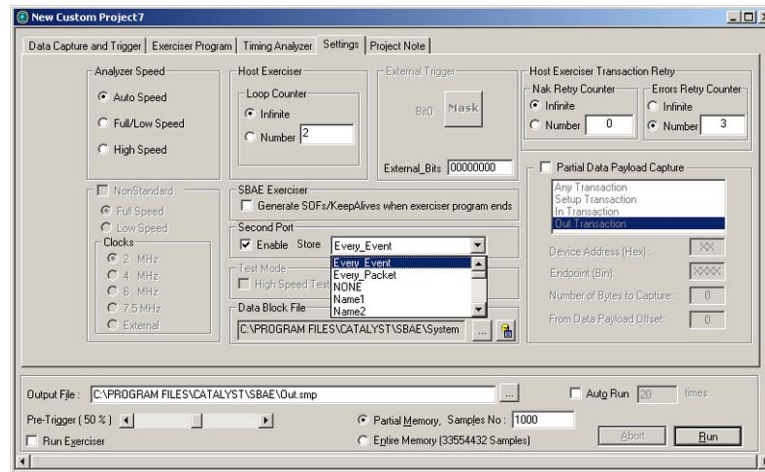


Figure 32 Auxiliary Port Data Capture Selection

Analyzer Only

When used for Hub testing, the Conquest M2 Auxiliary Port performs as an Analyzer only with pre-defined data capture options and has no triggering capability. Triggering is limited to the Main Analyzer Port.

Project Note

To enter a note about the current project, click the **Project Note** tab and enter the data to associate with this project.

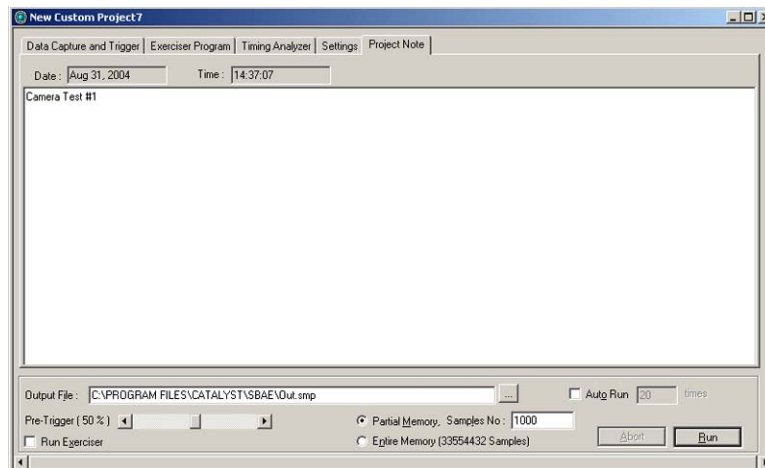
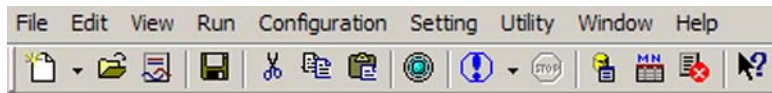


Figure 33 Easy Project Note

Advanced Mode (User-Defined)

This mode expands Analyzer capability by allowing you to program complex triggering and data capture projects. Such projects are programmed by defining USB transaction packets and then including them for data manipulation and trigger in a sequencer. Additionally, you have the capability to exercise the bus with pre-determined data by programming and invoking the exerciser.

Capture Data Project



Make sure that the configuration is set to use Advanced Analyzer. See “Creating Projects” on page 30.

Last Project

Clicking the **Green** button opens the last project saved.

New Project

To start a **New** project, select **File > New > Analyzer/Host Exerciser Project**.

For a custom project:

- Define a set of packets
- Program the Sequencer
- Program the Exerciser (Optional)

Note: To perform data capture and triggering with exerciser generated data you must use the Exerciser Port as shown in Figure 3.

Defining Packets

You can define up to eight different packets. To define a packet, click the **Packet** tab to display the packet definition dialog on top.

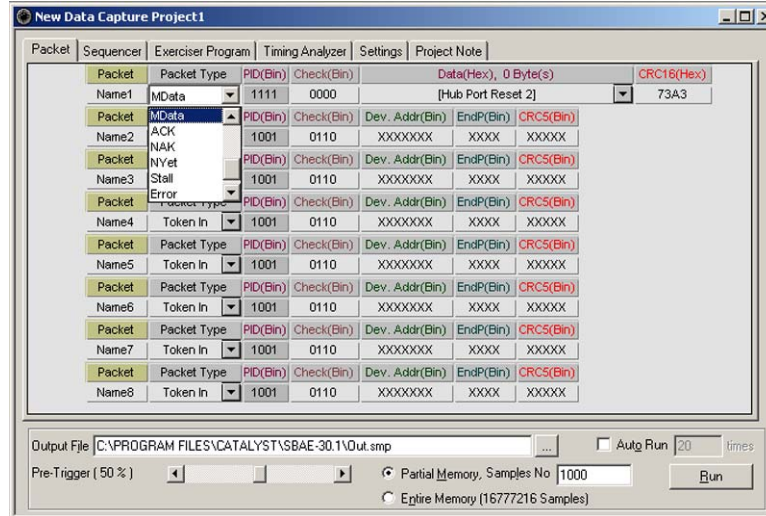


Figure 34 New Project Dialog Box

Protocol Analysis

1. Click the down arrow in the **Packet Type** list box and choose a packet type from the following choices:

Packet Types¹

SOF	Start of frame (Full Speed) or start of microframe (High Speed)
SPLIT	(Token) Defines split transaction type
IN	(Token) Requests information from a Device
OUT	(Token) Designates a data transfer to a device
SETUP	(Token) Indicates the start of a control transfer
PING	(Token) High-speed flow control probe for a bulk/control endpoint
DATA0	Data Packet PID even
DATA1	Data Packet PID odd
DATA2	Data Packet PID for High Speed, High bandwidth, Isochronous transactions
MDATA	Data Packet PID for High Speed Split or High bandwidth Isochronous transactions
ACK	Target received data without error
NAK	Target unable to accept data
NYET	No response yet from receiver
STALL	Indicates error preventing data transfer
PRE	(Token) Preamble, enables low speed port (Full/Low speed mode only)
ERR	Split transaction error handshake (High speed mode only)
	User-defined Arbitrary set of patterns that you define

2. Complete the enabled packet definition parameters for the selected packet Type
 - **Check:** Default value that you can change to search for other check values on the bus
 - **Device Address:** The address of the token packet to capture (0-7F)
 - **Endpoint:** The end point of the token packet (0-15) (Entered in binary)
 - **CRC5:** Default value that you can change to search for other CRC5 values on the bus. (5-bit CRC used by packets other than data)
 - **CRC16:** Default value that you can change to search for other CRC16 values on the bus. (16-bit CRC used by data packets)

1. For a detailed description of USB packets, see Universal Serial Bus Specification, Revision 2.0, April 27, 2000

Packet Payload Data

For packet types that include payload data, Conquest M2 offers a set of pre-defined or user-defined data blocks.

Data

Enter data directly into the data edit box, define a data block by clicking the **Data Blocks...** button, or open a previously created data block file. For detailed instructions on creating Data Blocks, see “Creating a Data Block” on page 90.

User Defined Dialog

When you choose a “User Defined” packet type, the following dialog allows you to define an arbitrary set of patterns by checking available choices.

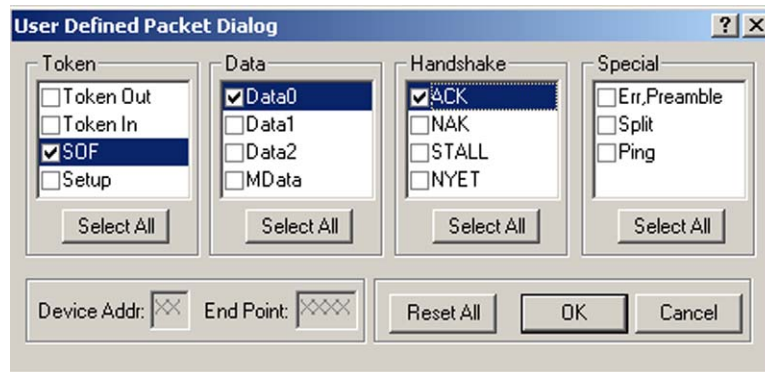


Figure 35 User Defined Packet Dialog Box

Protocol Analysis

Overlap Warning

In the event that you make user-defined packet choices that result in overlaps, you get the following warning message.



Figure 36 Overlap Warning

If you proceed with overlapping selections, you cannot edit the results, because of “don’t care” entries. For example, combining 1101 with 1001 results in 1X01.

The Sequencer

The Sequencer is for data capture manipulation, generating complex triggering, and starting the exerciser. Sequencer operation at each state controls the Analyzer capture functions in response to specified bus activity.

The USB Analyzer Sequencer allows you to define up to 32 states. The Sequencer always starts at “State0”. You can program each state to go to any other state, depending upon the occurrence of packets or events in that state. Jump to any state is conditional and the jump may occur to several other possible states depending on which of the specified conditions is met first.

You can set a trigger in the Sequencer, to occur at any state, on:

- Every packet (unconditional)
- Every event (unconditional)
- Occurrence of any packet or a boolean expressions of packets.
- Occurrence of Bus_Idle, Resume, Bus_Reset, Disconnect, Keep Alive, Suspend, Chirp, Session, SRP, Connect, HNP or Protocol Error

Sequencer Operation Overview

Whenever a project runs, Conquest starts monitoring data on the bus and starts the sequencer in “State0”. You can program “State0” to **start capture immediately** (Default) or to **wait for a specific occurrence** on the bus. You can select what to capture:

- Capture all events
- Capture all Packets
- Capture a specific event or events
- Capture a specific packet or packets

If a user-selected event, packet, or logical combination of these occurs, the sequencer transitions to another state you selected.

You can program each state to output an external bit pattern, set a trigger, and change what is to capture. If the capture involves the use of the exerciser you can only start it in “State0”.

Protocol Analysis

Programming the Sequencer

To program the Sequencer, click the **Sequencer** tab to display the sequencer programming dialog as shown in Figure 37.

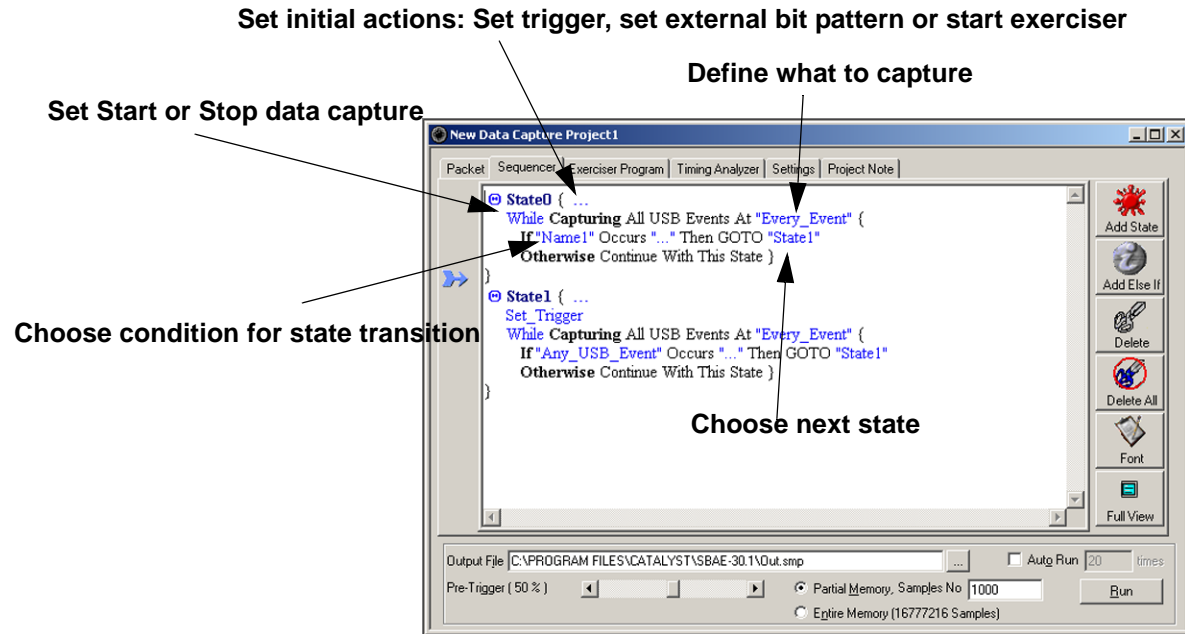


Figure 37 Sequencer Programming Dialog

The sequencer programming dialog opens with default settings and all of the programmable parameters displayed in blue.

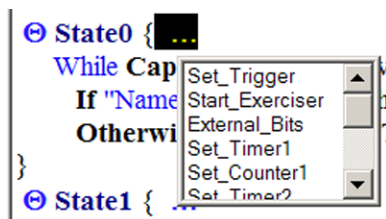


Click the **Add State** button to add additional states, up to 32.

Program State0

In "State0" you choose, if you want to set a trigger, set an external bit pattern, start the exerciser, when to start a data capture, what to capture and when to transition to another state

1. To set the initial sequencer actions, click the ellipses next to "**State0**"



2. Choose one or all of the available choices one at a time.

Counters

The sequencer incorporates two counters each of which you can configure as a timer or an event counter, with the following commands:

- Set_Timer1
- Set_Counter1
- Enable_Timer1
- Enable_Counter1
- Set_Timer2
- Set_Counter2
- Enable_Timer2
- Enable_Counter2

Set the counters with the Set_Timer and Set_Counter commands, with corresponding timer or counter values in any state.

```

Ⓞ State1 { ...
Set_Trigger & Set_Counter1 To : "99" Count: "Name1"
While Capturing All USB Events At "Every_Event" {
If "Any_USB_Event" Occurs "... " Then GOTO "State1"
Otherwise Continue With This State }
}

```

Whenever a state transition occurs, the timer or counter stops, unless an Enable_Timer or Enable_Counter command is set in the next state.

```

Ⓞ State0 {}
Ⓞ State1 { ...
Set_Trigger & Set_Counter1 To : "99" Count: "Name1"
While Capturing All USB Events At "Every_Event" {
If "Any_USB_Event" Occurs "... " Then GOTO "State1"
Otherwise Continue With This State }
}
Ⓞ State2 { ...
Enable_Timer1
While Capturing All USB Events At "Every_Event" {
If "Any_USB_Event" Occurs "... " Then GOTO "State2"
Otherwise Continue With This State }
}

```

Start Exerciser

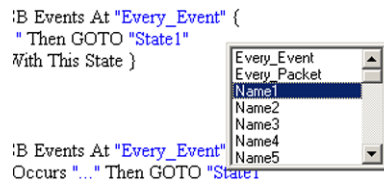
You can start the exerciser at “State0” only. To start the exerciser, click the ellipses next to “State0”. Note, however, that you must program the exerciser prior to running the project to get meaningful results. For instructions on programming the exerciser, see page 71.

External Outputs

You can set the eight external output bits on the output connector at any state by clicking the ellipses next to the state, choosing **External_Bits** from the drop-down list for that state, and then entering a bit pattern. See page 15 for external output connector pin assignment.

Protocol Analysis

To define what to capture, click **Every_Event** to display the capture choices.

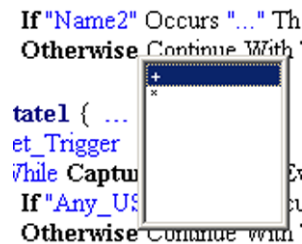


Every_Event is the default, however, you can choose every packet, a specific packet or event or create a boolean expression from the available choices.

Create Boolean Expression

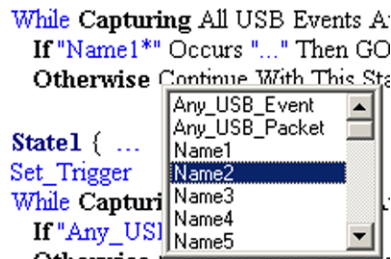
You can create a boolean expression by clicking the left mouse button immediately after entering the first parameter, to open the boolean list box.

1. Choose a boolean operator and then select the next parameter in the expression resulting e.g. Name2+Name3.



The operator convention used is + for "or" and * for "and".

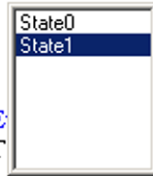
2. To transition to another state, click the blue **Name1** to display the transition choices.



Choose a packet or bus event or create a logical expression. When the bus detects this choice, the sequencer transitions to another state.

- Choose the next state by clicking the blue **State1** to display the available states, then choose the next state.

```
3 Events At "Every_Event" {
  Then GOTO "State1"
  With This State }
```

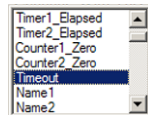


```
3 Events At "Every_E
)ccurs "..." Then GOT
With This State }
```

- Conquest starts capturing all bus activity at start by default. To delay the start of capture until some occurrence, click the blue **While** button to toggle it to Stop.

```
State0 { ...
  Set_Trigger
  Stop Capturing All USB Events {
    If "Name1" Occurs "..." Then GC
    Otherwise Continue With This S
  }
}
State1 { ...
```

- Define the operation of each additional state similarly.
- If you are testing for receipt of an expected packet, such as Data or Handshake, select **Timeout**. This is equivalent to a Truncated Transaction protocol error, PE 14.



- By clicking the ellipses after If "Packet" Occurs"... " and choosing either **Tag_Event** or **Discard**, you can identify a block of data to discard. Setting **Tag_Event** tags the start of a data discard. Setting **Discard** designates the point up to which to discard the data. Discarded data is between the last Tag_Event set and the Discard set.



Protocol Analysis

Add Else If

You can include Else If statements in your program, to allow transition to additional states.



Click the **Add Else If** button to add an else if line to the program.

To remove the added Else If, double-click it and choose **Delete** from the open list box.

To Delete a Parameter

Highlight it and push the **Delete** key on the keyboard.

Optimized Workspace

To avoid scrolling the display in long sequencer programs, you can minimize completed states by clicking the symbol next to the state to minimize. To expand the minimized state, click the state name. Alternatively, click the state name.



You can maximize the workspace to full screen by clicking the **Full View** button.

To delete a State

To delete a state, place the cursor in the text for that state and click the **Delete** button.

Note: You cannot delete a state if it is used in another state.

Manual Trigger

To use the Manual Trigger button on the front panel, remove all "Set Trigger" commands from the sequencer program.

Protocol Errors

You can set the sequencer to trigger on protocol errors at any state by clicking **Packet** and then choosing **Protocol_Error**. To exclude some protocol errors, set the protocol mask as described in the following section.



For a description of the detected protocol errors, see page 108.

Setting a Protocol Error Mask



To mask protocol errors, click the **Protocol Errors** button on the toolbar to open the Protocol Errors Dialog box. Uncheck the protocol errors to exclude and click **OK**.

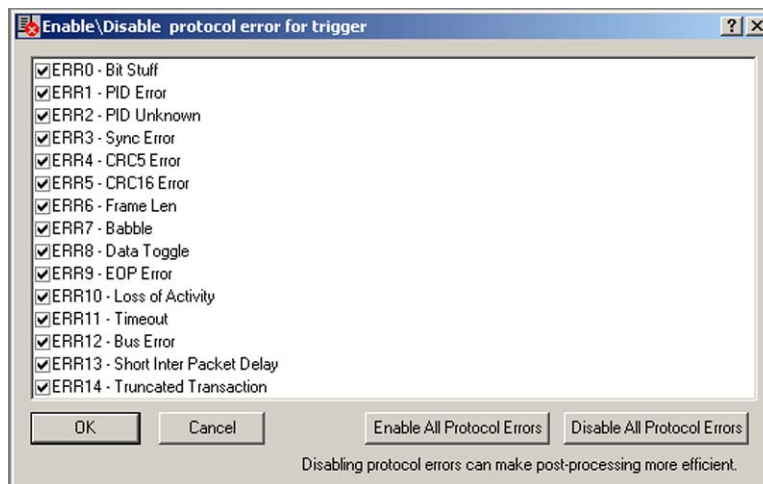


Figure 38 Protocol Error Mask

Host Exerciser (Optional)

Note: To perform data capture and triggering with exerciser generated data, you must use the Exerciser Port as shown in Figure 3.

Programming the Exerciser In Advanced Mode

To program the exerciser, click the **Exerciser Program** tab to open the exerciser programming dialog box, as shown in Figure 39

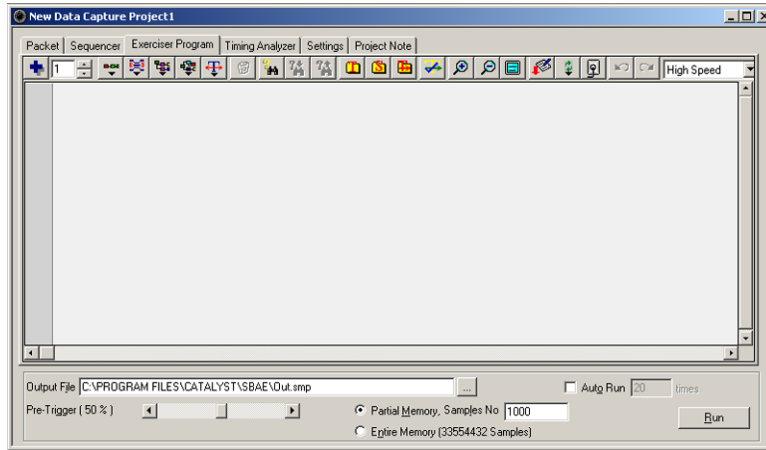


Figure 39 Exerciser Programming Dialog Box

Set the Host Speed First!

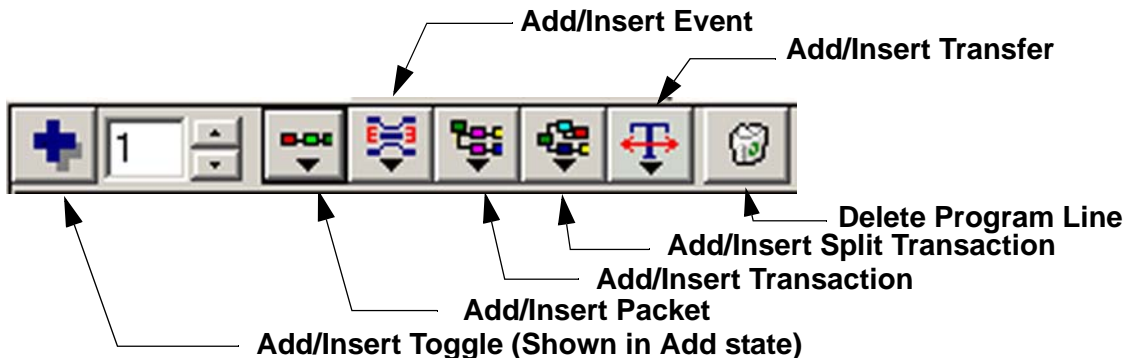
Click the down arrow for the speed selection drop-down box and choose an operating speed.

Maximum Program Lines

You can define up to 999 exerciser program lines. Each line is an event or packet transmitted on the bus.

Add Program Lines

To add program lines, enter the number of lines to add, select the type of program lines to add by clicking the appropriate category button from the choices shown below, and then select the line type.



Insert Program Lines

You can insert additional program lines by positioning the cursor on an existing line. Click **Add/Insert** to toggle to the insert state, enter the number of lines to insert, and click the appropriate line-type button and select the line type to insert. The specified number of lines inserts above the line where you positioned the cursor.

Delete Program Lines

To delete a program line, position the cursor in the line to remove and click the **Delete** button. You can also select and highlight multiple lines for deletion.

Programming with Transfers

Transfers are a powerful way to create an exerciser program that automates time consuming tasks such as generating several transactions that make up a transfer.

Start with Bus Reset

Insert a Reset followed by a SOF packet in the start of the program to assure that the timing and frame number are automatically set, allowing you to insert your own transactions between the SOF packets.

To add a Bus Reset Event, click the **Add/Insert Event** button and choose **Reset** from the drop-down list.

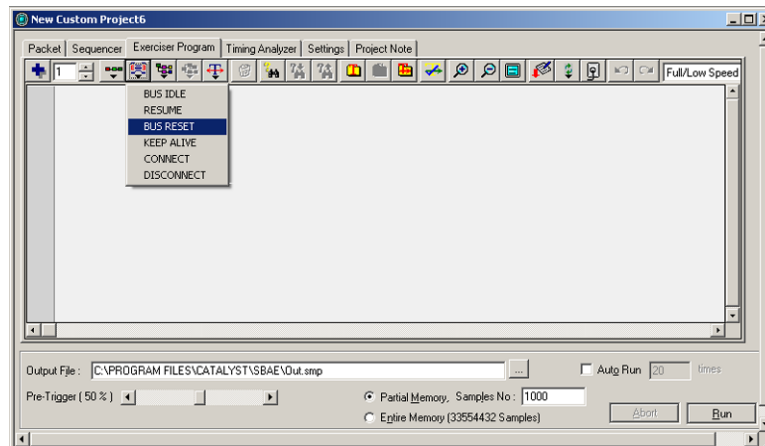


Figure 40 Adding a Bus Reset

Protocol Analysis

Add a Control Transfer(s)

To add a control transfer line, click the **Add/Insert Transfer** button and choose **Control** from the drop-down list.

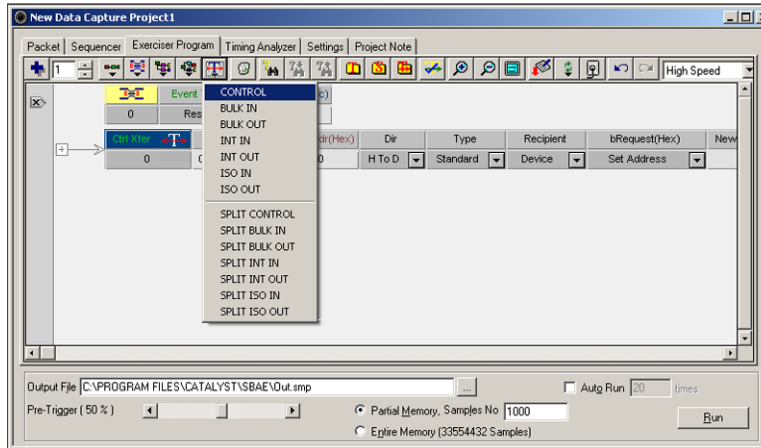


Figure 41 Adding a Control Transfer in High Speed Protocol

Control Transfers

Control Transfers transfer specific requests and commands to a Standard, Class or Vendor Device, Interface or Endpoint. Other user-defined setups are available. The most commonly used is the selection of a request or command for a Standard Device. These selections are made on the inserted Transfer line. See Figure 44.

Transfer Type

The default transfer type is **Standard**. However, you can choose Class, Vendor, or user-defined by clicking the down arrow under Type and choosing from the drop-down list.

To choose a Class, click the down arrow under Type and select **Class** to open the select class dialog.



Figure 42 Choosing Class



Figure 43 Select Class Dialog

Click the down arrow for Select class name, choose a pre-defined class, click **OK**, then click in the blank program entry area to register the choice.

Note that, once you register, all parameters for the transfer line are updated automatically for the request or command selected.

Specify a New Class

You can specify a new class by clicking **Classes** in the Select Class Dialog and proceed as described on page 77.

To choose a request or command, click the down arrow under **bRequest(Hex)**, choose a request or command, and click the blank program entry area to register. **Note:** After you register, all parameters for the transfer line are updated automatically for the request or command selected.

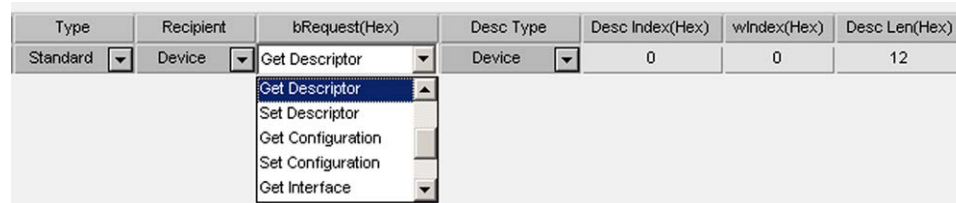


Figure 44 Choosing a Request or Command

Enter and set up additional Control Transfers, for example, Set Address to set up exercising session.

Protocol Analysis

Non Control Transfers

Non Control transfers provide for writing data to and reading data from a Device, Interface, or Endpoint. Non Control transfers are provided for Bulk, Interrupt, or Isochronous data transferring.

Add a Data Transfer(s)

Click the **Add/Insert Transfer** button and choose a data transfer from the drop-down list, for example, Bulk Out.

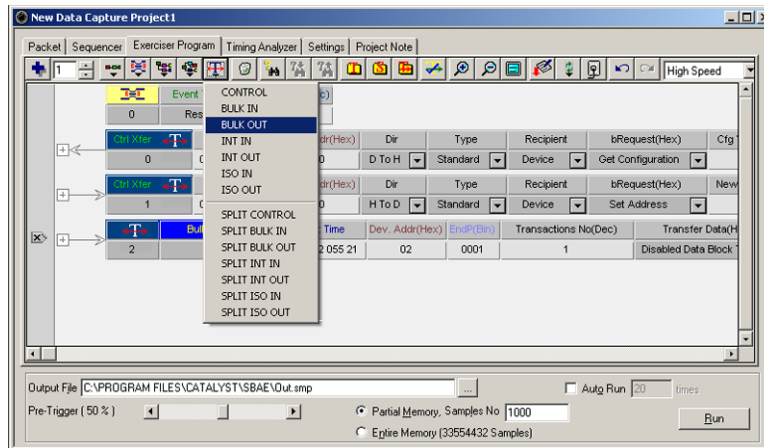


Figure 45 Adding a Data Transfer in High Speed Protocol

Set up Data Transfer

For each data transfer inserted, you must specify the appropriate Device and Endpoint addresses. Additionally, for Out Transfers, you can specify the data to transfer, as a pre-defined data block, manually entered data at the transfer level, or directly entered at the transaction level.

Dev. Addr(Hex)	EndP(Hex)	Transactions No(Dec)	Transfer Data(Hex)
02	1	1	Disabled Data Block Transfer

Figure 46 Transfer Setup Fields

Specify Endpoint number

To specify an endpoint number for the current data transfer, highlight the number (default 1) under **EndP(Hex)** and enter an endpoint number.

Note: The example shows data entry in Hex. However, you can right-click the **EndP(Hex)**, choose **Format**, and change the entry format to Binary or Decimal.

Define Endpoint Settings



1. Click the **Define/Modify endpoint settings** button to open the endpoint setting dialog.

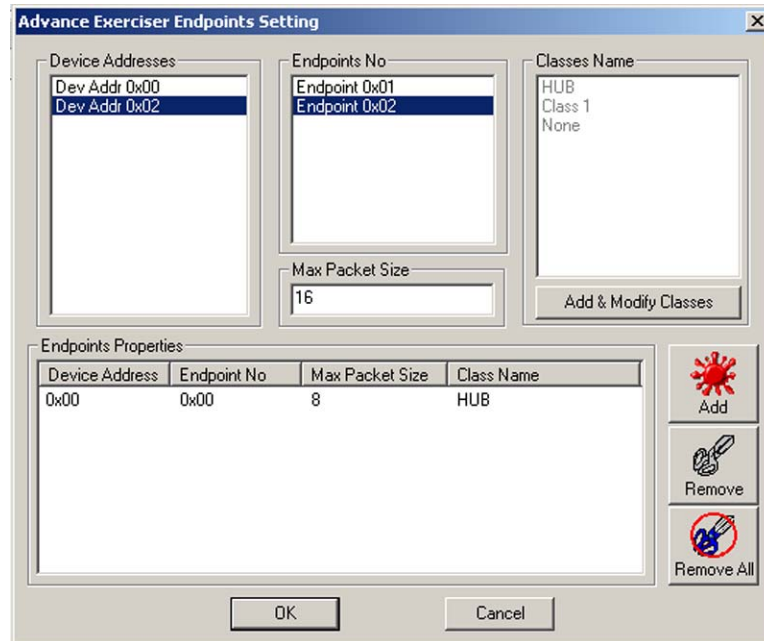


Figure 47 Endpoint Setting Dialog

2. Highlight the **Device Address**, highlight each endpoint, enter a packet size value in the **Max Packet Size** text box (for a value other than the default 8), and click the **Add** button until all endpoints appear in the **Endpoint Properties** box.
3. Click **OK**.

Specifying a New Class

1. Click the **Classes** button on the Select Class dialog to open the Class & Class Requests Definition Dialog.

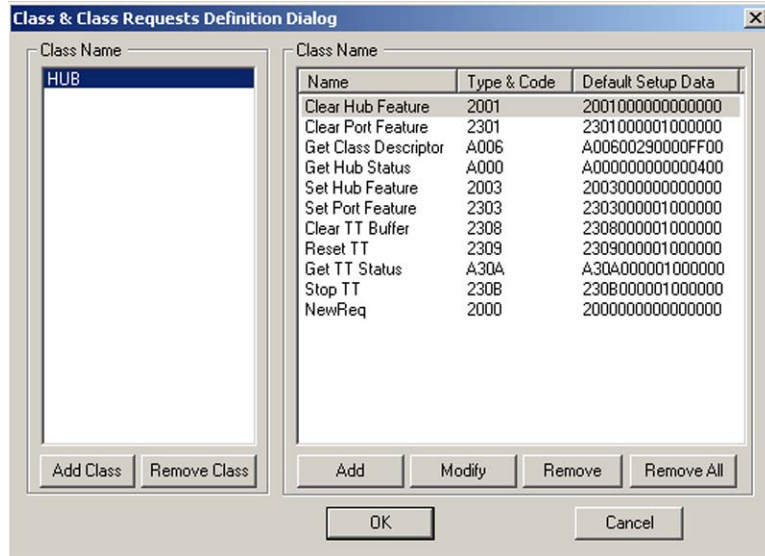


Figure 48 Class & Class Requests Definition Dialog

2. Click the **Add Class** button, enter the new class name in the Enter Class Name dialog, and click **OK**.

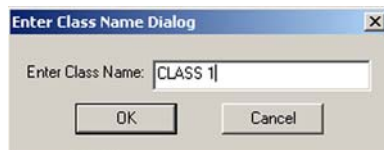


Figure 49 Enter Class Name Dialog

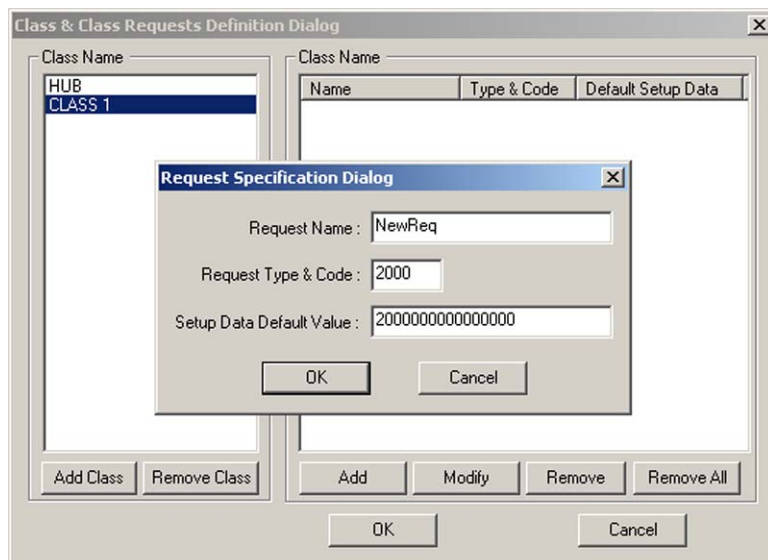


Figure 50 Defining Requests for a New Class

3. Highlight the new class name and click the **Add** button to open the Request Specification dialog.
4. Enter a request name, request type code and a data default value, and click **OK**.
5. Repeat for all requests for the new class.
6. To modify a previously entered request, click the **Modify** button, enter the new parameters in the Request Specification dialog, and click **OK**.

Specifying Data for Transfer

You can define data for transfer at the Transfer layer or at the Transaction layer.

Specify data in Transfer layer

Click **Transfer Data(Hex)** to enable the Block transfer field. Either click the down arrow next to this field and choose a pre-defined data block from the drop-down list, or highlight the field and type specific data that to use in this transfer.

Note: The entry format is selectable as described for **EndP(Hex)**

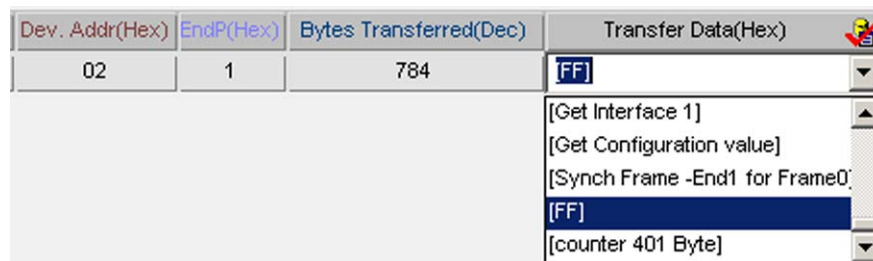


Figure 51 Choosing Data to Transfer

Note Important: A significant advantage of this method of Data Transfer setup is that the Exerciser program automatically generates the number of transactions, based on maximum packet size, required to transfer the data block. This is especially useful for large data blocks requiring perhaps hundreds of transactions.

When the Endpoint number and the data to transfer has been specified, click the blank program entry area to register the choice. Once the choice is registered, the **Bytes Transferred(Dec)** field displays the number of data out Transactions created.

You can add additional transfers.

Protocol Analysis

Specify data in Transaction layers

Transfers specified in Transaction layers are a flexible way to set up a Transfer with different data patterns in each transaction for that transfer.

1. Insert a **Transfer type**, click **Transactions No**, and enter a number of transactions. (The following example shows three entered).
2. Expand the transfer to display the created transactions.
3. Expand each transaction and enter data to transfer by that transaction.

Note: Data entered this way is limited to a maximum of 1024 bytes.

4. Specify the end point in the same way as described for byte transfers.

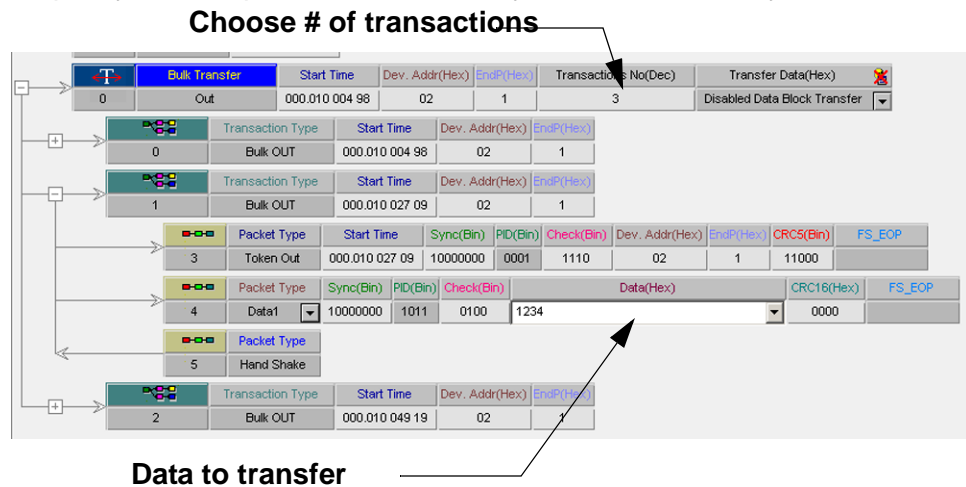


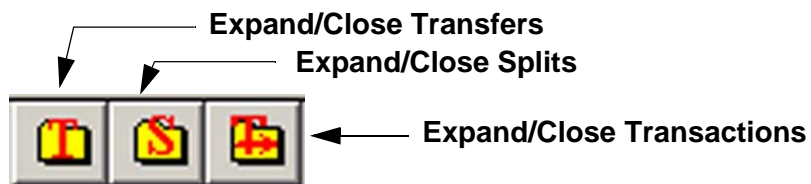
Figure 52 Transfer With 3 Transactions

Create a data block

For commands that involve data transfers, click the **Data Block** button and define a data block. See page 90 for how to create data blocks.

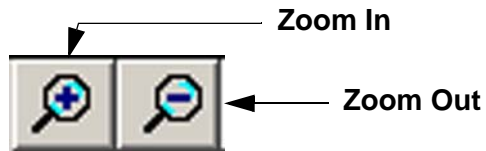
Expand/Close

You can quickly expand/close Splits or Transfers by clicking the corresponding button on the toolbar.



Zoom In/Zoom Out

Clicking a **Zoom** button on the toolbar optimizes the size of the displayed code lines.



Forcing Errors

You can introduce errors for any field of a packet by highlighting the field to change and making the change directly in the field. Bit Stuff error (unchecked has Bit Stuff) and force no EOP (unchecked has EOP) are available by right-clicking the packet.

Program Loops

Program loops allow you to choose a series of program steps to repeat when the program executes. A loop is defined by a start instruction and an end instruction.

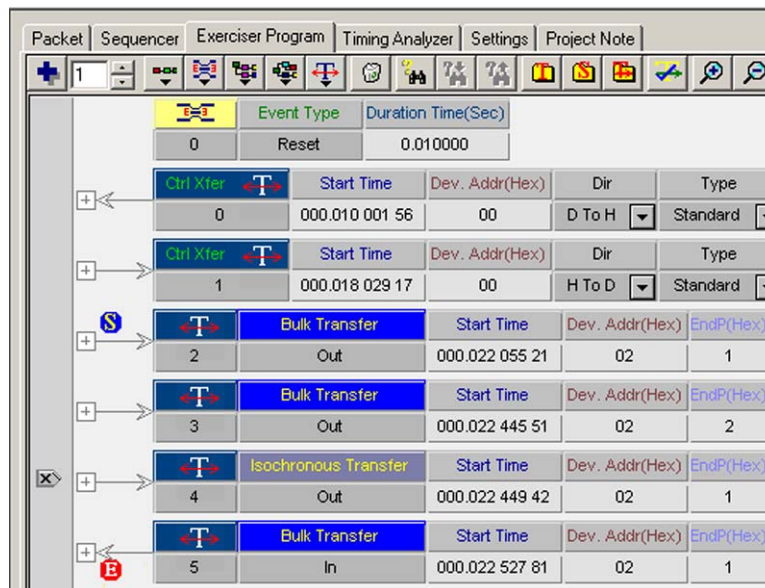
Protocol Analysis

Set up Loops

To set up a loop in the exerciser program, right-click next to the line to start the loop and choose **Start Loop**. Similarly right-click next to the line to end the loop and choose **End Loop**.



Figure 53 illustrates a programmed loop.



The screenshot shows the Exerciser Program interface with a table of events. The table has columns for Event Type and Duration Time (Sec). The events are as follows:

Event Type	Duration Time (Sec)
0 Reset	0.010000
Ctrl Xfer	Start Time
0	000.010 001 56
Ctrl Xfer	Start Time
1	000.018 029 17
Bulk Transfer	Start Time
2 Out	000.022 055 21
Bulk Transfer	Start Time
3 Out	000.022 445 51
Isochronous Transfer	Start Time
4 Out	000.022 449 42
Bulk Transfer	Start Time
5 In	000.022 527 81

Figure 53 Loop Start and End Set

Deleting Loops

To delete a loop, right-click the Start Loop or End Loop symbol and click the appropriate **S**tart Loop or **E**nd Loop in the menu to remove the check marks.

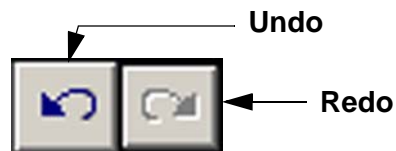


Loop count

Loop count is the number of times to execute a loop. You can set the loop to execute a fixed number of times or to run continually, on the **Settings** tab, as described on page 96.

Undo/Redo

You can exercise up to 200 levels of Undo/Redo steps in an exerciser program by clicking the corresponding button on the toolbar.



Reduced Bit Width

Full/Low Speed Mode Only

The reduced bit width feature is available for the Full/Low speed mode only.

