

ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

Custom engineering
so your equipment works exactly as you specify.

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

Expert team | Trust guarantee | 100% satisfaction

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

Visit our website - *Click HERE*

SBAE-30/CONQUEST

SBAE SERIES

USB 2.0/1.x

Analyzer/Exerciser/DC Verification



User's Manual

CATALYST ENTERPRISES INC.
WWW.getcatalyst.com
PCI, PCI-X, PCI Express
SAS, SATA, Fibre Channel, UWB

©2005-2006
073106 P/N 400-0104-001

Copyrights

"SBAE-30 and Conquest are trademarks of Catalyst Enterprises, Inc. Universal Serial Bus is a registered trademark of USB-IF. On-The-Go is a registered trademark of USB-IF. Microsoft, Microsoft Windows 2000, and Windows XP are either registered trademarks or trademarks of Microsoft Corporation. All other trademarks and registered trademarks are property of their respective owners.
©2004 Catalyst Enterprises, Inc. All rights reserved."

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 CATALYST FOR A COPY.

Copyright © 2003-2006, Catalyst Enterprises, Inc.

All rights reserved.

---

www.getcatalyst.com

Catalyst Enterprises, Inc.
6680 Via Del Oro
San Jose, CA. 95119
(408) 365-3846
support@getcatalyst.com

Supported Features and Options

This manual describes the use and operation of the SBAE series products that includes all of the available 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 info@getcatalyst.com.

Table of Contents

Supported Features and Options - iii

SBAE Versions - 1

SBAE-30 Versions - 1

CONQUEST vs SBAE-30 - 2

Common Software - 2

Introduction - 5

What's In this Manual - 5

Overview - 5

Analyzing USB Designs With the Basic Analyzer - 6

SBAE-30 Options - 7

Host Exerciser - 7

Device Emulation - 7

OTG Exerciser - 7

DC Compliance measurement - 7

Timing Analyzer - 7

SBAE-30 Interface - 8

Status LED Function Description - 8

Manual Trigger - 8

External Signals (Ext. Out, Ext. Trig In, and Ext. Clk) - 8

Ports - 8

Power In - 9

Analyzer/Exerciser Connection Identification - 9

Receiving Your Analyzer - 10

SBAE-30 - 10

Conquest - 10

Unpacking Your Analyzer - 10

Installing Your Analyzer - 11

Hardware Setup - 11

Exerciser Configuration Connections - 12

Analyzer/Exerciser in Hub testing Connections - 12
OTG Analysis Connections - 13
OTG Exerciser as DRD A-Device Connections - 13
OTG Exerciser as DRD or Peripheral Only B-Device Connections - - - - - - - - - - - 14
Device Emulation Connections - 14
Conquest Analyzer Connections - 15
Conquest Exerciser Connections - 15
SBAE-30 External I/O Connector Pin Assignment - - - - - - - - - - - - - - - - - 16
Conquest External I/O Connector Pin Assignment - 16
Software Installation - 17
Manual USB Driver Installation - 17
Updating the SBAE-30 USB Driver Manually - 18
Launching Your USB Analyzer - 19
Connecting via Ethernet - 20
Connecting to a Network - 20
Connecting via Hub, Switch or Similar device - 21
LPT Port Setting - 22
Operating in Simulation Mode - 22
Authorization - 23
Example Projects - 24
Run an example project - 26
Default Capture - 27
Creating Projects - 28
Capture and Timing Analysis Projects - 28
Performance Analysis Projects - 29
Device Emulation & OTG Advanced Exerciser - 29
Current & Voltage Measurements - 29
Protocol Analysis - 31
Easy Mode (Pre-Defined Setups) - 31
Easy Data Capture - 32
Trigger on Data- 34
Data Capture Options - 40
Custom Transaction - 41

Exercise and Capture (Optional) - 42
Programming the Exerciser in Easy Mode - 43
Settings - 46
Auxiliary Port- 48
Advanced Mode (User Defined) - 50
Capture Data Project - 50
Defining Packets - 50
Packet Payload Data - 52
The Sequencer - 54
Sequencer Operation Overview - 54
Programming the Sequencer - 55
Protocol Errors - 58
Setting a Protocol Error Mask - 59
Host Exerciser (Optional) - 60
Programming the Exerciser In Advanced Mode - - - - - - - - - - - - - - - - - 60
Programming with Transfers - 61
Specifying a New Class - 65
Specifying Data for Transfer - 66
Reduced Bit Width - 70
Exerciser Programming Shortcuts - 71
Creating an Exerciser Program by Importing - - - - - - - - - - - - - - - - - - 71
Advanced Refresh - 72
Creating an Exerciser Program by Copying and Pasting - - - - - - - - - - - - 73
Creating a Data Block - 75
Naming a Data Block- 76
Creating and Editing Data Blocks as Text - 80
Settings - 81
Auxiliary Port- 81
Running the Project - 82
Set The Data Capture Options - 82
Reports - 83
View Statistical Report - 83
View Data Report - 84

View Data Report Statistics - 86
Display Histogram - 87
Search a Data Report for a Pattern - 88
High Level Interpretation Assignment - 90
User Defined Decodes - 91
Protocol Errors - 93
Protocol Errors Detected (While capturing data) - 93
Post Process Protocol Errors Detected - 93
Pre-Trigger - 95
Performance Analysis- 97
Performance Analysis (Easy Mode) - 97
Performance Analysis (Advanced Mode) - 100
Real Time Analysis - 100
Perform a Pre-defined Analysis - 101
Create a New Analysis - 101
Define Packets - 102
Program The Exerciser- 102
Creating Analysis Expressions - 102
Saved Performance Analysis Review -108
Timing Analysis (Optional)- 111
Easy Mode Timing Analysis - 111
Advanced Mode Timing Analysis - 113
Device Emulation (Optional)- 115
Device Emulation (Easy Mode) - 116
Programming the Device - 116
Endpoint errors generated by OTG DRD & Device Emulation - - - - - - - - - - - 121
Device Emulation (Advanced Mode) - 122
Programming the Device - 122
OTG Exerciser (Optional) - 127
OTG Exerciser as an A-Device - 127
Programming the Device -128

Create Exerciser Program ----------129
Define OTG Device Configuration ----------130
Starting a Session ----------130
Dropping a Session ---------- 131
OTG Exerciser as a Peripheral B-Device ----------131
Define OTG Device Configuration ----------132
Create Exerciser Program ----------133
Requesting a Session ----------133
Terminating the Exerciser Program ----------134
Current Measurement (Optional) ---------- 135
Unconfigured Current Measurement ---------- 136
Operating Current Measurement ---------- 138
Making the VBus Measurement ---------- 141
VBus Droop Measurement ---------- 143
Inrush Current Measurement ----------145
Inrush Current Display Features ---------- 148
Suspend Current Measurement ---------- 150
Current Measurement Calibration Board ---------- 152
Display Manipulation---------- 155
Results Display Viewing Preferences ----------156
Compact View ---------- 157
Display Idle Time ---------- 158
Waveform Display ---------- 159
Filter ----------160
Smart on Screen Filtering ---------- 161
Save Display Settings ----------162
Timing Analysis Display---------- 163
Set Timing Display Viewing Options ---------- 164
View Timing Details ----------164
Using the Cursors and Bookmarks ----------165
Search---------- 167

Display Configuration - 172

Mnemonics - 175

Utilities - 176

Self Test - 176

View Scan Descriptors- 177

Attach/Detach Device - 178

Capture Screen - 179

Appendix A - 181

Advanced Script Language (ASL) - 181

Document conventions - 181

Language Elements -182

ASL Script Structure -184

Protocol Extraction Section -186

Protocol Decoding Section -191

DefineOptions Block - 192

ValidRanges Block - 193

Main Block - 194

USB Descriptor Section -200

Descriptor Block - 201

Functions -202

Samples 205

Script Editor 212

Creating the Script -213

Index - 217

SBAE Versions

SBAE-30 Versions

A new SBAE series version product, the **SBAE-30B** supersedes the **SBAE-30A** and the SBAE-30.

The **SBAE-30A** is identical to the SBAE-30 except for the following:

- Timing Analyzer supports High Speed.
- Higher resolution VBus current and voltage measurement.
- External Clock range of 400KHz - 8MHz for High Speed and 1KHz-100KHz, 400KHz - 8MHz for Full/Low speed.

The **SBAE-30A** is identified by serial numbers that are greater than **SBAE-400**

The **SBAE-30B** is identical to the **SBAE-30A** with the following changes:

- The interface has been changed from a Full Speed USB and Parallel Port to High Speed USB and Ethernet.
- The Host Exerciser memory has been increased from 128KB to 256KB.