Adjustable bit width of 40, 60, 80, and 100% is allowed for all packet fields, except data, for Device response testing. You can reduce bit width for all bits in a packet or individual bits in a packet.

To set the Bit Width

Right-click in a packet field and choose **Bit Width** to open the Set Bit Width dialog.

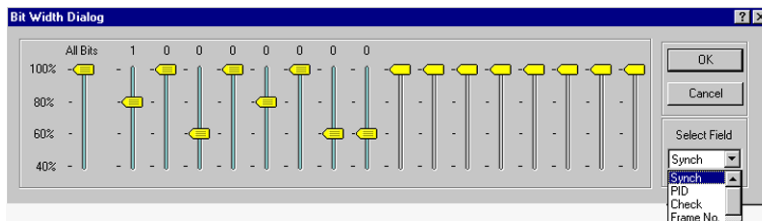
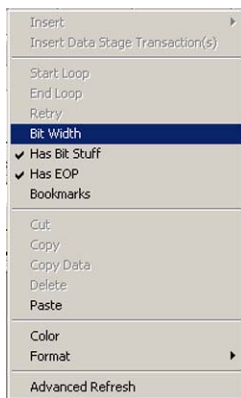


Figure 54 Set Bit Width Dialog

Select a field for bit width reduction, choose a percentage bit width reduction for either all of the bits in the field or each bit individually, and click **OK**.

Exerciser Programming Shortcuts

You can quickly create an exerciser program from an existing *.smp file. There are two ways to quickly create an exercise program. You can import a complete *.smp file or open an *.smp file and copy the lines.

Creating an Exerciser Program by Importing

Right-click in an open Exerciser Program window and choose **Import from SMP file** to open the File Import dialog.

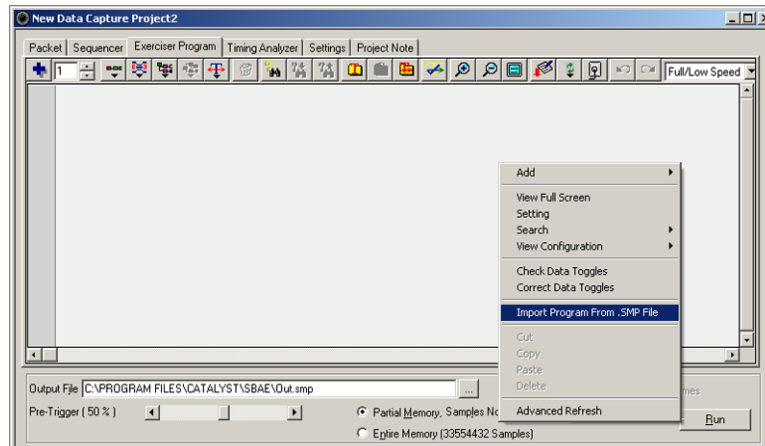


Figure 55 Import Exerciser Program

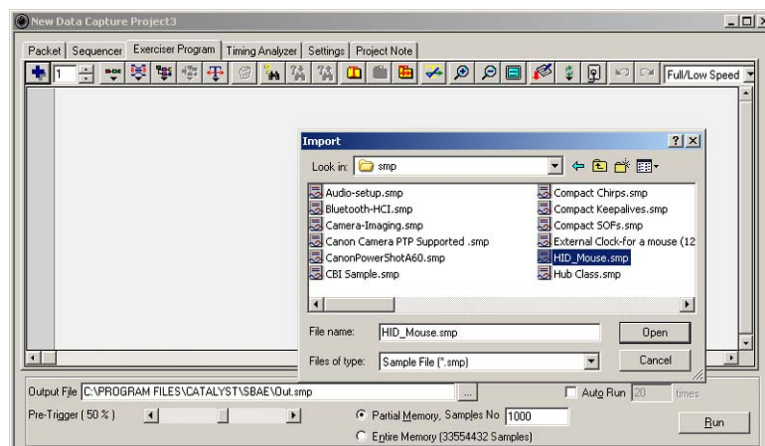


Figure 56 File Import Dialog

Choose an *.smp file to use for an exerciser program and click **Open**. You now have an exerciser program ready to run. It is recommended that you perform a refresh before running the program.

Protocol Analysis

Refresh the Program

Click the **Refresh Timing** button on the Exerciser Program toolbar. For Full/Low speed projects you must choose the operating speed from the refresh button's drop-down list.

Refresh Timing button



Auto Refresh

Choosing **Auto Refresh** adjusts the starting time of packets, transactions, and splits automatically after a paste operation.

Manual Refresh

Choosing **Manual Refresh** requires you to adjust start times yourself.

Auto Generate SOFs

Choosing **Auto Generate SOFs** causes the Host Exerciser to automatically insert SOF packets.

Advanced Refresh

Advanced Refresh specifies the refresh range and changes device addresses and endpoint numbers for all selected transactions.

Right-click in the exerciser program area and choose **Advanced Refresh** to open the advanced refresh dialog.

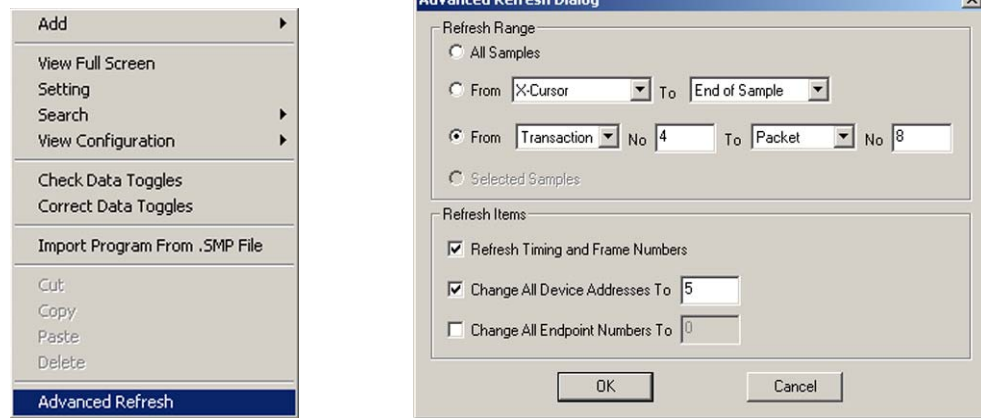


Figure 57 Advanced Refresh Dialog

Protocol Analysis

Creating an Exerciser Program by Copying and Pasting

To create an exerciser program by copying and pasting:

1. Open an *.smp file, hold down the left mouse button, draw a rectangle around a group of data lines with the mouse cursor, and then release the mouse button to select them.



P	T	Control Transfer	S	Dev. Addr(Hex)	Descriptor Type	Descriptor Index(Hex)	Interpreted Data
1	0	Get Descriptor	L	00	Device	0	Device Descriptor
1	3	Reset		000.000.991.15		10.947	
1	1	Set Address	L	00		1	
1	2	Get Descriptor	L	01	Device	0	Device Descriptor
1	3	Get Descriptor	L	01	Configuration	0	Configuration Descriptor
1	4	Get Descriptor	L	01	Configuration	0	Configuration Descriptor
1	5	Get Descriptor	L	01	String	0	String Descriptor
1	6	Get Descriptor	L	01	String	0	String Descriptor
1	7	Get Descriptor	L	01	Device	0	Device Descriptor
1	8	Get Descriptor	L	01	Configuration	0	Configuration Descriptor
1	9	Get Descriptor	L	01	Configuration	0	Configuration Descriptor
1	10	Set Configuration	L	01	Configuration Value(Hex)	0	

Figure 58 Lines Selected

Note: The appearance of red check marks next to the lines indicates that they have been selected.



2. Click the **Copy** button, or right-click the data area, and choose **Copy**.
3. Open an Advanced Project and click the **Exerciser Program** tab.



4. Click the **Paste** button, or right-click the exerciser program area, and select **Paste**.

Packet	Sequencer	Exerciser Program	Timing Analyzer	Settings	Project Note													
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>000.000 000 00</td> <td>00</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	0	000.000 000 00	00	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
0	000.000 000 00	00	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Event Type</th> <th>Duration Time(Sec)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reset</td> <td>0.010947</td> </tr> </tbody> </table>	Event Type	Duration Time(Sec)	0	Reset	0.010947											
Event Type	Duration Time(Sec)																	
0	Reset	0.010947																
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>000.027 964 05</td> <td>00</td> <td>H To D</td> <td>Standard</td> <td>Device</td> <td>Set Address</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	1	000.027 964 05	00	H To D	Standard	Device	Set Address			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
1	000.027 964 05	00	H To D	Standard	Device	Set Address												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>000.036 377 12</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	2	000.036 377 12	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
2	000.036 377 12	01	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>000.053 353 81</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	3	000.053 353 81	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
3	000.053 353 81	01	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>000.070 137 30</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	4	000.070 137 30	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
4	000.070 137 30	01	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>5</td> <td>000.087 500 38</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	5	000.087 500 38	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
5	000.087 500 38	01	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>6</td> <td>000.109 902 57</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	6	000.109 902 57	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
6	000.109 902 57	01	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>000.132 304 77</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	7	000.132 304 77	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
7	000.132 304 77	01	D To H	Standard	Device	Get Descriptor												
		<table border="1"> <thead> <tr> <th>Start Time</th> <th>Dev. Addr(Hex)</th> <th>Dir</th> <th>Type</th> <th>Recipient</th> <th>bRequest(Hex)</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>000.149 281 45</td> <td>01</td> <td>D To H</td> <td>Standard</td> <td>Device</td> <td>Get Descriptor</td> </tr> </tbody> </table>	Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)	8	000.149 281 45	01	D To H	Standard	Device	Get Descriptor			
Start Time	Dev. Addr(Hex)	Dir	Type	Recipient	bRequest(Hex)													
8	000.149 281 45	01	D To H	Standard	Device	Get Descriptor												

Figure 59 New Exerciser Program

Refresh

When copying and pasting program lines from different projects, it is recommended that you refresh the program window prior to running the program. This is required to ensure correct start time computation, which is dependent on consistent project speed definition.

Protocol Analysis

To Refresh the Program

Click the **Refresh** button on the Exerciser Program toolbar.

For Full/Low speed projects, choose the operating speed from the Refresh button's drop-down list.



Auto Refresh

Choosing **Auto Refresh** adjusts the starting time of packets, transactions, and splits automatically after a paste operation.

Manual Refresh

Choosing **Manual Refresh** requires you to adjust start times yourself.

Auto Generate SOFs

Choosing **Auto Generate SOFs** causes the Host Exerciser to automatically insert SOF packets.

You can now use this new exerciser program immediately in the current project or save it for use elsewhere.

Creating a Data Block

You can create the following four types of data blocks for use wherever data fields are used:

- Custom data pattern specifically for your application
- Random data pattern
- Counter
- Walking “1” or “0” pattern.



1. To create a data block, click the **Default Data Block** button on the Main toolbar to open the Data Block dialog box, as shown in Figure 60.

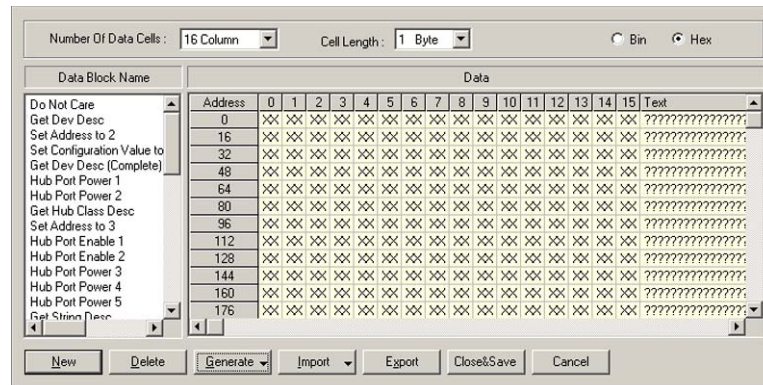


Figure 60 Default Data Block Dialog Box

2. To add another data block, click the **New** button in the Data Block dialog box.

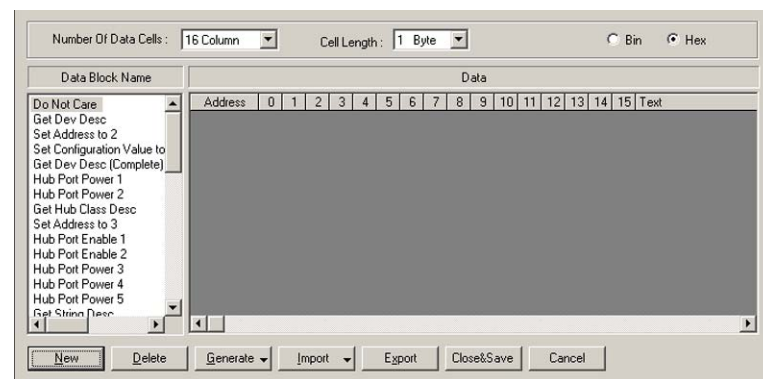


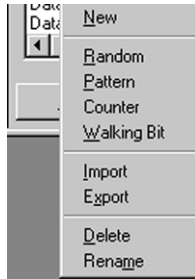
Figure 61 New Data Block Dialog Box

3. Choose the number of data columns (up to 16 Data Cells/Row) and the Cell length (up to 16 Bytes/Cell). This is a display function only.
4. Click either the **Bin** or **Hex** button to choose a number format.

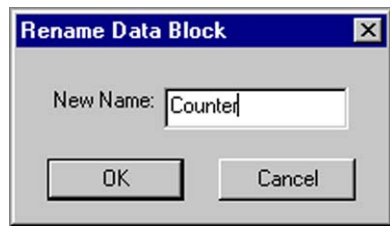
Protocol Analysis

Naming a Data Block

Each new data block is automatically assigned a sequential data block number as it is created. To assign a unique descriptive name to a data block, right-click the data block name to open the data block edit menu.



Choose **Rename**.



Enter a descriptive name in the **New Name** edit box and click **OK**.

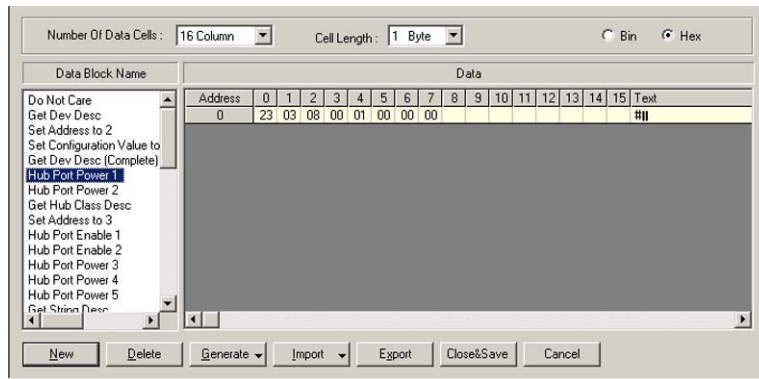


Figure 62 Sample Active Data Block Hub Port Power

You can enter data in the defined cell structure by choosing one of the four available methods. Define your own pattern, set a counter, choose a Random Pattern or choose a Walking Bit Pattern

Define Your Own Pattern

To define a pattern:

1. Click the **Generate** button and choose **Pattern** to open the Define Pattern dialog box, as shown in Figure 63
2. Enter a data pattern in the Data Pattern edit box.
3. Choose the number of times to repeat that pattern and click **OK**.

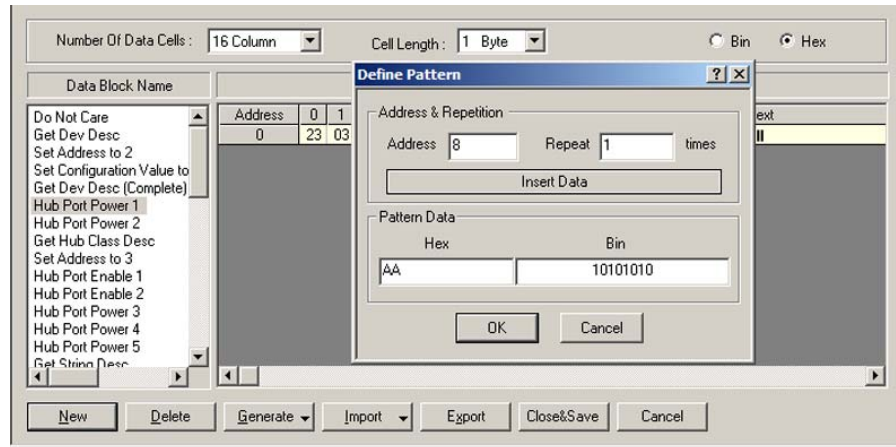


Figure 63 Define Your Own Data Pattern

Address

The cell address starts at 0 for the first data entry and automatically increments to the next available address as data is written. You can set it back to a previously defined address to modify its content or insert additional data at that point.

Insert/Overwrite Data

To define if the data in a previously defined cell is overwritten or new data is inserted after that cell, click the **Insert/Overwrite** button to toggle to an operation.

Protocol Analysis

Counter

To use a counter as data, click the **Generate** button, choose **Counter**, enter a **Starting Number** for the counter, enter the data address to which to count, and click **OK**.

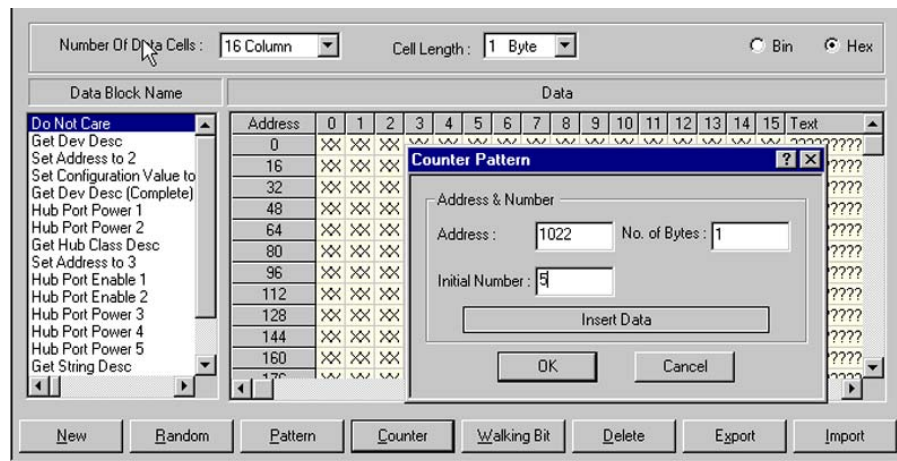


Figure 64 Set Counter as Data

Random Data Pattern

To use a random data pattern, click the **Generate** button, enter the number of times to repeat the pattern, and click **OK**.

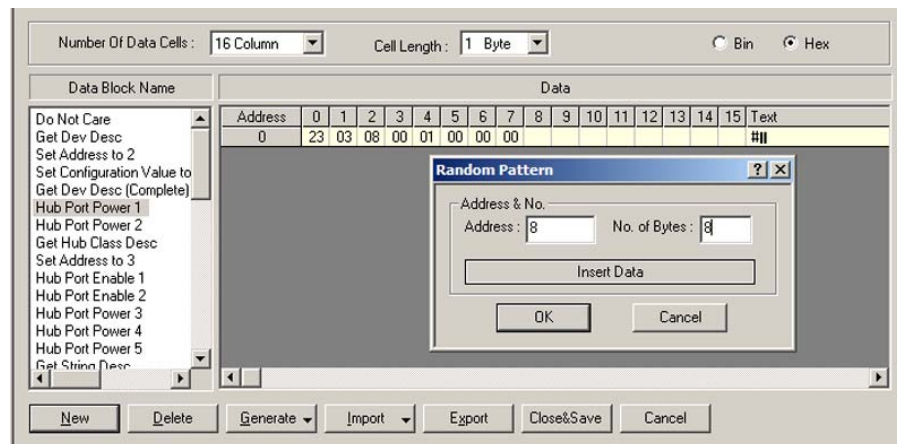


Figure 65 Choose a Random Pattern

Walking Bit Pattern

To use a walking bit pattern, click the **Generate** button. Choose a walking bit of “0” or “1”, the walk direction, the start position, and the number of times to repeat the pattern.

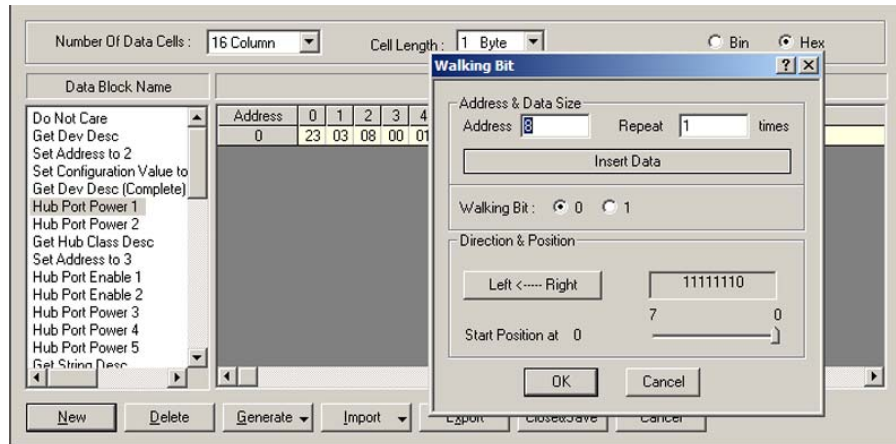


Figure 66 Define a Walking Bit Pattern

Sample Data Block

Figure 67 illustrates a data block with a defined pattern AA starting at address 0 and repeated four times, followed by a random pattern starting at address 4 and repeated four times, then a left-to-right walking 0 pattern starting at address 8 and repeated six times.

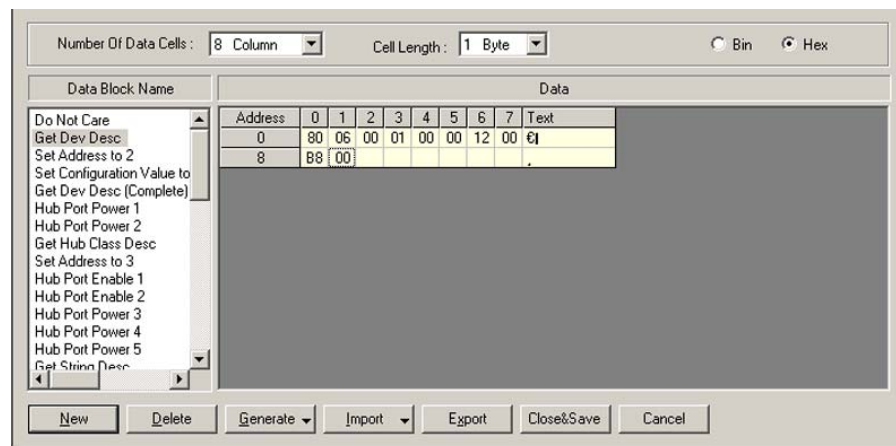


Figure 67 Sample Data Block Definition

Close & Save

When you have completed a data block definition, click the **Close&Save** button to save the newly created data block.

The saved data block automatically is assigned a sequentially numbered (DataBlock##) title. To rename for easier identification later, see “Naming a Data Block” on page 91.

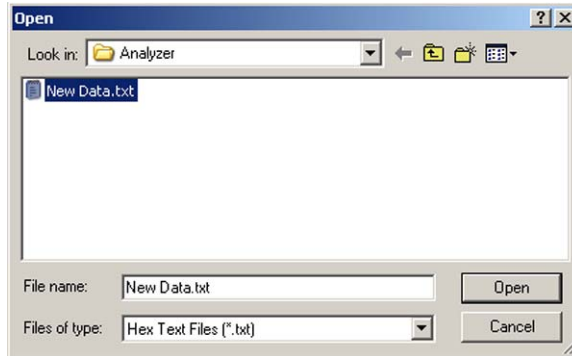
Protocol Analysis

Creating and Editing Data Blocks as Text

You can create and edit data blocks using a text editor such as Windows[®] Notepad. To create a data block in Notepad, launch Notepad and enter the data in space-delimited Hex format. Save as a **.txt** text file.

Import from Text File

To import Text Editor created data, click the **Import** button in the data block definition dialog and choose **Import from Text File** to open the Open dialog.



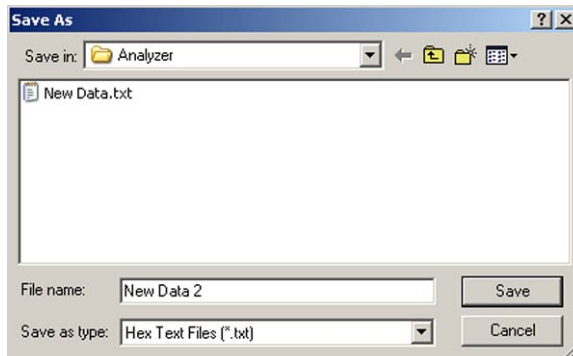
Choose a file and click **Open**.

Import from Clipboard

To import data that has been copied to the clipboard, click the **Import** button in the data block definition dialog and choose **Import from Clipboard**.

Modify existing data

To edit an existing data block to create a new data block using a text editor, select the data block to edit from the Data Block Name list and click **Export** to open the Save As dialog.



Assign a name to the new data block text file and click **Save**.

You can now edit the newly created text file using Notepad or any other text editor and then import it into the data block definition, as described above.

Settings

Click the **Settings** tab to open the Settings dialog. See “Settings” on page 54 for descriptions of settings options.

Analyzer Only

When used for Hub testing, the Conquest M2 Auxiliary port performs as an analyzer only.

Project Note

To enter a note about the current project, click the **Project Note** tab and enter the data to associate with the project.

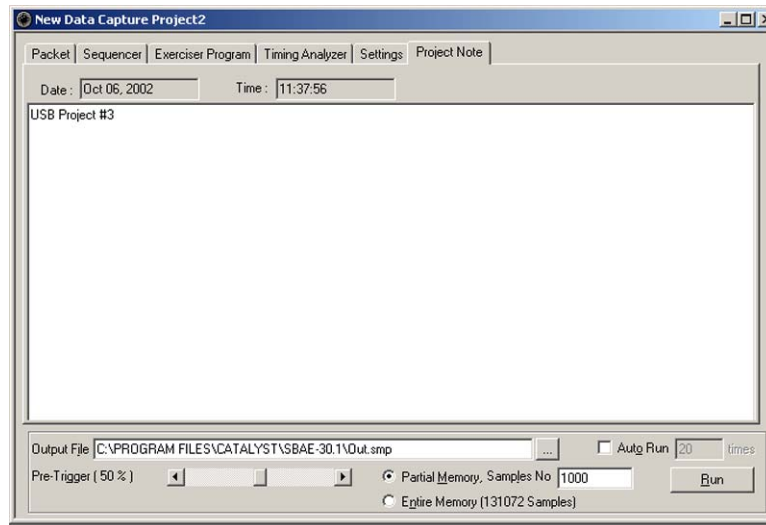


Figure 68 Informative Project Note

Running the Project

Set Data Capture Options

Capture to Memory

You can limit the captured data display to a specific number of samples by checking **Partial Memory** and entering the number of Samples to capture. Alternatively, you can check **Entire Memory** to allow capture for the entire memory. Choosing Partial Memory results in less time to display results.

Pre-Trigger

Pre-Trigger is set by default at 50%, which defines the percentage of data to capture before and after the triggering event. You can change this percentage by dragging the slider.

Pre-Trigger Data Limit - The capture of the specified percentage of the data prior to the triggering event cannot be guaranteed and may in some cases be 0. This can occur if the triggering event occurs before the required amount of pre-trigger event data can be stored. In these cases, the data display shows fewer than the specified data points prior to the triggering event. For more detail, see “Pre-Trigger” on page 111.

Auto Run

Continually captures data for the number of times specified. A separate data file is generated each time that a capture is performed.

Run Project

To run the project, click **Run** and wait for the occurrence of a trigger event to open the data capture result.

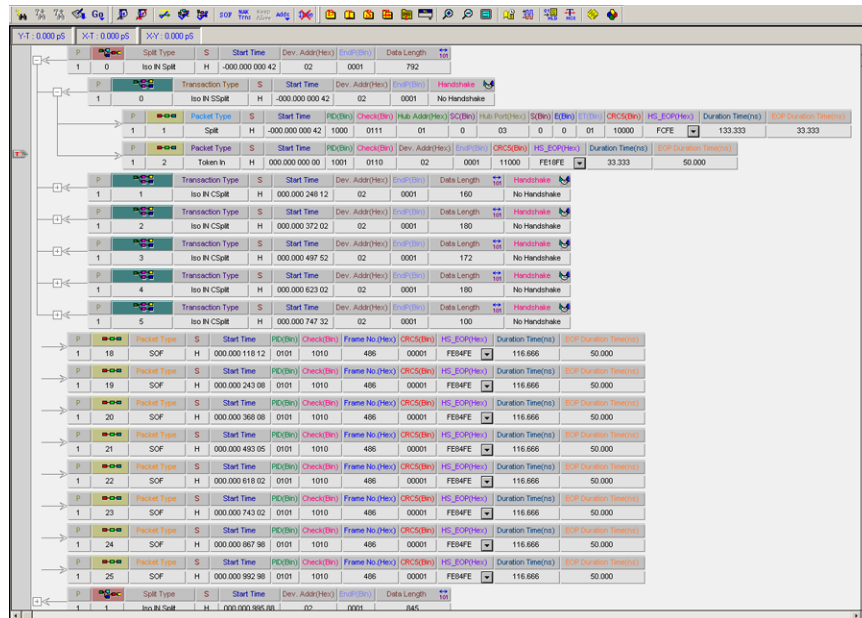


Figure 69 Data Capture Result

Display Manipulation

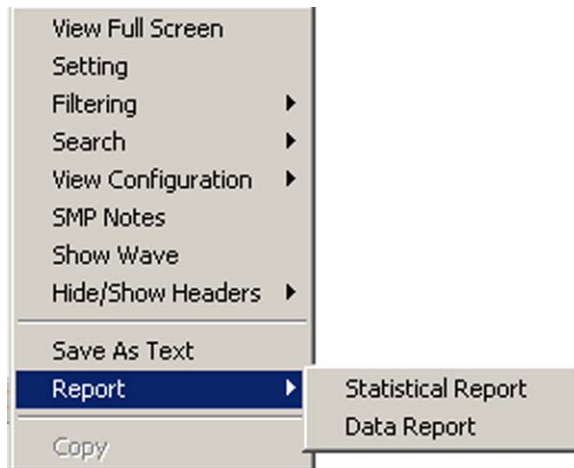
For details on configuring the data capture result display, see “Display Manipulation” on page 177.

Reports

When a data capture is in the sample viewer, you can view either a packet data summary Statistical Report or a Data Report.

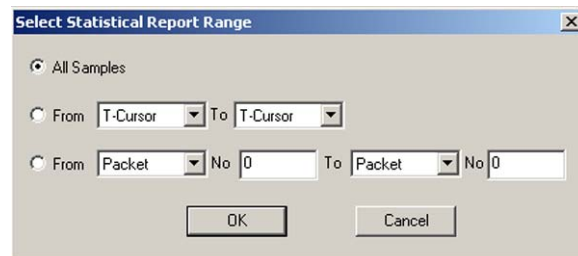
To View a Report

Right-click in the right side display area and choose **Report**. You can also access the reports from the sample viewer toolbar.



View Statistical Report

Choose **Statistical Report** to open the Select Statistical Report Range dialog. Select a report range and click **OK** to display the result as shown in Figure 70.



A screenshot showing the sample viewer toolbar and a statistical report table. The toolbar includes icons for file operations, navigation, and analysis. The report table is as follows:

Type	Port	Speed	Duration Time	Count	%
All	---	---	---	---	---
SOF	1	High	69.00 us	414	31.34
Split	1	High	51.83 us	311	23.54
Token In	1	High	53.92 us	324	24.53
MData	1	High	670.03 us	208	15.75
Data0	1	High	89.00 us	51	3.86
NAK	1	High	1.73 us	13	0.98
			935.52 us	1321	100.00

Figure 70 Statistical Report

Protocol Analysis

View Data Report

Choose **Data Report** to open the Select Report Type dialog.

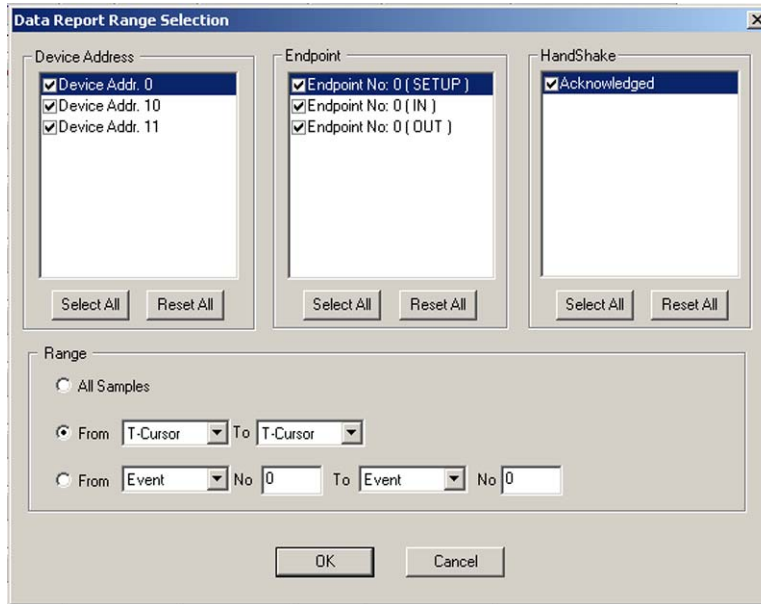


Figure 71 Select Report Type Dialog

Make selections to view the report on Transaction or Endpoint type Device Address(es), Handshake, or all. You can specify to report on all samples, between specific cursors, or between specific events. Make a choice and click **OK** to display the result as shown in Figure 72.

The Data Report display is Hierarchical, displaying the report in terms of Transactions, Transfers, and Data with clear delimiters.

Data packet color switches between transfers **ASCII Panel**

Transfer boundary

Transaction #

Data

Data address offset

Trans. No	Device Address : 0	Endpoint : 0	Direction : Control
0	00000000 : 80 06 00 01 00 00 00	Transfer : 0	Acknowledged
	00000008 : 12 01 00 01 00 00 00 08		Acknowledged
3	00000010 : 00 05 02 00 00 00 00 00	Transfer : 1	Acknowledged
5	00000000 : 80 06 00 01 00 00 12 00	Transfer : 2	Acknowledged
6	00000008 : 12 01 00 01 00 00 00 08		Acknowledged
7	00000010 : FA 04 01 42 A2 00 00 00		Acknowledged
8	00000018 : 00 01		Acknowledged
10	0000001A : 80 06 00 02 00 00 09 00	Transfer : 3	Acknowledged
11	00000022 : 09 02 A5 01 02 01 00 C0		Acknowledged
12	0000002A : 50		Acknowledged
14	0000002B : 80 06 00 02 00 00 FF 00	Transfer : 4	Acknowledged
15	00000033 : 09 02 A5 01 02 01 00 C0		Acknowledged
16	0000003B : 50 0B 04 00 00 00 01 01		Acknowledged
17	00000043 : 00 00 01 00 09 24 01 09		Acknowledged
18	0000004B : 00 72 00 01 01 0C 24 02		Acknowledged
19	00000053 : 01 01 01 00 02 03 00 00		Acknowledged
20	0000005B : 00 0C 24 02 02 01 06 00		Acknowledged
21	00000063 : 02 03 00 00 00 0C 24 02		Acknowledged
22	0000006B : 03 01 06 00 02 03 00 00		Acknowledged
	00000073 : 00 0A 24 06 05 09 01 01		Acknowledged

Figure 72 Data Report Hierarchy

Data Report Settings

You have a number of choices to set up the way the data report is displayed. To set up a data report view, right-click in the data report, then select **View Setting** and **Setting Dialog** to open the Data Report Setting dialog.

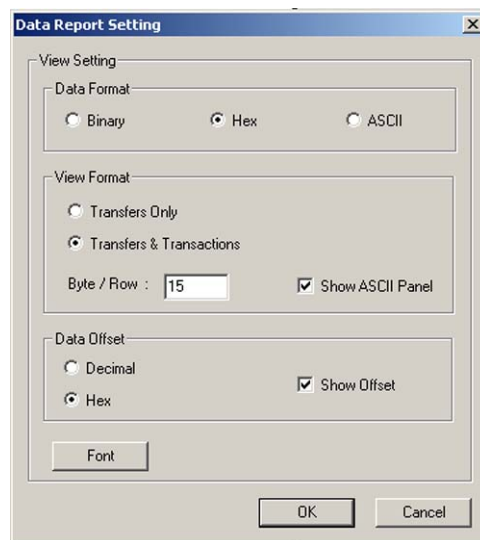
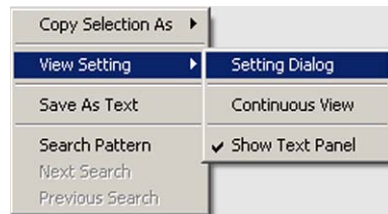


Figure 73 Data Report Setting Dialog

Data Format

You can display the data in the report as Binary or ASCII, instead of the default Hex, by checking the appropriate button.

View Format

You can display Transfers Only by checking the **Transfers Only** button and set the Byte/Row to a value that matches the endpoint size of the device, to allow a transaction to be on a single line without wrapping. Check the **Show ASCII Panel** to display an ASCII interpretation of notes.

Data Offset

The default data address offset display is Hex. You can change it to decimal by checking the **Decimal** button. You can also disable the data address offset in the display by unchecking the **Show Offset** box.

Protocol Analysis

View Data Report Statistics



Click the **Data Report Statistics** button on the data report toolbar to display the data report statistics.

Linked display

The screenshot shows a data report window with a table of transactions and transfers. A cursor is positioned on a 'Setup' transaction in the table. An arrow labeled 'Linked display' points from this row to the corresponding data in the main report window above. The table below is titled 'Data Report Statistics'.

Transaction	Device Address	Endpoint	Handshake	Payload size	Count
All	All	All	All	---	---
Setup	0	0	Acknowledged	8	2
In	0	0	Acknowledged	8	1
Setup	2	0	Acknowledged	8	68
In	2	0	Acknowledged	8	299
In	2	0	Acknowledged	2	3
In	2	0	Acknowledged	1	5
In	2	0	Acknowledged	7	2
In	2	0	Acknowledged	5	2
Out	2	0	Acknowledged	8	4
Out	2	0	Acknowledged	4	4
Out	2	0	Acknowledged	3	2
Out	2	1	No Handshake	176	4858
Out	2	1	No Handshake	180	550
Out	2	1	No Handshake	192	247
In	2	2	No Handshake	5	22
					6069

Figure 74 Data Report Statistics Displayed

Linked Displays

Clicking a Transaction/Transfer type in the Report Statistics window causes a cursor to go to the first instance of that Transaction/Transfer type in the Data Report window. You can position the linked cursor to any item in the Data Report by using the **Go to Next**, **Go to Previous**, and **Go to item #** buttons on the Data Report Statistics toolbar.

The toolbar contains the following buttons and controls:

- Export to Excel
- Save as Text
- Transaction # (input field showing '1' of 68 Setup Transactions)
- Go to Transaction/Transfer #
- Go to Next Transaction/Transfer
- Go to Previous Transaction/Transfer
- Report display settings
- Print
- Print preview

Figure 75 Data Report Statistics Toolbar

Customize the Display

You can select for display only specific items for a specific project. To set up the display, click the **Report Display Settings** button on the Data Report Statistics Toolbar.

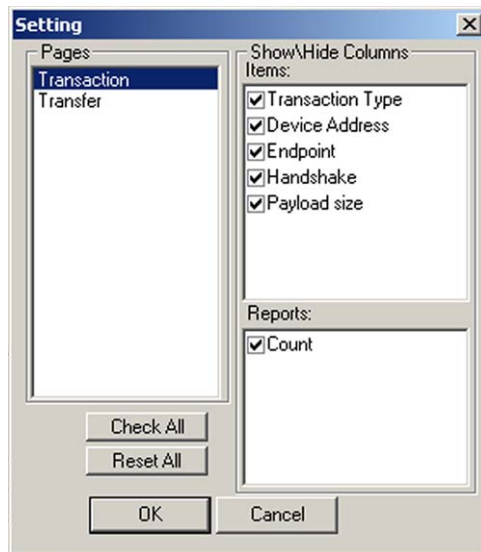


Figure 76 Report Display Settings Dialog

Uncheck the items to exclude from the display.

Display Histogram



Click the **Histogram** button on the data report toolbar to display a histogram of data byte values

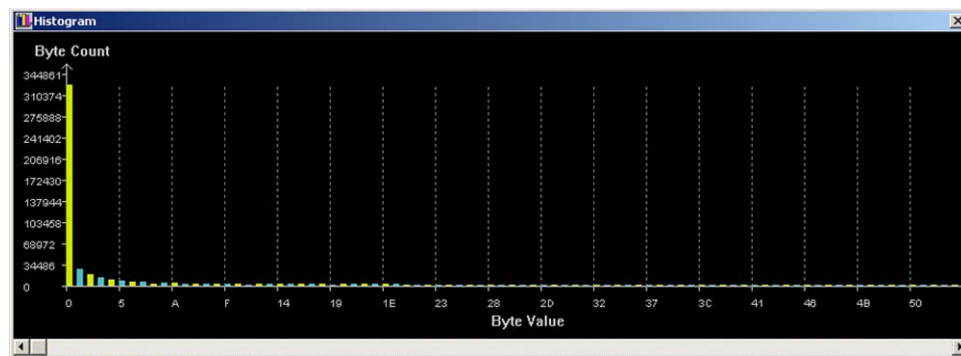


Figure 77 Data Byte Value Histogram

Position the mouse cursor over byte value to display a numeric message for that byte value.

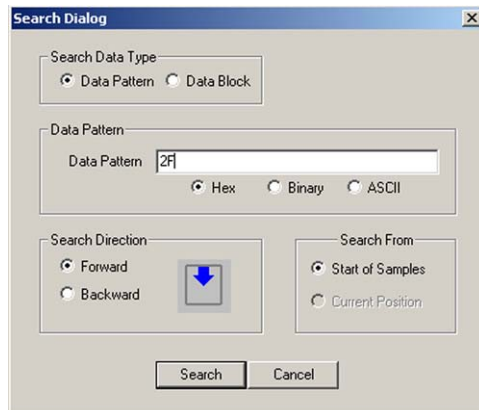
Protocol Analysis

Search a Data Report for a Pattern

When a data report displays, you can quickly search for a data pattern in Hexadecimal, Binary, or ASCII, or a pre-defined data block.



To search for a data pattern or a data block in a data report, click the **Search** button on the toolbar to open the Search for Data Pattern dialog.



To search for a data pattern, enter the data pattern in the Data Pattern text box and click **OK**. Continue the search within the data report for the same data by

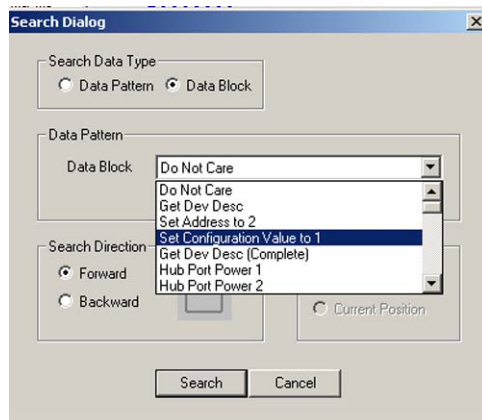


clicking either the **search forward** button or the **search backward**



button, or repeat with a different data pattern.

To search for a data block, check the **Data Block** option button, click the down arrow next to the Data Block drop-down list box, and select a pattern for which to search.



Perform the search by clicking the search buttons to find the specified data pattern.

Interpret Data

You can display a decoded descriptor or request data by selecting **Interpret Data**.



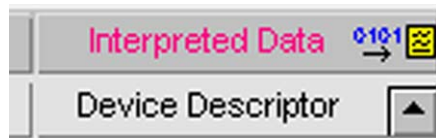
Click the **Interpret Data** button on the toolbar to display the data interpretation for all transfers.

The screenshot shows a software interface with a toolbar at the top and a main display area. The main display area is divided into two sections. The top section shows a list of events with columns for Event Type, Start Time, and Duration Time (ms). The bottom section shows a detailed view of a Device Descriptor with columns for Offset, Field, Size, Value, and Description.

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of this descriptor is 18 Bytes
1	bDescriptorType	1	0x01	DEVICE Descriptor Type
2	bcdUSB	2	0x0100	USB Specification Release Number is 1.00
4	bDeviceClass	1	0x00	Each interface within a configuration specifies its own class
5	bDeviceSubClass	1	0x00	This field like bDeviceClass must also be reset to zero.
6	bDeviceProtocol	1	0x00	The device does not use class-specific protocols on a device basis.
7	bMaxPacketSize0	1	0x08	Maximum Packet Size for endpoint zero is 8 bytes
8	idVendor	2	--	Not Received
10	idProduct	2	--	Not Received
12	bcdDevice	2	--	Not Received
14	iManufacturer	1	--	Not Received
15	iProduct	1	--	Not Received
16	iSerialNumber	1	--	Not Received
17	bNumConfigurations	1	--	Not Received

Figure 78 Interpret Data Configuration Display


To close a data interpretation for a transfer, click the **Device Descriptor** field. Click the field again to open it.



High Level Interpretation Assignment

High level interpretation assignment is a results analysis tool that allows decode and identification of upper level protocols used by a known device. Standard requests are interpreted automatically. However, Class requests require assignment of an appropriate protocol.

To assign a protocol for a High Level interpretation in an open results window,

click the  **NCX HLD** button on the main toolbar to open the High Level Interpretation Assignment dialog.

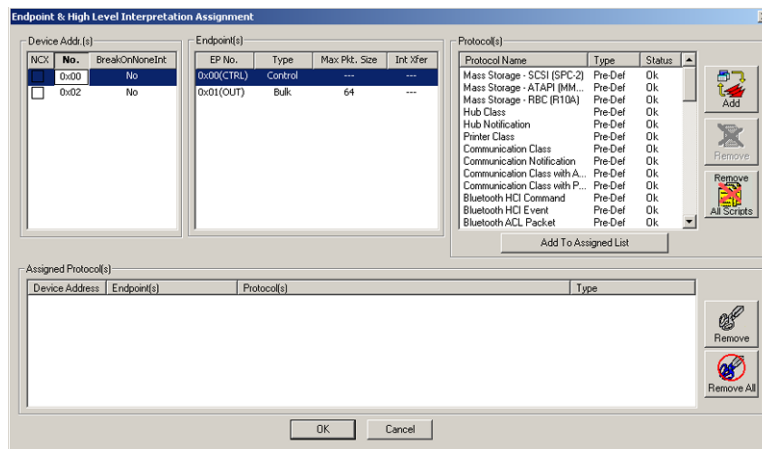


Figure 79 Upper Level Interpretation Assignment Dialog

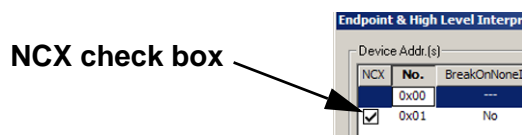
The global Upper Level Interpretation dialog opens with all of the active Devices and endpoints identified.

To assign a protocol for the device you are testing:

1. Highlight an endpoint.
2. Choose a protocol appropriate for that endpoint in the Modifiable Protocols list box.
3. Click the **Add to Assigned List** button.
4. Repeat for all endpoints and click **OK**.

5. Click the  **Show/Hide Non Control Transfers** button to toggle the display.

Note: To Show/Hide Non Control Transfers, check the **NCX** box.



To view the interpretation for a transaction, click the down arrow for the interpretation flag on that transaction.

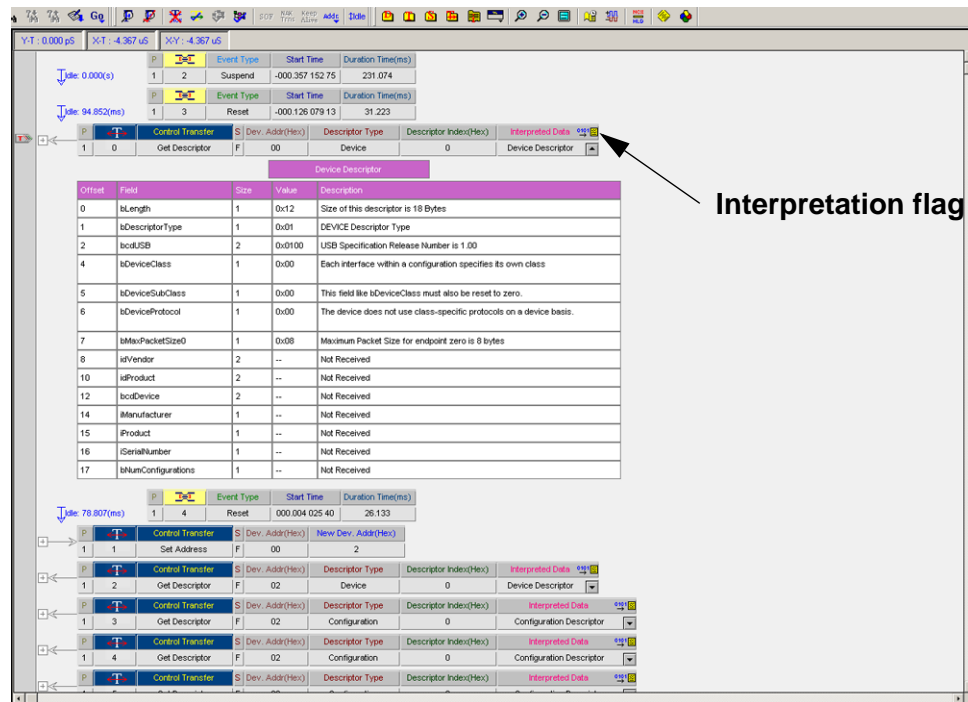



Figure 80 Upper Level Interpretation Result

User-Defined Decodes

The user-defined decode feature allows you to create custom Upper Level interpretation tables by using a text based scripting language and to save them as *.asl files. For a syntax definition of the scripting language, see “Appendix A” on page 205.

To add a previously created custom script file as a new protocol, open the High Level Interpretation Assignment dialog.

Protocol Analysis

Click  **Add** to access previously created protocol script files.

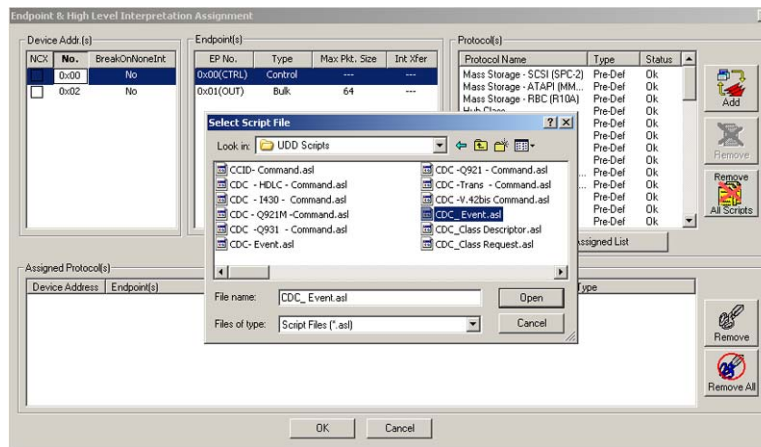


Figure 81 Choosing a Custom Script File

Choose a script file from the Select Script File dialog, then click **Open** to add this file to the Defined Protocols list. Repeat for additional protocols.

You can now assign this protocol as described on page 105.

Note: Two types of High Level decodes are in the Assigned Protocols window:

- Pre-defined protocols included with the system
- Custom protocols that you create using the ASL script

Errors in User-Defined Scripts

Errors in user-defined scripts are logged as a result of script file parsing the first time that the script is run on the system. An appropriate error message is displayed and a text file of errors is generated and saved in the same folder as the source script file.

Note: The first time you open the software after installation, the software compiles the decoding files in the **SystemData\UDDcfg** and **SystemData\Cfg** folders. If you remove decoding files from these directories, by deleting or by moving files to another folder, the software recompiles the decoding files the next time you open the software. If you do not remove decoding files from the **SystemData\UDDcfg** and **SystemData\Cfg** directories, the software does not compile the decoding files again.

You can modify decoding files directly by using any text editor or the application tool. Example source decoding files are in the **Examples\UDDScripts** folder. For examples of script modification using the software tool, you can open and modify the **.asl files** in the **Examples\UDDScripts** using the application. If you modify decoding files in the application, you must verify them. It is recommended that you save the example file before using the UDD script. For information on how to modify or create new scripts, see “Appendix A” on page 205.

Protocol Errors

Conquest monitors and captures a variety of real time protocol errors every time that a data capture is made. The Conquest Protocol Suite software also detects post-process protocol errors.

Protocol Errors Detected While Capturing Data

- Err0** Bit Stuff (packet corrupted, stuff bit not present)
- Err1** PID Err (packet identifier check failure)
- Err2** PID Unknown (packet with undefined PID)
- Err3** Sync Error
- Err4** CRC5 Error (IN, OUT, SETUP packet error)
- Err5** CRC16 Error (DATA0, DATA1, DATA2, MDATA payload error)
- Err6** Frame Len (the space between SOF is not per USB specification)
- Err7** Babble (Bus not idle at end of frame error)
- Err8** Data Toggle (detected incorrect data packet toggle bit)
- Err9** EOP Error (incorrect width of EOP signal or incorrect EOP occurrence)
- Err10** Loss Activity (packet transfer interrupted by constant state on bus)
- Err11** Time Out (distance between token or data packet and corresponding response greater than USB specified)
- Err12** Bus Error indicating that both D+ and D- signals were set to one (SE1). This error can be detected only in Full or Low speed mode.
- Err13** Short Inter Pkt Delay
- Err14** Truncated Transaction. This error occurs if an expected packet is not received at all. When capturing isochronous transactions, disable this protocol error.

Protocol Analysis

Protocol Errors Detected Post-Process

Packet Protocol Errors Detected

- Err16** SOF at Low speed. Start of frame packet, SOF, cannot be used in low speed mode. Instead of this packet, a keep-alive event is expected.
- Err17** Packet Size is big. Indicates that the total number of bits received for a given packet exceeds the expected number.
- Err18** Truncated Packet. Indicates that the total number of bits received for a given packet is less than the expected number.
- Err19** Invalid speed of packet. This packet is not allowed at this speed.
- Err20 - Err31** Reserved for future use.

Transaction Protocol Errors Detected

- Err32** Invalid Idle after Packet. Indicates that the required Idle time after a packet was less than expected.
- Err33** Timeout after Packet. Indicates that the required Idle time after a packet was more than expected.
- Err34** Reserved
- Err35** ACK not found. Indicates that the required ACK handshake was not received in this transaction.
- Err36** Data Payload Error. Indicates that the expected 8-byte data payload for setup transaction was incorrect.
- Err37** DATA0/1 not found. DATA0/1 is expected but not found in its position.
- Err38** Handshake not found. Handshake is expected in this transaction but not found.

Split Protocol Errors Detected

- Err39** Start Split has smash. Start Split transaction of this Split has smash: Transaction has not been completed successfully.
- Err40** Complete Split has smash. Complete Split transaction of this Split has smash: Transaction has not been completed successfully.
- Err41** Endpoint Halt has occurred. Endpoint halt occurs when three retries have been done with errors in High speed or Full/Low Speed.
- Err42** Full/Low Speed side smash. Transaction smash in Full/Low speed side. Full/Low speed transaction has not been completed successfully.

Transfer Protocol Errors Detected

- Err43** Invalid Data payload. Data payload of the transaction during this transfer is invalid.
- Err44** Status Stage has error.
- Err45** Status Stage for this Transfer not found.
- Err46** Data Payload Speed Error. Speed for Data Payload has error.
- Err47** Status Stage Speed Error. Speed for Status Stage has error.

Pre-Trigger

You can set the amount of data to capture before and after the trigger as a percentage of pre-trigger, between 1% and 99%, by positioning the pre-trigger slider. This feature allows evaluation of bus activity leading up to and after the triggering event. The operation of the pre-trigger in the data memory is conceptually illustrated in Figure 82.

Capture of the specified percentage of data prior to the triggering event cannot be guaranteed and may in some cases be 0. This can occur if the triggering event occurs before the required amount of pre-trigger event data can be stored. In these cases, the data display shows fewer than the specified data points prior to the triggering event.

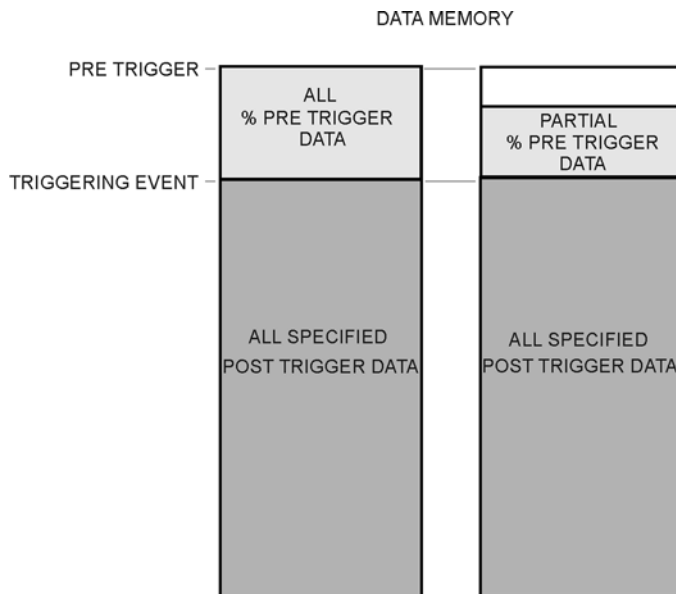


Figure 82 Pre-Trigger Example, 20% Pre-Trigger

If the trigger does not occur prior to a full memory, then new data wraps around to overwrite the captured memory from the beginning.

Performance Analysis

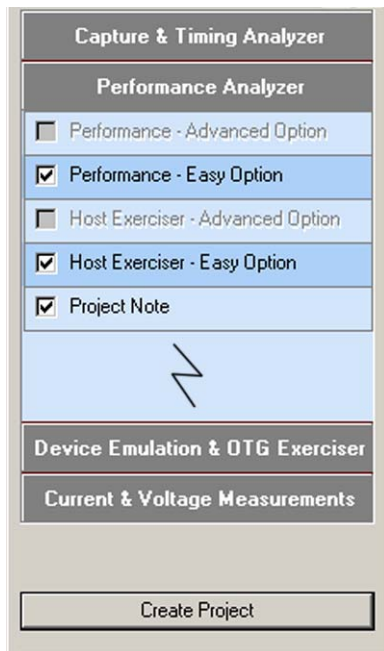
Conquest has two modes of operation:

- An Easy Mode for immediate Performance Analysis with a minimum of setup.
- An Advanced Mode allowing you to program a custom performance analysis.

Performance Analysis (Easy Mode)

The Easy Mode offers you the ability to perform a quick Performance Analysis with a minimum of setup or programming. You can perform a performance analysis on normal bus traffic or program the exerciser to measure performance with exerciser generated data.

Click the **New Project** button on the main toolbar to open the Project Setup dialog and click **Performance Analyzer**.



Check the **Performance - Easy** option and then the **Create Project** button to open a Performance Analysis project.

Note: Easy mode allows you to choose to run with either the Host Exerciser - Easy Option or the Host Exerciser - Advanced Option.

Set up

- Click the **Performance Options** tab to display the Performance Options dialog box.

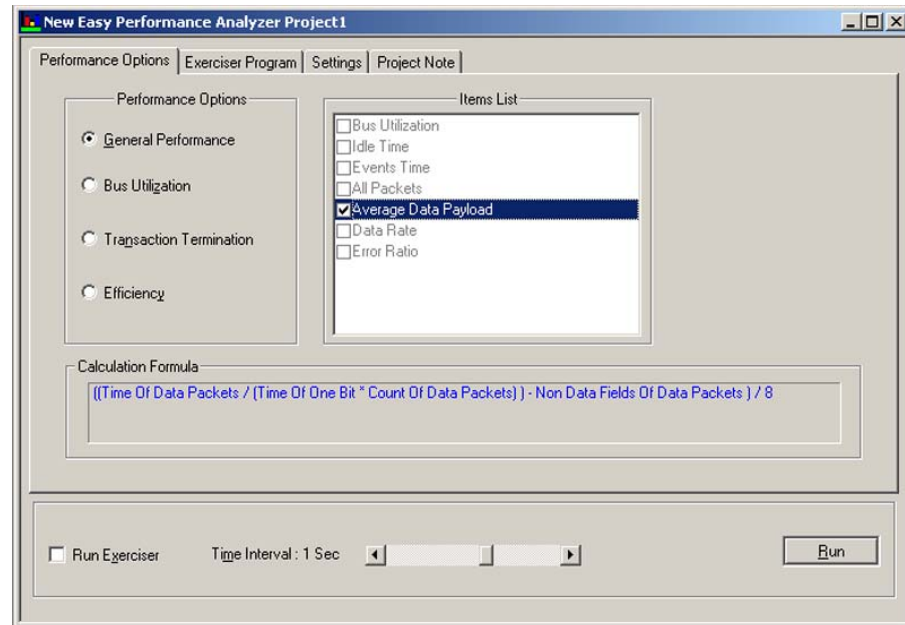


Figure 83 Performance Analysis Options Dialog Box

- Choose a Performance Option by clicking the appropriate option button and then check an appropriate item check box in the Items List.

Note: Calculation Formula - The relationship used for the selected Performance Option and Item is displayed in blue in the Calculation Formula box.

- To perform an immediate Performance Analysis on bus traffic, click **Run**.
- To perform a Performance Analysis using the exerciser, click the **Exerciser** tab to open the Exerciser program dialog box.
- Program the Exerciser as described in “Programming the Exerciser in Easy Mode” on page 51 or “Programming the Exerciser In Advanced Mode” on page 71, to match the Exerciser mode selected.

Performance Analysis

Run Exerciser

Note: Make sure to check the **Run Exerciser** check box.

Click the **Settings** tab to open the settings dialog box and click the **Infinite** option button to loop continually or the **Number** option button and enter the number of times to run the program.

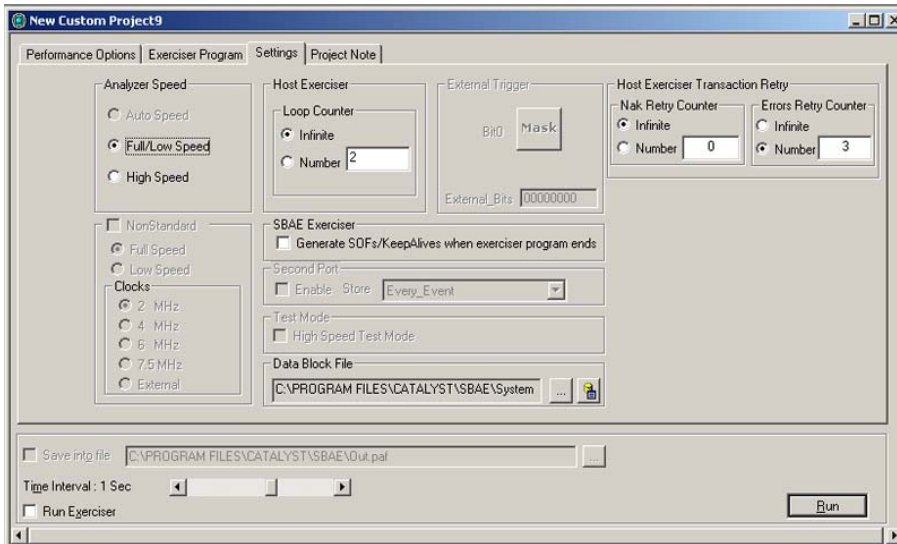


Figure 84 Settings Dialog Box

Click **Run**.

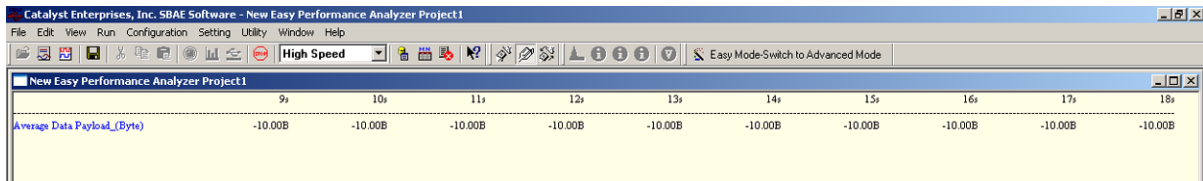


Figure 85 Performance Analysis Result

Performance Analysis (Advanced Mode)

Conquest M2 has two ways to do Performance Analysis: Real-time and Statistical Trace analysis.

Real Time analysis uses hardware counters and dual ported FIFOs. These counters continually count the events and interface with the CPU via the FIFO to pass the results. This method results in the most accurate measurement, since the counters never stop for reading.

Advantage Continuous Real-Time measurement.

Disadvantage Measurement results are based on an average.
Minimum and maximum cannot be reported in this mode.

Statistical Trace analysis is done by capturing the data into memory and then post processing it with software. In this mode, Conquest M2 allows capture to occur after an occurrence(s) of events such that the data includes mostly interesting transactions and not unrelated bus activity.

Advantage Many more measurements are possible than in the Real-Time analysis. It can measure Min, Max, and Average counts.

Disadvantage Non-Real-Time software post-processing of data.

Real Time Analysis

To do a Real-Time performance analysis, open an existing project file or define a new analysis project by defining:

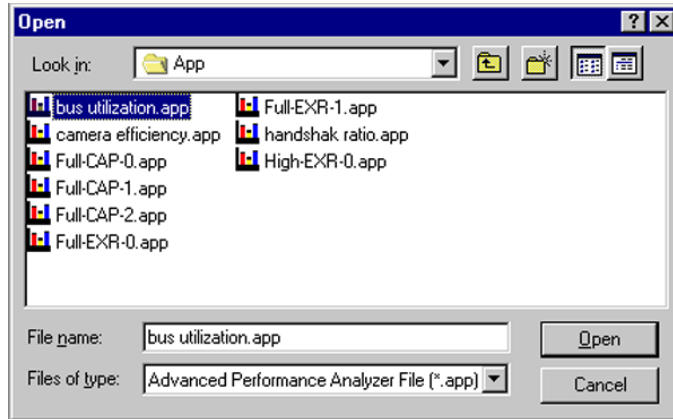
- Packets to use in the analysis
- An Exerciser program, to measure performance with specially generated bus traffic
- An analysis expression or expressions, to measure multiple performance parameters
- Results display configuration
- Performance analysis project options

Performance Analysis

Perform a Pre-defined Analysis

To analyze:

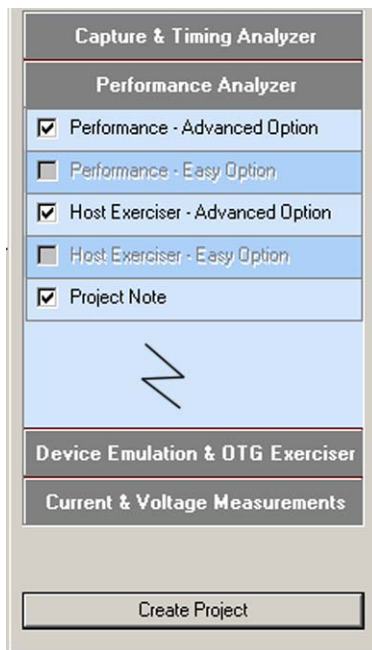
1. Select **F**ile on the main toolbar and choose **O**pen.



2. Select a pre-defined analysis file and click **O**pen to load the project.
3. Click **R**un to perform the analysis.

Create a New Analysis

Click **N**ew **P**roject button on the main toolbar to open the Project Setup dialog and click **P**erformance **A**nalyzer.



Check the **P**erformance - **A**dvanced option and then the **C**reate **P**roject button to open a Performance Analysis Project.

Define Packets

Select the **Packets** tab.

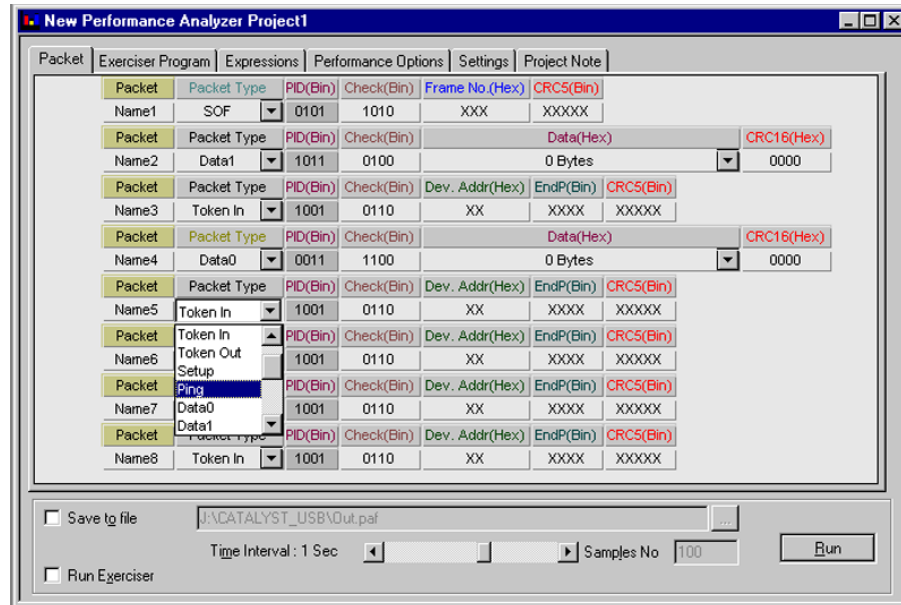


Figure 86 Performance Analysis Dialog Box

Define the packet types to use in the analysis from the Packet Type list box for each packet. See “Defining Packets” on page 60.

Program The Exerciser

To use the exerciser, click the **Exerciser** program tab. Program the Exerciser as described in “Programming the Exerciser in Easy Mode” on page 51 or “Programming the Exerciser In Advanced Mode” on page 71 to match the Exerciser mode selected. Be sure to check the **Run Exerciser** check box.

Performance Analysis

Creating Analysis Expressions

The Analyzer includes four counters and four timers to use to create Analysis expressions. To create an analysis expression, you must first assign a packet, event, or a logical expression to a counter and/or timer included in the Analysis expression. To perform the assignment:

1. Click the **Performance Options** tab to view the Performance Options setting dialog box.

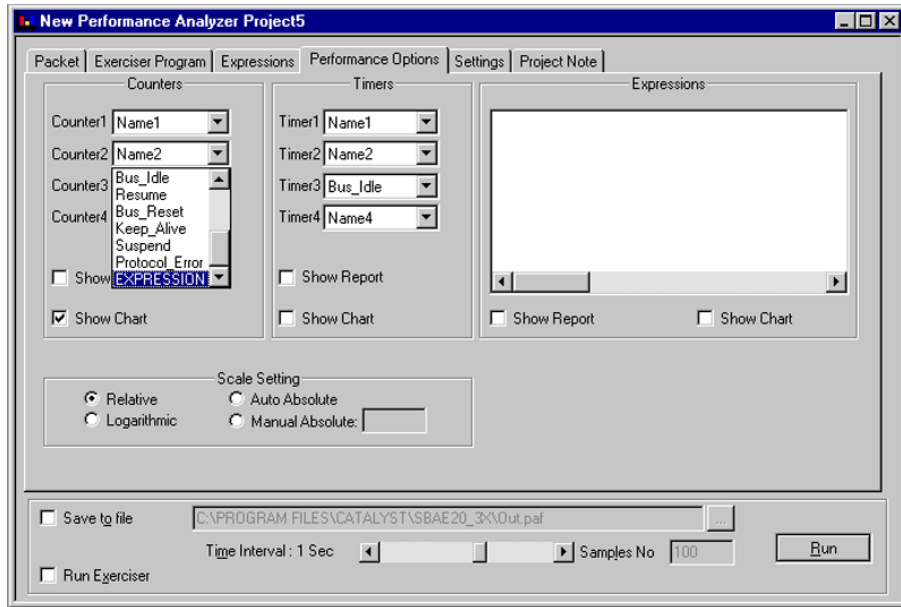
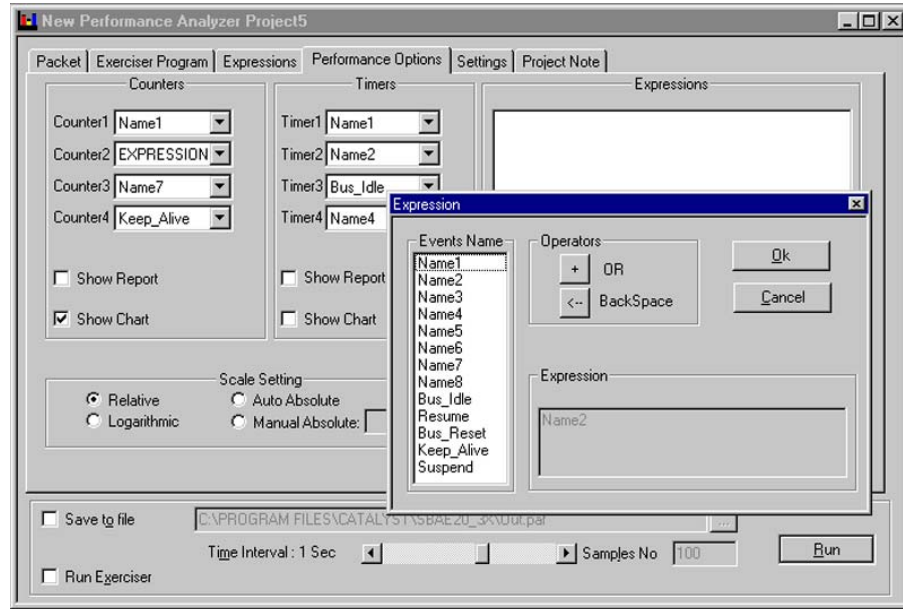


Figure 87 Performance Options Setting Dialog Box

2. Click the down arrow next to a counter and timer to associate a packet event or logical expression with it.

To Define a Logical Expression

1. Choose **EXPRESSION** in the drop-down list next to a counter or timer to open the logical Expression dialog box.



2. Double-click the **Name** that to include in the logical expression, click the **OR** operator, double-click the next name for the expression, and click **OK**.
3. Click the **Expressions** tab to view the analysis expression dialog box and create an analysis expression using the counters and/or timers set up with parameters.

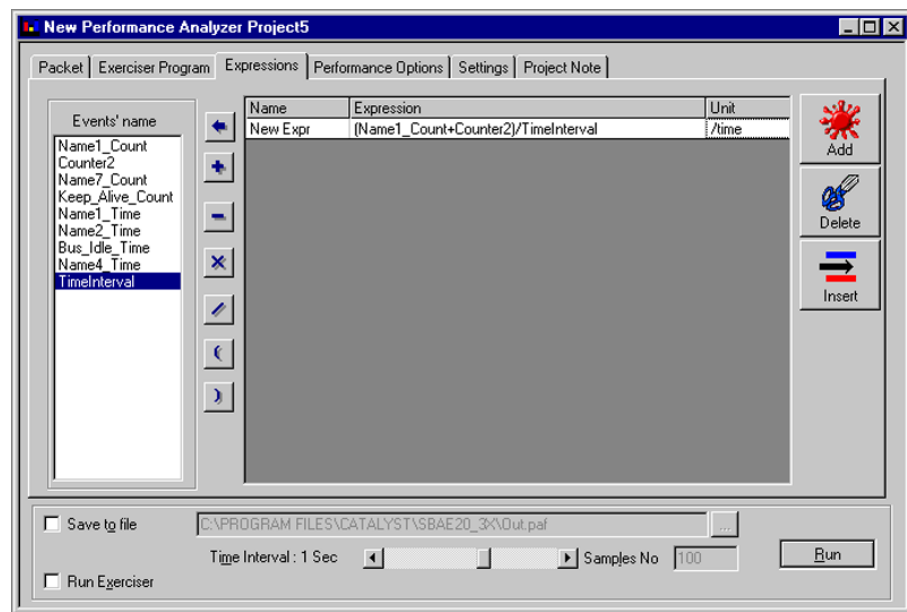


Figure 88 Creating an Expression

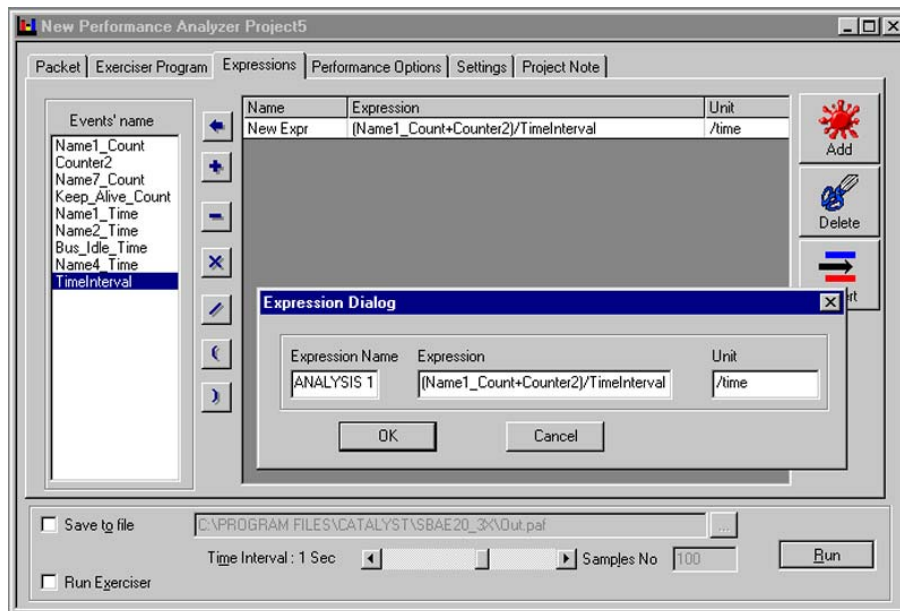
Performance Analysis

Create an expression

Click the **Add** button to open a blank expression line. You can enter the expression by double-clicking the event's name and an operator or by double-clicking the blank expression line to open the Expression Dialog and typing an expression directly.

Name the Expression

Double-click the **New Expression** name field, enter an expression name in the Expression dialog, and click **OK**.



Repeat for additional expressions.

Choose Results Display

Click the **Performance Options** tab to view the Performance Options setting dialog box.

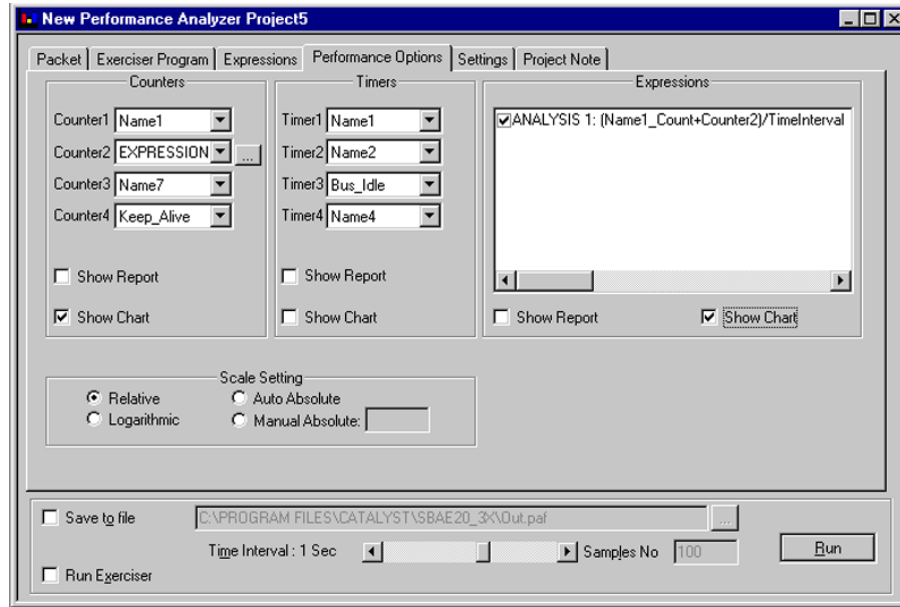


Figure 89 Choosing Display Options

4. Choose the counters, timers, or expressions for display and the type of display, for example, report, chart, or both.
5. Click the **Setting** tab to view the Performance Analysis settings dialog box.

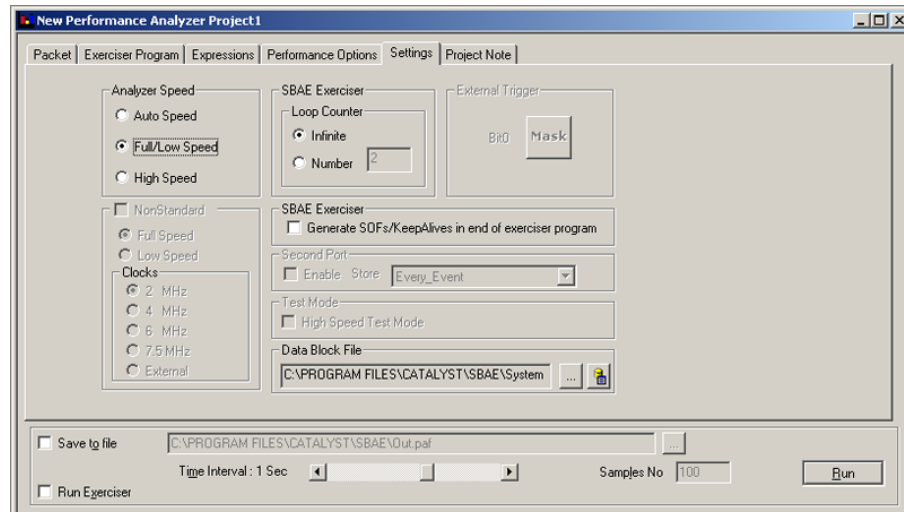


Figure 90 Performance Analysis Project Settings

Run Exerciser

Click the **Run Exerciser** check box if you are using an exerciser program.

Performance Analysis

Choose Time Interval

Drag the **Time interval** scroll bar to set the time interval for the performance analysis.

Loop Exerciser

Choose exerciser looping by checking **infinite** or **number** with a number of loops entered in the adjacent edit box.

Project Note

To attach a descriptive note to the project, click the **Project Note** tab and enter information about the project.

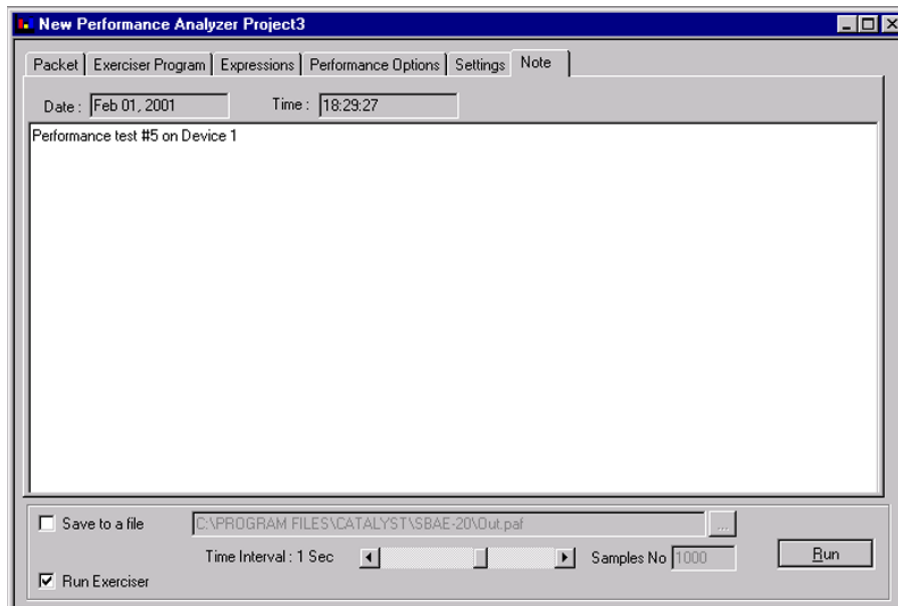
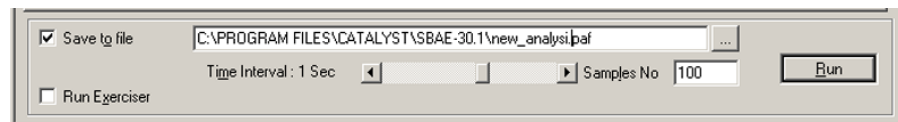


Figure 91 Project Annotation Dialog Box

Save to file

Click the **Save to file** checkbox to save the Performance Analysis in a file for later review.



Unique File

By default, the output is saved to **Out.paf** every time that a performance analysis is run, providing that the **Save to file** check box is checked. To save to a unique file, highlight the file name and enter a unique file name.

Run

Click **Run** to start the analysis and the real time results display.

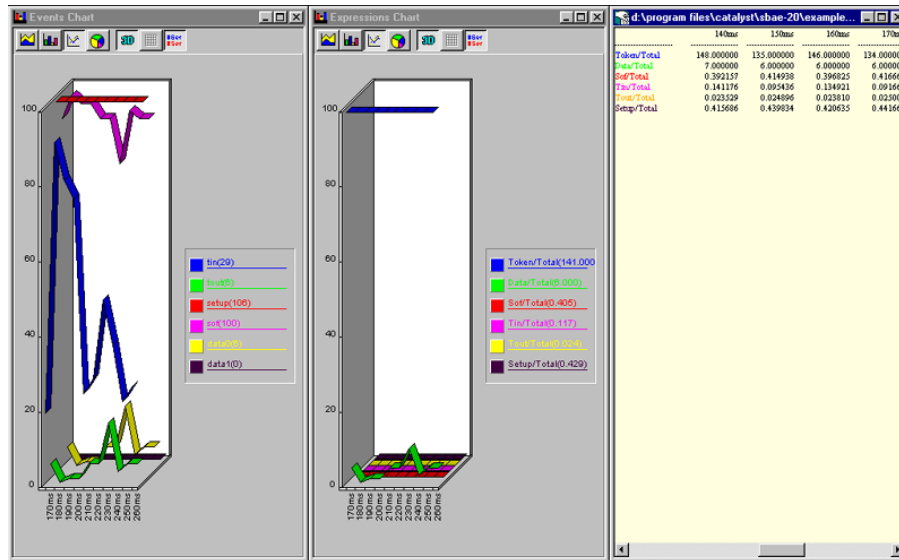


Figure 92 Performance Analysis Real Time Display

To stop the performance analysis, click the



Stop button on the main toolbar.

Alternate Display Format

You can choose to display the result as 2D, 3D, and so on, by clicking the corresponding **Graphics Setting** on the Performance Analysis display toolbar.

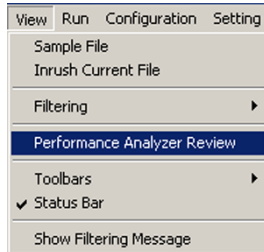


Performance Analysis

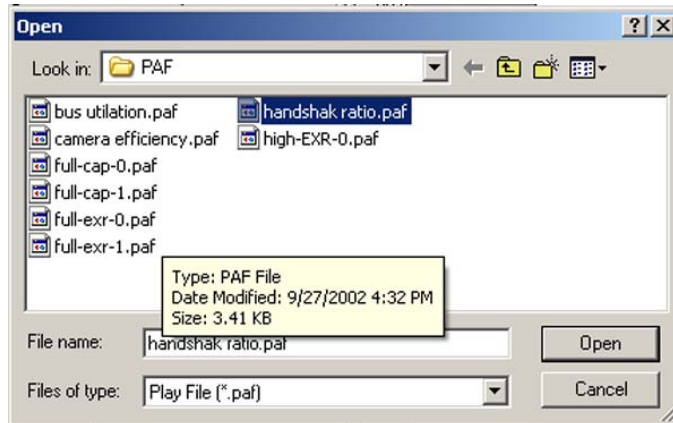
Saved Performance Analysis Review

Conquest M2 can review the results of previously performed and saved performance analyses.

Click **View** on the toolbar and then select **Performance Analyzer Review**.