The **SBAE-30B** is identified by serial numbers that are greater than **SBAE-600.**

CONQUEST vs SBAE-30



Figure 1 Conquest

Conquest is a product that supports a subset of features of the SBAE-30. The following are not supported on Conquest:

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

Common Software

The SBAE software supports both, the SBAE-30 and the Conquest with limited capabilities with features not available on the Conquest grayed out.

The following table summarizes the supported features of the SBAE-30 and Conquest

Table 1 SBAE-30 vs Conquest Supported Feature Summary

Feature	SBAE-30	Conquest
Advanced - Device Emulation Typically there is no difference between SB30A and SB30B	√	Not Supported
Easy - Device Emulation Typically there is no difference between SB30A and SB30B	√	Not Supported
OTG Exr. Typically there is no difference between SB30A and SB30B	√	Not Supported
Advanced - Performance Analyzer Typically there is no difference between SB30A and SB30B	√	Not Supported
Easy - Performance Analyzer Typically there is no difference between SB30A and SB30B	√	Not Supported
DCC- Inrush current Sampling resolution of and old SB30 PHY boards is 8-bit and in new SB30 PHY boards is 12-bit.	√	Not Supported
DCC- Suspend current	√	Not Supported
DCC- Unconfigured current Sampling resolution of old SB30 PHY boards is 8-bit and in new SB30 PHY boards is 12-bit.	√	Not Supported
DCC- Operating current Sampling resolution of old SB30 PHY boards is 8-bit and in new SB30 PHY boards is 12-bit.	√	Not Supported
DCC- VBUS Sampling resolution of old SB30 PHY boards is 8-bit and in new SB30 PHY boards is 12-bit.	√	Not Supported
DCC- VBUS Droop Sampling resolution of old SB30 PHY boards is 8-bit and in new SB30 PHY boards is 12-bit.	√	Not Supported
First Analyzer Features **Not Supported** on CONQUEST: 1. Trigger on VBUS. 2. Trigger on Data Pattern (Length, Pattern). 3. Trigger on operating current. 4. OTG (Trigger and Capture). 5. Trigger on Bus Error. 6. Trigger on Chirp 7. Auto speed detection. 8. Partial data payload capture. 9. Non Standard clock. 10. Capture custom transaction. 11. Test Mode. 12. Connect and disconnect events.	√	√

Table 1 SBAE-30 vs Conquest Supported Feature Summary

Feature	SBAE-30	Conquest
Second Analyzer Limitations: 1. OTG is not supported. 2. External clock is not supported. 3. Low speed TTL level is not supported. 4. Sampling Memory is limited to 128K 5. Protocol error memory is lower than sampling memory, so it may cause to overwrite Protocol Error memory during capture (it causes some invalid PEs on Packets).	√	Not Supported
Exerciser Features **Not Supported** on CONQUEST: 1. Transaction retry. 2. Auto generate SOF. 3. Host Exerciser loop is limited to 255. 4. High speed project should be started with Reset and SOF. 5. Connect/Disconnect. 6. Exerciser Memory of CONQUEST is two times larger than and SB30A. 7. Not SOF between retries. 8. No Retry for Split Transactions.	√	√
High Speed Timing Analyzer There is no difference between SB30A and SB30B.	√	Not Supported
Full Speed Timing Analyzer There is no difference between, SB30A and SB30B.	√	Not Supported

Introduction

What's In this Manual

This manual describes the installation and operation of your Catalyst Enterprises **SBAE-30 USB Analyzer/Exerciser** and **Conquest Analyzer**. Conquest users please note that functional descriptions for features common with the SBAE-30 appear under SBAE-30 headings. Examples of some typical applications are included.

Overview

The SBAE-30 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 SBAE-30 capabilities if all options are installed.

The SBAE-30 incorporates two modes of operation, **Easy** and **Advanced**. Each of the installed options except the OTG Exerciser may be operated in any mode combination to support your specific project need. As an example, you may 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 your USB data capture and triggering with a minimum of programming. In the **Easy Mode** you may quickly:

- Capture and Trigger of 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 complex data capture and triggering. In the **Advanced Mode** you may program the Analyzer/Exerciser to:

- Capture all bus traffic or specific packets and to 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** that can be triggered as defined in the Sequencer or, on a user specified bus signal pattern.

An extended set of capabilities allows you to:

- Program the SBAE-30 as a **Device Emulation** for testing your 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 Basic Analyzer

The SBAE-30 basic Analyzer allows you to perform comprehensive USB analysis by capturing data and triggering on desired 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.

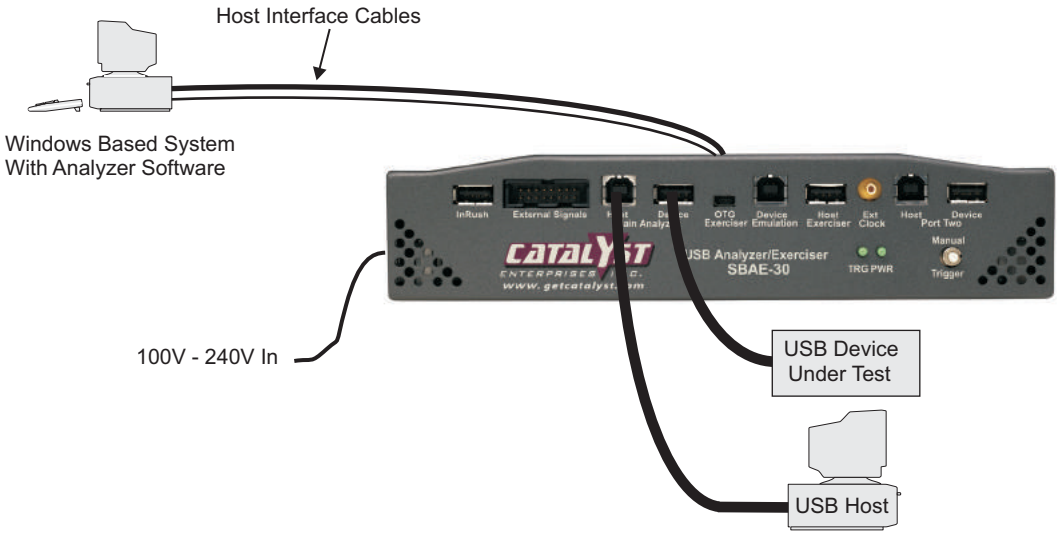


Figure 2 Typical USB Test Setup of Basic Analyzer

SBAE-30 Options

Depending on your needs, you may expand the 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 the SBAE-30 as a Device for testing your USB development system. You may define each Device to include up to 3 configurations with 2 interfaces each and assign up to 7 endpoints to these interfaces. The Device may be defined as a Low, Full, High or a High/Full speed Device.

OTG Exerciser

This option allows you to operate the SBAE-30 as an OTG Dual-Role Device. In this mode the SBAE-30 is a combination of Device Emulation and Host Exerciser. A DRD is emulated such that it 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 SBAE-30 Timing Analyzer samples bus signals at different sampling rates depending on the bus speed selected.

SBAE-30 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)

Your SBAE-30 Analyzer can output a unique 8 bit, user defined CMOS level when triggered. These outputs may be used for activating external Devices such as other types of test equipment when certain events are detected in the USB data capture.

Additionally your SBAE-30 Analyzer provides you with an External Trigger Input that allows you to trigger the Analyzer from externally generated sources. The trigger input may 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 47.

The SBAE-30 offers a number of internally generated pre-defined low frequency clock settings, however you may 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.

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.
Port Two	Auxiliary port for analysis only, no triggering. Capable of analysis with pre-defined data capture only. Most often used for Hub testing**.**
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 the SBAE-30 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.