The **Open** dialog box opens.



Select a previously generated performance analysis file *.paf and click **OK** to open the performance analysis window for the saved analysis.

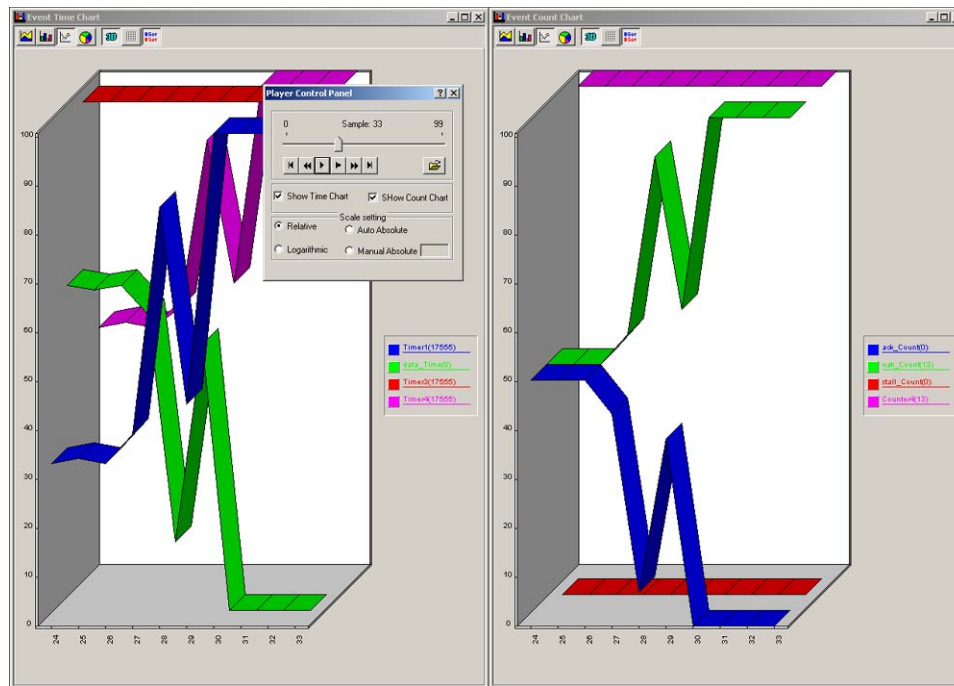


Figure 93 Review of Previously Saved Performance Analysis

To perform the review, you can drag the scroll bar on the Play Control Panel to a location, or command it to step through the analysis automatically.

Timing Analysis (Optional)

Note: High Speed Timing Analysis is not available in Conquest. It is available in Conquest M2 (see “Conquest M2 versus Conquest” on page 1).

Easy Mode Timing Analysis

You can perform a timing analysis only or together with a protocol analysis. To perform a Timing Analysis:

1. Define a trigger on the **Data Capture and Trigger Page** or check the **Trigger on Timing Pattern** and specify pattern parameters.
2. Click the **Timing Analyzer** tab and check either the **Timing Analyzer** or the **Both Analyzers** button in the Enable area as shown in Figure 94.
3. Click the **Run** button.

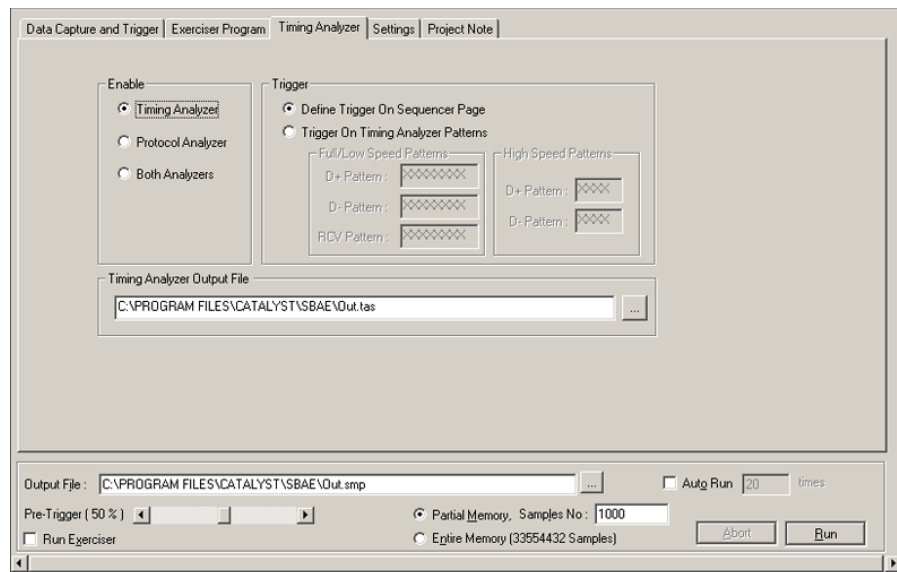


Figure 94 Enabling Timing Analysis

4. Click Run.

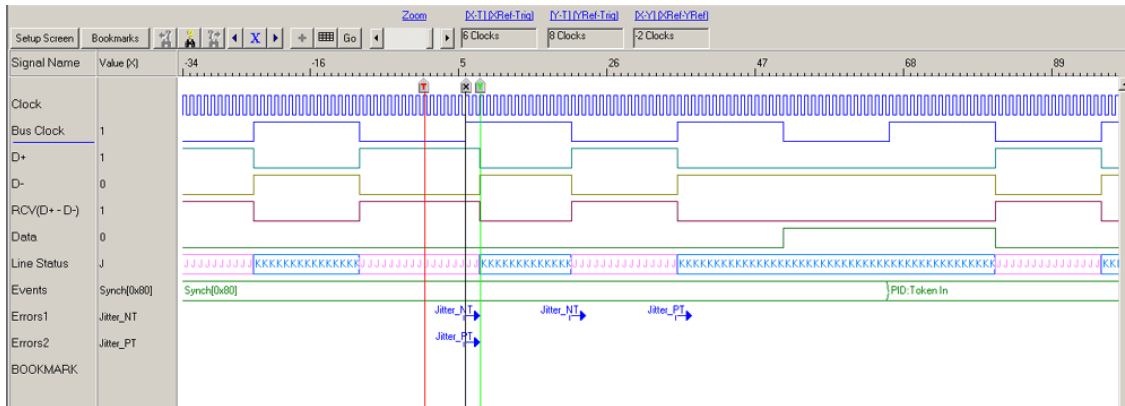


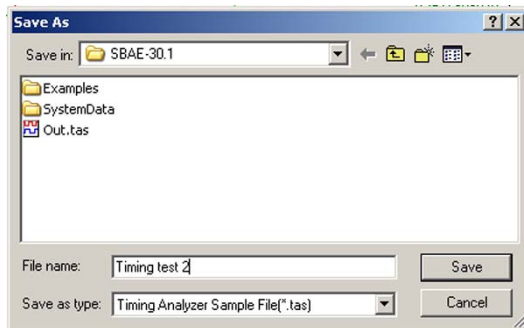
Figure 95 Timing Analysis Result

Using Cursors

You can use the cursors to make measurements within the display. For a description of using the cursors, see “Using the Cursors and Bookmarks” on page 188.

Saving the Result

To save the timing analysis result for later review, click **File** on the main toolbar and choose **Save As** to open the Save As dialog.



Enter an appropriate file name and click **Save**.

Advanced Mode Timing Analysis

You can perform a timing analysis only or together with a protocol analysis. To perform a Timing Analysis:

1. Define a trigger on the **Sequencer** Page or check the **Trigger on Timing Pattern** and specify pattern parameters.
2. Click the **Timing Analyzer** tab and check either the **Timing Analyzer** or **Both Analyzers** button in the Enable area, as shown in Figure 96.
3. Click the **Run** button.

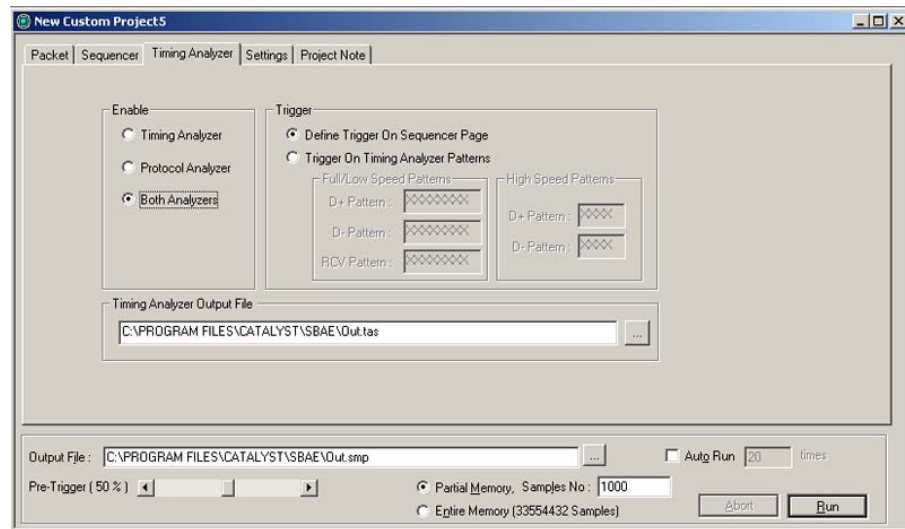


Figure 96 Advanced Mode Timing Analysis Dialog

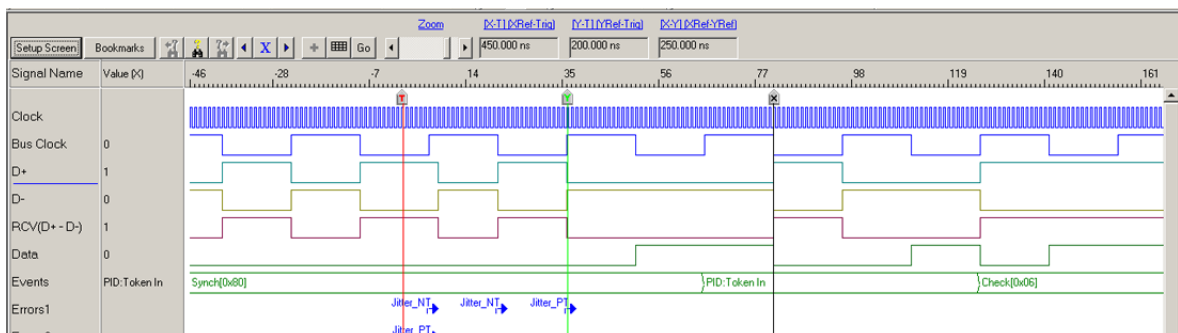


Figure 97 Timing Analysis Result

RCV in Full/Low Speed only

The RCV Pattern specification is not available in High speed mode.

Results Display

You can customize the results display by adding or removing signals from the display and using cursors and bookmarks to make timing measurements and mark results for later review.

For ways to customize the results display, see “Timing Analysis Display” on page 185. For filtering, see “Filter” on page 182. For using cursors and bookmarks, see “Using the Cursors and Bookmarks” on page 188.

Timing Analyzer Errors

Invalid EOP	FS/LS: Detected short EOP or long EOP.
False EOP	Detected misplaced EOP.
Invalid Bit Width	FS/LS: Width of bit is outside nominal range, accounting for Jitter_NT.
Bit Stuff	Detected seven consecutive 1s.
Bad Sync	FS/LS: Synch value is not 80h.
Jitter_NT (Next Transition Jitter)	FS/LS: Jitter measured from each transition to next transition exceeds maximum.
Jitter_PT (Paired Transition Jitter)	FS/LS: Jitter measured for any set of paired (JK-to-next JK transition or KJ-to-next KJ transition) exceeds maximum.
Inter-packet Delay	FS/LS: Delays measured between packets moving in the same direction are not within limits. HS: Delays measured between packets moving in the same direction are not within limits.
End-to-end Delay	FS/LS: Packet response exceeds limit.
Skew	D+ and D- transitions did not occur simultaneously.

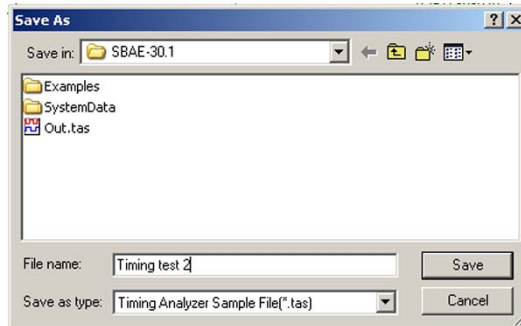
Limitations

Slightly different rising and falling edge thresholds of D+ and D- signals may display transitions, that ideally occur simultaneously, as one clock period apart. Similarly, this also may occur in the RCV signal.

Timing Analysis (Optional)

Saving the Result

To save the timing analysis result for later review, click **File** on the main toolbar and choose **Save As** to open the Save As dialog.



Enter an appropriate file name and click **Save**.

Device Emulation (Optional)

This mode allows you to operate Conquest M2 as a Device for testing the USB host operation. Conquest M2 can be a Low, Full, High, or a High/Full speed Device. Each Device can include up to three configurations with two interfaces each. You can assign a total of seven endpoints to these interfaces. Figure 98 shows a functional connection for the Conquest M2 operating as a Device.

Conquest M2 allows two ways to set up as a device:

- **Easy Mode:** Requires minimum setup.
- **Advanced Mode:** Offers more powerful device emulation by allowing you to define custom responses to different commands.

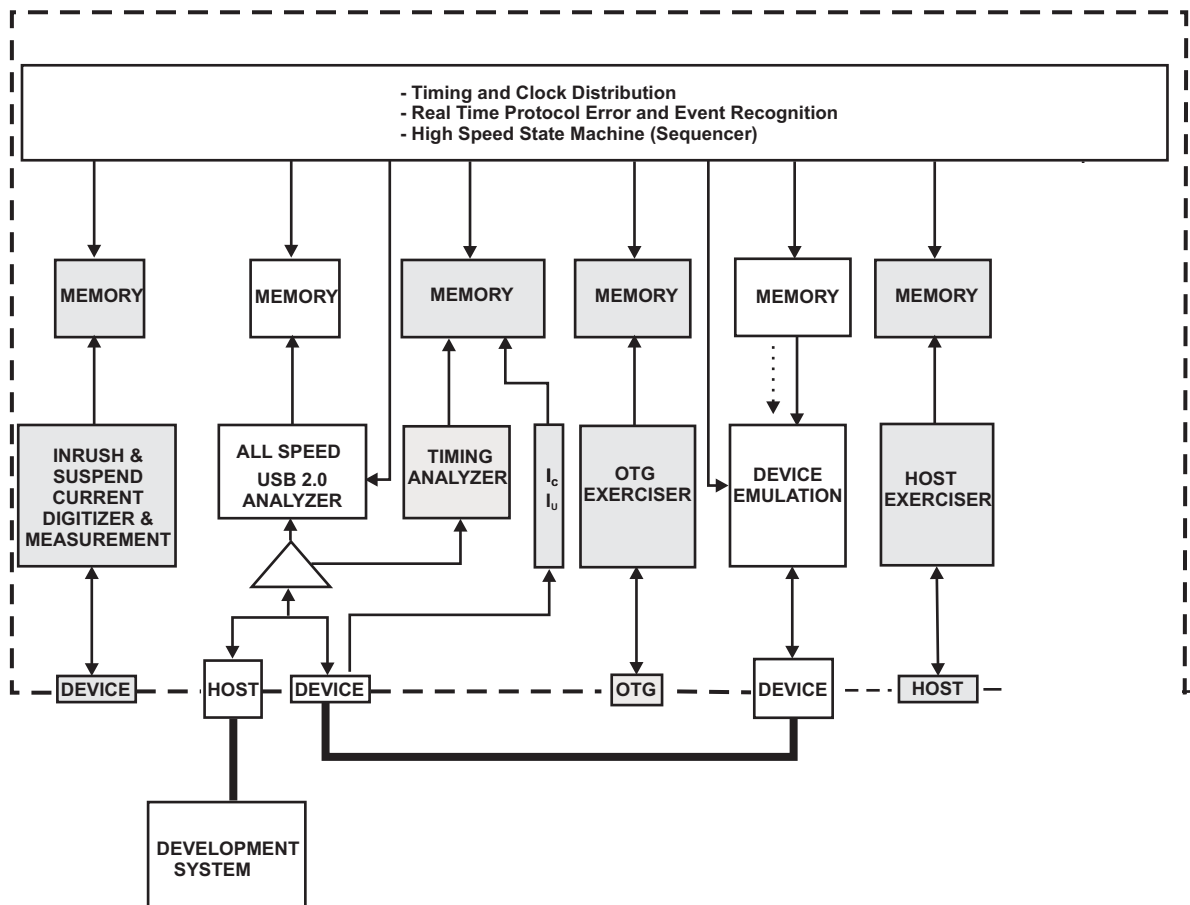


Figure 98 Conquest M2 Operating as a Device Functional Connection

Device Emulation (Easy Mode)

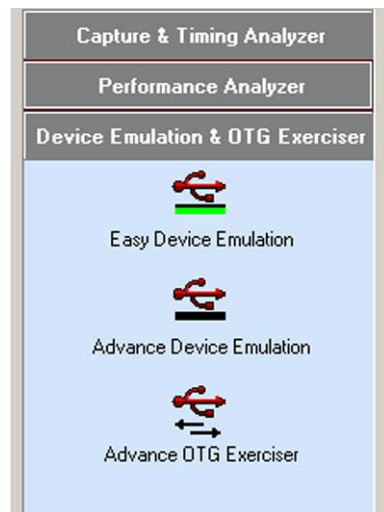
Programming the Device

To open a new Easy Device Emulation project, select **File > New** and choose **Easy Device Emulation**.

Analyzer/Host Exerciser Project	Alt+F1
Advanced Device Emulation	Alt+F12
Easy Device Emulation	Alt+F3
Advanced OTG Dual-Role Device Project	Alt+F11
Data Blocks	
Script	

You can also open a default Easy Device Emulation Project.

To open the default project, click the **New Project** button on the main toolbar to open the Project Setup dialog and then click **Device Emulation & OTG Exerciser**.



Click the **Easy Device Emulation** icon to launch the default device emulation.

Device Emulation (Optional)

The Device Emulation dialog opens with a default hierarchy as shown in Figure 99.

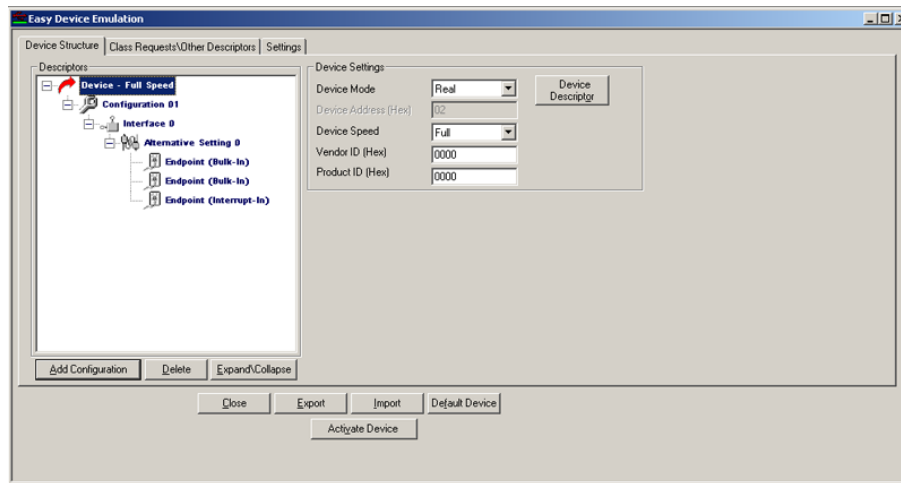


Figure 99 Device Mode Setup Dialog

You can add configurations to a total of three. Each configuration by default opens with one interface. You can add interfaces to total two per configuration. Each interface by default opens with one Alternative Setting with one endpoint. You can add Alternative Settings to total four for each interface, but no more than seven total for the Device. A Device can have a total of seven endpoints. You can enter descriptor values at each level of the hierarchy.

In addition to support for most standard requests defined for USB, you can configure the Device to respond with up to four arbitrary standard request data descriptors and up to seven String Descriptor strings and to acknowledge up to eight class requests.

Define the Device

1. Assign the Device speed from the Device Speed drop down list as **Low**, **Full**, or **High/Full**. Note that in High/Full speed, you must set up two independent Devices.
2. Enter **Vendor** and **Product ID** in the appropriate text box.

3. Define Device Descriptors. Click the **Device Descriptor** button and enter the parameters for the Device in the Descriptor dialog, as shown in Figure 100.

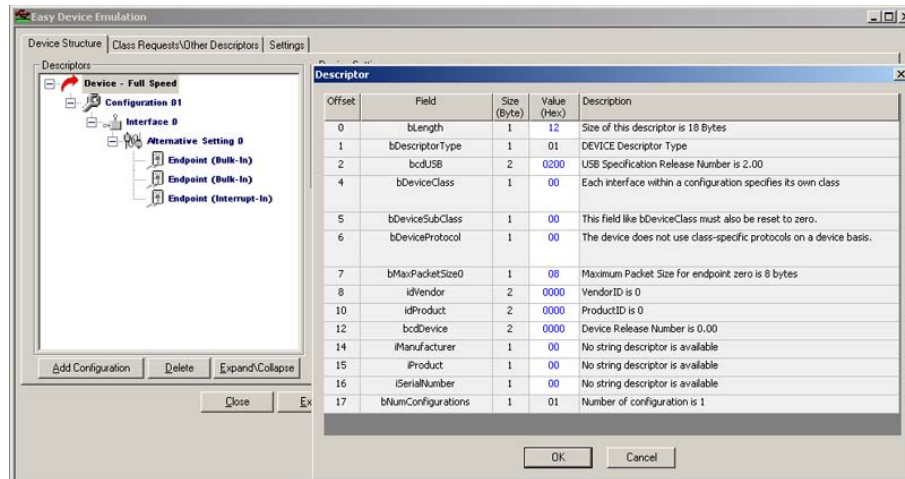


Figure 100 Defining Device Descriptors

Define Configuration

A Device initially opens with one configuration. You can add additional configurations, up to a total of three, by clicking a Device then the **Add Configuration** button.

To define configuration descriptors for each configuration, click the **Configuration Descriptor** button and enter parameters for the configuration in the corresponding descriptor dialog.

Enable Configuration Class Descriptor

Clicking this checkbox returns additional data as part of the Configuration Descriptor. (up to 128 bytes).

Define Interface

Each configuration opens by default with one interface comprised of one Alternative Setting with one endpoint. You can add additional interfaces up to four per configuration, but not exceeding seven total for a Device, by clicking a configuration then the **Add Interface** button.

To define alternative setting descriptors for each alternative setting, click the **Alternative Setting** button and enter parameters for the alternative setting in the corresponding descriptor dialog.

Enable Alternative Class Descriptor

Clicking this checkbox returns additional data as part of the Alternative Descriptor (up to 128 bytes).

Device Emulation (Optional)

Define Endpoint

A Device allows up to a total of seven endpoints under defined alternative settings. You can add endpoints by clicking an alternative setting then the **Add Endpoint** button.

To define the endpoints, select an endpoint and click the **Endpoint Descriptor** button to display the Endpoint Definition dialog, shown in Figure 101.

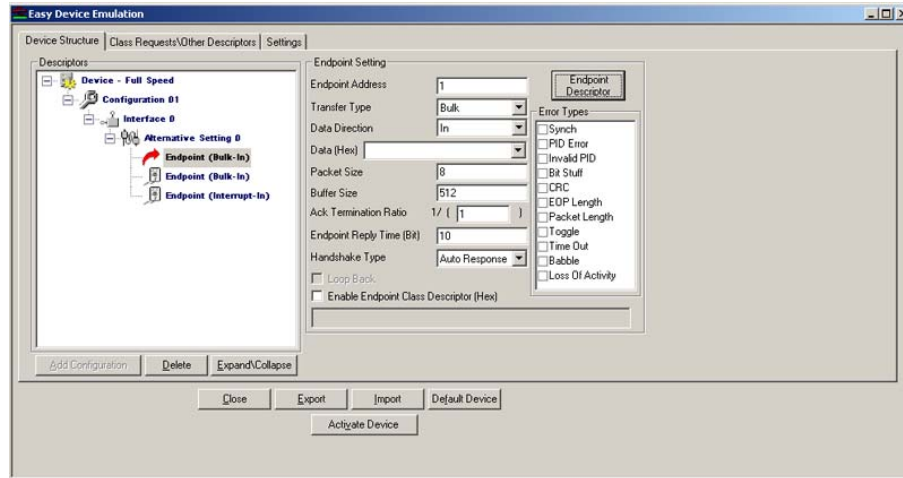


Figure 101 Endpoint Definition Dialog

Transfer Type

Click the down arrow next to the Transfer Type item list and choose **Isochronous**, **Bulk** or **Interrupt**.

Data Direction

Click the down arrow next to the Data Direction item list and choose **IN** or **OUT**.

Data (Hex)

For transfer types using data, enter data values in hex or choose from pre-defined data by clicking the down arrow next to the Data(Hex) item list and making a choice.

Packet Size

Enter a packet size (bytes/packet) for the device to returned. (Default is 64.) Data returns to the Host in N bytes/packet increments.

Buffer Size

Enter a buffer size in bytes. (Default is 512). This defines how much data the endpoint can accept.

ACK Termination Ratio

Defines the number of NAKs returned before an ACK is returned. (Default is 1)

Endpoint Reply Time (bit)

Enter a delay time between the Host packet and the Conquest M2 Device reply packet.

Handshake Type

Click the down arrow next to the Handshake Type and choose an appropriate response type. This defines the response of the Conquest M2 Device for the selected endpoint.

Error Types

Check **error type** boxes from the available errors listed in the Error Types group box for errors to introduce.

Loopback

Checking this option for OUT endpoint causes the data received from the Host for the OUT endpoint to return to the Host on the IN endpoint with the same endpoint number. The IN endpoint is automatically selected when you check Loopback. Do not define another IN Endpoint, because checking Loopback does it automatically. **Packet Size** controls the data packet size for the IN endpoint. **Buffer Size** controls how many bytes to buffer for the OUT endpoint. If this limit is exceeded, the Endpoint responds with a NAK.

Enable Endpoint Class Descriptor

Clicking this checkbox returns additional data as part of the Endpoint Descriptor.

Define Additional Requests

Additional requests include requests for Independent Descriptors, String Descriptors, and Class Requests. You can define Device Emulation responses to these requests by clicking the **Class Requests\Other Descriptors** tab.

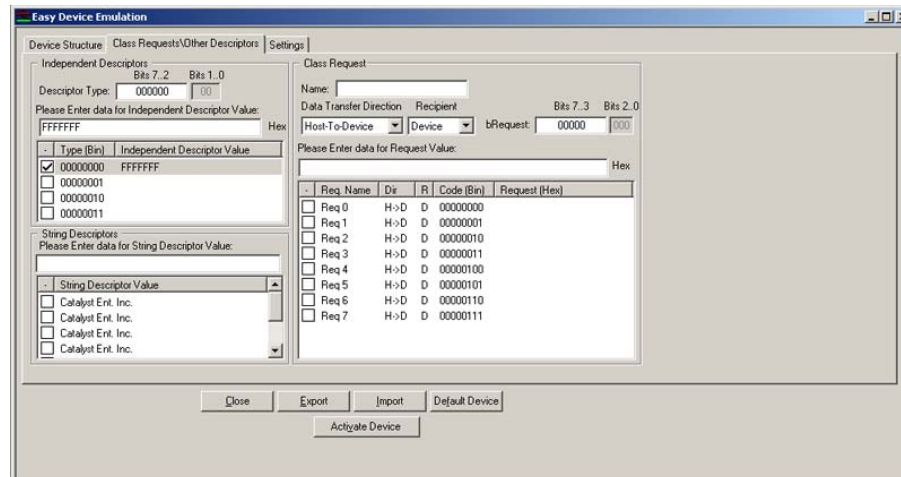


Figure 102 Additional Requests Response Definition

Device Emulation (Optional)

Independent Descriptors

To define Independent descriptors, enter a binary **Descriptor Type**, click the check box for that **Descriptor Type**, and enter the data (up to 256 bytes) to return in response to a Get Descriptor request. Responses support requests with index 0.

Note: You can set only the six most significant bits of the Descriptor Type. **Do not** set the descriptor type to the same value as any standard USB descriptor.

String Descriptors

To define String Descriptors, enter a value string (up to 128 characters) in the String Descriptors text box and click a check box to assign it.

Class Requests

To define a Class Request, click a request name check box and enter the data for the Request Value. Choose the Data Transfer direction and Recipient.

Importing Scan Descriptors

You can import previously defined Scan Descriptor *.sdv files by clicking the **Import** button and choosing an appropriate file from the SDV folder.

Non Standard Clocks

If the device is to operate at a non-standard clock rate, click the **Settings** tab to display the clock selection dialog.

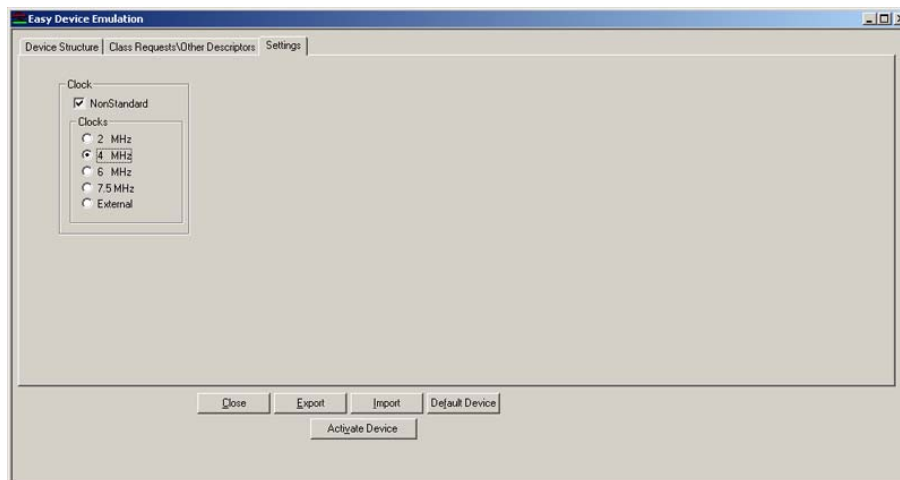


Figure 103 Using Non Standard Clocks

To use a non-standard clock, click the **Nonstandard** check box in the Clocks area and then choose one of the pre-set clock values available or check **External** and apply an external clock.

You can apply an external clock in ranges of 400KHz - 8MHz for High Speed and 1KHz-100KHz, 400KHz - 8MHz for Full/Low speed device emulation.

Activate Device

Click the **Deactivate Device** button to activate the Device. Note the “Device: Active” indication on the taskbar displays green.



Note 1 After setup is complete, you do not have to activate the device immediately. You can close the dialog, perform other tasks, and then activate the device.

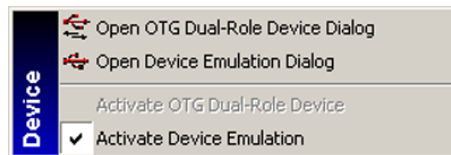
Note 2 To make the just-defined Project the Default Project, click the **Save** button on the main toolbar. However, to leave a previously defined Default Project intact, select **File** on the main toolbar and choose **Save As** to save the project with a new name.

Delayed Device Activation

Click the down arrow next to the Device button on the Task Bar



to open the Device dialog and then check the **Activate Device Emulation** check box.



Note that this menu also applies to OTG Exerciser activation.

Set Up Analysis

Set up an analysis in either Easy mode (see page 37) or Advanced mode (see page 59) and click **Run**.

Connect to Device Port

Connect the cable to the Device port or click the **Attach Device** button on the main toolbar.

Device Emulation (Optional)

Endpoint Errors Generated by OTG DRD & Device Emulation

Synch	Generates the KJKJKJJJ pattern for Synch field. Not available in HS
Pid	Check field contains the same bits as PID field.
Invalid Pid	Uses Reserved PID field value of 0000b and Check field value 1111b.
Bit stuffing	Disables bit stuffing. Not available in HS
CRC	Incorrect CRC for data packets Not available in OUT endpoints
EOP Length	FS/LS: The transmitted SE0 is approximately 1 bit time in width. HS: first bit of EOP is not inverted.
Packet Length	IN: Short data packet without CRC OUT: Handshake packet appended with extra bytes
Toggle	Does not toggle the data packets bit (always 1). Not available for OUT endpoints
TimeOut	Response is sent after a period, if time is longer than the interpacket period specified by USB.
Babble	Generates a long packets that cross frame boundaries.
Loss of Activity	EOP is not transmitted, bus returns to idle at end of packet. Not available in HS

Device Emulation (Advanced Mode)

Programming the Device

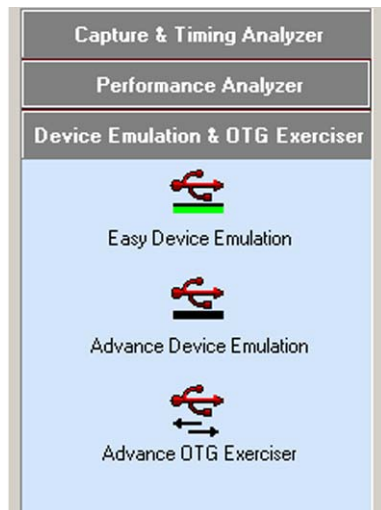
To overcome some of the limitations of Easy Device Emulation, such as a constant data payload for **In** endpoints, Advance Device Emulation provides the means to respond with different pre-defined data to host requests by using Command Sets.

To open a new Advanced Device Emulation project, select **File > New** and choose **Advanced Device Emulation**.

Analyzer/Host Exerciser Project	Alt+F1
Advanced Device Emulation	Alt+F12
Easy Device Emulation	Alt+F3
Advanced OTG Dual-Role Device Project	Alt+F11
Data Blocks	
Script	

You can also open a default Advanced Device Emulation Project.

To open the default project, click the **New Project** button on the main toolbar to open the Project Setup dialog and then click **Device Emulation & OTG Exerciser**.



Device Emulation (Optional)

Click the **Advance Device Emulation** icon to launch the default Advanced device emulation.

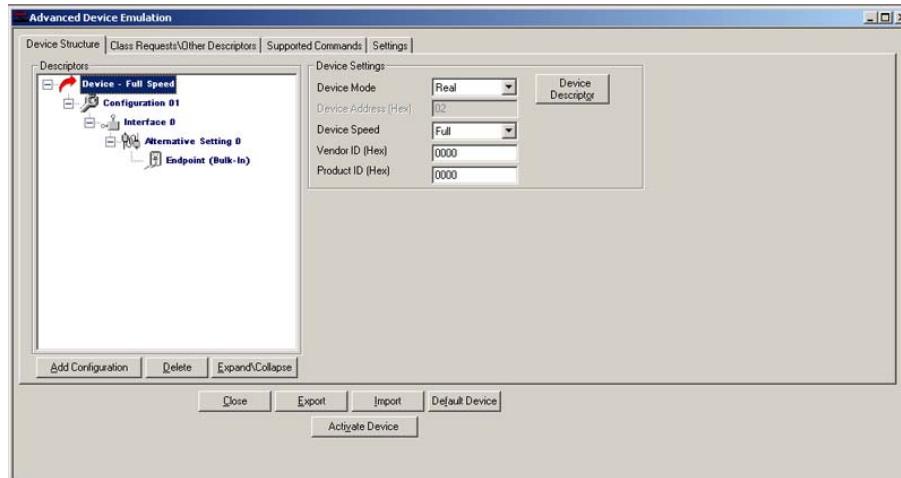


Figure 104 Advanced Device Mode Setup

Device structure and supported class requests are defined in the same manner as Easy Device Emulation as described starting on page 135. Note however that all of the endpoint behavior settings are grayed out because the endpoint responses are programmed in the supported commands page.

Define Command sets on the Supported Commands page. Each Command Set includes a command stage, which has to be a **setup** or **out** transaction to a non-zero endpoint. This data stage can have one or more in or out transactions to any endpoint and may contain one last in or out transaction as a status stage.

The 248 KB memory of the Advanced Device Emulation is divided into 31 command sets. Each command set occupies 8 KB that includes all the fields of the packets, like PID, data payload and CRC. In cases where a Command Set is larger than 8 KB, it occupies more than one set, so the total number of Commands Sets is reduced. For instance, if one command uses 12 KB, 2 command sets are assigned to it, so the total commands number is reduced to 30 from 31. Defining a Command Set

1. Click the **Supported Commands** tab to open the Supported Commands Entry dialog, as shown in Figure 105.

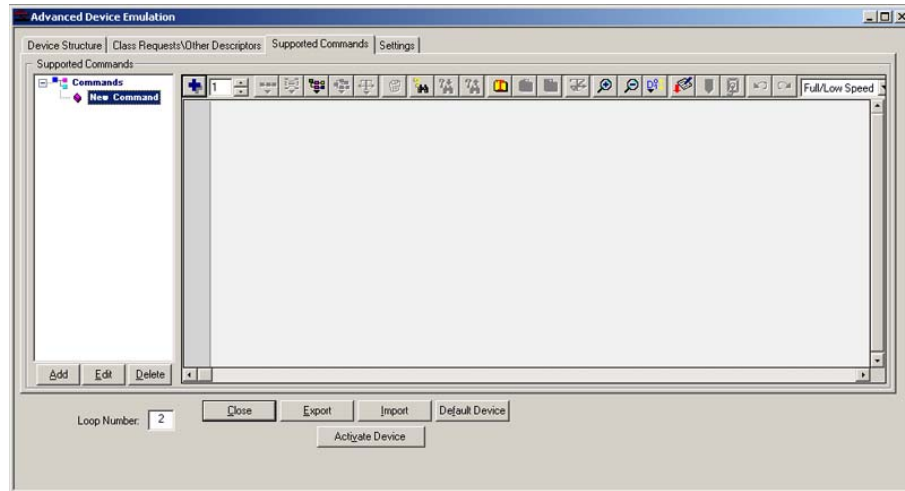


Figure 105 Supported Commands Entry Dialog

2. Click the **Add** button to open the command name dialog.

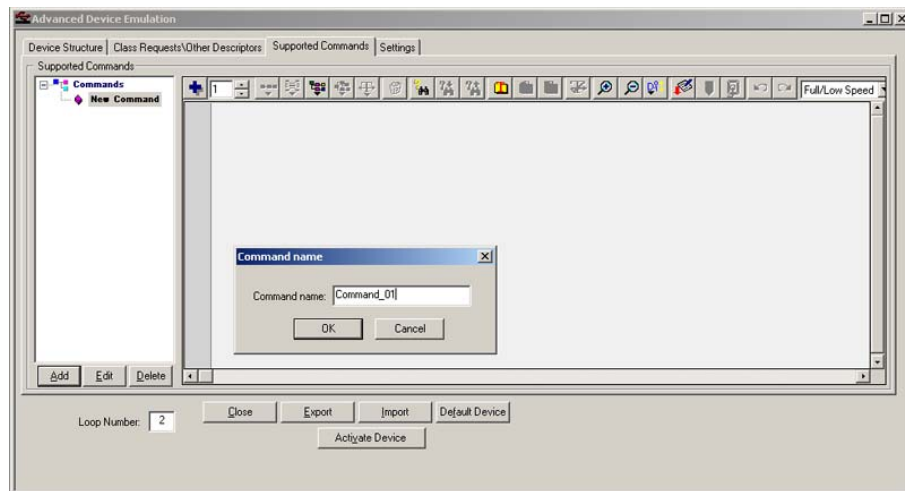


Figure 106 Adding a New Command

3. Enter the command name and click **OK**.
4. You can also select the default **New Command** and rename it.
5. Click the transaction button on the dialog toolbar to display the transaction drop-down list.
6. Choose a **Setup** or **Out** transaction as the command stage and make sure to change the endpoint number to a non-zero one.
7. Edit the data payload of the data packet to a value.
8. Complete the data and status stage of the command set by adding addition in or out transaction to any endpoint.

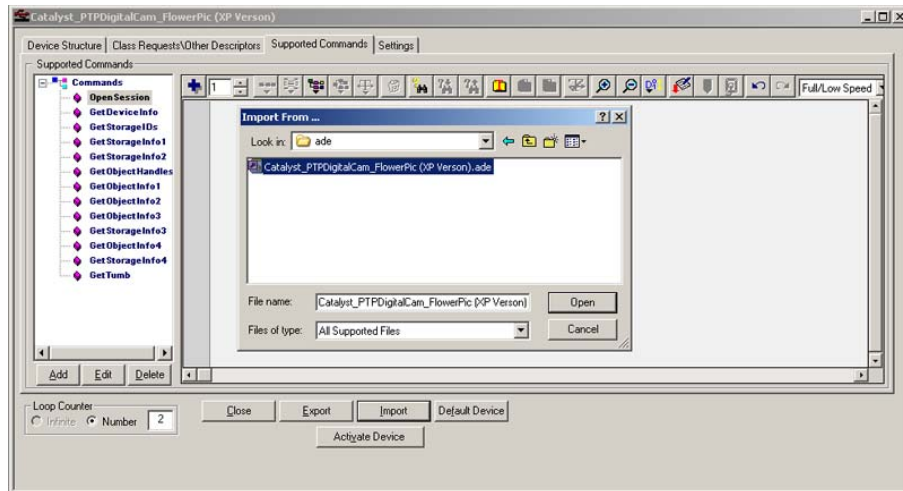
Device Emulation (Optional)

9. You can define additional command sets by repeating the above steps.

Note To make the just defined Project the Default Project, click the **Save** button on the main toolbar. To leave a previously defined Default Project intact, click **File** on the main toolbar and choose **Save As** to save the project with a new name.

Importing a Pre-defined Project

1. Click the **Import** button to open the Import from dialog.



2. Choose a previously defined Project and click **OK**.

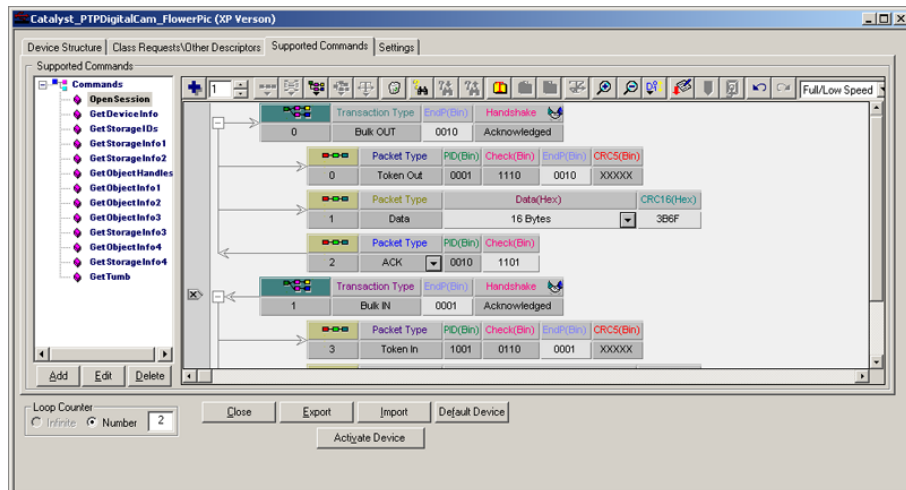


Figure 107 Previously Defined Command Set



Click the **Auto Data Toggle** button on the main toolbar and choose **Auto Toggle** to toggle sent data per specification or **Manual Toggle** to send data as programmed.

OTG Exerciser (Optional)

This option provides for testing OTG development devices. The OTG Exerciser essentially is a Full/High speed OTG Dual-Role Device that can act as a host or a peripheral.

OTG Exerciser as an A-Device

To operate the OTG Exerciser as a DRD A-Device, connect it with a DUT OTG B-Device as shown in Figure 108. To operate as an A-Device, plug the Mini-A side of the cable into the OTG exerciser port.

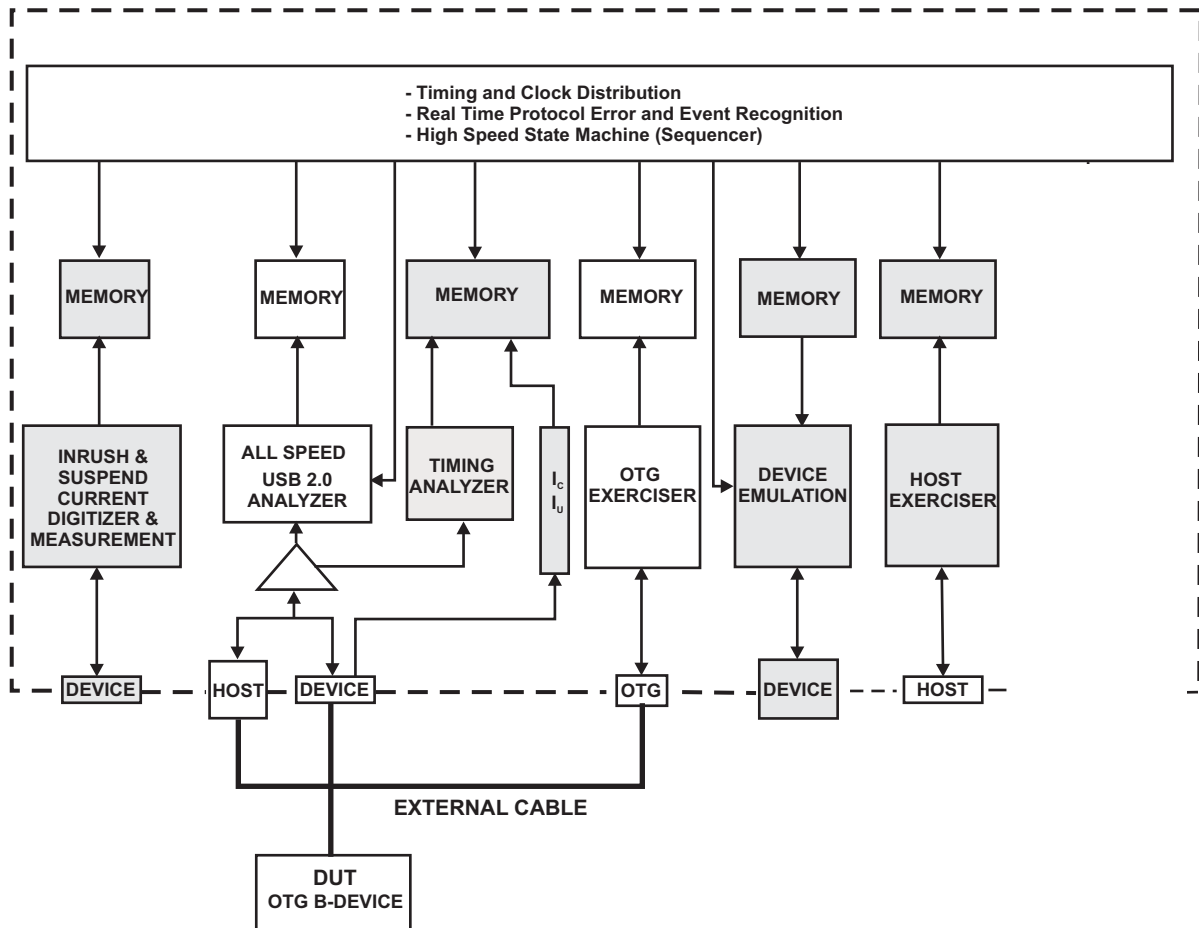
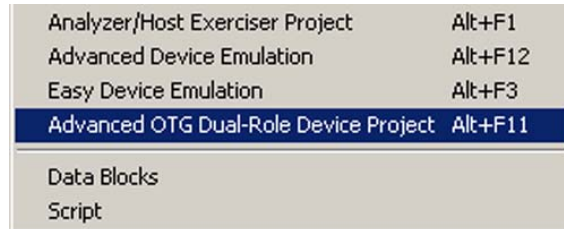


Figure 108 OTG Exerciser as DRD Device A-Device Connection

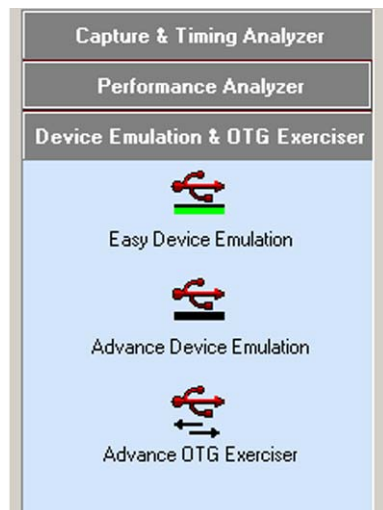
Programming the Device

To open a new Advanced OTG Dual-Role Device Project, select **File > New** and choose **Advanced OTG Dual-Role Device Project**.



You can also open a default Advanced OTG Dual-Role Device Project.

To open a default project, click the **New Project** button on the main toolbar to open the Project Setup dialog and click **Device Emulation & OTG Exerciser**.



OTG Exerciser (Optional)

Click the **Advance OTG Exerciser** icon to launch the OTG Exerciser.

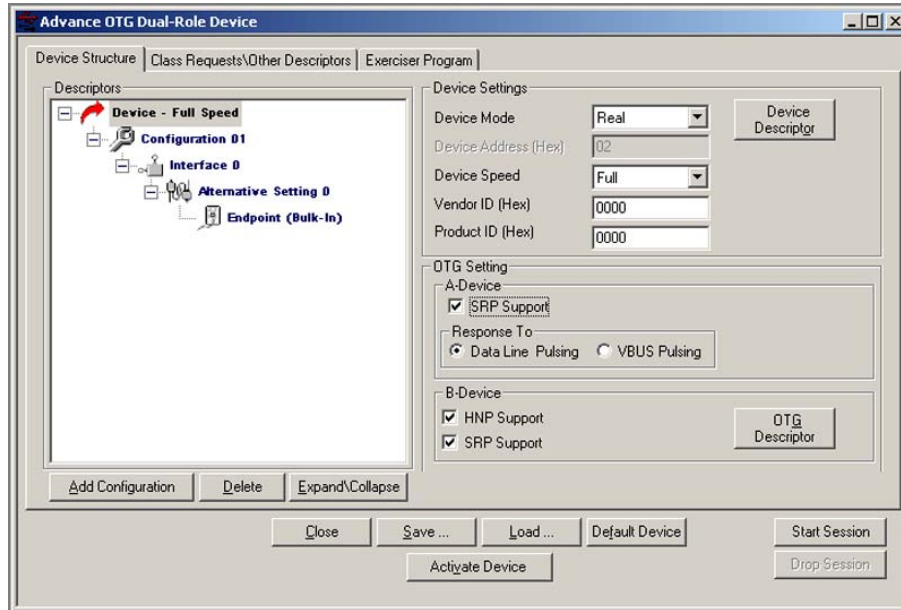


Figure 109 OTG DRD Setup Dialog

Define the Device

1. Enable/Disable OTG Exerciser response to SRP by checking or unchecking the A-Device **SRP Support** check box (Normally enabled).
2. If SRP is enabled, choose the response type as either **Data Line Pulsing** or **VBUS Pulsing**.
3. Assign the Device speed from the Device Speed drop down list as Full, or High. Note that the speed selection must match that of the DUT.

Create Exerciser Program

To program the OTG Exerciser, select the Program Exerciser Tab. Programming the OTG Exerciser is identical to programming the Host Exerciser as described starting on page 71. The following differences apply:

1. Not all of the commands available for the Host Exerciser are available for the OTG Exerciser. Commands not applicable to OTG are grayed out.
2. The OTG Exerciser speed is set in the Device Structure Dialog (See Figure 109).

- The OTG Exerciser loop counter is set in the Exerciser Program dialog.

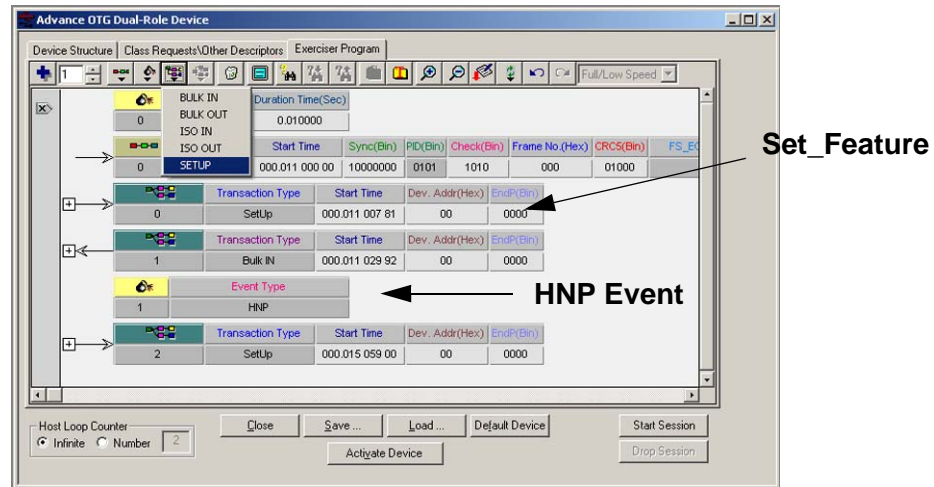


Figure 110 Inserting a Setup Transaction

Note: To enable HNP in the exerciser program, you must incorporate program lines as shown in Figure 110. The exerciser code example shown in Figure 110 specifies the DUT SetFeature(b_hnp_enable) and sets up a role switch (HNP Event). After the HNP is executed by both, the OTG Exerciser and the DUT, the DUT becomes the host and the exerciser becomes the peripheral. Additional exerciser code inserted after the lines shown is executed when the OTG Exerciser switches back to being a host. The SetFeature(b_hnp_enable) request is defined in a Data Block to use as data for the Setup Transaction.

Both Bus Idle and HNP cause the bus to suspend. However, for HNP, if the OTG exerciser detects a disconnect from the B-Device DUT, it automatically disconnects, regardless if the SetFeature(b_hnp_enable) is in the program or not.

Avoid using Hubs

It is not recommended to use Hub(s) between the DUT and the OTG Exerciser. To perform more extensive testing of DUT functionality, you can use the Host Exerciser

OTG Exerciser (Optional)

Define OTG Device Configuration

Programming the OTG device is performed in the dialogs of the Device Structure and Class Request\Other Descriptors tabs. The procedural steps for programming is identical to that described for Device Emulation starting on page 134.

Differences from Device Emulation

The **OTG DRD Setup** dialog opens with a default hierarchy as shown in Figure 109. You can add configurations to total 3. Each configuration by default opens with one interface. You can add interfaces to total 2 per configuration. Each interface by default opens with one Alternative Setting with one endpoint. You can add Alternative Settings to total 3 for each interface, but no more than 3 total for the Device. A Device is limited to a total of 4 endpoints. You can enter descriptor values at each level of the hierarchy. In addition to support for most standard requests defined for OTG, you can configure the Device to respond with up to four arbitrary standard request data descriptors and up to seven String Descriptor strings and to acknowledge up to eight class requests.

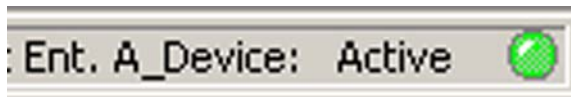
Note To make the just-defined project the Default Project, click the **Save** button on the main toolbar. However, to leave a previously defined Default Project intact, click **File** on the main toolbar and choose **Save As** to save the project with a new name.

Starting a Session

Once you have configured the device and completed an exerciser program, you must activate the device by clicking the **Activate Device** button.



Note: After a session starts, the status bar displays:



When a device is active and you change any parameter on any tab, you must reactivate the device, by deactivating and then reactivating the device.

Note: Device Emulation and the OTG Exerciser may not be active at the same time. To start a session, click the **Start Session** button or Wait for DUT (B-Device) to initiate SRP (If SRP has been checked).

Dropping a Session

A-Device controls the VBus and may drop it at any time. A session for the OTG Exerciser in this mode ends when the exerciser program completes. When a session is in progress and VBus is asserted, the **Drop Session** button can be active. Stop a session by clicking this button.

Note: There may be cases where the **Drop Session** button appears never to be enabled. This occurs in cases where a session is completed in a very short time. In general this button is enabled in cases with long program loops.

OTG Exerciser as a Peripheral B-Device

To operate the OTG Exerciser as a DRD Peripheral, connect it with a DUT OTG A-Device, as shown in Figure 111. To operate as a B-Device, plug the Mini-B side of the cable into the OTG exerciser port.

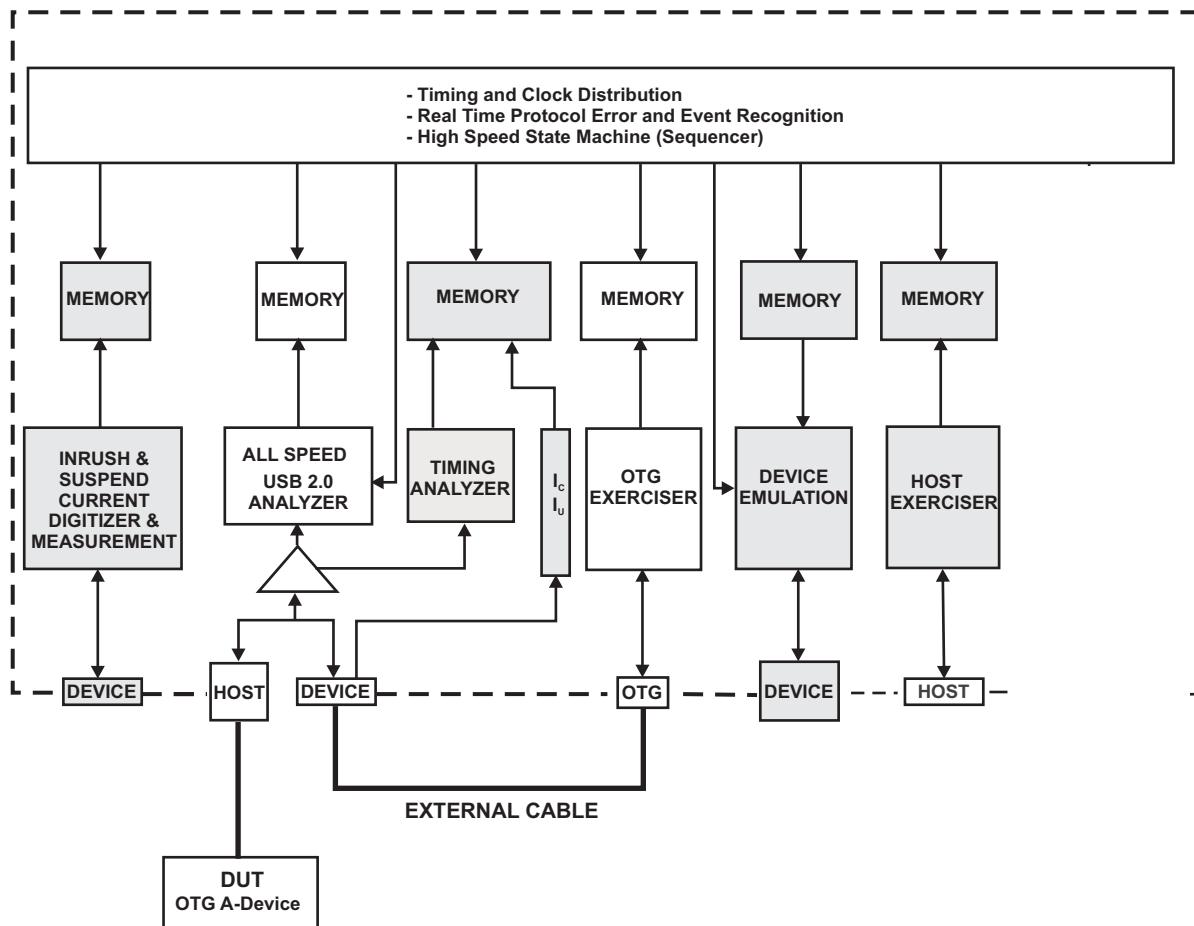


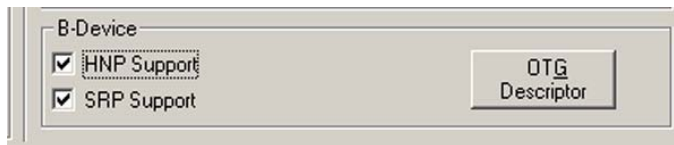
Figure 111 OTG Exerciser as DRD or Peripheral Device B-Device Connection

OTG Exerciser (Optional)

Define the Device

You can define the OTG Exerciser to be an SRP only capable peripheral or to include HNP support as well. To define the device:

1. Enable/Disable OTG Exerciser B-Device SRP and/or HNP response by checking or unchecking the corresponding check boxes in the Device Mode Dialog.



You can alternatively set these responses in the OTG Descriptor dialog by changing the bit assignment directly. Disabling **HNP Support** also causes the OTG Exerciser to stall the SetFeature(b_hnp_enable) request. Disabling **SRP Support** has no side effects.

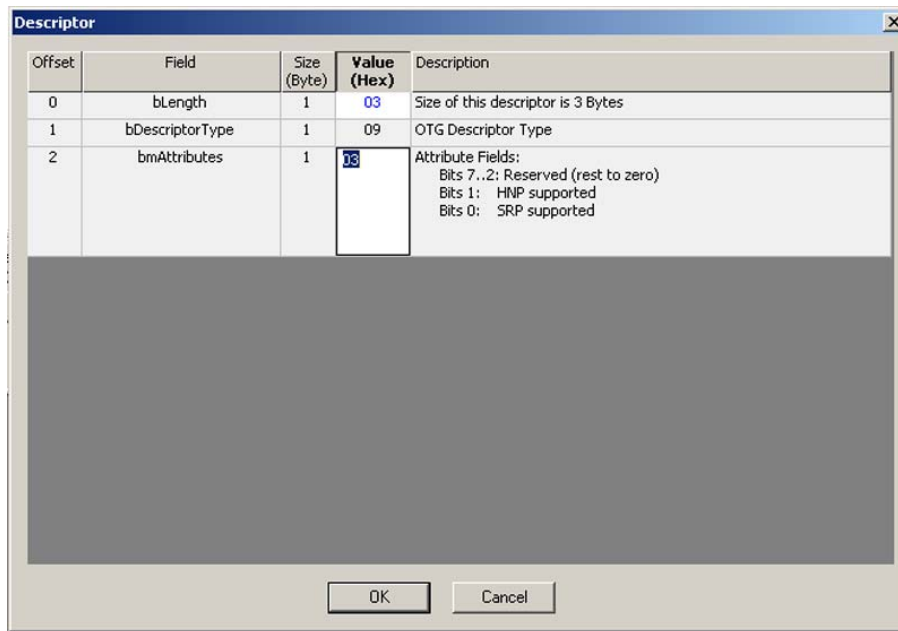


Figure 112 Selecting SRP/HRP in the OTG Descriptor Dialog

2. Assign the Device speed from the Device Speed drop down list as Full, or High. Note that the speed selection must match that of the DUT.

Define OTG Device Configuration

Programming the OTG device is performed in the dialogs of the Device Structure and Class Request\Other Descriptors tabs. The procedural steps for programming is identical to that described for Device Emulation starting on page 134.

Differences from Device Emulation

The **OTG DRD Setup** dialog opens with a default hierarchy as shown in Figure 109. You can add configurations to total 3. Each configuration by default opens with one interface. You can add interfaces to total 2 per configuration. Each interface by default opens with one Alternative Setting with one endpoint. You can add Alternative Settings to total 3 for each interface, but no more than 3 total for the Device. A Device is limited to a total of 4 endpoints. You can enter descriptor values at each level of the hierarchy. In addition to support for most standard requests defined for OTG, the Device can be configured to respond with up to four arbitrary standard request data descriptors and up to seven String Descriptor strings and to acknowledge up to eight class requests.

Create Exerciser Program

To program the OTG Exerciser, select the Program Exerciser Tab. Programming the OTG Exerciser is identical to programming the Host Exerciser as described starting on page 71. The following differences apply:

- Not all of the commands available for the Host Exerciser are available for the OTG Exerciser. Commands not applicable to OTG are grayed out.
- The OTG Exerciser speed is set in the Device Structure Dialog (See Figure 109).
- The OTG Exerciser loop counter is set in the Exerciser Program dialog.

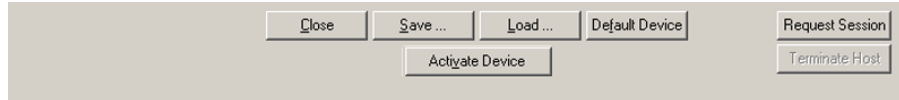
Note: This step is not necessary if you choose to operate the OTG Exerciser as an SRP capable peripheral only.

Note: You do not need to set an HNP event to perform HNP. This is done automatically when the OTG Exerciser is conditioned for HNP by the A-Device. After completing the exerciser program as a host, the OTG Exerciser automatically returns control back to the A-Device.

OTG Exerciser (Optional)

Requesting a Session

Once you have configured the device and completed an exerciser program (if required), you must activate the device by clicking the **Activate Device** button.



If at this time the OTG Exerciser does not detect presence of VBus, the Request Session button is enabled. You can now click the **Request Session** button and wait for DUT (A-Device) to initiate SRP.

Note: After a session starts, the status bar displays:



When a device is active and you change any parameter on any tab, you must reactivate the device, by deactivating and then reactivating the device.

Note: Device Emulation and the OTG Exerciser may not be active at the same time.

Terminating the Exerciser Program

If **HNP Support** is checked, then, when the OTG Exerciser B-Device becomes the host it executes the exerciser program. If the exerciser program is long or set up as a continuous loop, you can manually terminate it and return control back to the A-Device.

Note: There may be cases where the **Terminate Host** button appears never to be enabled. This occurs in cases where exerciser program is completed in a very short time. In general, this button is enabled with long program loops.

Host Negotiation Protocol (HNP)

A Host Negotiation Protocol (HNP) transfers control of a connection from the default Host (A-device) to the default Peripheral (B-device). The A-device conditions the B-device to allow it to control the bus, and then the A-device presents an opportunity for the B-device to take control of the bus.

To condition the B-device, the A-device sends a **SetFeature(b_hnp_enable)** command. Then the A-device suspends the bus to signal the B-device that it may now take control of the bus. If the B-device wants to use the bus at that time, it signals a disconnect to the A-device. If the A-device has enabled the B-device to become Host, then the A-device interprets the disconnect signal during suspend as a request from the B-device to become Host. The A-device completes the handoff by turning on its pull-up resistor on D+.

In the OTG Exerciser page, adding the event in the following figure causes the Conquest Pro to switch back to Device mode and Device to Host mode.

A screenshot of the OTG Exerciser event log. It shows a table with two columns: 'Event ID' and 'Event Type'. The first row contains the number '1' in the 'Event ID' column and 'HNP' in the 'Event Type' column. The 'Event Type' column header is highlighted in pink.

Event ID	Event Type
1	HNP

Session Request Protocol (SRP)

To conserve power, the OTG supplement allows an A-device to leave VBUS turned off when the bus is not being used. If the B-device wants to use the bus when VBUS is turned off, the B-device can use the OTG-supplement Session Request Protocol (SRP) to request the A-device to supply power on the VBUS.

The SRP has two methods that the B-device uses to request that the A-device begins a session: **data-line pulsing** and **VBUS pulsing**. The two signaling methods allow maximum latitude in the design of A-devices. An A-device is only required to respond to one of the two SRP signaling methods. When initiating SRP, a B-device must use both methods, to insure that the A-device responds. Any A-device, including a PC or laptop, can respond to SRP. Any B-device, including a standard USB peripheral, can initiate SRP. Dual-role devices can initiate and respond to SRP.

A session is the time period that VBUS is above the **Session Valid** threshold of a device. The A-device threshold is within the range defined by **VA_SESS_VLD**, and the B-device threshold is within the range defined by **VB_SESS_VLD**, as shown in the following table.

Parameter	Symbol	Conditions	Min	Max	Units
Input Levels:					
A-device VBus Valid	VA_VBUS_VLD		4.4		V
A-Device Session Valid	VA_SESS_VLD		0.8	2.0	V
B-Device Session Valid	VB_SESS_VLD		0.8	4.0	V
B-Device Session End	VB_SESS_END		0.2	0.8	V

When a session starts, the A-device defaults to the role of Host. During a session, the Host Negotiation Protocol (see preceding section) can transfer the role of Host back and forth between the A-device and the B-device any number of times. The session ends when VBUS falls below the A-device **Session Valid** threshold.

How to Turn Off the VBUS (in an OTG Script)

To turn off the VBus, click the **Drop Session** button in the OTG Exerciser GUI.

How Long Does Conquest Take to Drop D+ Pull-up?

After detecting the sequence **Chirp K-J-K-J-K-J**, the USB Analyzer disconnects the D+ pull-up resistor in 500 microseconds.

Current Measurement (Optional)

Current measurements are performed using the USB 2.0/1.x Analyzer, Exerciser, and Inrush Current connector and require unplugging and reconnecting the Device cable in accordance with on-screen commands.

Figure 113 shows the functional connection used for measuring Unconfigured and Suspend current.

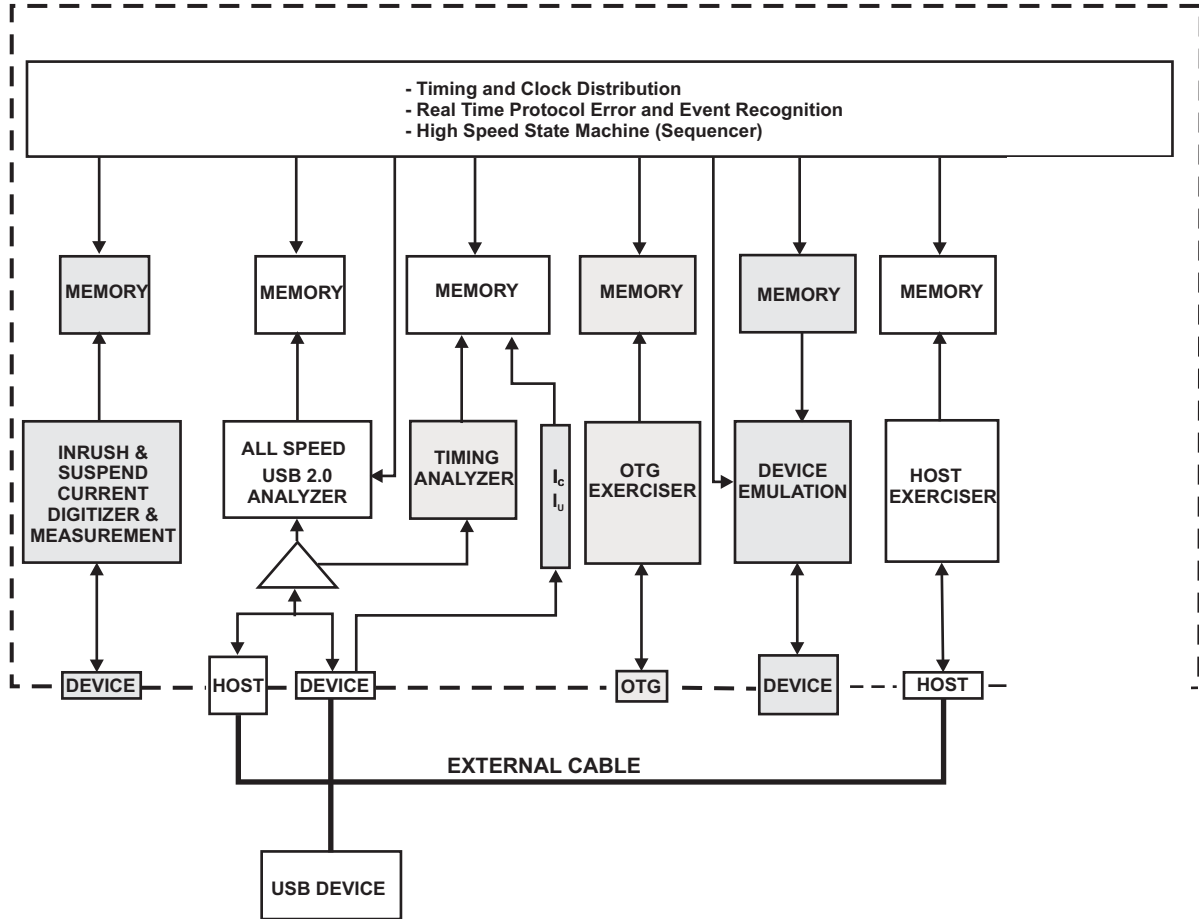


Figure 113 Unconfigured and Suspend Current Measurement Functional Connection

Unconfigured Current Measurement

The Unconfigured current measurement requires the interaction of the exerciser to place the Device in the unconfigured state. Prior to making the measurement you must connect the external cable provided between the USB 2.0/1.x Analyzer Host connector and the Exerciser Port. See Figure 113 on page 157.



Click the black **current measurement** button in the Current & Voltage Measurements project selector to open the Unconfigured Current dialog.

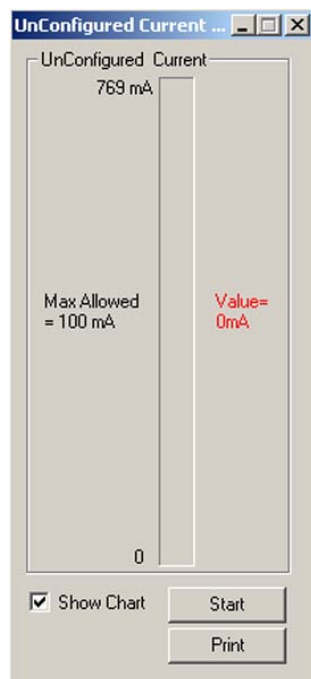


Figure 114 Unconfigured Current Measurement Dialog

To display the unconfigured current as a function of time, check the **Show Chart** check box.

Click **Start**. After a brief time, the following message appears:



Connect, then click **OK**.

Current Measurement (Optional)

The system performs the measurement and displays the result.



Figure 115 Unconfigured Current Measurement Results Display

Stop the Measurement

In the graphical display, to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

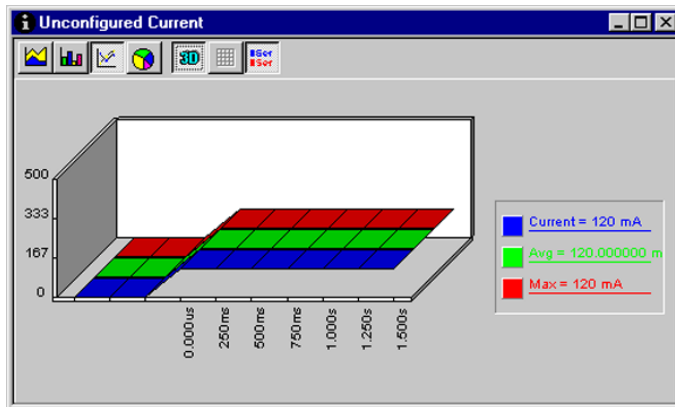


Figure 116 Unconfigured Current Chart Display

Print Result

You can print a summary of the current measurement by clicking **Print** in the Measurement Results dialog.

Operating Current Measurement

To perform this measurement, connect the USB Device and PC to the USB 2.0/1.x Analyzer port

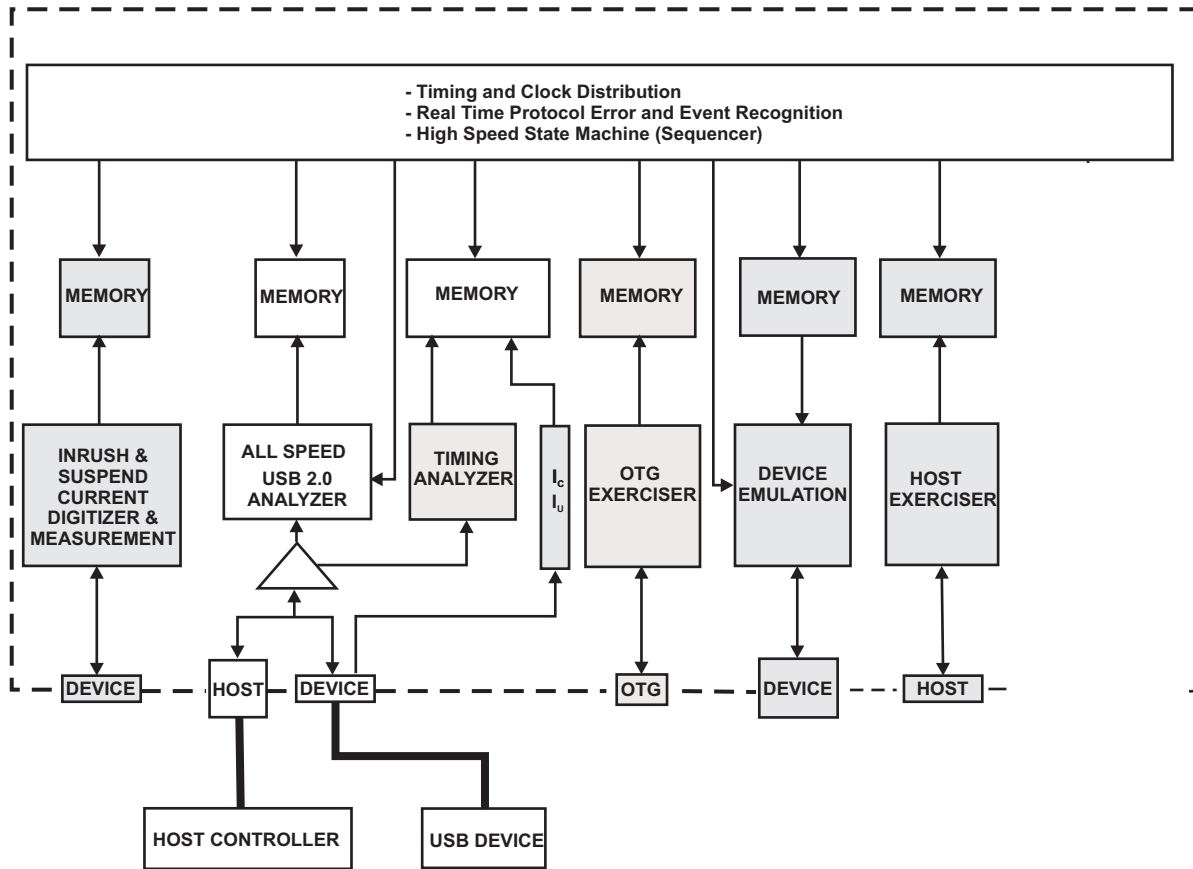


Figure 117 Operating Current Measurement Functional Connection

Make sure that the Device for the current test is unplugged from the Analyzer.

Current Measurement (Optional)



Click the **Green** button in the Current & Voltage Measurements project selector to open the Operating Current Measurement dialog.

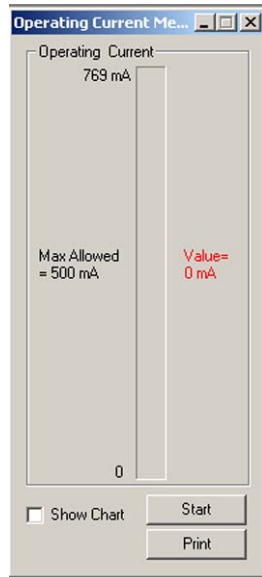


Figure 118 Operating Current Measurement Dialog Box

To display the suspend current as a function of time, check the **Show Chart** check box.

Click **Start**. After a brief time, the following message appears:



Current Measurement (Optional)

Connect, click **OK**, and wait for the measurement result to appear.



Figure 119 Operating Current Measurement Results Display

Stop the Measurement

In the graphical display, to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

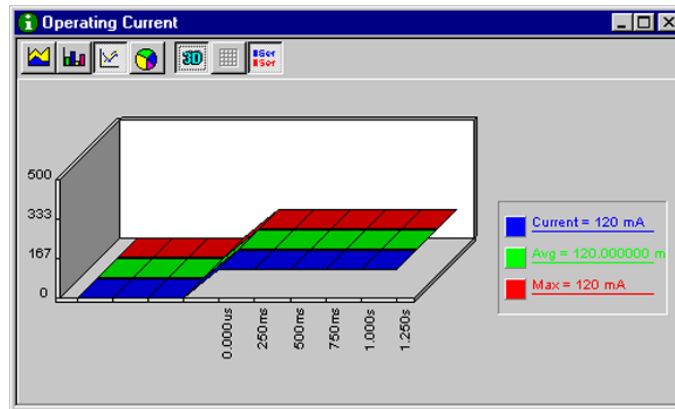


Figure 120 Operating Current Chart Display

Print Result

You can print a summary of the current measurement by clicking **Print** in the Measurement Results dialog.

Current Measurement (Optional)

Making the VBus Measurement

Make sure that the host is plugged in the Analyzer port.



Click the blue **VBus** button in the Current & Voltage Measurements project selector to open the VBus Measurement dialog.

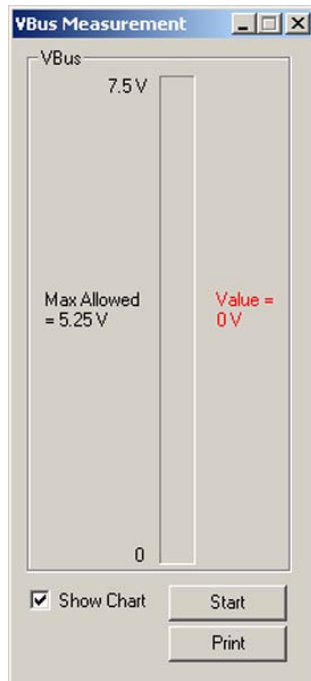
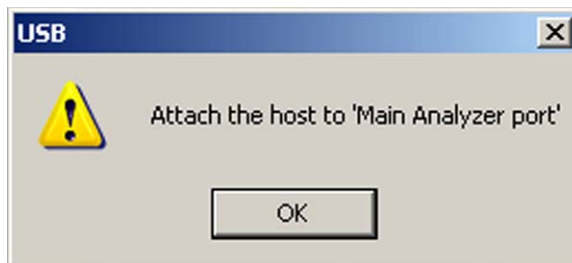


Figure 121 VBus Measurement Dialog Box

To display the VBus measurement as a function of time, check the **Show Chart** check box.

Click **Start**. After a brief time, the following message appears:



Current Measurement (Optional)

Click **OK**, and wait for the measurement result to appear.

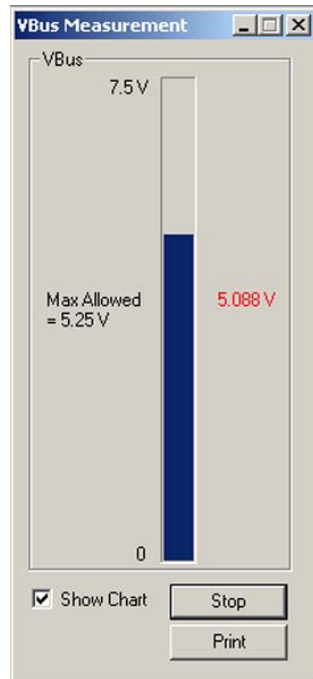


Figure 122 VBus Measurement Results Display

Stop the Measurement

In the graphical display, to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

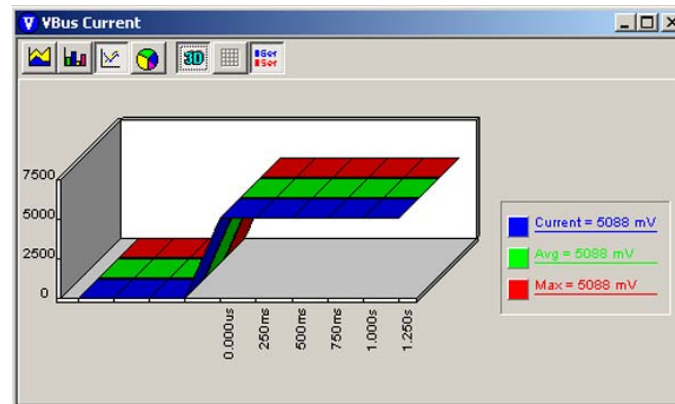


Figure 123 VBus Measurement Chart Display

Print Result

You can print a summary of the VBus measurement by clicking **Print** in the Measurement Results dialog.

Current Measurement (Optional)

VBus Droop Measurement

The VBus droop measurement determines the VBus voltage drop when a new device (load) is attached to the USB bus. The VBus droop measurement is performed with the analyzer and a USB host connected as shown in Figure 124.

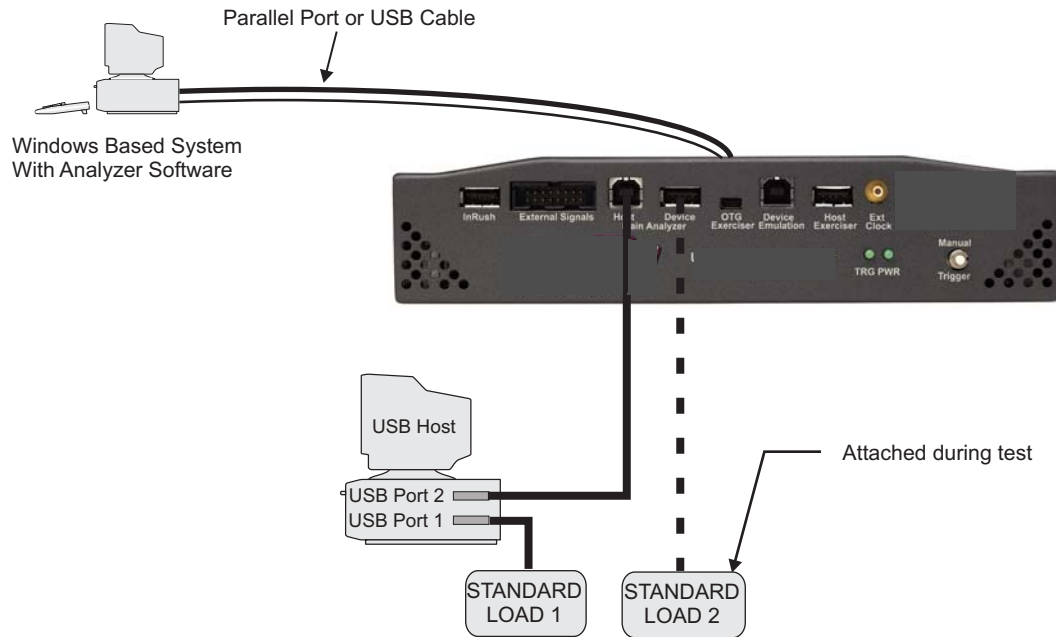


Figure 124 VBus Droop Test Setup

Performing the Measurement



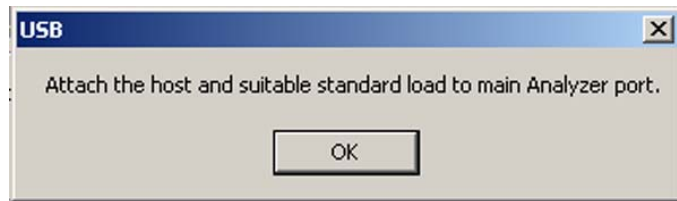
Click the **VBus droop** button in the Current & Voltage Measurements project selector to start the VBus Droop measurement.



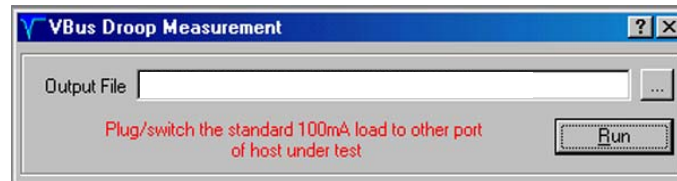
Each time this measurement is performed, the result is written to the default **Out.vsb** file, overwriting the previously saved test. To retain the current result, click the **ellipses** next to the output file name and enter a new file name.

Current Measurement (Optional)

To start test, click **Run**. This opens a reminder prompt to make sure that the initial test setup is in place.



Click **OK** to continue the test. A flashing prompt asks you to connect the Standard Load #2 to the Analyzer.



Once the Standard Load #2 has been connected the test proceeds to display the measurement result.