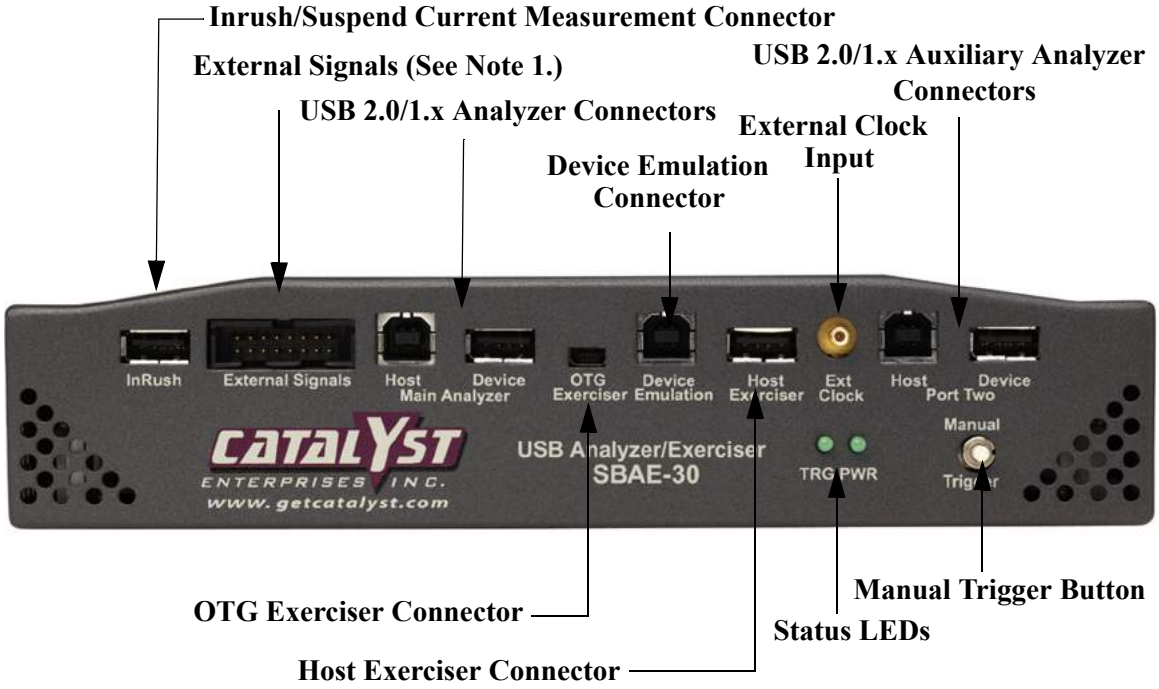
USB Host Interface Interface connection, located on the back of the unit, for connection between the Analyzer and the system hosting the GUI software.

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

Power In

The SBAE-30 operates on 100V - 240V 50/60Hz power that is connected on the back of the SBAE-30.

Analyzer/Exerciser Connection Identification



Note 1. For External I/O connector pin assignment see **page 16**.

Receiving Your Analyzer

SBAE-30

Your SBAE-30 Analyzer includes the following components:

Carrying Case
SBAE-30 identified in the packing list
Software on the CD Rom
DB-25 -> SCSI II 26, Parallel Port connection. (SBAE-30 only)
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 (SBAE-30B only)
External I/O Ribbon and External Clock SMB to BNC Cables
Current Measurements Calibration Board
Power cable
User's Manual

Conquest

Your 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's Manual

Unpacking Your 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 Catalyst Enterprises. Retain all shipping materials for shippers inspection.

Installing Your Analyzer

Hardware Setup

Your SBAE-30 Analyzer/Exerciser may be set up in any one of the following configurations:

1. As an Analyzer to analyze USB Host and Device transactions.
2. As an Exerciser to exercise a USB Device.
3. As an Analyzer/Exerciser for OTG Device testing.
4. As an Analyzer/Exerciser in Hub testing.
5. To measure Inrush current, Configured current, Unconfigured current, Suspended current and VBus measurement
6. 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 3 (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.

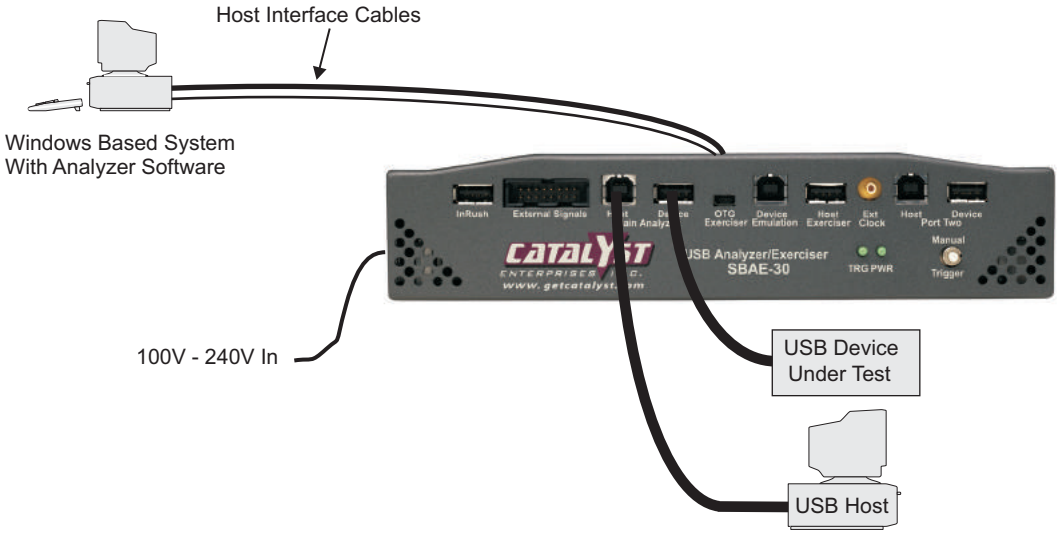


Figure 3 SBAE-30 Configured as an Analyzer

1. Connect your SBAE-30 to a Parallel, USB or Ethernet Port to system running analysis software with the appropriate cable as applicable.
2. Connect the USB Device to the Main Analyzer Port on the SBAE-30.
3. Connect the USB Host to the Main Analyzer Port on the SBAE-30, using a USB cable.

Exerciser Configuration Connections

Figure 4 shows the SBAE-30 used to exercise a USB Device. In this configuration the the SBAE-30 acts as the USB Host.

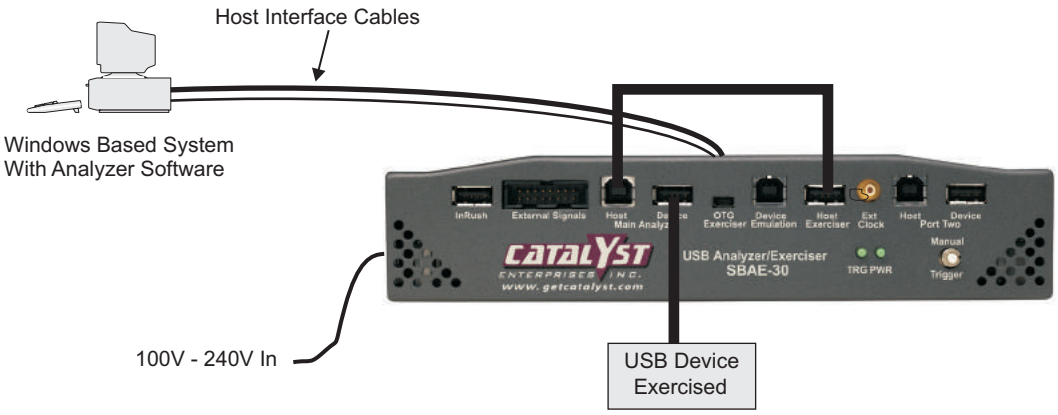


Figure 4 SBAE-30 Configured as an Exerciser

1. Connect your SBAE-30 to a Parallel, USB or Ethernet Port to system running analysis software with the appropriate cable as applicable.
2. Connect the USB Device to the Main Analyzer Port on the SBAE-30.
3. Connect the Main Analyzer Port to the Host Exerciser Port using a USB cable.

Analyzer/Exerciser in Hub testing Connections

Figure 5 shows the SBAE-30 used to test a Hub.