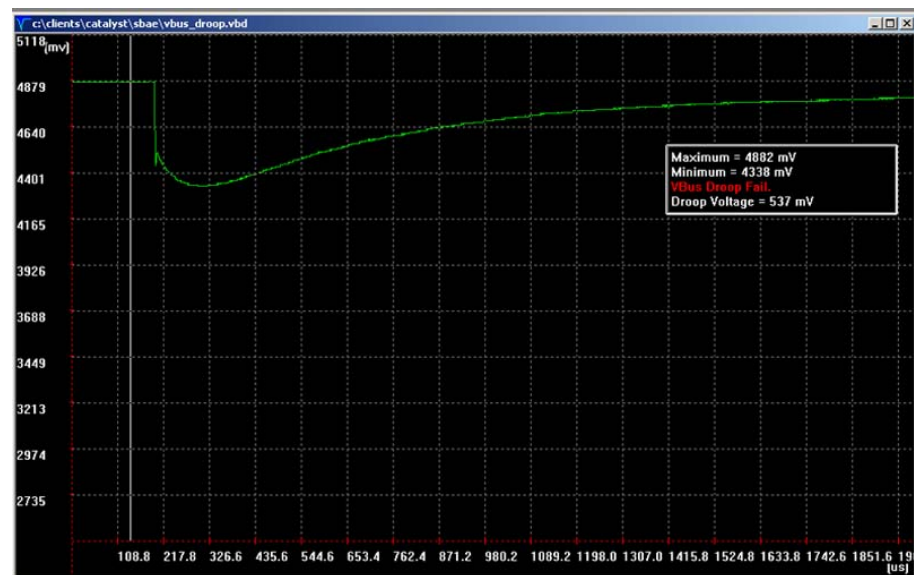


Figure 125 VBus Droop Test Result

Current Measurement (Optional)

Inrush Current Measurement

This test measures inrush current for the connected Device over the first 10 milliseconds of Device activation. The current measurement result is saved in a default file “**Out.irc**” which is overwritten every time that you perform a measurement unless you specify a new file name for each measurement made. Figure 126 shows the functional connection for inrush current measurement.

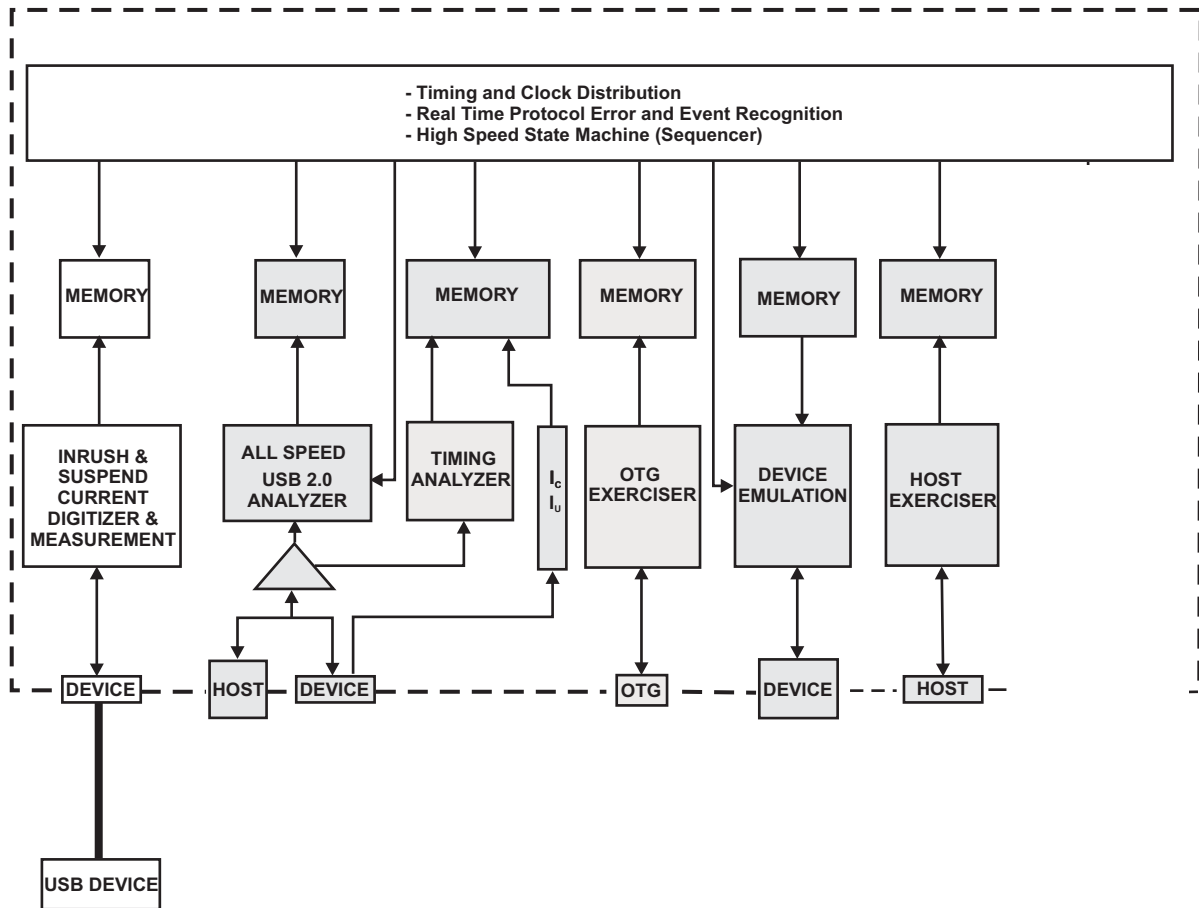


Figure 126 Inrush Current Measurement Functional Connection

To measure inrush current, click the  **Inrush** button in the Current & Voltage Measurements project selector to open the Inrush Current Measurement dialog as shown in Figure 127.

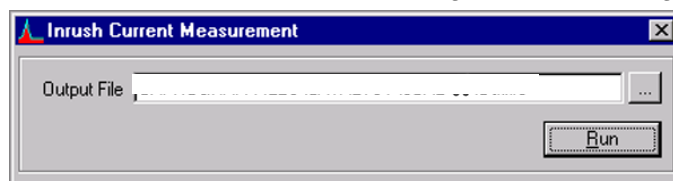


Figure 127 Inrush Current Measurement Dialog Box

Current Measurement (Optional)

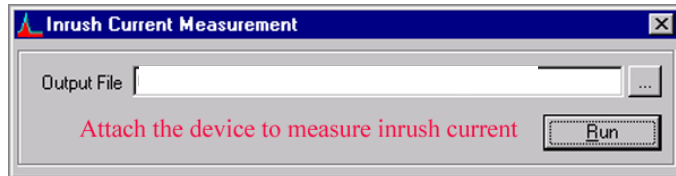
Click the **Run** button and follow the on-screen prompts.



Plug the Device into the Inrush port, click **OK**, and wait for the next prompt.



Unplug the Device from the inrush port, click **OK**, and wait for the next prompt.



Plug the Device back into the Inrush port. If this action does not open the result display, the inrush current is below the minimum threshold detected by Conquest M2. In this case, you can perform the measurement again by clicking **Run**.

You may receive the following message:



Current Measurement (Optional)

This is not an error, but an indication that Conquest M2 needs to switch range. To continue with the measurement, click **OK** followed by **Run**.

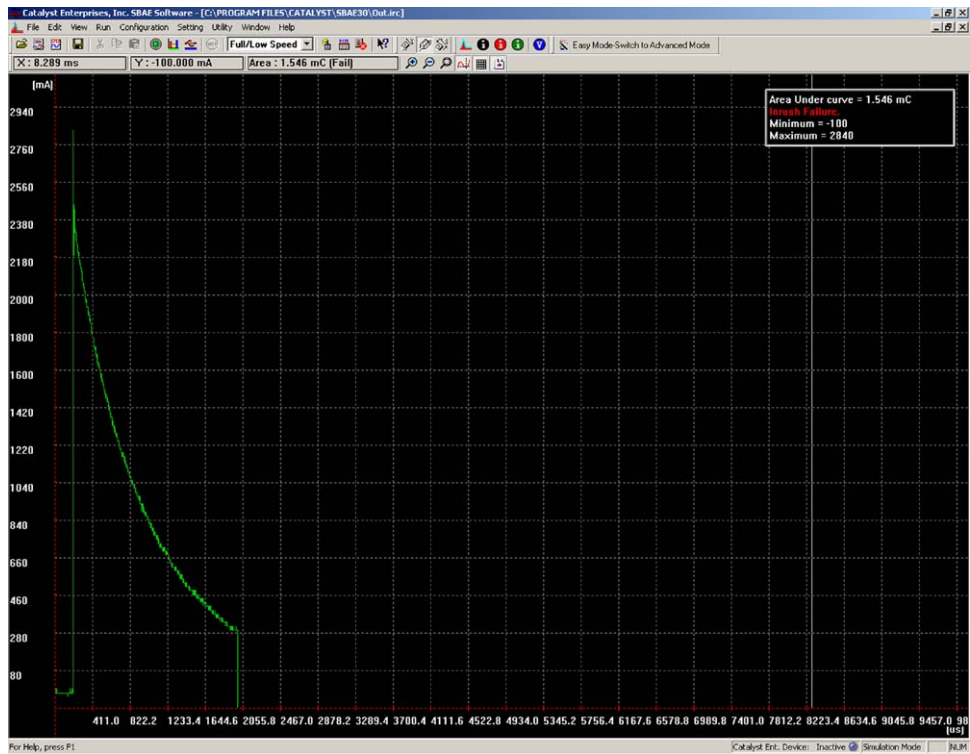


Figure 128 Inrush Current Measurement Result Display

Inrush Current Display Features

A number of display features allow a detailed examination of the inrush current display.



Click the **cursors** button on the inrush current display toolbar to enable the cursors.

Inrush pass/fail

The Inrush pass/fail is determined by the area under the curve and above the $Y=100$ mA horizontal line. If the area is less than $50 \mu\text{C}$, the test is reported as a pass.

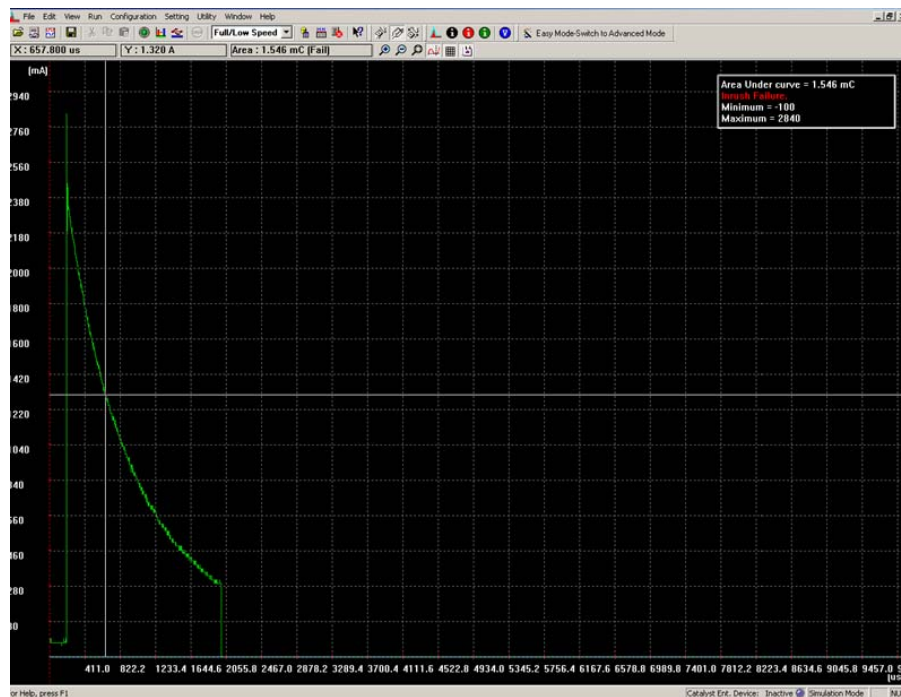
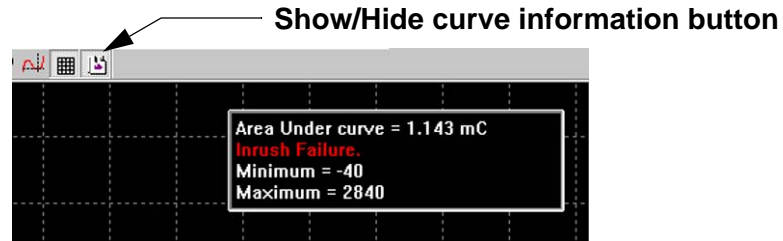


Figure 129 Inrush Current Display With Cursors

Enabling the cursors allows you to make precise measurements in the display. The time and current values at the intersection of the X(Time) and Y(Current) cursors are displayed on the toolbar as well as the area under the curve representing the energy in micro coulombs.

Positioning the Cursors

To position the cursors within the display, click the pointer at the point to place a cursor.

Current Measurement (Optional)

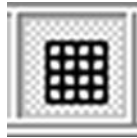
Optimize the Display

To optimize the display for viewing, click the **Best Fit** button on the display toolbar.

Zoom In/Zoom Out

To zoom in or out around the cursor position, click the **Zoom In** or **Zoom Out** button.

Grid



Click the **Grid** button to enable a grid in the display.

Suspend Current Measurement

The Suspend current measurement requires the interaction of the exerciser to place the Device in the suspended state. Prior to making the measurement you must connect the external cable provided between the USB 2.0/1.x Analyzer Host connector and the Exerciser Port. See Figure 113 on page 157

The device is partially enumerated by the Host Exerciser that sets the address and configuration and then initiates suspend by stopping all traffic to the device.



Click the **Red** button in the Current & Voltage Measurements project selector to open the Suspend Current Measurement dialog.

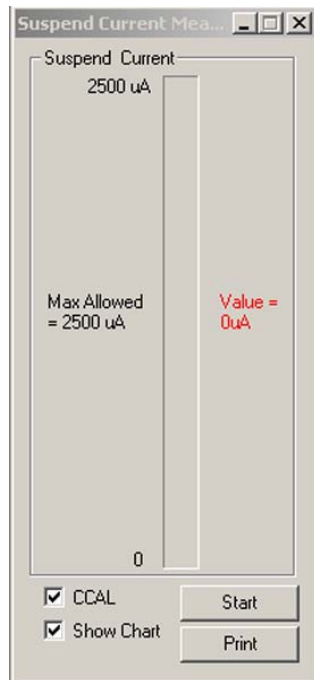


Figure 130 Suspend Current Measurement Dialog Box

Check the **CCAL** checkbox if using the Current Measurement Calibration Board as described on page 174. To display the suspend current as a function of time, check the **Show Chart** check box.

Click **Start**. After a brief time, the following message appears:



Current Measurement (Optional)

Connect, then click **OK**.

The system performs the measurement and displays the result.

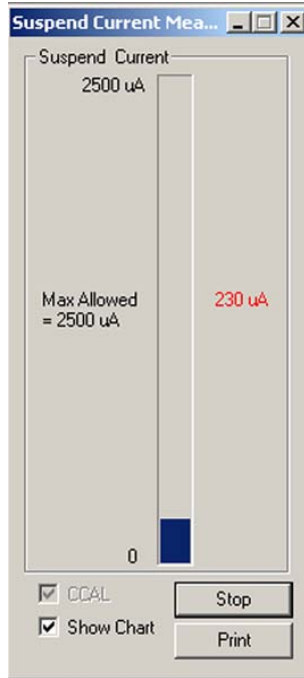


Figure 131 Suspend Current Measurement Results Display

Stop the Measurement

In the graphical display, to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

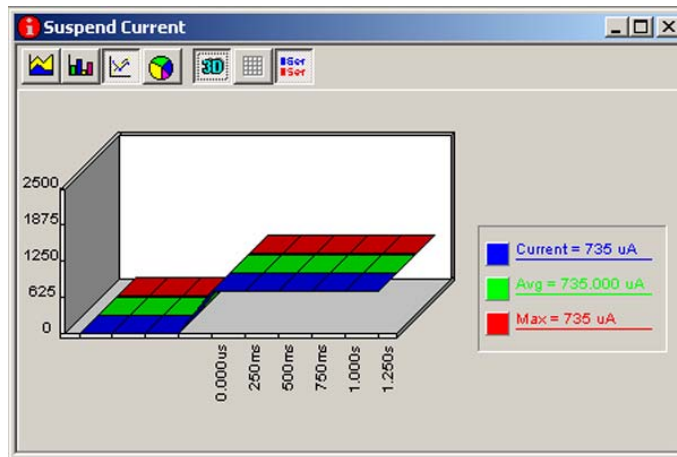


Figure 132 Suspend Current Chart Display

Print Result

You can print a summary of the current measurement by clicking **Print** in the Measurement Results dialog.

Current Measurement Calibration Board

The Current measurement calibration board is provided to verify the current measurement functionality of Conquest M2.

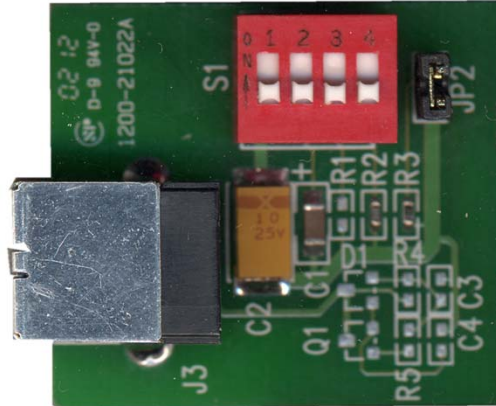


Figure 133 Current Measurement Calibration Board

Table 6 shows the connections to the Analyzer, calibration board switch settings and pass/fail criteria for the tests.

Table 6 Measurement Calibration Parameters

Current Test	Pass Limit	Connection	S1-1	S1-2	S1-3	S1-4	Conquest M2 Measurement
Inrush 1 (5A range)	50 μ C	Inrush	ON	OFF	OFF	OFF	40 μ C***
Inrush 2* (10A Range)	50 μ C	Inrush	ON	ON	OFF	OFF	50 μ C***
Unconfigured	100mA	Analyzer/Exerciser	OFF	OFF	ON	OFF	66mA
Suspend	500 μ A	Analyzer/Exerciser	OFF	OFF	OFF	ON	107 μ A
Operating	500mA	Analyzer/Host	OFF	OFF	ON	OFF	66mA
VBus**	5.25V	Analyzer/Exerciser	N/A	N/A	N/A	N/A	5.176V

* This test must be run twice to change from 5A to 10A range.

** The calibration board is not used for this test. Only connect the host exerciser port to the Main Analyzer Port.

*** There is a $\pm 3 \mu$ C tolerance on the measurements limited by the components on the calibration board.

Current Measurement (Optional)

Note: When performing inrush current measurements, use the 4" (10cm) supplied cable and repeat the measurement until you get a clean result without any extra spikes, as shown in Figure 134.

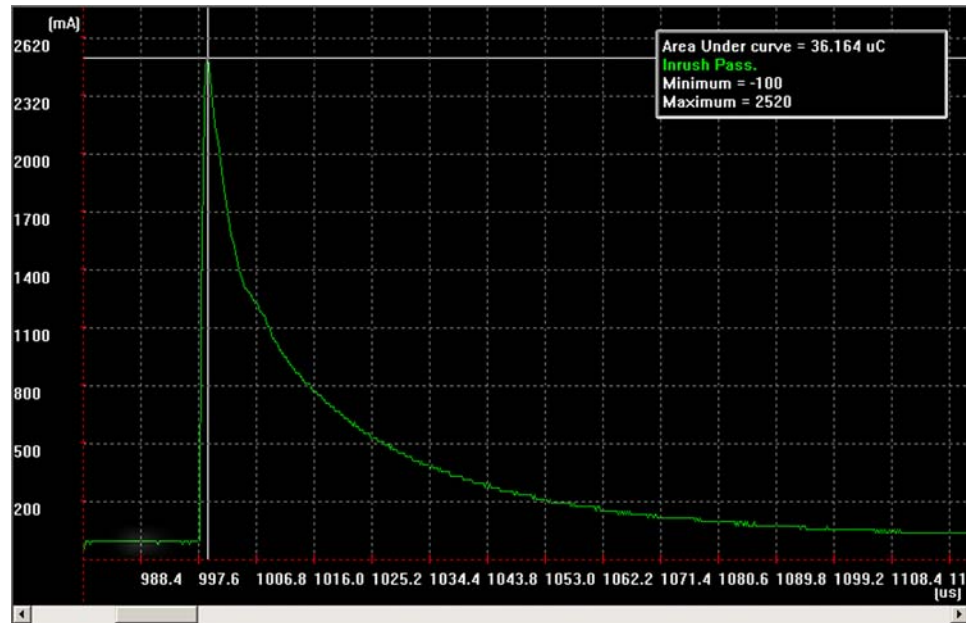


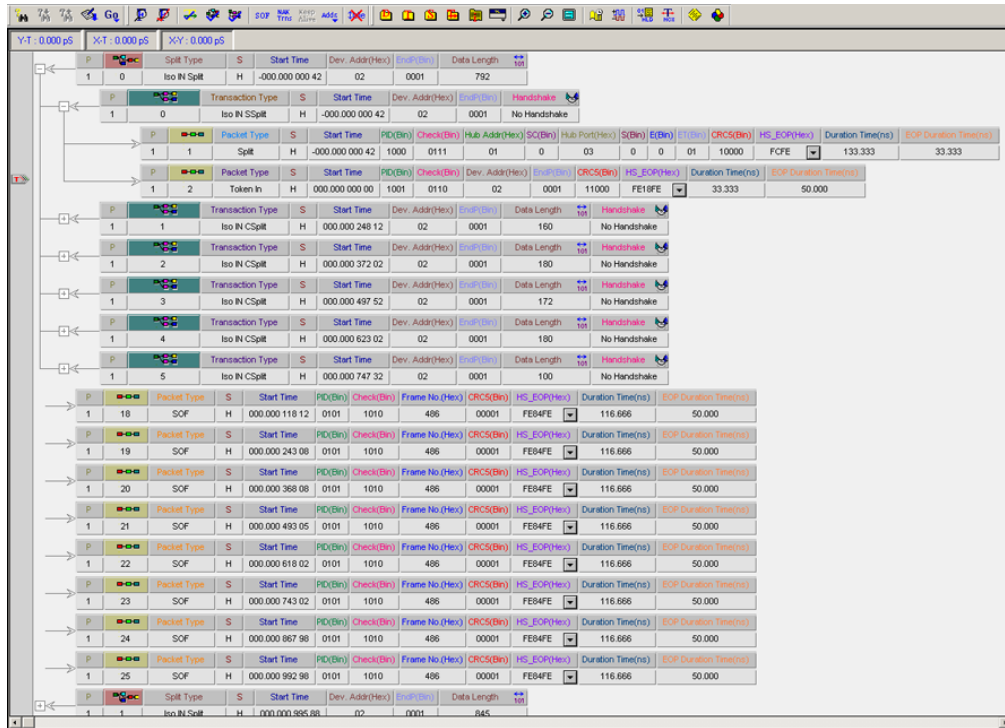
Figure 134 Clean Inrush Measurement Result

Note: There is a $\pm 10\%$ tolerance on the measurements limited by the components on the calibration board.

Current Measurement (Optional)

Display Manipulation

You can configure the captured data display test and viewing preferences.

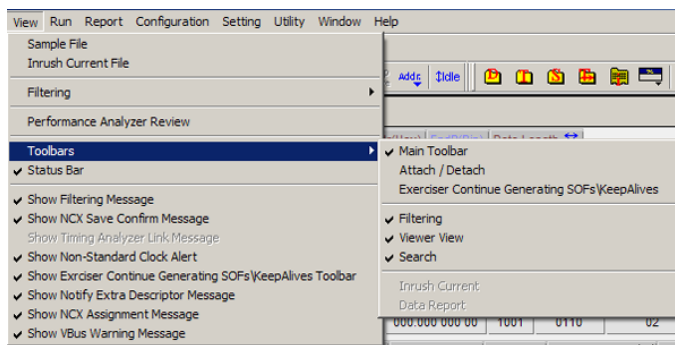


Transaction Type	S	Start Time	Dev. Addr(Hex)	End(Bin)	Data Length	Handshake								
Iso IN Split	H	-000.000 000 42	02	0001	792									
Iso IN S Split	H	-000.000 000 42	02	0001		No Handshake								
Packet Type	S	Start Time	PID(Bin)	Check(Bin)	Hub Addr(Hex)	SC(Bin)	Hub Port(Hex)	S(Bin)	E(Bin)	Color	CRCS(Bin)	HS_EOP(Hex)	Duration Time(ns)	EOP Duration Time(ns)
Split	H	-000.000 000 42	1000	0111	01	0	03	0	0	01	10000	PCFE	133.333	33.333
Token In	H	000.000 000 00	1001	0110	02	0001	11000						33.333	50.000
Transaction Type	S	Start Time	Dev. Addr(Hex)	End(Bin)	Data Length	Handshake								
Iso IN CSplit	H	000.000 248 12	02	0001	160	No Handshake								
Iso IN CSplit	H	000.000 372 02	02	0001	180	No Handshake								
Iso IN CSplit	H	000.000 497 52	02	0001	172	No Handshake								
Iso IN CSplit	H	000.000 623 02	02	0001	180	No Handshake								
Iso IN CSplit	H	000.000 747 32	02	0001	100	No Handshake								
Packet Type	S	Start Time	PID(Bin)	Check(Bin)	Frame No (Hex)	CRCS(Bin)	HS_EOP(Hex)	Duration Time(ns)	EOP Duration Time(ns)					
SOF	H	000.000 118 12	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 243 08	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 368 08	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 493 05	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 618 02	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 743 02	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 867 98	0101	1010	486	00001	FE84FE	116.666	50.000					
SOF	H	000.000 992 98	0101	1010	486	00001	FE84FE	116.666	50.000					
Split Type	S	Start Time	Dev. Addr(Hex)	End(Bin)	Data Length									
Iso IN Split	H	000.000 995 88	02	0001	845									

Figure 135 Captured Data Display

Simplify Tool Bar

To simplify the working area, you can temporarily hide toolbars for features that you are not using. Click **View > Toolbars**



Uncheck the toolbars to temporarily hide. You can restore them by following the same procedure and checking them for display.

Results Display Viewing Preferences

You can vary the level of detail presented by using some of the following display modification features:



The **Full Screen** button expands hides the main toolbar and increases the data display area on the screen.



The **Show/Hide Splits** button expands or collapses a data line to show or hide splits.



The **Show/Hide transfers** button hides or displays transfers.



The **Show/Hide transactions** button expands or collapses a data line to show or hide transactions.



The **Expand/Close Splits** button expands or collapses a split display.



The **Expand/Close Transactions** button expands or collapses a transaction display.



The **Expand/Close Data Fields** button expands or collapses a captured data stream display.



The **Expand/Close Transfers** button expands or collapses displayed transfers.

Display Manipulation

Compact View

You can view the results display in a more compact form by hiding the headers in the results display.



Clicking the down arrow on the **Show/Hide Headers** button allows you to Show all Headers, Hide All Headers, or to Hide Repeated Headers.

Figure 136 shows a results display with all headers hidden.

Transfer ID	Transfer Action	Packet ID	Data	Status	Address	Port	Length	Checksum	Device	Device Descriptor		
1	Transfer	0	Get Descriptor	F	00	Device	0	Device Descriptor				
1	Transaction	0	Setup	F	-000.000.004.37	00	0000	8	Acknowledged			
1	Packet	0	Setup	F	000.000.004.37	1101	0010	00	0000	01000	2.667	183.333
1	Packet	1	Data0	F	000.000.000.00	0011	1100	80 06 00 01 00 00 40 00		BB29		6.717
1	Packet	2	ACK	F	000.000.007.23	0010	1101	1.333				166.666
1	Transaction	1	IN	F	000.000.995.37	00	0000	8	Acknowledged			
1	Transaction	2	OUT	F	000.002.994.82	00	0000	0	Acknowledged			
1	Event	4	Reset		000.004.025.40				26.133			
1	Transfer	1	Set Address	F	00	2						
1	Transfer	2	Get Descriptor	F	02	Device	0	Device Descriptor				
1	Transfer	3	Get Descriptor	F	02	Configuration	0	Configuration Descriptor				
1	Transfer	4	Get Descriptor	F	02	Configuration	0	Configuration Descriptor				
1	Transfer	5	Get Descriptor	F	02	Configuration	0	Configuration Descriptor				
1	Transfer	6	Get Descriptor	F	02	Device	0	Device Descriptor				
1	Transfer	7	Get Descriptor	F	02	Configuration	0	Configuration Descriptor				
1	Transfer	8	Get Descriptor	F	02	Configuration	0	Configuration Descriptor				
1	Transfer	9	Set Configuration	F	02	1						
1	Transfer	10	Set Interface	F	02	1	0					
1	Transfer	11	Get Descriptor	F	02	Device	0	Device Descriptor				
1	Transfer	12	Set Interface	F	02	1	1					
1	Transfer	13	Set Interface	F	02	1	0					
1	Transfer	14	Class Type Request	F	02							
1	Transfer	15	Class Type Request	F	02							
1	Transfer	16	Class Type Request	F	02							
1	Transfer	17	Class Type Request	F	02							
1	Transfer	18	Class Type Request	F	02							
1	Transfer	19	Class Type Request	F	02							
1	Transfer	20	Class Type Request	F	02							
1	Transfer	21	Class Type Request	F	02							
1	Transfer	22	Class Type Request	F	02							
1	Transfer	23	Class Type Request	F	02							
1	Transfer	24	Class Type Request	F	02							

Figure 136 Results Display With All Headers Hidden

Display Idle Time



Click the **Show/Hide idle time** button on the toolbar to display time between events. The button is shown in the Hide Idle time state.

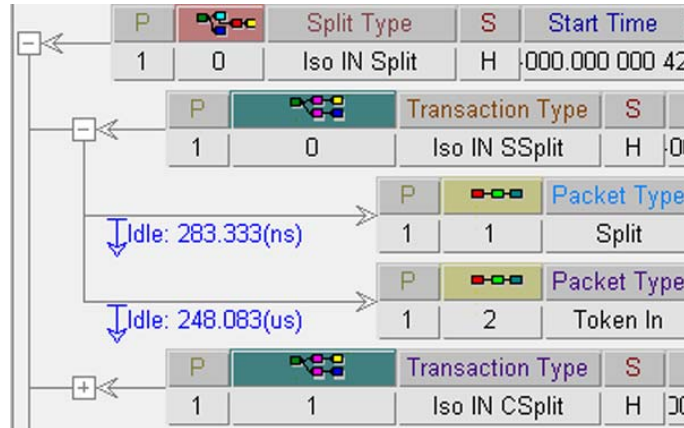


Figure 137 Time Between Events Display Enabled



Click the **Show/Hide idle time** button on the toolbar to remove the time display between events. The button is shown in the Show Idle time state.

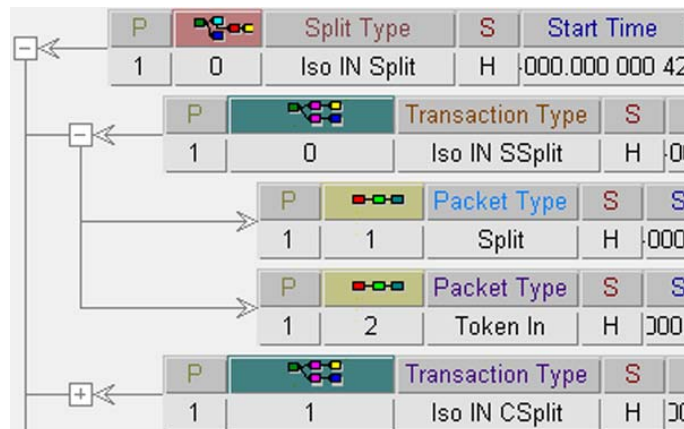


Figure 138 Time Between Events Hidden

Waveform Display

You can open a Waveform display that corresponds to the packet at the X-cursor position by right-clicking in the data display area and choosing **Show Wave**.

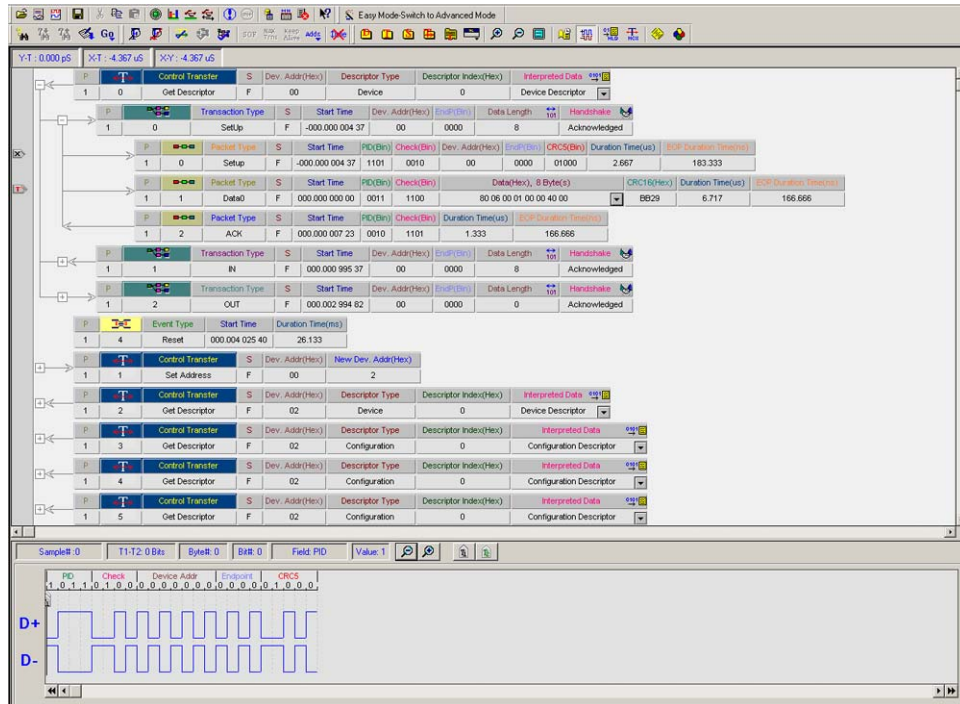



Figure 139 Wave Display Selected

Clicking the  **Show/Hide Waveform** button displays or hides the waveform display.

T1, T2 Pointers

The Waveform display shows each transmitted bit level and incorporates T1 and T2 pointers that allow the measurement of the number of bits between different locations in the waveform display.

Lost Pointers

When scrolling through a long wave display, you may lose the T1 and T2 pointers. To rapidly return to a pointer, click either the **T1** or **T2 Scroll to Pointer Page** button on the wave display tool bar.



Filter

Filtering allows you to modify data in the display to exclude a set of user-defined patterns and save the result in a new file. Available filtering patterns include Fields, Packets, Transactions and Events.



Click the **Filter** button on the display toolbar to set up the filtering patterns.

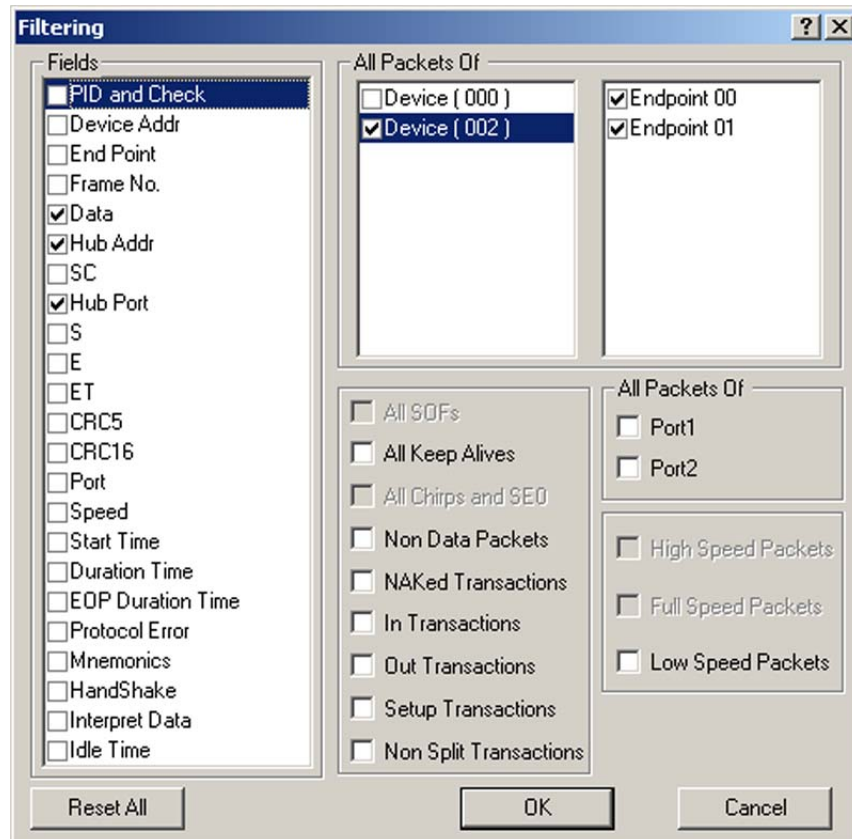


Figure 140 Filter Setup Dialog

Click **OK** and note that the selected filtering patterns are not in the data display.

Smart on Screen Filtering

In addition to the normal filtering function, Conquest M2 offers a quick way to filter the result. You can quickly filter out Device-related-, Keep Alives, SOF, and NAK transactions. To perform Smart filtering, right-click the **results area** and choose a filtering action.

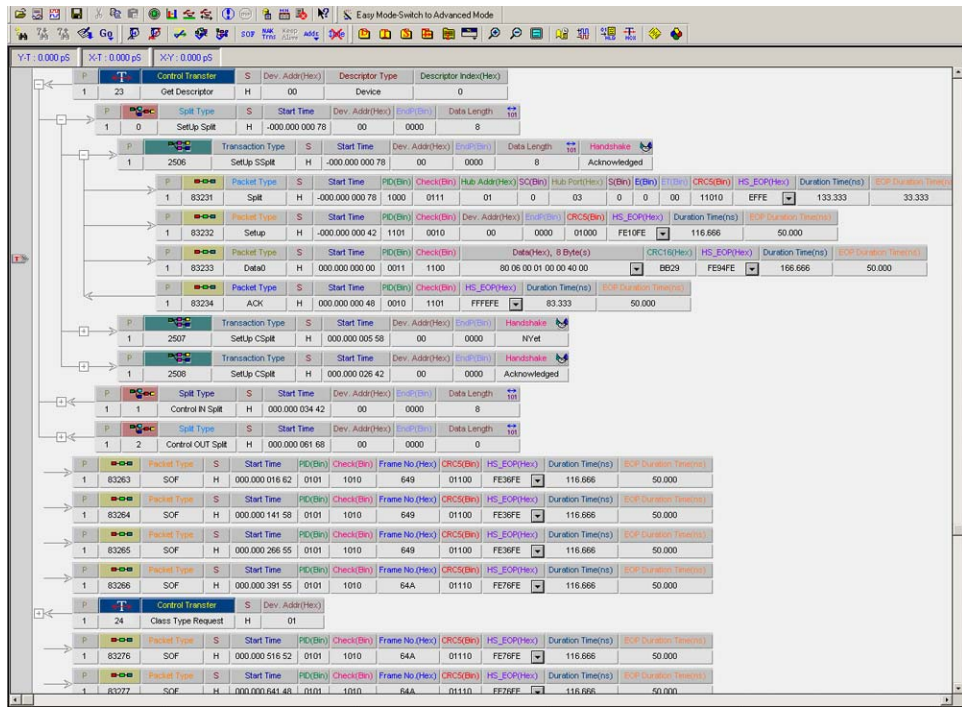


Figure 141 Filtering out All SOF Transactions

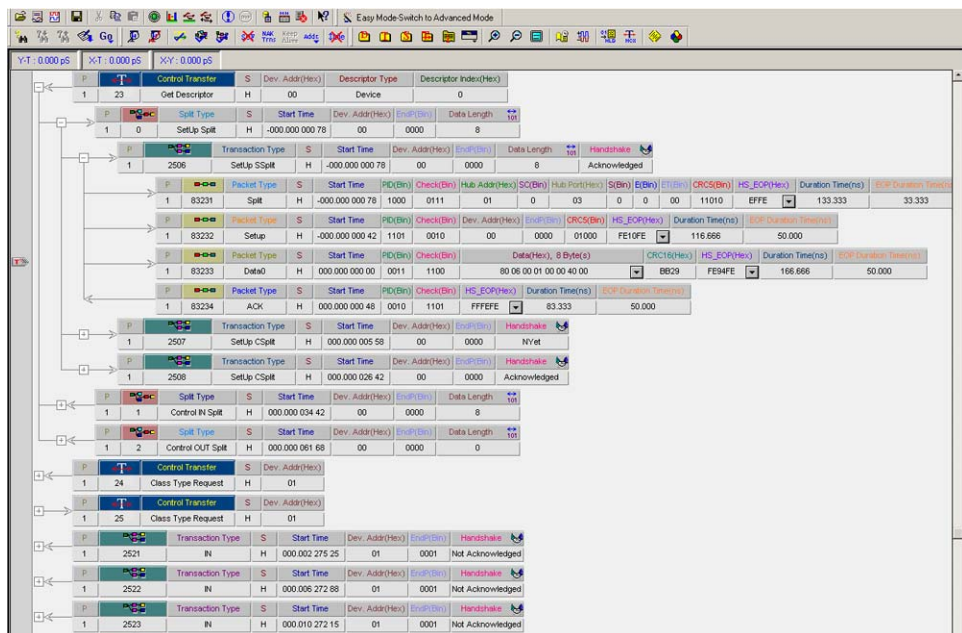


Figure 142 All SOF Transactions Filtered

Filter from Toolbar

You can quickly filter SOF, NAK, Keep Alive, and Device Transactions by clicking the corresponding button on the main toolbar.



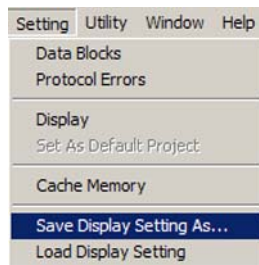
Only Transactions Present

Filtering buttons are enabled only for the transactions that occur in the results display.

Save Display Settings

Once you have set all display settings, you can save them in a file for use later on similar applications.

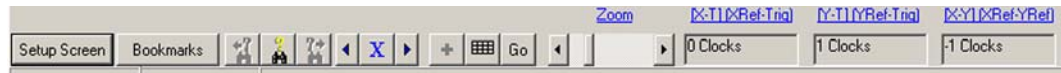
To save the display settings, click **Setting** on main toolbar and choose **Save Display Settings As.**



To retrieve these settings later, click **Setting** on main toolbar and choose **Load Display Setting** to open the saved settings file.

Timing Analysis Display

All USB signals are always captured and stored on disk, but may not be displayed in the results display, unless you select them for display.



Click the **Setup Screen** button on the timing results display toolbar to open the Active Signals Dialog box.

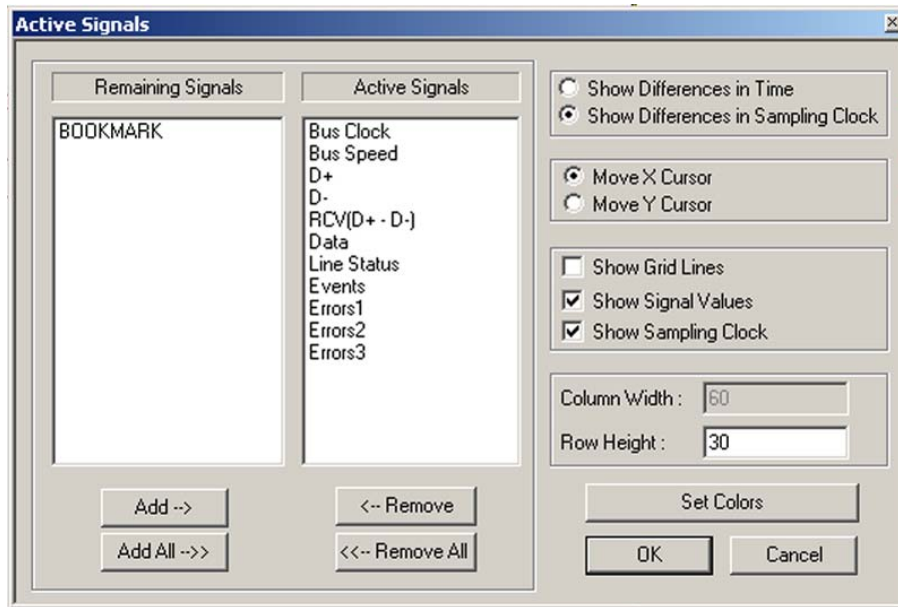


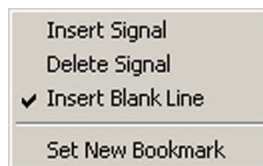
Figure 143 Active Signals Dialog Box

Select a signal in the **Active Signals** or the **Remaining Signals** list and use the **Add->** and **<-Remove** buttons to set up signals for display.