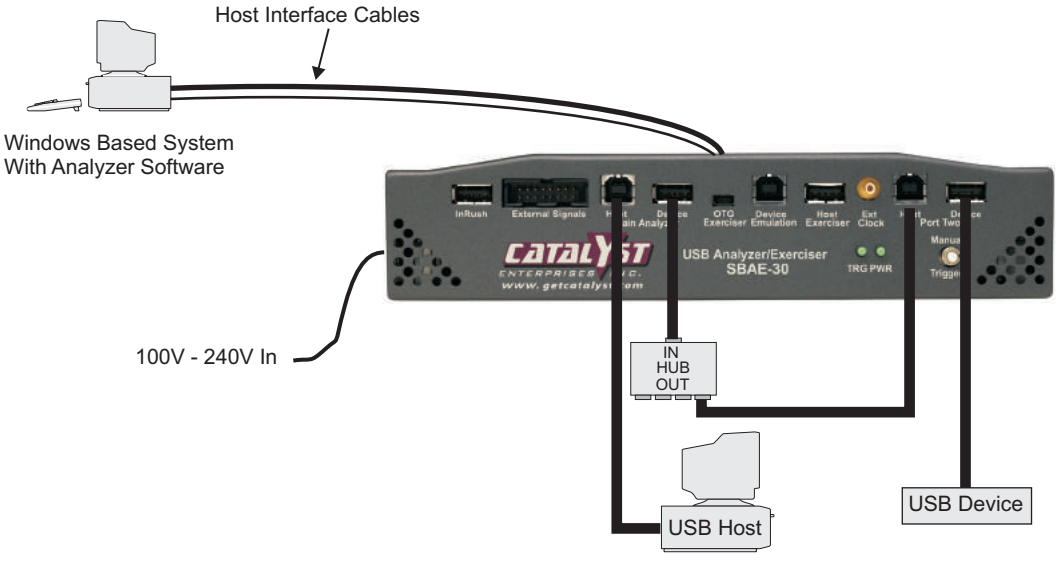


Figure 5 SBAE-30 Connected to Test a Hub

OTG Analysis Connections

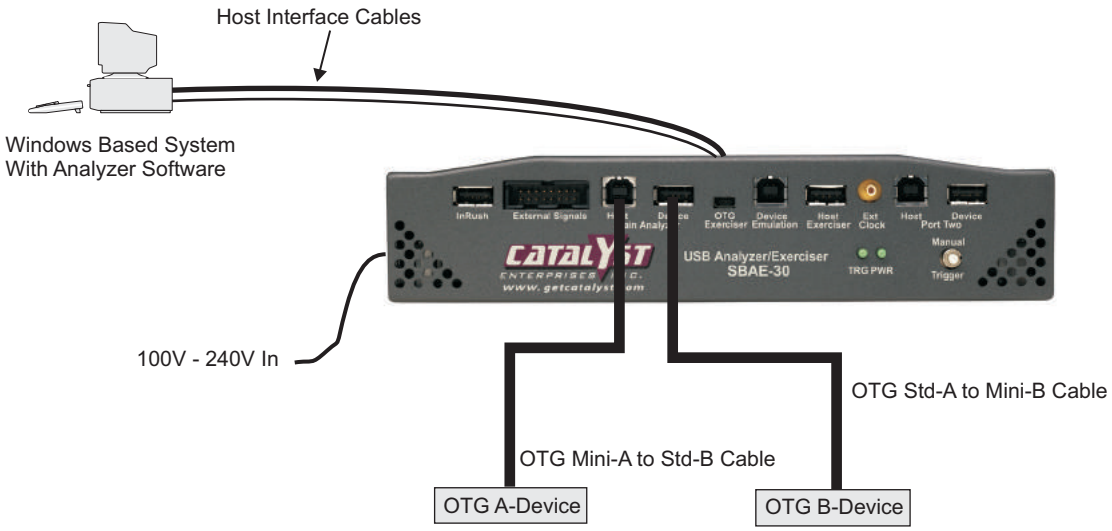


Figure 6 SBAE-30 Connections for OTG Analysis

OTG Exerciser as DRD A-Device Connections

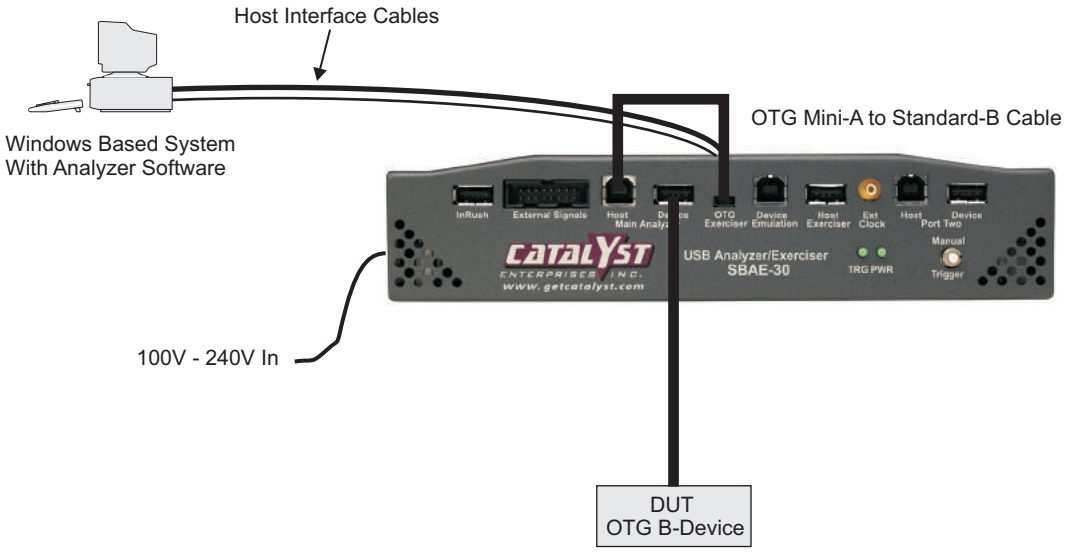


Figure 7 OTG Exerciser as DRD A-Device

OTG Exerciser as DRD or Peripheral Only B-Device Connections

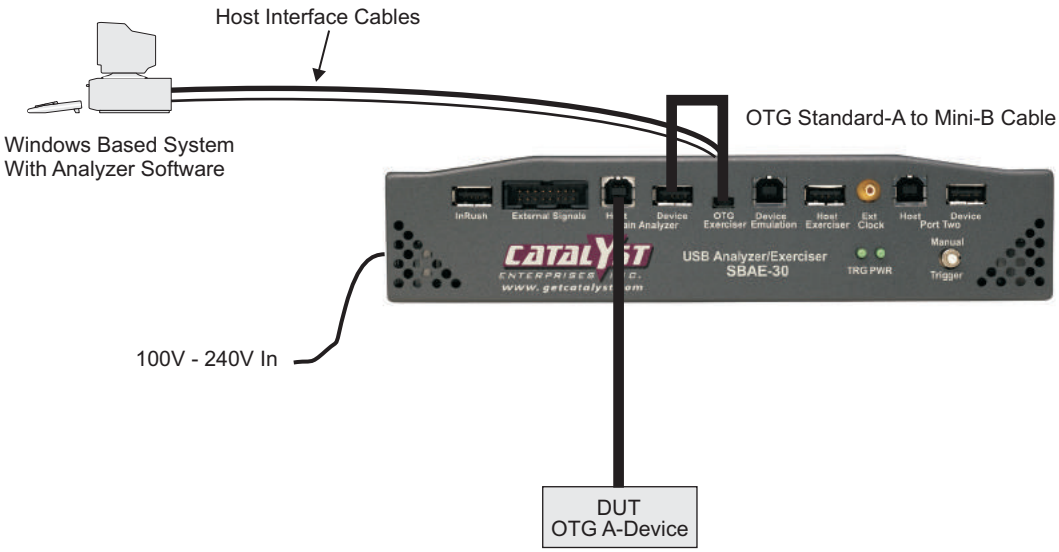


Figure 8 OTG Exerciser as DRD or Peripheral Only B-Device

Device Emulation Connections

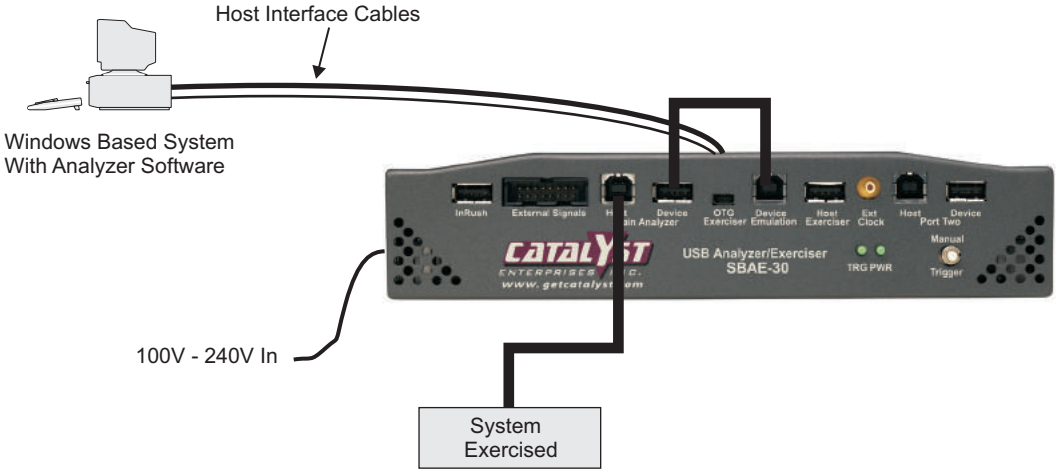


Figure 9 SBAE-30 Connections for Device Emulation

Conquest Analyzer Connections

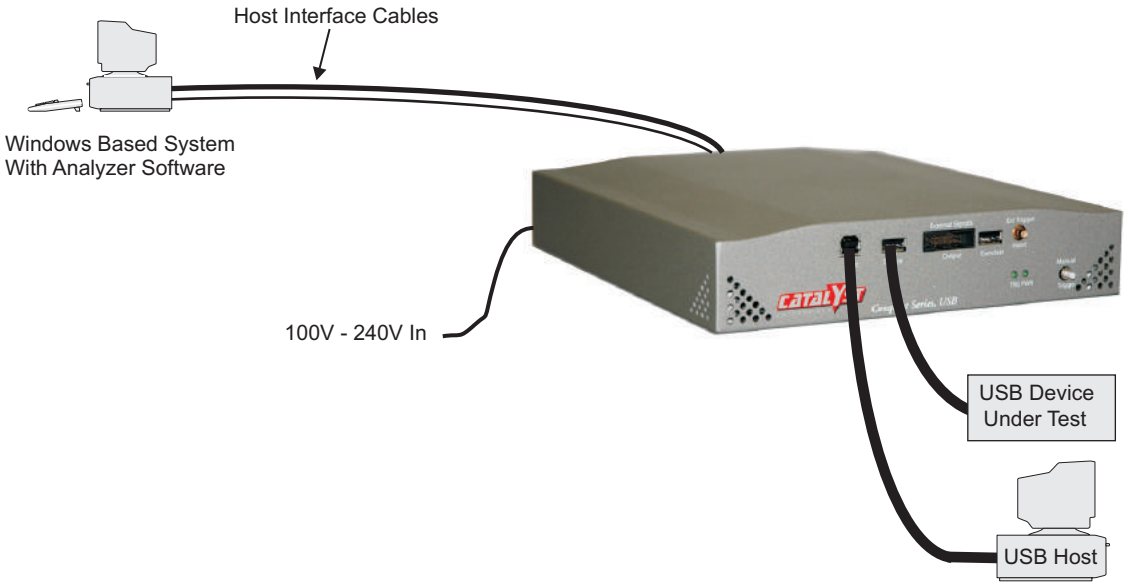


Figure 10 Conquest Connected as an Analyzer

Conquest Exerciser Connections

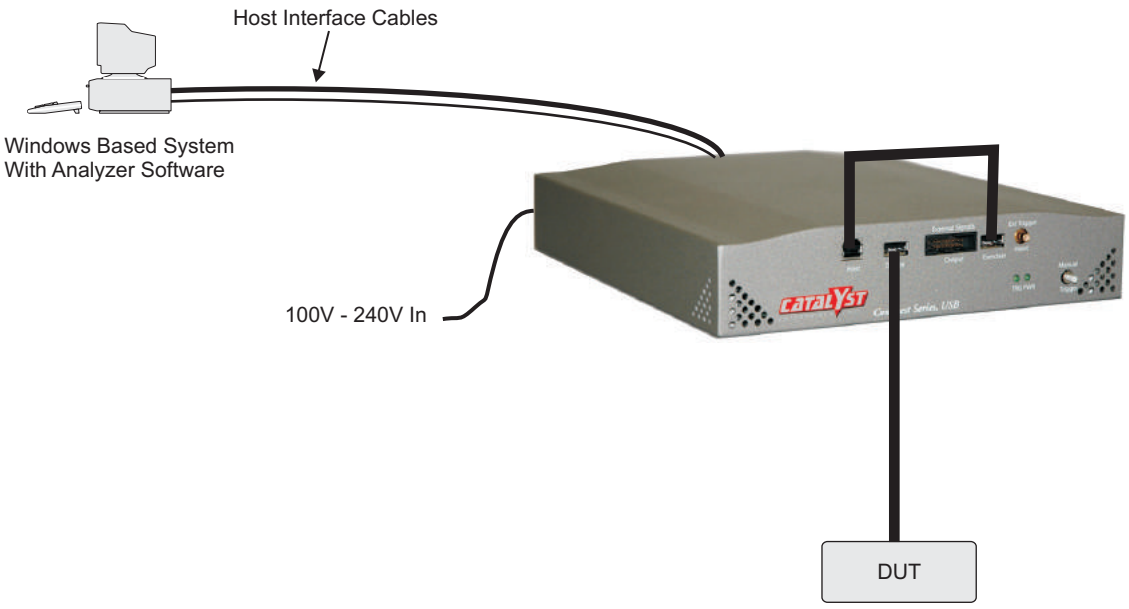


Figure 11 Conquest Connected as an Exerciser

SBAE-30 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 2 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 3 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).

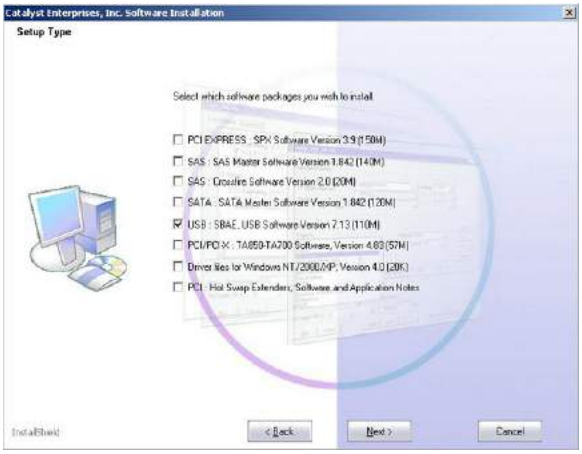
Note 2. The signal “External out 0” (LSB) is not externally available.

Software Installation

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

Do not connect Analyzer Do not connect the Analyzer to your host system until the software installation is complete.

1. Insert the CD Rom in the CD Rom drive.
2. The installation will automatically start the setup, unless the Auto Run is turned off. In this case select the CD Rom from "My Computer" and click on setup.
3. After the warning to close all other programs and before starting the installation, the Install component selection will open as shown below**.**



4. Select the desired components for installation.
5. Click **Next** to complete the installation.

System restart You must restart your computer before you can use your Analyzer software.

Error Message If you get an error message during installation of the drivers for 2000 or XP consult your system administrator. Your system may be setup to only allow an administrator level to copy such driver files.

Manual USB Driver Installation

To manually install the USB driver perform the following:

1. Power Up the SBAE-30
2. Connect the USB cable between the SBAE-30 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\catalyst\SBAE\usb driver\win2k” folder.
    - Choose Ibusb.inf
    - Click OK
6. Click Next and then Finish to complete the SBAE-30 USB port driver installation.

Updating the SBAE-30 USB Driver Manually

1. Click Start: Settings: Control Panel: System and click the Device Manager tab.
2. Find the **Catalyst Test Tools** entry.
3. Select and right click the **Catalyst SBAE-30** 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\catalyst\SBAE\usb driver\win2k” folder.
    - Choose Ibusb.inf
    - Click OK
8. Click Next and then Finish to complete the USB driver update.

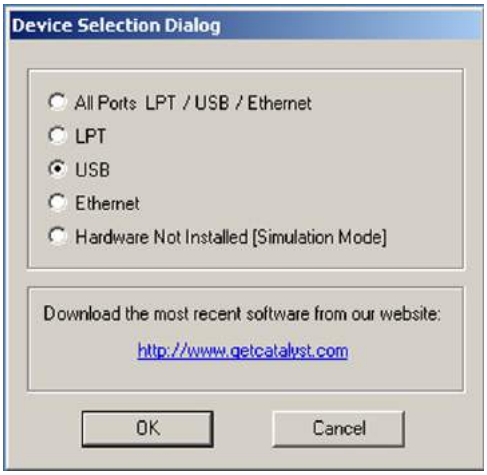
Launching Your USB Analyzer



To launch the USB Analyzer software, double click the SBAE button in the Program Manager Window

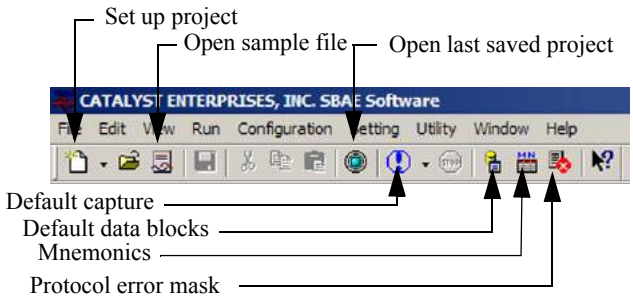
Each time you run your software it will search for a default host interface and, if it is found, the software will launch. If no interface is found the software will launch in simulation mode.

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



Click **OK** and the Analyzer will launch and display the Analyzer Tool bar as shown below.

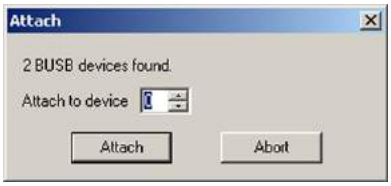
Note: The very first time that you run your software, the USB Class Decodes will be loaded. This process takes approximately 1 minute.



In the event that the SBAE-30 is not detected, the software will display the **Hardware Not Detected** message.



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



Connecting via Ethernet

You may use the ethernet connection using any one of the following 3 supported configurations:

1. The SBAE-30B connected to a network via a hub, switch, or similar device.
2. The SBAE-30B connected to the host computer (machine running the application software), via a hub, switch or similar device.
3. The SBAE-30B connected directly to the host computer using a crossover cable.