Insert blank line

To enhance readability, you can insert a blank line between displayed signals by right-clicking in the **timing display area** between signal lines.

Choose **Insert Blank Line**.



Similarly, you can Insert additional signal lines, delete signal lines, or set a bookmark.

Set Timing Display Viewing Options

The Active Signals Dialog allows you to set up additional options for viewing the timing analysis result.

Show Differences in Time/Clock

Click the **Show differences in Time** button to display cursor position in time units or the **Show differences in Clock** button to display cursor differences in number of clocks.

Show Grid Lines

Click the **Show Grid Lines** check box to enable grid lines in the results display.

Show Signal Values

Click the **Show Signal Values** check box to enable the Value(x) column in the results display.

Show Clock

Click the **Show Clock** checkbox to display the Conquest M2 sampling clock.

View Timing Details

To assist in evaluating the timing result you can position the cursors incrementally one sample at a time as well as to zoom around each cursor to display a level of detail.

Move X/Y Cursor

Click the **X** or **Y** toggle button to enable positioning either the X or Y cursor. Move one sample at a time to the right or left by clicking the **right** or **left** arrow.



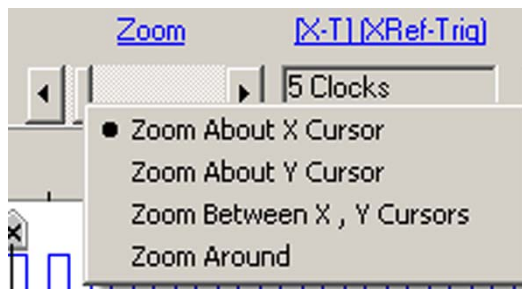
Display Manipulation

Zoom around cursors

To zoom around either the X or Y cursor, click the **cursor** to make it active and then slide the zoom slider right or left for a magnification. For incrementing or decrementing the magnification, click the **right** or **left arrow** on the zoom slider, respectively.



To quickly access additional zoom options, right-click the zoom slider to display the additional zoom choices.



Using the Cursors and Bookmarks

The captured data display incorporates three cursors labeled **X**, **Y** and **T** (See Figure 135). All of the cursors are initially overlaid and positioned at location 0, which is the trigger position of the display. The Trigger, or **T**, cursor is the measurement reference and is always locked at location 0 in the display.

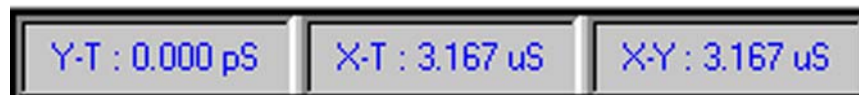
Positioning the X Cursor

To position the X-Cursor within the captured data display, click the line in which to place the cursor.

Positioning the Y Cursor

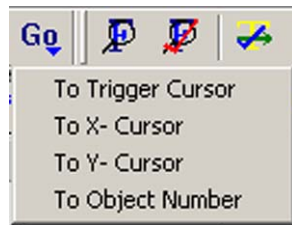
To position the Y-cursor within the captured data display, click the point at which to position the cursor.

Time differences between the cursors are displayed on the main toolbar.



Locate Cursors

To quickly locate any of the cursors within the result, click the **Go** button on the toolbar and choose the cursor to locate.

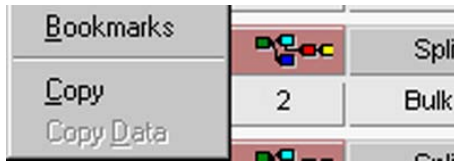


Display Manipulation

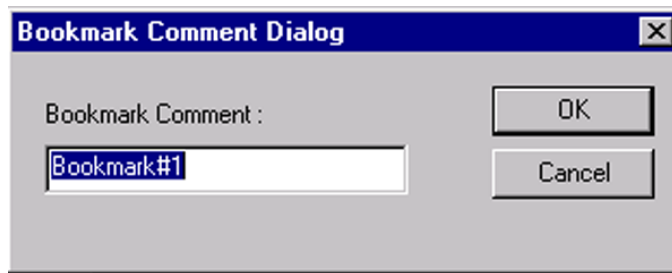
Bookmarks

Bookmarks is a convenient way to mark a point in the results display by name such that you can rapidly return to that point by that name. To create a bookmark:

Right-click the mouse in the signal display area in which to place the bookmark.



Click **B**ookmarks to open the Bookmark Comment Dialog.



Enter an identifying name for the bookmark and click **OK**. Repeat for additional bookmarks.

Finding a Bookmark



Click the **Go To Bookmark** button on the display toolbar to open the Go to Bookmark dialog.

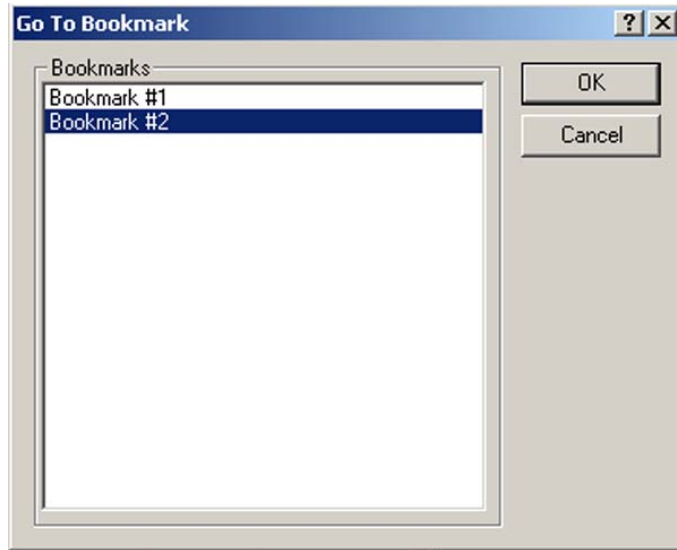


Figure 144 Go To Bookmark Dialog Box

Highlight the bookmark to which to go and click **OK**.

P	Transaction Type	S	Start Time	Dev. Addr	EndP
1	3	Setup	F	000.103 996 07	00 0000
1	9	Setup	F	000.103 996 07	1101 0010 00 0000 01000
1	10	Data0	F	000.103 999 23	0011 1100 0005030000000
1	11	ACK	F	000.104 007 85	0010 1101 Mnemonics Ack
1	4	Async IN	F	000.104 995 20	00 0000

Figure 145 Bookmark Found

Search

The search option permits you to examine any captured data output file and to quickly locate packets or bus conditions including search patterns that are defined as Mnemonics.

Note: Search operations include online software and hardware search and offline search in saved files.

Whenever an output file is displayed, the toolbar also displays the additional **Search Next** and **Search Previous** buttons.



To perform an initial search, click the **Search** button to open the search pattern definition dialog box, as shown in Figure 146.

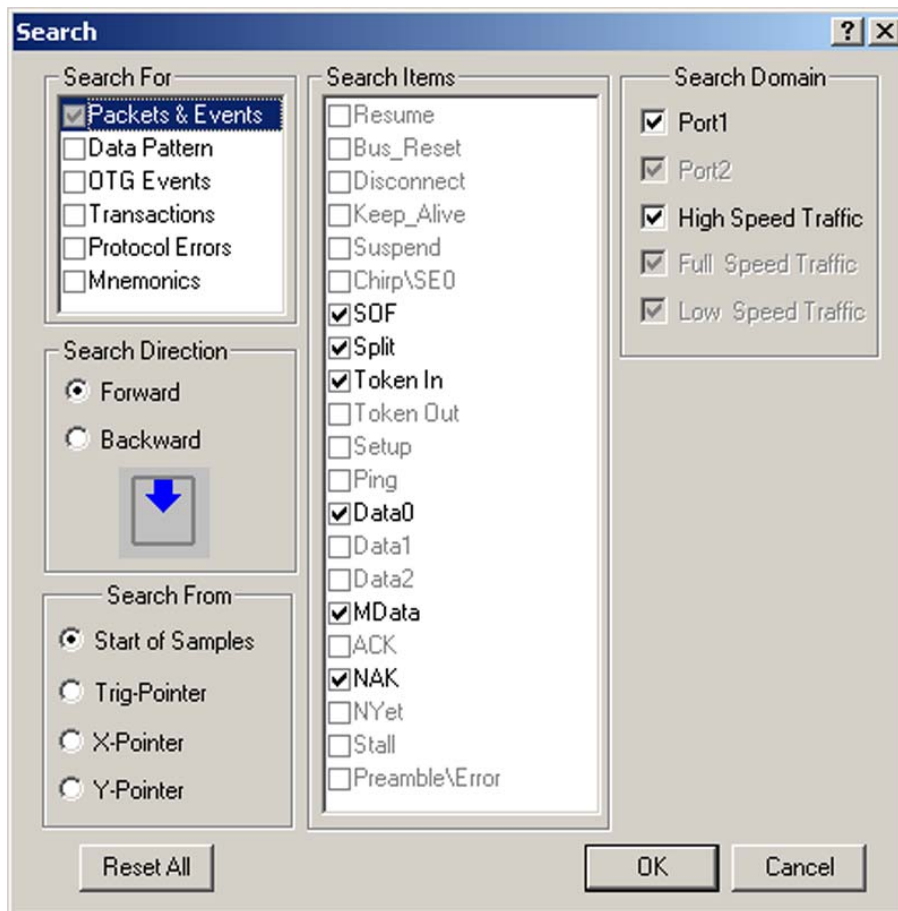


Figure 146 Search Parameter Definition Dialog Box

Check search parameters and click **OK** to perform the search.

You can continue to search the output file using **Next Match** or **Previous Match** for the same pattern until you redefine the search parameters.



To continue the search for the same pattern, click the **Next Match** or **Previous Match** buttons on the results display toolbar.

Search for Protocol Errors

To search for protocol errors, check the **Protocol Errors** box and check the protocol errors to search for.

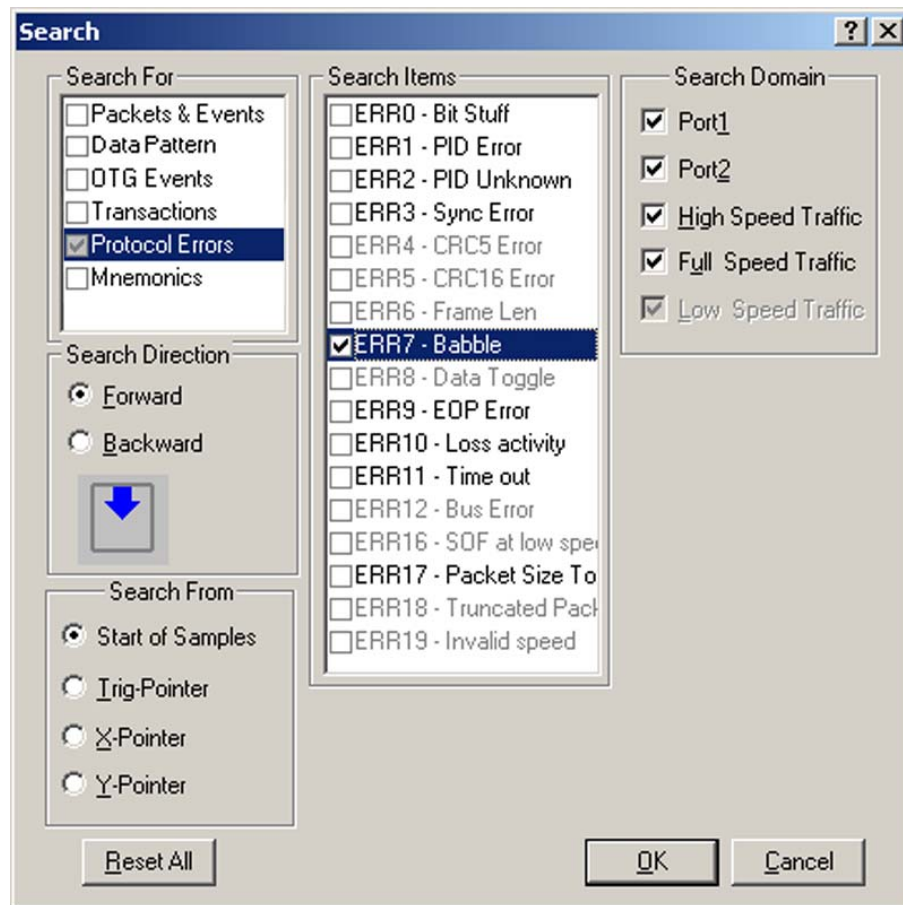


Figure 147 Expanded Search Parameter Definition Box

Display Manipulation

Search for Data Pattern

Choose **Data Pattern**, choose the data format, enter the data pattern for which to search, and click **OK**. You can also search for data packets of payload length jointly or independently of the data pattern.

Note: A data pattern is found only if it is within a data packet boundary. However, if the data is in multiple Data packets, perform the search in the Data Report, as described on page 103.

The screenshot shows the 'Search' dialog box with the following configuration:

- Search For:** Data Pattern (checked)
- Search Direction:** Forward (selected)
- Search From:** Start of Samples (selected)
- Look For:** Data (pattern/payload length)
- Data Pattern:** 01XX0011
- Format:** Binary (selected)
- Logic:** And (selected)
- Data Payload Length:** 11 Byte(s)

Search with a Mnemonic

In order to perform a search using a Mnemonic, you must first define a Mnemonic as described in "Mnemonics" on page 199. Once Mnemonics are defined proceed similarly as for searching for Protocol Errors.

Search a Timing Analysis Result

You can perform a search in a timing analysis results display for Packets, Packet Fields, Errors and Events that have been captured.



To perform an initial Timing Result search, click the **Search** button to open the search pattern definition dialog box as shown in Figure 148

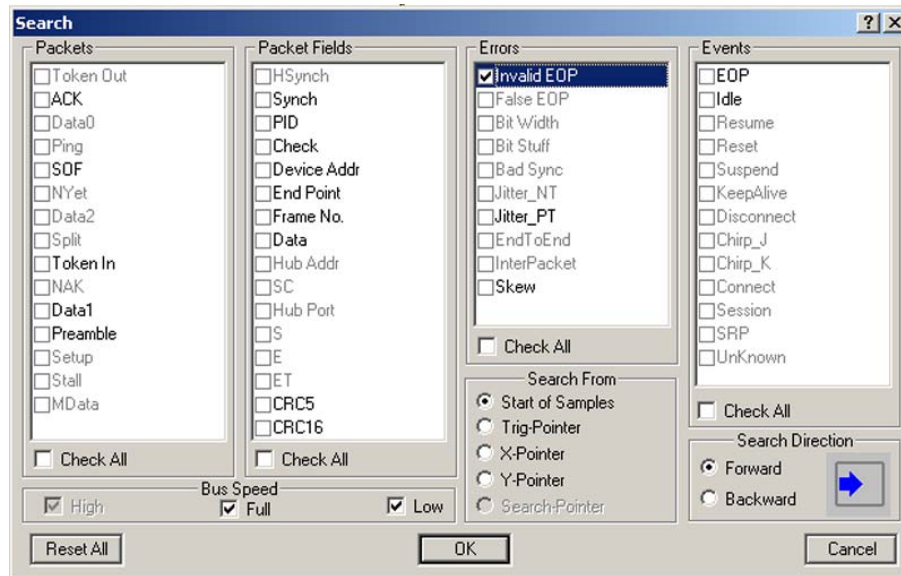


Figure 148 Timing Result Search

Note that only items present in the current data capture are enabled for selection.

Choose Item(s) for search

Select all items for which to search and click **OK**.

The search takes you to the first occurrence of any of the items in the search direction chosen.

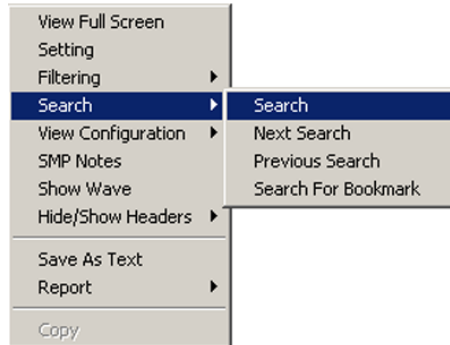


To continue the search for the same set of items, click the **Next Match** or **Previous Match** buttons on the results display toolbar.

Display Manipulation

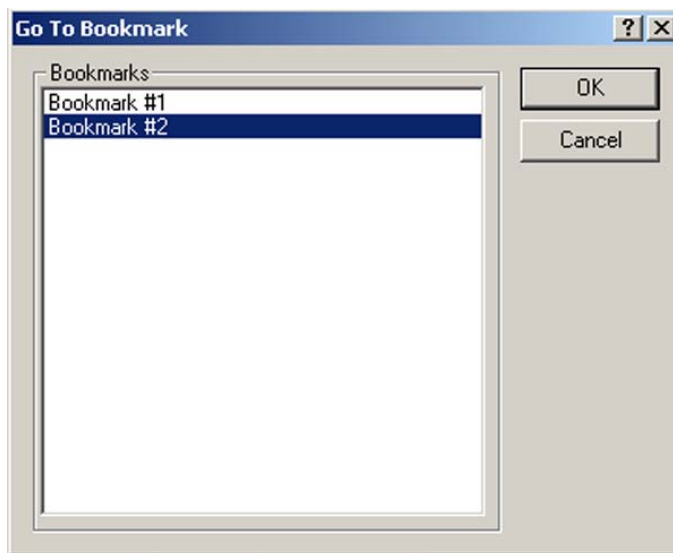
Additional Search Options

You can access additional search options by right-clicking the right side of the results display.



Search for Bookmark

Choose **Search for Bookmark** to open the Go to Bookmark dialog, highlight the bookmark to find, and click **OK**.



Access from toolbar

You can quickly access the Search for Data Patterns or Search for Bookmarks by clicking the corresponding buttons on the main toolbar.



Display Configuration

Conquest M2 ships with a default display font and color configuration. You can, however, define your own fonts and color scheme for a particular testing scenario. You can set different foreground (Text color) for each cell type in the display.



To customize your display, click the **Configuration** button on the main toolbar to open the Display Configuration dialog box.

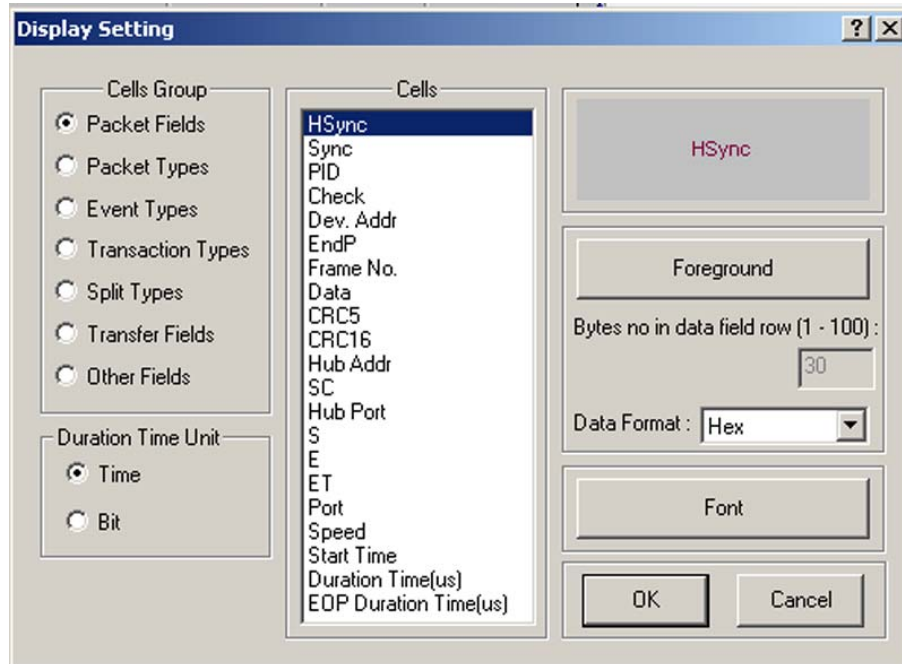


Figure 149 Display Configuration Dialog Box

Display Manipulation

To Change Cell Color

1. Select a **Cells** group and highlight the item for which to set the color.
2. Click the **Foreground** button to open the color palette dialog box.



3. Choose an appropriate color and click **OK**.

To change Display Fonts

1. Click the **F**ont button to open the Font dialog box.

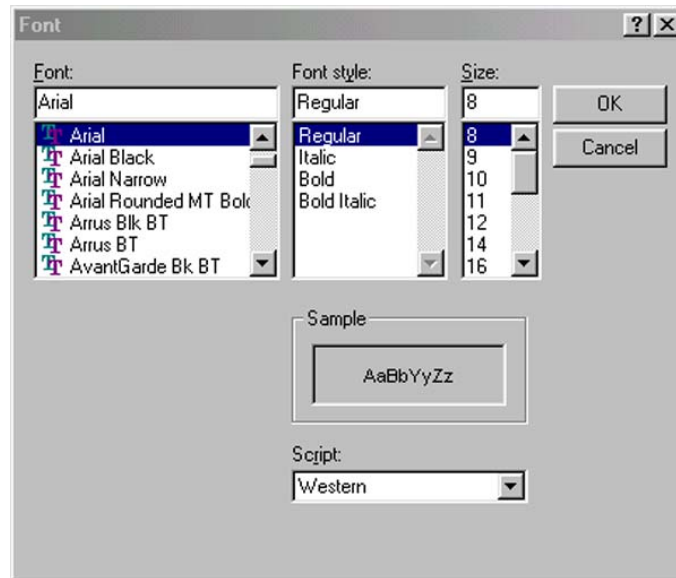


Figure 150 Display Font Definition Dialog Box

2. Choose a font, font style, and size, then click **OK**.
3. When finished, click **OK** to save changes and close the Display Configuration dialog.

Mnemonics

Mnemonics is a convenient debugging tool that allows you to assign names to data packet types. You can use these names:

- In the captured data display next to matching patterns for quick identification.
- In a search for a pattern.
- To filter previously generated data files to include or exclude certain data patterns.



To define mnemonics, click the **Mnemonics** button on the toolbar to open the Mnemonics Definition Dialog box.

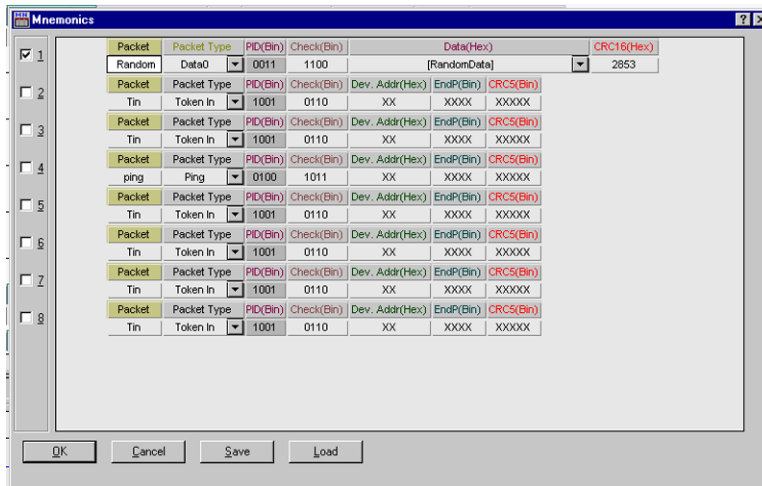


Figure 151 Mnemonics Definition Dialog Box

To assign a unique name to a mnemonic, highlight the Packet “Name” field and type a name.

To complete the mnemonic definition set each of the data fields just as if you were defining packets as described in “Defining Packets” on page 60.

Not a Measurement

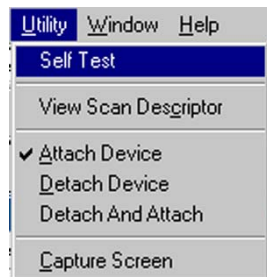
While similar in appearance and action to a Packet definition dialog box, the mnemonics definition is only an assignment of names to matching patterns that are used to search and highlight already collected data and is not used to perform any measurements.

Utilities

Self Test

You can perform two levels of self test on the Conquest M2 hardware. A Quick Test performs a RAM check with a cursory algorithm. The Advanced Test performs a RAM check with a more thorough algorithm.

Click **Utility** and then choose **Self Test** to open the self test dialog.



Choose the type of test to perform and click **Start** to run the test.

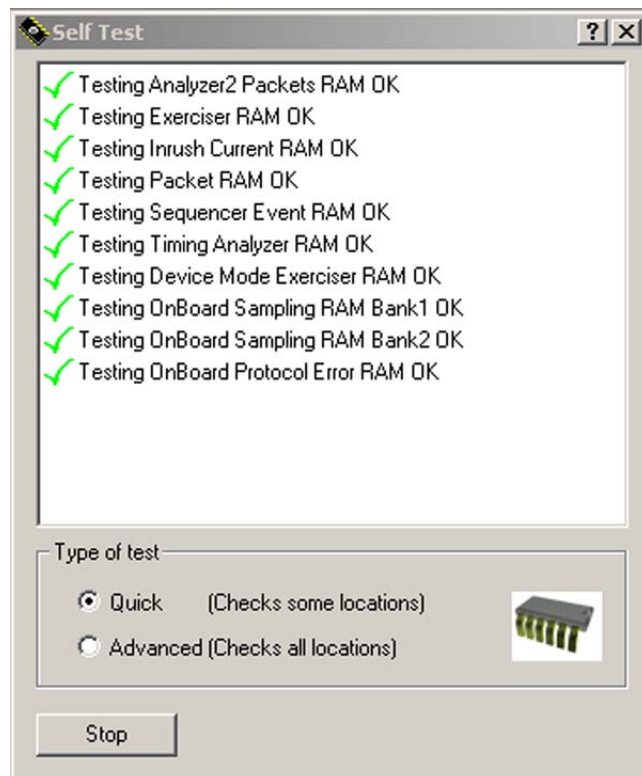
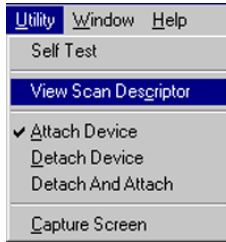


Figure 152 Self Test Dialog With Results Shown

View Scan Descriptors

You can view all the Device Descriptors of all the Devices that are connected to the Exerciser port of Conquest M2.

Click **Utility** and then choose **View Scan Descriptor**.



The View Scan Descriptors dialog opens.

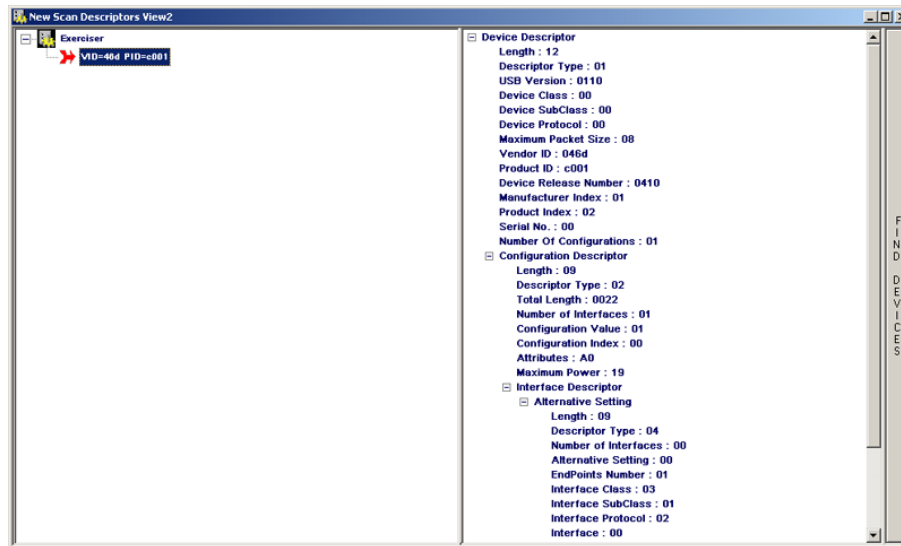


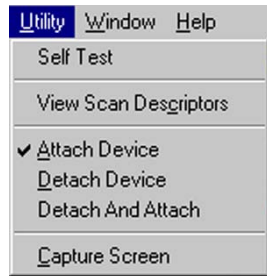
Figure 153 View Scan Descriptor Display

Click the **Find Devices** button to display the scan descriptors.

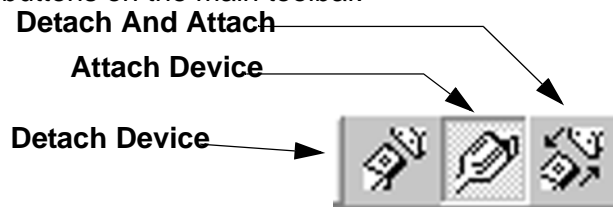
Attach/Detach Device

This feature allows you to attach and detach a device from the Analyzer without physically disconnecting the cable.

Click **Utility** and then choose **Attach Device**, **Detach Device** or **Detach And Attach** to perform an action.



You can alternatively perform the same actions by clicking the corresponding buttons on the main toolbar.



After you click **Detach And Attach**, the Device is detached for five seconds, with the remaining detached time countdown displayed.



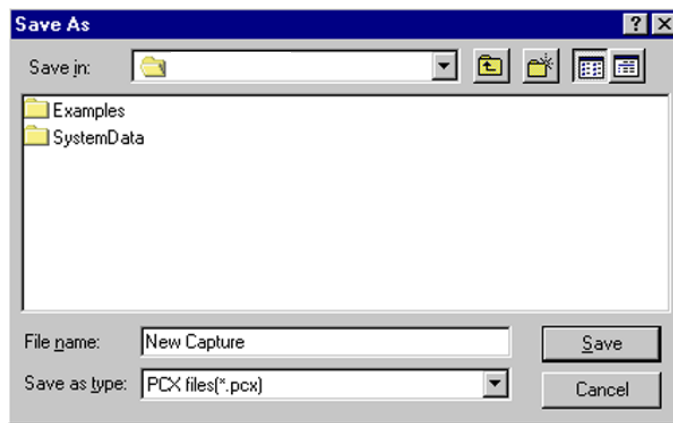
Capture Screen

You can perform a screen capture at any time and save it as a *.pcx graphics file for later review or inclusion in a report.

To perform a screen capture, click **Utility** and choose **Capture Screen**.



Position the crosshair cursor at a point from which to define the capture area and, with the left mouse button depressed, drag the cursor to define the capture area. Release the mouse button to open the **Save As** dialog.



Assign a file name to the screen capture to save and click **Save**. The capture progress bar opens. When complete, you can open the *.pcx file for viewing using a graphics program or import it in a word processing program such as Microsoft® Word.

Appendix A

Advanced Script Language (ASL)

The Advanced Script Language (ASL) is an extension of the Upper Level Protocol Decodes that allows you to define custom upper level protocol decodes in addition to those provided with the product.

The ASL enables you to extract and interpret arbitrary data from the USB stream by creating custom scripts.

You can use the ASL protocol decodes to decode any class or vendor-specific requests, descriptors, protocols, or any structured data transferred over the USB bus or to complement pre-defined protocols.

This appendix includes definitions, code snippets, and complete examples. You can write the decode script files in any text editor or use the Conquest Protocol Suite script editor (see example on page 242). You must files saved with an **.asl** file extension.

Document Conventions

This appendix uses the following convention to describe the script syntax.

Text Style	Meaning
bold	Functions, statements, or other reserved keywords
<i>italics</i>	Identifiers
<u>underscored</u>	Expressions
< >	Code enclosed in "<" and ">" is optional
normal	Integer constants
" "	Strings
parameter1 ,..., parameterN	Multiple entries
option1 option2	Choice of options

Language Elements

Integers

ASL accepts integer values in decimal, binary, or hexadecimal formats. Floating point and negative numbers are not accepted.

Examples:

Hexadecimal: 0x21, 0x0001, 0xA5A5

Binary: 0b00100001, 0b1, 0b101001011010010110100101

Decimal: 33, 1, 42405

Strings

Strings in ASL can be static character strings or can contain variable values that are calculated at runtime.

```
"<ConstantString1> <%d | %x | %b, ..., %d | %x | %b>
<ConstantString2>, <Value1,..., ValueN>"
```

ConstantString1, ConstantString2:

Character strings. These strings do not change at runtime.

%d, %x, %b, Value:

The value of the expressions is replaced by the corresponding strings in either decimal (%d), hexadecimal (%x), or binary (%b) formats.

Examples:

```
"This is a static string"
```

```
"Field length is %d, LengthOf(F1)"
```

Appendix A

Operators

The following lists the operators accepted by ASL. Operator precedence is that same as in C.

Operator	Function
()	Associative
and	Logical And
or	Logical Or
not	Logical Not
>	Greater than
<	Less than
<>	Equal to
=	Assignment
+	Arithmetic Addition
-	Arithmetic Subtraction
*	Arithmetic Multiplication
/	Arithmetic Division

Expressions

The expressions consist of one or more Integers, Functions, or Operators. For more detail about functions see page 231.

Examples:

```
0x2A and LengthOfF1
(ValueOf(F1)<2) or not (LengthOf(F2)<>10)
(4 + LengthOf(F1))* ValueOf(F2)
```

Comments

Comments are enclosed in `'/'` and `*/` and can span multiple lines.

```
Examples:
/* This is a single line comment */
/* This
   is a multi-line
   comment */
```

ASL Script Structure

An ASL Script file includes three types of sections:

- Protocol Extraction Section
- Protocol Decoding Section
- USB Descriptor Section

Each ASL script can include only one of each section type. Not every script file requires both *Protocol Decoding* and *USB Descriptor* sections, however as minimum each script file must contain a *Protocol Extraction* section, and one of *Protocol Decoding* or *USB Descriptor* sections.

Each section serves a specific function during the decoding process.

Section Type	Function
Protocol Extraction	Instructs the decoder on how to extract the high-level protocol data from USB transactions.
Protocol Decoding	Breaks the extracted data stream into protocol fields and decodes the data per each field.
USB Descriptor	Decodes descriptor information that is transferred by standard GetDescriptor and SetDescriptor requests.

A section consists of one or more blocks. A block is a procedure that is formed with a series of statements that are enclosed within the block. Each block has a dedicated function within the section and follows one of two formats as shown:

```

Blockname
    Statement1;
    Statement2;
    ....
    StatementN;
EndBlock
or
{
    Statement1;
    Statement2;
    ....
    StatementN;
}

```

Appendix A

Block Type	Function	Section(s)
ProtocolTransferRequirement	Defines the device endpoints used by the protocol.	Protocol Extraction
ProtocolTransferDefinition	Specifies the mechanism that is utilized for data transfer.	Protocol Extraction
Main	Splits the data stream into protocol fields.	Protocol Decoding
DefineOptions	Assigns interpretation values to the protocol or descriptor fields.	Protocol Decoding, USB Descriptor
ValidRange	Specifies valid ranges of values for the protocol fields.	Protocol Decoding, USB Descriptor
Descriptor	Splits the data stream into descriptor fields.	USB Descriptor

The statements within a block perform specific actions on the blocks of data, such as to assign a group of bytes to a protocol field.

Protocol Extraction Section

The *ProtocolExtraction* Section specifies how the data is extracted from traffic to one or more endpoints. This section is required in every script file and must precede all other sections. The *ProtocolExtraction* section begins with the *[ProductName=]* and ends with the *End* keywords. Two blocks are contained in this section, *ProtocolTransferRequirement* and *ProtocolTransferDefinition*.

[ProductName = SBAE]

ProtocolName = "NameString"

ProtocolTransferRequirement

```

{
  < USBControlTransfer
  {
    ControlType = Class | Vendor
    <ControlRequest = number1 <, ..., numberN>>
  } >
  < Endpoint = EndpointId1
  {
    EndpointNumber = number
    EndpointType = Bulk | Interrupt | ISO
    MaximumPacketSize = number
    Direction = In | Out
  } >
  .
  .
  .
  < Endpoint = EndpointIdN
  {
    EndpointNumber = number
    EndpointType = Bulk | Interrupt | ISO
    MaximumPacketSize = number
    Direction = In | Out
  } >
}
< ProtocolTransferDefinition
{
  ProtocolCommand = USBControlTransfer | DataStageUSBOfControlTransfer | EndpointId <MCS = number>
  <ProtocolData = EndpointId1 <, EndpointId2> <MDS = number>
  <IsOptional>>
  <ProtocolStatus = EndpointId <MSS = number> <IsOptional>>
  <ToggleEndpoints = EndpointId1, ..., EndpointIdN >
} >
End

```

Appendix A

ProductName

The product name for ASL files must be **SBAE**. This statement is required in every **.asl** file.

ProtocolName

The string *NameString* is the name of the implemented protocol that is shown in the User-defined page of the Upper Level Interpretation assignment dialog. This statement is required in every **.asl** file.

ProtocolTransferRequirement Block

This block identifies all endpoints involved in the transfer of the high-level protocol data. The data can be from either a control endpoint zero, non-control endpoints, or both. Each type of source has additional properties that completely specify the endpoint.

USBControlTransfer

This group of statements is required if the protocol data is transferred over the bus using the control transfer mechanism (for example, requests to endpoint zero) defined by USB.

ControlType

Specifies the type of USB device request used by the protocol, one of *Class* or *Vendor*. This statement corresponds to encoding in the "Type" field of the "bmRequestType" bitmap of a control transfer.

ControlRequest

If the protocol uses the data stage for some requests, this statement lists the request values of those requests. These values correspond to the "bRequest" values of a control transfer.

Endpoint

This group of statements identifies the attributes of non-control endpoints, if any, utilized by the protocol. Assign the described endpoint an identifier *EndpointId* to reference in the *ProtocolTransferDefinition* to specify the involvement of the endpoint in the data transfer.

EndpointNumber

This statement identifies the address of the described endpoint.

EndpointType

This statement identifies the transfer type of the endpoint, one of *Bulk*, *Interrupt*, or *ISO*.

MaximumPacketSize

This statement identifies the value of the “wMaxPacketSize” parameter associated with this endpoint.

Direction

This statement describes the direction of the endpoint, one of *In* or *Out*. Note that In and Out endpoints with same endpoint address must be two distinct endpoints using two *Endpoint* statement groups.

ProtocolTransferDefinition

In general, the mechanism by which high-level protocols are transferred over USB can consist of Command, Data, and Status stages. This block specifies how the Command, Data, and Status stages are arranged across the assigned endpoints. Depending on the protocol, you can omit the Data and Status stages. **Note:** Do not confuse the function of this block with the setup, data, and status stages of USB control transfers.

ProtocolCommand

This statement identifies the source of the Command stage. This must be one of *USBControlTransfer* (refers to setup stage of control transfers), *DataStageOfUSBControlTransfer* (refers to data stage of control transfers), or one of previously defined *EndpointIds*. To optimize the decoding, you can optionally assign the maximum number of bytes of the Command stage, using the **MCS** keyword.

ProtocolData

This statement identifies the source of the Data stage. This must be one or two previously defined *EndpointIds*. To optimize the decoding, you can optionally assign the maximum number of bytes of the Data stage, using the **MDS** keyword. The **IsOptional** keyword specifies that the Data stage is optional for some commands.

ProtocolStatus

This statement identifies the source of the Status stage. This must be a previously defined *EndpointIds*. To optimize the decoding, you can optionally assign the maximum number of bytes of the Status stage, using the **MSS** keyword. The **IsOptional** keyword specifies that the Status stage is optional for some commands.

ToggleEndpoint

This statement specifies if the source of the data stream varies between multiple *EndpointIds* listed.

Appendix A

Notes on Protocol Extraction Section

- If the stages have a clearly defined limitations in sizes, it is preferred to mention these sizes using the **MCS**, **MDS**, **MSS** keywords. This assists the decoder in detecting these stages correctly.
- It is valid to assign the same *EndpointId* to more than one stage. For example, both Data and Status stage.
- If the protocol does not obey the USB “Short Packet” or “Zero Length” packet mechanisms to identify the end of a transfer block, the protocol transfer may not be detected correctly. Please refer to sections 5.5.3, 5.7.3, and 5.8.3 of the USB Specification for more details about these mechanisms.
- If Data or Status stages are independent from Command stages, they must be separate protocols. For example, Hub Class (carries hub class requests) and Hub Class Notification (indicates change on any of hub ports) are separate pre-defined protocols. This rule often applies to any device with an Interface that includes an Interrupt endpoint. You must make this distinction when creating a protocol decode.

Example 1

```

[ProductName = SBAE]
ProtocolName = "Audio Class"
ProtocolTransferRequirement
{
    USBControlTransfer
    {
        ControlType = Class
    }
}
ProtocolTransferDefinition
{
    ProtocolCommand = USBControlTransfer
}
End

```

Example 2

```

[ProductName = SBAE]
ProtocolName = "CBI-SCSI"
ProtocolTransferRequirement
{
    Endpoint = Out2
    {
        EndpointNumber = 2
        EndpointType = Bulk
        MaximumPacketSize = 64
        Direction = Out
    }
    Endpoint = In2
    {
        EndpointNumber = 2
        EndpointType = Bulk
        MaximumPacketSize = 512
        Direction = In
    }
    USBControlTransfer
    {
        ControlType = Class
        ControlRequest = 0x0
    }
}
ProtocolTransferDefinition
{
    ProtocolCommand = DataStageOfUSBControlTransfer
    ProtocolData = In2, Out2 IsOptional
}
End

```

Appendix A

Example 3

```
[ProductName = SBAE]
ProtocolName = "Bluetooth HCI-Command"
ProtocolTransferRequirement
{
    USBControlTransfer
    {
        ControlType = Class
        ControlRequest = 0xE0
    }
}
ProtocolTransferDefinition
{
    ProtocolCommand = DataStageOfUSBControlTransfer
}
End
```

Protocol Decoding Section

The *ProtocolDecoding* Section details the decoding of the data stream into protocol fields. The section may include *DefineOptions* and *ValidRanges* blocks and must include the *Main* block. The beginning of this section is delimited by the *[Decode]* and *End* keywords.

[Decode]

<DefineOptions

```
Statement1;  
Statement2;  
...  
StatementN;
```

EndOptions >

< ValidRanges

```
Statement1;  
Statement2;  
...  
StatementN;
```

EndValidRanges >

```
{  
    Statement1;  
    Statement2;  
    ...  
    StatementN;  
}  
End
```

DefineOptions Block

This optional block allows you to assign meaningful interpretations of values in each field, to help you interpret the raw value by assigning a string that describes that value. The block consists of one or more *DefineOptions* statements.

DefineOptions

```
fieldId1 = ( "DescriptionString1", Value1 < - Value2> ; <
            "DescriptionString2", Value3 < - Value4> ; <
            .... >> );
.
.
.
fieldIdN = ( "DescriptionString3", Value5 < - Value6> ; <
            "DescriptionString4", Value7 < - Value8> ; <
            .... >> );
```

EndOptions

DefineOptions Statements

Remark:

Assigns description strings to the value or value ranges of the field. You can assign a unique description for some field values, all values, or ranges of values. There must be a separate *DefineOptions* statement for each field for which to assign a description. That applies to all fields and subfields added in the *Main* block.

Input Parameters:

fieldId:	Specifies the field. This identifier is the output of a <i>AddField</i> statement.
DescriptionString:	This string is a description for the specified value or range of values.
Value:	Specifies the field value for which to apply the given description string, or optionally specify the range of values, by identifying the lower range and upper range integers;

Example:

```
/* F3, F4 and BlockStatus are fields */
DefineOptions
  F3 = ("First Option", 0x0000-0x1000;
       "Second Option", 0x1001-4352);

  F4 = ("Option 1 for F4", 0x0-14,
       "Option 2 for F4", 0x0f);

  BlockStatus = ("Option for BlockStatus", 0-0x1100);
EndOptions
```

ValidRanges Block

This optional block specifies the legal range of values for the field. At run time, if the values of the field do not fall within the valid range, a Protocol Error is generated. The block consists of one or more *ValidRanges* statements that applies to all fields and subfields added in the *Main* block.