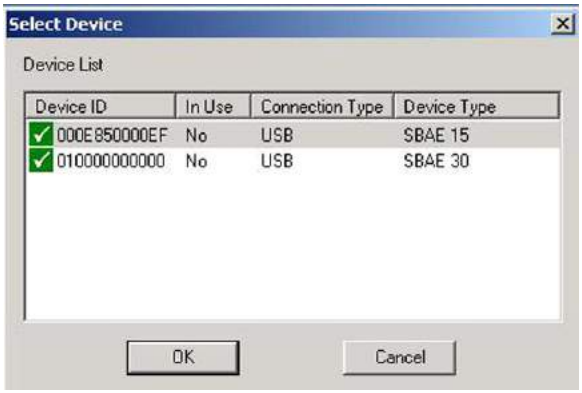
Connecting to a Network

When connected to a network the SBAE-30B must communicate with the DHCP server to establish a connection. The DHCP server will continually send the next available IP address to the SBAE-30B until the SBAE-30B software is started.

When the user starts the software, the user may be prompted if they wish the software to automatically use the offered IP address or if they wish to assign a specific IP address (the assigned IP address needs to be on the same network segment as the host computer). The menu also allows the user to save the selected option (automatic or specific address). If the assigned IP address is not available, the OS will notify the user of an IP address conflict.



After the user clicks 'OK' the software will search for all SBAE-30B units connected to the network, and will display a list of available SBAE-30B units. After the user selects the desired SBAE-30B unit, the software will assign the IP address to the selected unit, completing the connection and will launch 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 Catalyst board must communicate with the host computer to establish a connection. The host computer will continually broadcast the next available IP address to the Catalyst Board until the Catalyst software is started.

When the software starts, the user may be prompted if they wish the software to automatically use the offered IP address or if they wish to assign a specific IP address (the assigned IP address needs to be on the same network segment as the host computer). The menu also allows the user to save the selected option (automatic or specific address). If the assigned IP address is not available, the OS will notify the user of an IP address conflict.

After the user clicks 'OK' the software will search for all Catalyst boards connected to the network, and will display a list of available Catalyst boards, after the user selects the desired Catalyst board, the software will assign the IP address to the selected board, completing the connection and will launch the software.

LPT Port Setting

Your system Parallel Port may not necessarily be set to a Bi-directional mode even if your system supports this mode.

Port Not Configured If you continue to get a **Hardware Not Detected** message, even if you have set the required LPT port, please check your system setting for your Parallel Port.

Set The Parallel Port Mode

1. Restart your computer and during boot, press F1 key to enter Setup (some systems may use a different key than F1, please consult your system manual).
2. Once in setup select the Advanced, then Peripheral Configuration and then Parallel Port Mode.
3. Select the Parallel Port to be configured. If your system supports Bi-directional mode you must have choices for at least one of the following:

   Bi-directional, ECP or EPP, preferably Bi-directional first then ECP and last EPP.

   Note that Standard or Compatible modes are not a Bi-directional mode.
4. When finished with the configuration press **ESC** to exit setup and save the new configuration.

Operating in Simulation Mode

Your system will operate in the Simulation Mode as default if the hardware is not detected, however, you may operate in Simulation Mode directly without installing the Analyzer hardware. To operate without hardware, select **Hardware Not Installed (Simulation Mode)** in the LPT Port Setting dialog box and click **OK.**

The Analyzer software will launch and display the tool bar, but with the limitation that the Analyzer will operate 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** to open the Authorization Code dialog.

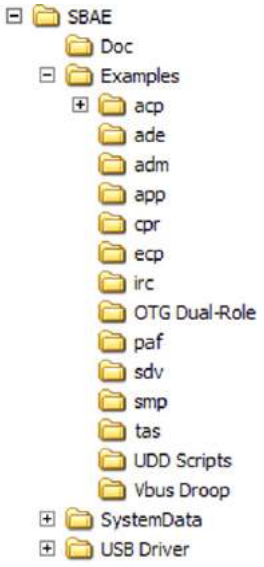


Enter the authorization code for the product(s) purchased and click **OK**.

Example Projects

Your SBAE-30 includes a number of pre-defined example projects that you may use to perform an immediate analysis without any setup.

The SBAE-30 system software comes with a pre-defined folder (Directory) structure for storing all files. All pre-defined example files are stored in the Examples Folder.



It is strongly recommended that you open some of these files to get an introduction to the types of projects that can be created for the SBAE-30.

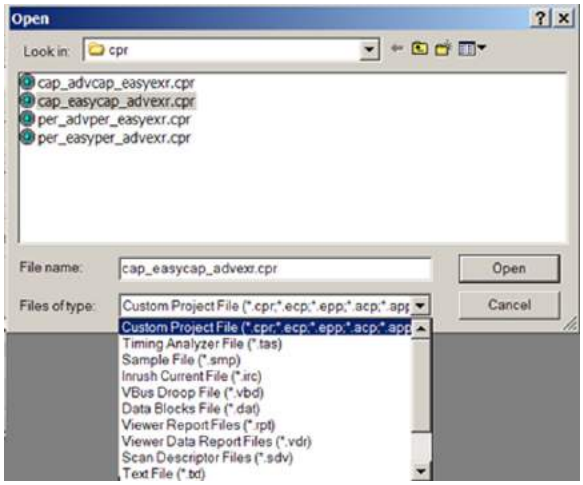
Project file type definition

*.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

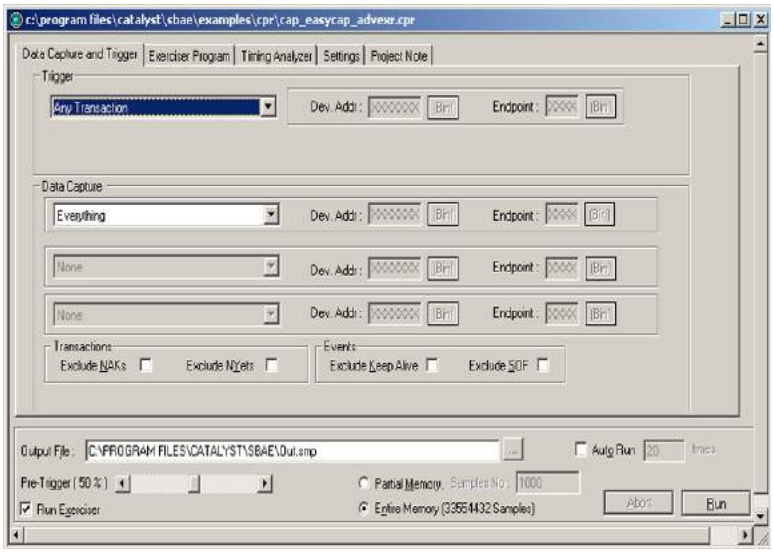
*.irc	Previously performed inrush current measurement file
*.paf	Easy performance analysis project file
*.rpt	Viewer report file
*.sdv	Scan descriptor file
*.smp	Previously performed project results file
*.tas	Previously performed timing analysis file
*.txt	Text file
*.vbd	VBus Droop file
*.vdr	Viewer data report file

Run an example project

1. Click **File** on the main toolbar and then choose **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 pre-defined example.

Default Capture

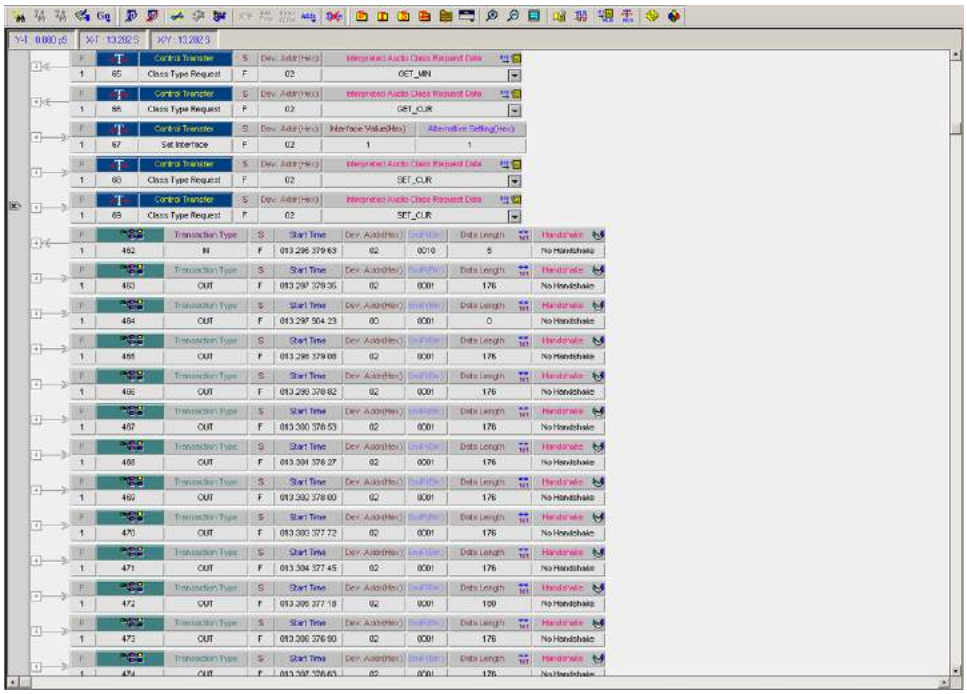
The default capture offers you one button activation of a default project.

Your software includes a pre-defined default capture, however, you may elect to 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 a display of current bus activity will display.



Creating Projects

Your 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 31 for setup of an **Easy Mode Capture**.

See page 50 for setup of an **Advanced Mode Capture**.

See page 42 for setup of an **Easy Mode Exerciser**.

See page 60 for setup of an **Advanced Mode Exerciser**.

See page 111 for setup of **Easy and Advanced Mode Timing Analysis**.

Performance Analysis Projects

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

See page 97 for setup of an Easy Mode Performance Analysis.

See page 100 for setup of an Advanced Mode Performance Analysis.

Device Emulation & OTG Advanced Exerciser

Check the Emulation to be used to open the corresponding project setup dialog.

See page 116 for setup of an Easy Mode Device Emulation.

See page 122 for setup of an Advanced Mode Device Emulation.

See page 127 for setup of the OTG Advanced Exerciser.

Current & Voltage Measurements

Check the desired measurement type to start the measurement.

See page 145 to perform an Inrush Current measurement.

See page 136 to perform an Unconfigured Current measurement.

See page 150 to perform an Suspend Current measurement.

See page 138 to perform an Operating Current measurement.

See page 141 to perform a VBus measurement.

See page 143 to perform a VBus droop measurement.

Protocol Analysis

Easy Mode (Pre-Defined Setups)

This mode allows you to operate the SBAE-30 with a minimum of setup. You may perform a Trigger and Data capture only, or program an Exerciser to generate bus traffic for triggering and data capture. Figure 12 shows a functional Capture and Trigger setup.

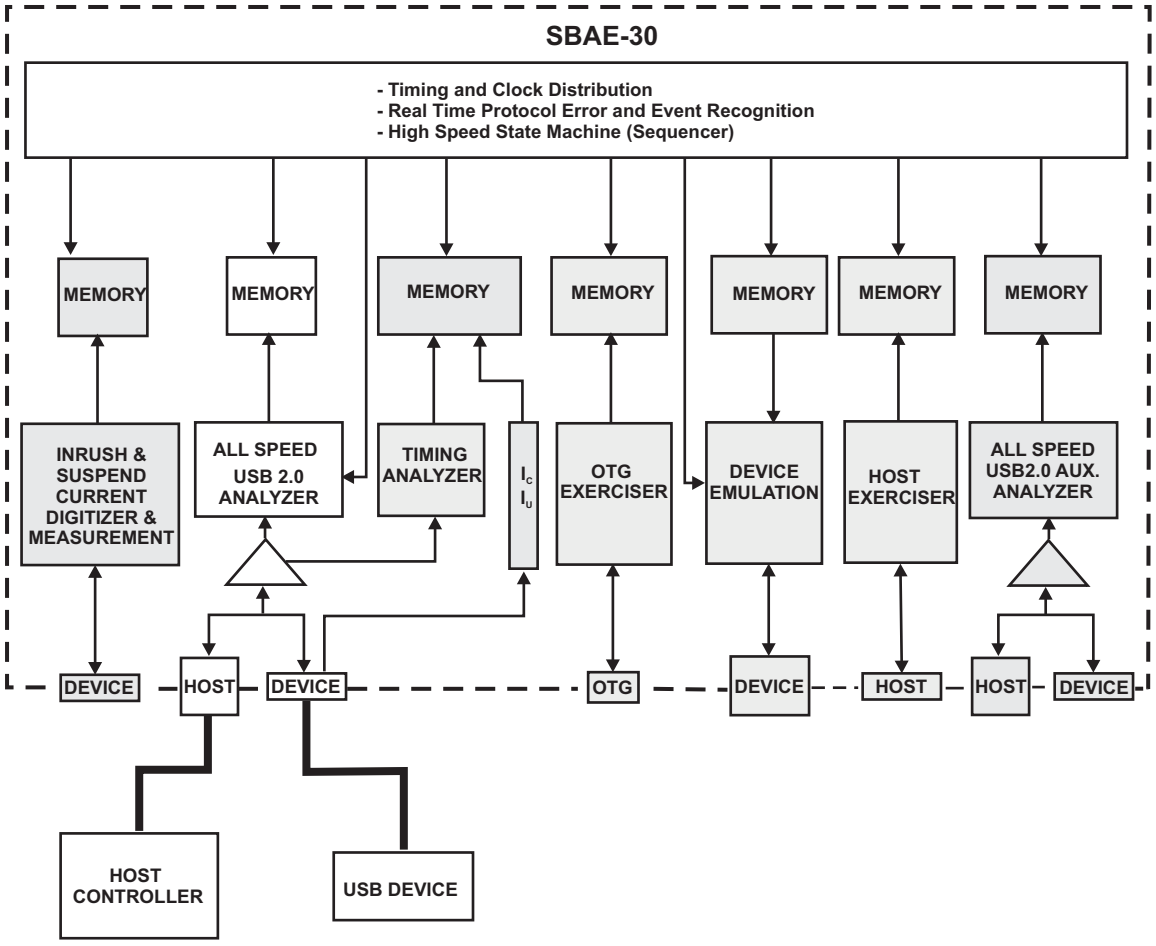


Figure 12 Capture and Trigger Functional Connection

Easy Data Capture



Make sure that the configuration is set to use Easy Analyzer. See “Creating Projects” on page 28.



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

New Project

To start a **New** project, click **File**, **New** and choose Analyzer Host Exerciser Project.

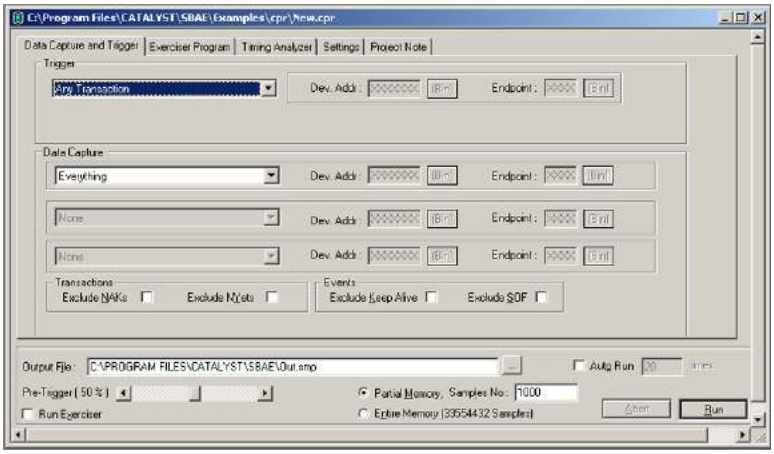


Figure 13 Trigger and Capture Data Dialog Box

1. Click the drop-down arrow next to the Trigger drop-down list box and choose a trigger. Table 4 on page 39 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.
2. If known, you may 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**.**

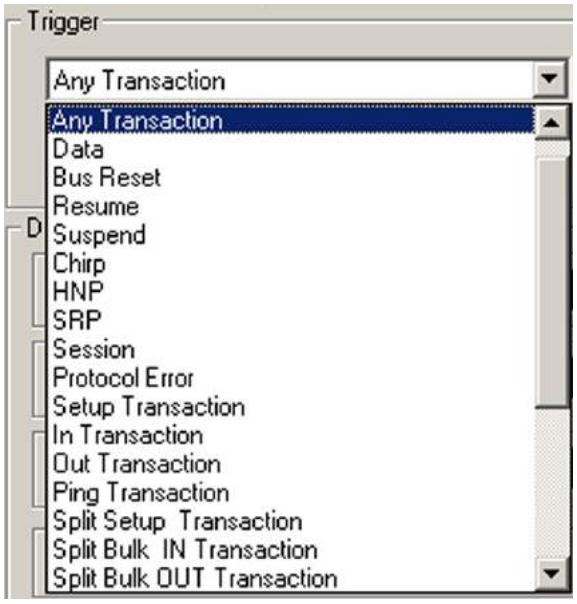


Figure 14 Trigger Choices in High Speed Mode

3. Choose up to 3 Data Capture selections from the 3 available Data Capture drop-down list boxes. Table 5 on page 40 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.