ValidRanges

```
fieldId1 = ( Value1 - Value2 );
```

```
.
```

```
.
```

```
.
```

```
fieldIdN = ( Value3 - Value4 );
```

EndValidRanges

ValidRange Statements

Remark:

Specifies the ranges of values expected for the field. There must be a separate *ValidRanges* statement for each field for which to assign a description.

Input Parameters:

fieldId:	Specifies the field. This identifier is the output of a <i>AddField</i> statement.
Value:	Specifies the field values for which to apply the given description string. You must specify the range of values by identifying the lower range and upper range integers.

Example:

```
/*F4, F1, BlockID are fields */
ValidRanges
  F4=(0x00, 0xAA);

  BlockID = (0,32);

  F1=(0b00001,0b0101010);

EndValidRanges
```

Appendix A

Main Block

This block contains the statements that break the raw data stream into protocol fields, allowing the decode of a wide range of protocols. Using branching statements, you can decode “dynamic” protocols, the fields for which, such as size or encoding, depend on the value(s) of other field(s).

```
{  
  
    Statement1;  
    Statement2;  
    .  
    .  
    .  
    StatementN;  
}
```

Main Block Statements

AddField

```
fieldId = AddField ( StartBit, Length, "Name", "Description", "Abbreviation",
                    < MSBLEFT | LSBLEFT > );
```

Remark:

This statement adds a field to the packet with the given specification. The value returned by the *CURPOS* function is automatically increased by the value of *Length* parameter. For more details about *CURPOS* function see page 231.

Output Parameter:

fieldId: Identifier assigned to the field.

Input Parameters:

StartBit: Specifies a zero-based start position of the field in the data, in bits. This parameter can be an integer or an expression.

Length: Specifies the length of the field, in bits. This parameter can be an integer or an expression.

Name: This string is displayed as the name of the field in the Interpreted Data table.

Description: This string is displayed as the description of the field in the Interpreted Data table.

Abbreviation: If the *Name* parameter exceeds the permitted length for the Interpreted Data table, this abbreviation is used instead. This parameter is not used in the current version of ASL but must be included.

MSBLEFT, LSBLEFT: These keywords specify the bit order of the decoded data, as either MSb->LSb or LSb->MSb. Note that bytes of the data stream are presented to the decoder in the same manner as they are transferred over USB, i.e LSB->MSB. This parameter is optional, and if not specified *MSBLEFT* is used by default.

Example:

```
/* This statement adds a field of 3 bits in length starting at bit 2 of packet.
   The data of this field is read from Lowest Bit first. */
```

```
F1 = AddField(2, 3, "Flags bitmap",
              "This field contains the flags bitmap.",
              "Flgs ", LSBLEFT);
```


Appendix A

BitStuff

BitStuff (*fieldId*, bit);

Remark:

This statement performs a bit-unstuffing algorithm on the field before the field is interpreted. This statement is only required for high-level data of some protocols. Do not confuse it with the low-level bit-unstuffing of USB. The hardware bit-unstuffs USB traffic before presenting the data stream to the decoder.

Input Parameters:

fieldId:	This is the identifier that specifies the field. This identifier is the output of a <i>AddField</i> statement.
bit:	Specifies the number of consecutive 1's before data is bit-stuffed with a zero by the protocol. For example, specifying this statement with this value being 5 converts the following stream "111110111b" into "11111111b".

Example:

```
/* run the bit-unstuff algorithm on 5 bits of continuous 1s */  
Data = AddField(8, 16,"Data","Data","");  
  
BitStuff(Data,5);
```

GetData

```
DataId = GetData(StartBit, Length);
```

Remark:

This statement assigns an identifier to a block of data from the data stream. You can reference the complete data block elsewhere in the script using the returned identifier. Unlike the *AddField* statement, this statement does not increase the value returned by *CURPOS* function. Typically this statement is used to retrieve the value of a field without adding that field to the decode.

Output Parameters:

DataId: This is the identifier assigned to a data block.

Input Parameters:

StartBit: Specifies a starting position of the field in the data, in bits. This parameter can be an integer or an expression.

Length: Specifies the length of the field, in bits. This parameter can be an integer or an expression.

Example:

```
/*This function links the Data from Bit#0 to Bit#9 in a dataId named  
DataSegment.*/  
DataSegment = GetData (0,9);
```

Appendix A

If, Else If, Else

```
if Expression1 then
    Main block 1
< Else If Expression2 then
    Main block 2 >
< Else If Expression3 then
    Main block 3 >
...
<Else
    Main block 4 >
```

Remark:

If, Else If, If statements, collectively known as branch statements, permit the execution of different blocks of statements based on the evaluation of the expressions. Following the convention of other programming languages, if the expression evaluates to true the immediate block is executed. Otherwise the control is passed to the next statement. Nesting of branch statements within other branch statements is allowed.

Example

```
If CURPOS > 5 and ValueOf(F1) < 8 then
{
    H1 = AddField(CURPOS,8,"H1","Field H1","H1");
}
Else If(CURPOS < 9) then
{
    H2 = AddField(CURPOS,8,"H2","Field H2","H2");
}
Else
{
    H4 = AddField(CURPOS,548,"H4","Field H4","H4");
}
```

Repeat**Repeat** Expression

Main block

Remark:

This statement repeats all statements in the block while Expression evaluates to TRUE.

Example

```
F1 = AddField(0, 20, "Field1", "", "F1");  
  
/* Loop repeats until current position >= 100 */  
Repeat (CURPOS < 100)  
{  
  SubF1 = AddField(CURPOS, 23, "SubField of field F1",  
    "subfield of F1", "SubF1");  
}
```

Appendix A

Repeat Count

Repeat Count = Expression

Main block

Remark:

This statement repeats all statements in the block Expression number of times.

Example

```
F1 = AddField(0, 20, "Field1", "", "F1");
```

```
/* Loop runs twice */
```

```
Repeat Count = 2
```

```
{
```

```
  F1 = AddField(CURPOS, 23, "SubField of field F1",  
               "subfield of F1", "SubF");
```

```
}
```

SubFieldOf**SubFieldOf (fieldId <, Separated >)**

Main Block

Remark:

This statement adds subfield(s) to the previously added field. All *AddField* statements that exist within the body of this statement are relative only to the specified field. Up to 8 levels of nested subfield statements is permitted.

Input Parameters:

fieldId: This is the identifier that specifies the parent field. This identifier is the output of a *AddField* statement.

Separated: This keyword specifies the display of subfield(s) in the Interpret Data tables. If you do not include the *Separated* keyword, then the subfield(s) are shown in the tables as a subset of the parent field and can be expanded or collapsed with an arrow. If you include the *Separated* keyword, a separate table is created for the subfield.

Example

```
F1 = AddField(2, 3, "Field1", "It starts form bit# 2", "F1");
F2 = AddField(5, 3, "Field2", "This field is used for..", "F2");

SubFieldOf(F1)
{
  s1 = AddField(0, 1, "Subfield1", "This is subfield of F1", "s1");

  s2 = AddField(1, 2, "s2", "a Subfield of f1",
               "subfield f1");
}
/* Viewer shows these subfields in a separate table */
SubFieldOf(F2, Separated)
{
  s21 = AddField(0, 1, "Subfield1", "This is subfield of F2", "s1");

  s22 = AddField(1, 2, "s2", "Subfield of F2",
                "subfield of F1");
}
```

Appendix A

SetTableHeader

SetTableHeader ("Title" | *OptionalfieldId*);

Remark:

This statement sets the header title of the Interpreted Data table in the Viewer. Either a string constant or the value from a previously defined field can be the title of the table.

Input Parameters:

Title:	Specifies the string to show in the header of the table.
OptionalfieldId:	This identifier specifies the field from which to get the header string. It is the output of a <i>AddField</i> statement, and must have a description assigned in the DefineOptions block.

Examples

```
OpCode = AddField (0, 8, "Op Code", "Op Code",  
                  "Op Code");  
If (ValueOf (OpCode) <> 0x01) then  
{  
    SetTableHeader ("Read");  
}
```

SetFieldTableHeader

SetFieldTableHeader (fieldId,"Title" | OptionalfieldId);

Remark:

If you specified a subfield to use a separate table, this statement sets the header title of the Interpreted Data table for that subfield. Either a string constant or the string from a previously defined field can be the title of the table.

Input Parameters:

fieldId:	Specifies the subfield with a dedicated table. This identifier is the output of a <i>AddField</i> statement.
Title:	Specifies the string to show in the header of the table.
OptionalfieldId:	Specifies the field from which to get the header string. This identifier is the output of a <i>AddField</i> statement, and must have a description assigned in the DefineOptions block.

Examples

```
Type = AddField (CURPOS, 8, "Type", "Type", "Type");

SubFieldOf(Type, Separated)
{
    SetFieldTableHeader(Type, "Sub fields of Type");
    T1= AddField (0, 4, "T1", "T1", "T1");
    T2= AddField (4, 8, "T2", "T2", "T2");
}
```


USB Descriptor Section

The *USBDescriptor* Section simplifies the decoding of non-standard descriptors. Any class or vendor-specific descriptor that is returned by the standard GetDescriptor or SetDescriptor USB requests, is decoded in this section. The section may include *DefineOptions* and *ValidRanges* blocks and must include the *Descriptor* block. The beginning of this section is delimited by the *[USB Decode]* and *End* keywords. Please see the description of *DefineOptions* and *ValidRanges* block in the *ProtocolDecode* section.

[USB Descriptors]

<DefineOptions

```
Statement1;  
Statement2;  
...  
StatementN;
```

EndOptions >

< ValidRanges

```
Statement1;  
Statement2;  
...  
StatementN;
```

EndValidRanges >

< *InterfaceClassId* = **GetInterfaceClass**; >

< *InterfaceSubClassId* = **GetInterfaceSubClass**; >

< *InterfaceProtocolId* = **GetInterfaceProtocol**; >

Descriptor(Type)

```
{  
Statement1;  
Statement2;  
...  
StatementN;  
}
```

End

GetInterfaceClass, GetInterfaceSubClass, GetInterfaceProtocol Statements

These statements are available for assigning identifiers to the “bInterfaceClass”, “bInterfaceSubClass”, and “bInterfaceProtocol” fields of the standard Interface descriptor, respectively.

Descriptor Block

This block is similar to the *Main* block of the *ProtocolDecode* Section in that it allows you to assign meaningful interpretations of values in each field, to interpret the raw value by assigning a string that describes that value. The block consists of one or more statements that have the same usages as those in the *Main* block of the *ProtocolDecode* section. Notice that this block has a *Type* input parameter. This must be a one-byte integer that corresponds to the Descriptor Type ("wValue" field) parameter of GetDescriptor() and SetDescriptor() requests.

Descriptor(Type)

```
{
    Statement1;
    Statement2;
    ...
    StatementN;
}
End
```

Example

```
[ProductName = SBAE]
ProtocolName = "Audio Class Descriptor"
End
[USB Descriptors]
InterfaceSubClass = GetInterfaceSubClass;
Descriptor (0x25)
{
If (ValueOf (InterfaceSubClass) <> 0x02) then
{
/*Decode statements*/
}
}
End
```

Appendix A

Functions

CURPOS

CURPOS

Remark:

This function returns the current position of the decoder cursor in bits. *CURPOS* is initialized to zero at the beginning of the decode and increases by *Length* parameter in each instance of *AddField* statement. *CURPOS* has a local scope within the *SubFieldOf* statements. In each *SubFieldOf* statement the *CURPOS* is a local variable that is initialized to zero, and the incremented value applies only within the braces of that block.

Example

```
F1 = AddField(0, 8, "F1", "F1", "F1");

/* CURPOS is equal to 8 and becomes 16 following this statement */
F2 = AddField(CURPOS, 8, "F2", "F2", "F2");

SubFieldOf(F2)
{
    F11 = AddField(0, 2, "F11", "F11", "F11");

    /* CURPPOS = 2 */
    F12 = AddField(CURPOS, 2, "F12", "F12", "F12");

    /* CURPPOS = 4 */
    F13 = AddField(CURPOS, 4, "F13", "F13", "F13");

    /* CURPPOS = 16 */
    F3 = AddField(CURPOS, 2, "F3", "F3", "F3");
}
```

EOD

EOD

Remark:

If *CURPOS* has reached the end of the data stream, this function returns a TRUE. Otherwise, a FALSE is returned. You can use this function to detect unexpected data at the end of the decode.

INPUTDATA**INPUTDATA**Remark:

The *INPUTDATA* is a symbolic representation of the entire data stream that is passed to the decoder.

Find**Find (StartBit, Pattern)**Remark:

The function searches for a data pattern in the data stream. It returns the bit location of the first instance of the data pattern, or the value equivalent to `LengthOf (INPUTDATA+1)` if the data pattern is not found.

Input Parameters:

StartBit: Specifies a zero-based start position of the field in the data, in bits. This parameter can be an integer or an expression.

Pattern: Identifies the data pattern for which to search. This parameter can be an integer or an expression.

LengthOf**LengthOf (DataId | INPUTDATA)**Remark:

The function returns the length of data in bits

Input Parameters:

DataId Identifier that specifies data block. This identifier is the output of a *GetData* statement.

INPUTDATA: The *INPUTDATA* function.

Example:

```

/* Add unknown field if excess data detected. Demonstrates use of EOD,
   CURPOS, LengthOf, and INPUTDATA functions */
if (not EOD)
{
  Unexpected = AddField(CURPOS,
    LengthOf(INPUTDATA) - CURPOS,
    "Unexpected", "Unexpected Data",
    "Unexpected");
}

```

Appendix A

StartOf

StartOf (*fieldId*)

Remark:

The function returns the starting bit position of given field.

Input Parameters:

fieldId: This is the identifier that specifies the field. This identifier is the output of a *AddField* statement.

ValueOf

ValueOf (*Identifier*)

Remark:

The function returns the value of the field or data.

Input Parameters:

Identifier: Specifies the field or the data block. The identifier the output of a *AddField* statement or the output of a *GetData* statement.

Samples

Sample1: Portion of Communication Class

```

[ProductName = SBAE]
ProtocolName = "Communication Class"
ProtocolTransferRequirement
{
    USBControlTransfer
    {
        ControlType = Class
    }
}
ProtocolTransferDefinition
{
    ProtocolCommand = USBControlTransfer
}
End
[Decoding]
DefineOptions
    bRequest = ("SEND_ENCAPSULATED_COMMAND", 0x00;
               "GET_ENCAPSULATED_RESPONSE", 0x01;
               "SET_COMM_FEATURE", 0x02;
               "GET_LINE_PARAMS", 0x35;
               "GET_ATM_VC_STATISTICS", 0x53;
               "RESERVED ", 0x54-0xFF);

    CallStateValue = ("Call is idle.",0x00 ;
                     "Typical Dial Tone",0x01 ;
                     "Interrupted Dial Tone",0x02 ;
                     "Dialing is in progress",0x03 ;
                     "Ringback.",0x04 ;
                     "Connected.",0x05 ;
                     "Incoming call.",0x06 );

    CallStateActiveFlag = ("No Activity on the line", 0x00 ;
                           "Line is active", 0x01);

EndOptions
{
bRequestType = AddField(0, 8, "bmRequestType", "bmRequestType Field", "bRequestType",
                        MSBLEFT);

bRequest = AddField(8, 8, "bRequest", "bRequest Field", " bRequest",
                    MSBLEFT);

wValue = AddField(16,16," wValue" ," wValue Field","wValue",MSBLEFT);

if (ValueOf(bRequest) = 0x35) then /*GET_LINE_PARAMS*/
{
    wInterface = AddField(32,16,"wInterface" ," wInterface Field"," wInterface",MSBLEFT);
    wLength = AddField(48,16," wLength" ," Size of structure"," wLength", MSBLEFT);
}

If ValueOf(wLength) > 0 Then

```

Appendix A

```
{
    wInfoLength = AddField(64,16," wInfoLength", " Size of this structure", "
        wInfoLength",MSBLEFT);

    dwRingerBitmap = AddField(80,32," dwRingerBitmap", "Ringer Configuration bitmap for this
        line.", "wInfoLength",MSBLEFT);

    dwLineState = AddField(112,32," dwLineState", "Define current state of the
        line.", "wInfoLength",MSBLEFT);

    SubFieldOf(dwLineState)
    {
        Index = AddField(0,8,"Index", " Index of active call on this
            line.", "Index", MSBLEFT);

        Reserved = AddField(CURPOS, 23, " Reserved", "Reserved",
            "Reserved",MSBLEFT);

        AciveFlag = AddField(CURPOS,1,"Active Flag", "Active Flag",
            "AciveFlag", MSBLEFT);
    }

    Repeat (CURPOS - StartOf(wInfoLength)) < ValueOf(wInfoLength)
    {
        CallStateValue = AddField(CURPOS,8," CallStateValue", "Call
            State Value", "CallStateValue", MSBLEFT);

        CallStateChangeValue =AddField(CURPOS,8,"CallStateChangeValue",
            "Call State change value",
            "CallStateChangeValue", MSBLEFT);

        Reserved = AddField(CURPOS,15,"Reserved" "Reserved", "Reserved",
            MSBLEFT);

        CallStateActiveFlag = AddField(CURPOS,1, "ActiveFlag",
            "ActiveFlag", "ActiveFlag", MSBLEFT);
    }
}
End
```

Sample2: LCP (PPP)

```

[ProductName = SBAE]
ProtocolName = "LCP"
ProtocolTransferRequirement
{
    USBControlTransfer
    {
        ControlType = Class
    }
}
ProtocolTransferDefinition
{
    ProtocolCommand = USBControlTransfer
}
End

[Decoding]
{
    /*
    The script of other protocols goes here.
    */

Repeat (not EOD)
{
    Data1 = GetData(CURPOS, 16);

    /* 0xc021 is the LCP code in PPP Protocol. */
    If (ValueOf(Data1) = 0x21 ) Then
    {
        ProtocolHeader = AddField(CURPOS, 16, "ProtocolHeader", "The
                                protocol is LCP", "LCP");

        Code = AddField(CURPOS, 8, "LCP Code", "LCP Code", "LCP Code");
        Id = AddField(CURPOS, 8, "Id LCP", "Id LCP", "Id LCP" );
        Length = AddField(CURPOS, 16, "Length", "Length", "Length");

        Repeat (CURPOS - StartOf(ProtocolHeader) < ValueOf(Length)
        {
            Type = AddField(CURPOS, 8, "Type", "Type", "Type");
            TypeLength=AddField(CURPOS,8,"TypeLength","TypeLength",
                               "TypeLength");
            If (ValueOf(Type) = 0x26 ) Then
            {
                Repeat CURPOS - StartOf(Type) < ValueOf(Length)
                {
                    LengthElision=AddField(CURPOS,8,"LengthElision",
                                           "LengthElision", "LengthElision");
                    Elision = AddField(CURPOS,ValueOf(LengthElision),
                                       "Elision", "Elision", "Elision");
                }
            }
            Else If (ValueOf(Type) = 0x28 ) Then
            {
                Internationalization = AddField(CURPOS,
                                                16,"Internationalization",
                                                "Internationalization",
                                                "Internt");
            }
        }
    }
}

```


Appendix A

```
        If (ValueOf(Length) > 32) then
        {
        Tag = AddField(CURPOS, ValueOf(Length)-32, "Tag", "Tag",
        "Tag");
        }
        }
    }
}
/* Here are the remaining protocols. */
}
}
}
End
```

Sample3: HID Class

```

[ProductName = SBAE]
ProtocolName = "HID"
ProtocolTransferRequirement
{
    USBControlTransfer
    {
        ControlType = Class
    }
}
ProtocolTransferDefinition
{
    ProtocolCommand = USBControlTransfer

}
End

[USB Descriptors]
DefineOptions

    bDescriptorType = ("HID",0x21;
                       "Report",0x22;
                       "Physical",0x23;
                       "Reserved",0x24-0x2F);

    bCountryCode = ("Not Supported",0;
                   "Arabic",1;
                   "Belgian",2;
                   "Canadian-Bilingual",3;
                   "Canadian-French",4;
                   "Czech Republic",5;
                   "Danish",6;
                   "Finnish",7;
                   "French",8;
                   "German",9;
                   "Greek",10;
                   "Hebrew",11;
                   "Hungary",12;
                   "International (ISO)",13;
                   "Italian",14;
                   "Japan (Katakana)",15;
                   "Korean",16;
                   "LatinAmerican",17;
                   "Netherlands/Dutch",18;
                   "Norwegian",19;
                   "Persian (Farsi)",20;
                   "Poland",21;
                   "Portuguese",22;
                   "Russia",23;
                   "Slovakia",24;
                   "Spanish",25;
                   "Swedish",26;
                   "Swiss/French",27;
                   "Swiss/German",28;
                   "Switzerland",29;

```

Appendix A

```
"Taiwan",30;
"Turkish-Q",31;
"UK",32;
"US",33;
"Yugoslavia",34;
"Turkish-F",35;
"Reserved",36-255);

Bias = ("Not applicable",0; "Right hand",1;"Left
hand",2;"Both hands",3;
"Either hand",4;"Reserved",5-7 );
Qualifier = ("Not applicable",0;"Right",1;
"Left",2;"Both",3; "Either",4;
"Center",5;"Reserved",6-7);

bDesignator =("None",0x00;
"Hand",0x01;
"Eyeball",0x02;
"Eyebrow",0x03;
"Eyelid",0x04;
"Ear",0x05;
"Nose",0x06;
"Mouth",0x07;
"Upper lip",0x08;
"Lower lip",0x09;
"Jaw",0x0A;
"Neck",0x0B;
"Upper arm",0x0C;
"Elbow",0x0D;
"Forearm",0x0E;
"Wrist",0x0F;
"Palm",0x10;
"Thumb",0x11;
"Index finger",0x12;
"Middle finger",0x13;
"Ring finger",0x14;
"Little finger",0x15;
"Head",0x16;
"Shoulder",0x17;
"Hip",0x18;
"Waist",0x19;
"Thigh",0x1A;
"Knee",0x1B;
"Calf",0x1C;
"Ankle",0x1D;
"Foot",0x1E;
"Heel",0x1F;
"Ball of foot",0x20;
"Big toe",0x21;
"Second toe",0x22;
"Third toe",0x23;
"Fourth toe",0x24;
"Little toe",0x25;
"Brow",0x26;
"Cheek",0x27;
"Reserved",0x28-0xFF);
```

EndOptions

```

Descriptor (0x21)
{
    bLength = AddField(CURPOS,8,"bLength","Numeric expression that is the
        total size of the HID descriptor","bLength");

    bDescriptorType = AddField(CURPOS, 8,"bDescriptorType", "Constant
        name specifying type of HID descriptor",
        "bDescriptorType");

    bcdHID = AddField(CURPOS, 16,"bcdHID", "Numeric expression
        identifying the HID Class Specification release",
        "bcdHID");

    bCountryCode = AddField(CURPOS, 8,"bCountryCode", "Numeric expression
        identifying country code of the localized
        hardware","bCountryCode");

    bNumDescriptors = AddField(CURPOS, 8,"bNumDescriptors", "Numeric
        expression specifying the number of class
        descriptors (always at least one i.e. Report
        descriptor.)","bNumDescriptors");

    Repeat Count = ValueOf(bNumDescriptors)
    {
        bDescriptorType = AddField(CURPOS, 8,"bDescriptorType",
            "Constant name specifying type of HID descriptor",
            "bDescriptorType");

        wDescriptorLength = AddField(CURPOS, 16,"wDescriptorLength",
            "Numeric expression that is the total size of the
            Report descriptor", "wDescriptorLength");
    }
    If(Not EOD) Then
    {
        Unexpected = AddField(CURPOS, LengthOf(INPUTDATA)-
            CURPOS,"Unexpected", "Unexpected", "Unexpected");
    }
}
Descriptor (0x23)
{
    bNumber = AddField(CURPOS, 8, "bNumber", "Numeric expression
        specifying the number of Physical Descriptor sets. Do not
        include Physical Descriptor 0 itself in this
        number", "bNumber");

    bLength = AddField(CURPOS, 16, "bLength", "Numeric expression
        identifying the length of each Physical
        descriptor", "bLength");

    bPhysicalInfo = AddField(CURPOS,8, "bPhysicalInfo", "bPhysicalInfo",
        "bPhysicalInfo");

    SubFieldOf(bPhysicalInfo)
    {
        Preference = AddField(0, 5, "Preference", "The Preference field
            indicates whether the descriptor set contains preferred or
            alternative designator information. A vendor defines the
            Preference value of 0 for the most preferred or most typical set
            of physical information. Higher Preference values indicate less
            preferred descriptor sets", "Preference");

        Bias = AddField(5, 3, "Bias", "The Bias field indicates which
            hand the descriptor set is characterizing. This may not
            apply to some devices", "Bias");
    }
    Repeat Count = ValueOf(bNumber)
    {

```

Appendix A

```
dPhysical = AddField(CURPOS,16,"dPhysical","dPhysical",
                    "dPhysical");

SubFieldOf(dPhysical)
{
    bDesignator = AddField(0, 8, "bDesignator","Designator
value; indicates which part of the body affects the
item", "bDesignator");

    bFlags = AddField(CURPOS,8,"bFlags","bFlags", "bFlags");

    SubFieldOf(bFlags)
    {
        Effort = AddField(0, 5, "Effort", "The Effort field
indicates how easy it is for a user to access the
control. A value of 0 identifies that the user can
affect the control quickly and easily. As the value
increases, it becomes more difficult or takes
longer for the user to affect the control",
"Effort");

        Qualifier = AddField(CURPOS, 3, "Qualifier", "The
Qualifier field indicates which hand (or half
of the body) the designator is defining. This
may not apply to for some devices",
"Qualifier");
    }
}

if(Not EOD) Then
{
    Unexpected = AddField(CURPOS, LengthOf(INPUTDATA)- CURPOS,
                        "Unexpected", "Unexpected", "Unexpected");
}
}

End
```

Note: In case of discrepancies between the sample code shown and that described by the text, the text description shall take precedence.

Script Editor

The following example illustrates the creation of a Mass Storage Class request script.

Table 7 shows the Class request descriptor field interpretation that is used in the script example.

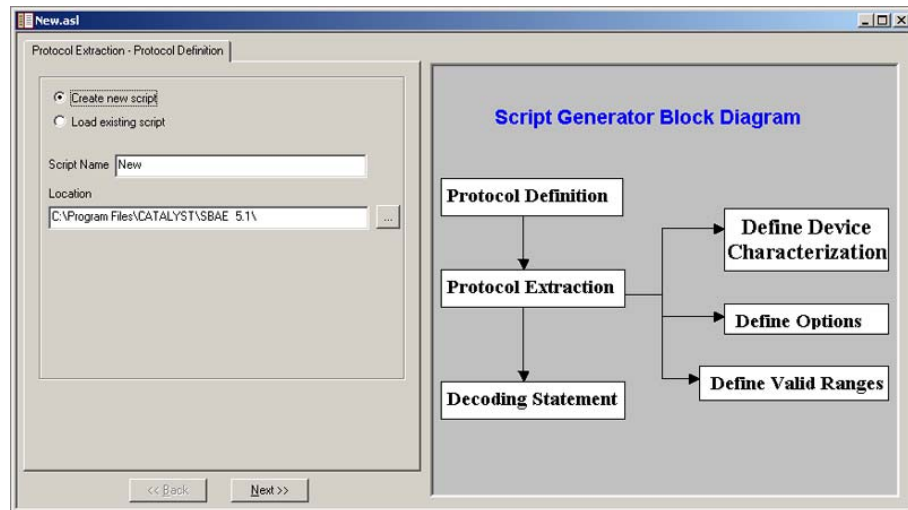
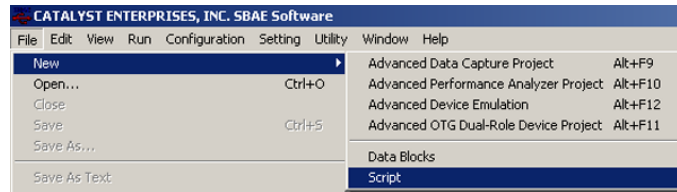
Table 7. Class Request Descriptor

Table 7. Class Request Descriptor				
0	bmRequesType	1	Bits 0...4 (Recipient)	Interpretation
			0b00000(0x00)	"Device"
			0b00001(0x01)	"Interface"
			0b00010(0x02)	"Endpoint"
			0b00011(0x03)	"Other"
			0x04-0x1F	"Reserved"
			Bits 6...5 (Type)	
			0x0	"Standard"
			0x1	"Class"
			0x2	"Vendor"
			0x3	"Reserved"
			Bit 7 (Direction)	
			0x0	"Host to device"
			0x1	"device to Host"
1	bRequest	1	Bits 0...8 (Request)	
			0x00-0xFD	"Unknown"
			0xFE	"Get Max LUN"
			0xFF	"Mass Storage Reset"
2	Wvalue	2		No field option needed. (Same value appears.)
3	Windex	2		No field option needed. (Same value appears.)
4	wLength	2		No field option needed. (Same value appears.)

Note: You must assign an option name to all fields that have multiple interpretations based on their values.

Creating the Script

1. Select **File > New** and choose **Script** to open the script generator dialog.



2. Enter a **Name** for the Script, enter a path to a location in which to save it, and click **Next**.

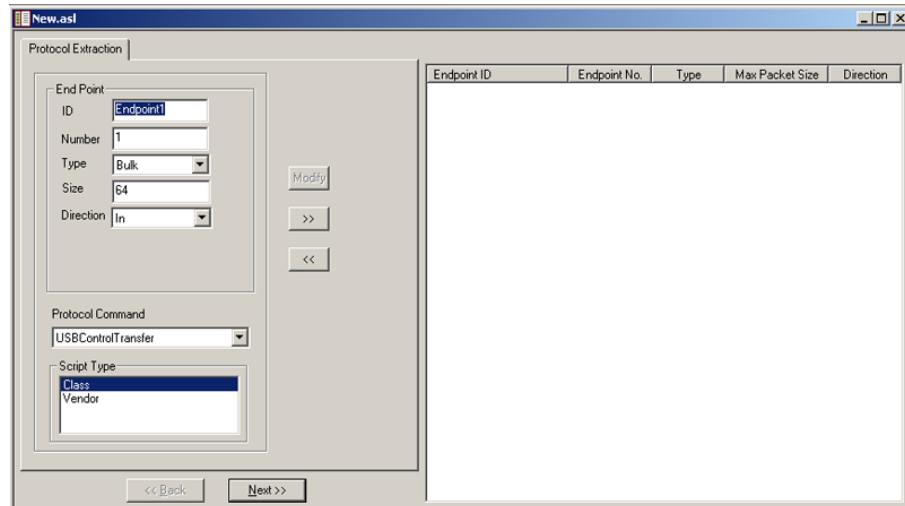


Figure 154 Choosing Command and Class

- Chose **USBControlTransfer** as protocol command and **Class** as Script Type, then click **Next**.

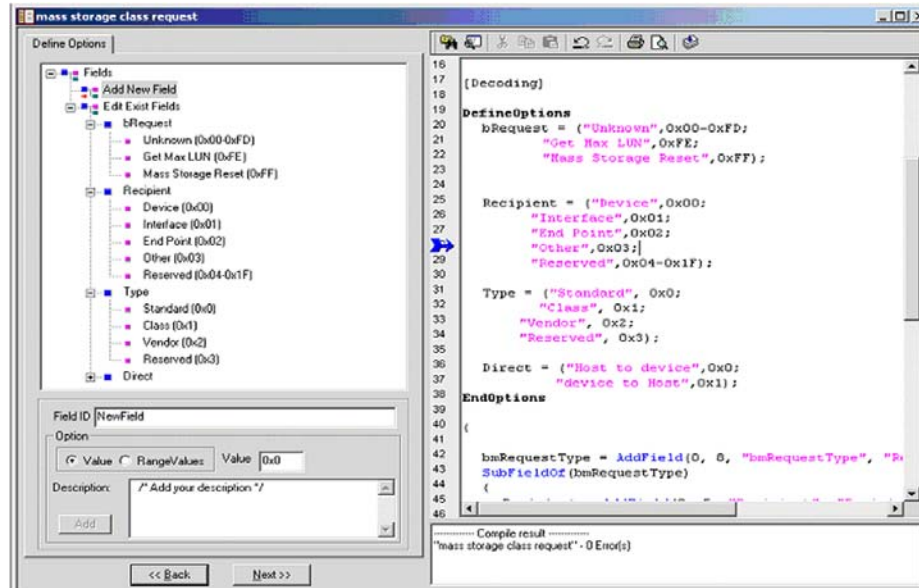


Figure 155 Defining Field Options

- Define the Field Options listed in Table 7 .
 - Select **Add New Field** from Tree and define proper ID name and option Value (or Value range)+ Description.
 - To add a new option to a field, select the field from the tree, define a value and a Description, and then click the **Add** button.
 - To modify an Added option, select it from tree and modify its descriptions or values from Edit boxes below the tree control. (You can modify it from the editor directly)
- Click **Next**.
- Add a Valid Range to your script, if needed. (You do not need to define any valid range for this example script.) (Press the **Next** button to go to the Next page.)
- Add your decoding statements to the decoding area by defining your command and adding it as a decode command.

Appendix A

Note: The command is placed at the cursor location in the decoding area. If the cursor is above the decoding area, the command is placed at the top. If it is under the decoding area, it is added at the end of the decoding area.

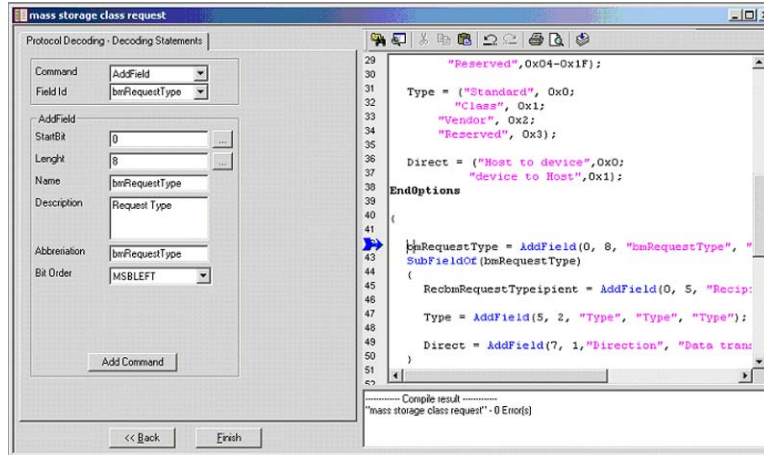


Figure 156 Command Location

China Restriction of Hazardous Substances Table

The following tables are supplied in compliance with China's Restriction of Hazardous Substances (China RoHS) requirements:

部件名称	有毒有害物质和元素					
	铅 (Pb)	汞 (Hg)	镉 (Cd)	六价铬 (Cr ⁶⁺)	多溴联苯 (PBB)	多溴二苯醚 (PBDE)
PCBAs	X	O	X	X	X	X
机械硬件	O	O	X	O	O	O
金属片	O	O	X	O	O	O
塑料部件	O	O	O	O	X	X
电源	X	X	X	O	X	X
电源线	X	O	X	O	X	X
保护外壳(如有)	O	O	O	O	X	X
电缆组件(如有)	X	O	X	O	X	X
风扇(如有)	X	O	X	O	X	X
交流滤波器和熔丝组件(如有)	X	O	X	O	O	O
外部电源(如有)	X	X	X	O	X	X
探头(如有)	X	O	X	O	X	X
O: 表明该有毒有害物质在该部件所有均质材料中的含量均在 SJ/T11363-2006 标准规定的限量要求之下。						
X: 表明该有毒有害物质至少在该部件的某一均质材料中的含量超过 SJ/T11363-2006 标准规定的限量要求。						

EFUP (对环境友好的使用时间) 使用条件:

温度: 5摄氏度到40摄氏度

湿度: 5% - 95%最大相对湿度 (无冷凝)

高度: 最高2000米

Part Name	Toxic or Hazardous Substances and Elements					
	Lead (Pb)	Mercury (Hg)	Cadmium (Cd)	Hexavalent Chromium (Cr ⁶⁺)	Polybrominated Biphenyls (PBB)	Polybrominated Diphenyl Ethers (PBDE)
PCBAs	X	O	X	X	X	X
Mechanical Hardware	O	O	X	O	O	O
Sheet Metal	O	O	X	O	O	O
Plastic Parts	O	O	O	O	X	X
Power Supply	X	X	X	O	X	X
Power Cord	X	O	X	O	X	X
Protective Case (if present)	O	O	O	O	X	X
Cable Assemblies (if present)	X	O	X	O	X	X
Fans (if present)	X	O	X	O	X	X
AC Filter/Fuse Assy (if present)	X	O	X	O	O	O
Ext Power Supply (if present)	X	X	X	O	X	X
Probes (if present)	X	O	X	O	X	X
O: Indicates that this toxic or hazardous substance contained in all of the homogeneous materials for this part is below the limit requirement specified in SJ/T11363-2006.						
X: Indicates that this toxic or hazardous substance contained in at least one of the homogenous materials used for this part is above the limit requirement specified in SJ/T11363-2006.						

EFUP (Environmental Friendly Use Period) Use Conditions:

Temperature 5C to 40C

Humidity 5% to 95% max RH (non-condensing)

Altitude Up to 2000 meters

Index

A

- activate
 - device mode 140
- additional requests
 - in device emulation 138
- advanced refresh
 - in exerciser program 86
- Advanced Script Language 205
- analysis
 - new 117
 - pre-defined 117
 - real time 116
- analysis Expressions
 - creating 119
- analyzer options 5
- analyzer speed 54
- ASL
 - functions 231
 - language elements 206
 - protocol decoding 216
 - protocol extraction 210
 - script structure 208
 - USB descriptor section 229
- authorization code
 - for upgrades 24
- auxiliary Port 57

B

- bit width
 - adjustable 83
- bookmarks 189
 - finding 190
- boolean expression
 - in sequencer 67

C

- capture
 - partial data payload 56
- capture screen 203
- class
 - specifying new 74
- clock
 - non standard 54
- Conquest
 - as analyzer 14
 - as exerciser 14
 - included components 9

- Conquest M2 1
- control transfers 73
- counter
 - as data 93
- counters
 - in sequencer 66
- create a new analysis 117
- current measurement 157
 - calibration 174
 - inrush compliance 167
 - suspend 172
 - unconfigured 158
- cursors
 - inrush current display 170
 - locating 188
 - positioning 188

D

- data
 - decoded 104
 - in transaction layer 79
 - in transfer layer 78
 - interpreting 104
 - specifying for transfer 78
- data block
 - counter data 93
 - custom pattern 92
 - defining 90
 - editing as text 95
 - naming 91
 - random pattern 93
 - walking bit 94
- data report 99
 - settings 100
- DC Compliance 5
- decodes
 - user defined 106
- default capture 29
- descriptors
 - independent 139
 - string 139

- device
 - attach/detach 202
 - configuration define 136
 - definition 135
 - emulation 133
 - mode 133
- device emulation 5
 - advanced mode 142
 - connections 13
- device interface
 - define 136
- display
 - cell color 197
 - fonts 198
 - inrush current 169
 - viewing preferences 178
- display Configuration 196
- display manipulation 177
- E**
- easy mode
 - pre-defined setups 37
- enable exerciser
 - easy mode 53
- endpoint
 - define 137
 - settings 76
- endpoint errors
 - definition of 141
- error message
 - startup 16
- errors
 - forcing 80
 - in user defined scripts 107
 - protocol 108
- ethernet
 - connecting with 21
- example
 - files 25
 - folder 25
- exclude
 - KeepAlive events 39
 - NAK transactions 39
 - NYet transactions 39
 - protocol errors 45
 - SOF events 39
- exercise and capture
 - setup 50
- exerciser
 - configuration 11
 - programming in advanced mode 71
 - programming in easy mode 51
- exerciser program
 - copying and pasting 87
 - importing 84
 - shortcuts 84
 - terminating in OTG 155
- expand/close
 - splits and transactions 79
- expressions
 - for analysis 119
- external
 - signals 6
- external I/O Connector
 - pin assignment 15
- external outputs 66
 - programming 66
- external Trigger
 - triggering options 55
- external trigger
 - enable 44
- F**
- file type
 - definition 26
- filter 182
 - access from menu bar 184
- filtering 182
 - smart, on screen 183
- forcing errors 80
- H**
- hardware setup 10
- headers
 - show/hide 179
- high level interpretation 105
- high level interpretation
 - protocol 105
- Host Exerciser 5
- host exerciser
 - programming with transfers 72
- I**
- idle time
 - show/hide in display 180
- inrush current compliance 167
- insert
 - blank line in timing display 185

- insert line
 - in exerciser program 52
- installing your analyzer 10
- interface
 - establishing 20
- interpretation
 - high level assignment 105
- interpretation assignment 105
- L**
- loops
 - in exerciser program 53, 81
- M**
- Manual Driver
 - installing 17
- manual Trigger
 - advanced mode 69
- manual trigger 6, 44
 - with sequencer 69
- mnemonics 199
- multiple analyzers 21
- N**
- new analysis
 - advanced mode 117
- new class
 - specifying 77
- new project
 - advanced mode 59
 - easy mode 38
- O**
- operating speed
 - for refresh 89
- options 5
- OTG
 - analysis connection 11
- OTG device
 - definition 149, 153
- OTG Exerciser 5
 - as DRD device-A 12
 - as DRD or peripheral only device-B 12
- OTG VBUS
 - response type 149
- outputs
 - external 66
- overview 3
- P**
- packet
 - defining 60
 - payload data 62
 - type 61
 - user defined 62
- packet types 61
- parameters
 - deleting in sequencer 69
- performance analysis 116
 - advanced mode 116
 - easy mode 113
- pointers
 - in waveform display 181
 - lost 181
- ports 7
- pre-defined analysis 117
- pre-defined data capture selections 48
- pre-defined trigger selections 46
- pre-trigger 111
- program lines
 - maximum in exerciser 71
- program loops
 - in exerciser program 80
- programming
 - exerciser 71
- project
 - file type definition 26
 - importing 145
- protocol
 - user defined decodes 106
- protocol error mask 70
- protocol errors 70
 - detected 108
 - exclude 45
 - post process 109
 - triggering on 108
- R**
- random data pattern 93
- reduced bit width 83
- refresh
 - advanced 86
 - auto 89
 - exerciser program 88, 89
 - manual 89
- reports 98
- results display
 - viewing preferences 178

Index

run the project 97

S

saving

performance analysis result 123

SBAE

versions 1

SBAE-30 1

included components 9

SBAE-30 external I/O Connector

pin assignment 15

scan descriptors

viewing 201

search

access from menu bar 195

data report 103

for bookmark 195

for data pattern 193

for protocol errors 192

using mnemonic 193

search data report

for data pattern 103

select

components for installation 16

self test 200

separate systems 10

sequencer

description 64

operation overview 64

programming 65

search data report

for data block 103

shortcuts

exerciser program 84

simulation mode 23

software installation 16

start exerciser

advanced mode 66

easy mode 53

statistical report 98

status LED 6

suspend current

measurement 172

T

test mode

high speed 55

high speed usage 55

timing analysis

easy mode 127, 129

Timing Analyzer 5

timing analyzer

errors 130

timing details

viewing 186

timing display

cursor positioning 186

viewing options 186

tool bar

simplifying 177

transaction retry 55

transfers

control 73

non control 75

trigger

external enable 44

manual 44

on data packet 41

on data pattern 41

on protocol error 45

over multiple transactions 41

trigger mask 55

U

unconfigured current
measurement 158

Undo/Redo

in exerciser program 82

unpacking 9

USB Driver

manual install 17

update 17

User Defined

packet 62

using the cursors 188

V

VBus droop 165

VBus measurement 163

view as report 98

view scan descriptors 201

viewing preferences

results display 178

W

walking bit pattern 94

waveform

display 181

workspace optimizing 69

Z

Zoom in exerciser program 80