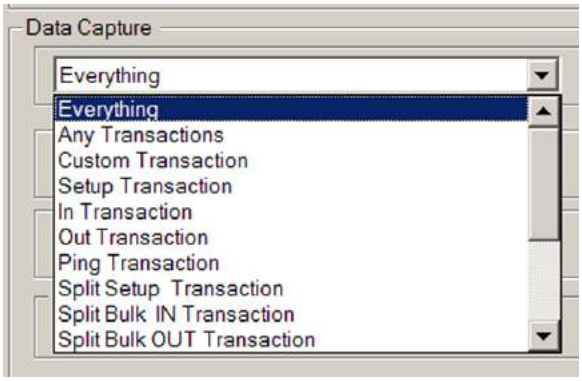


Figure 15 Data Capture Choices in High Speed Mode

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.

4. Click **Run** to capture and trigger immediately and wait for the results to be displayed as shown in Figure 16.

For ways to enhance your results analysis in the results display, see “Display Manipulation” on page 155.

Data Capture Options The capture and triggering is performed with default Data Capture Options. For explanation on setting the Data Capture Options see page 40.

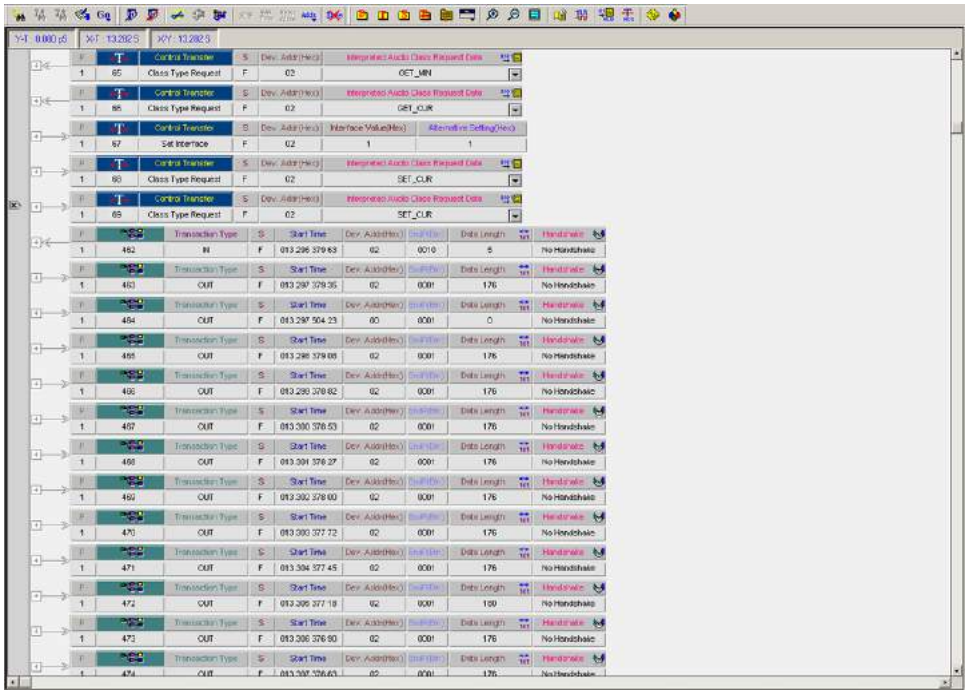


Figure 16 Captured Results Display

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

Trigger on a Data Packet This is the default setting when trigger on Data is selected.



Figure 17 Trigger on a Data Packet Dialog

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

Trigger on a Data Pattern To trigger on a data pattern check the Pattern button. A pattern may be spread across up to 8 similar transaction types by checking the **Multiple Transaction** check box.

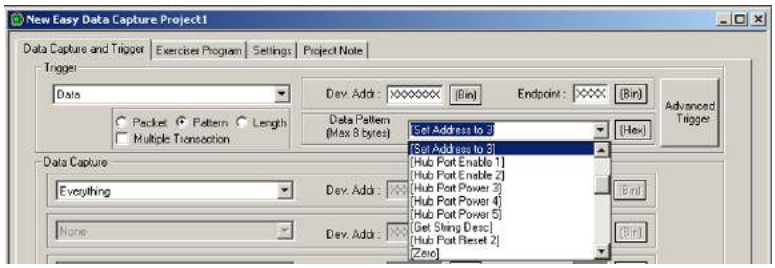


Figure 18 Trigger on Data Pattern Dialog

Click the down arrow next to the **Data Pattern** list box and select a pre-defined pattern, or enter a pattern directly in the list box (8 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.

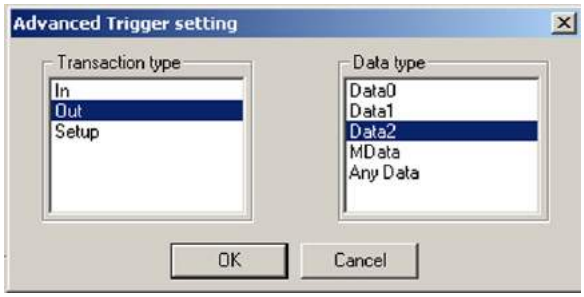
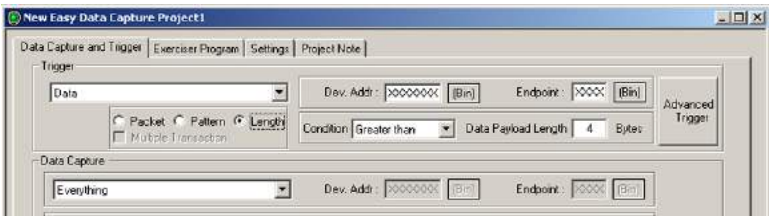


Figure 19 Advanced Trigger Setting Dialog

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 and make selection and enter a Payload Length value.



Sample Capture Result for trigger on data pattern

Figure 20 shows a capture result with a trigger on a data pattern identified.



Figure 20 Trigger on Data Pattern

To view details of the trigger click the **Data Pattern Triggered On** button to open the Triggering Transactions dialog as shown in Figure 21.

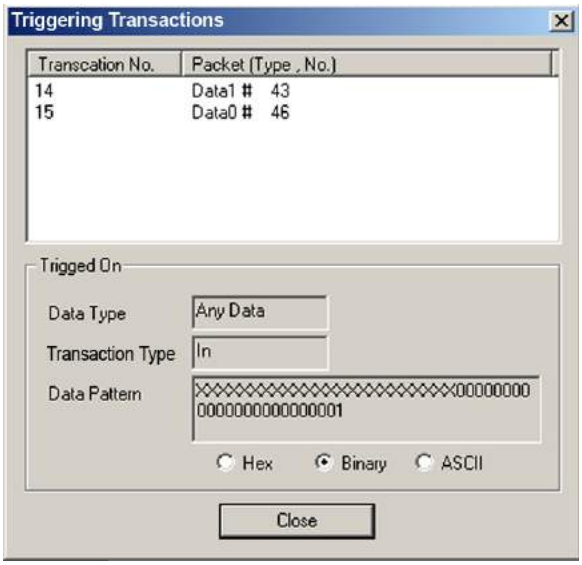


Figure 21 Triggering Transactions

Manual Trigger

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

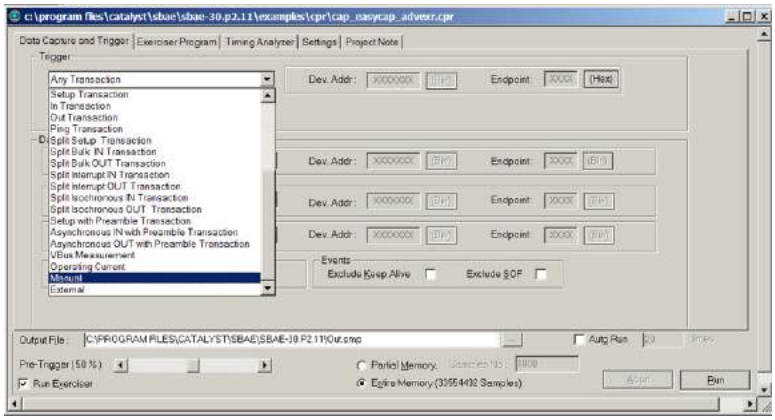
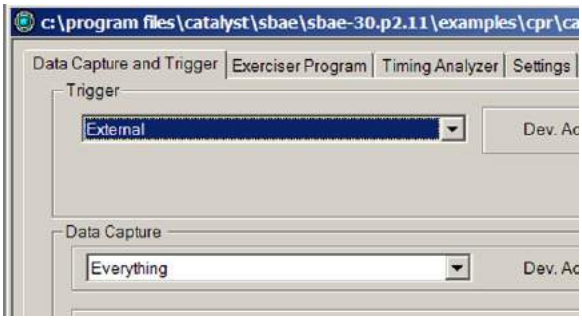


Figure 22 Enabling Manual Trigger

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.



To trigger on PE

Select **Protocol Error** from the Trigger drop-down list. For a definition of protocol errors see page 93.

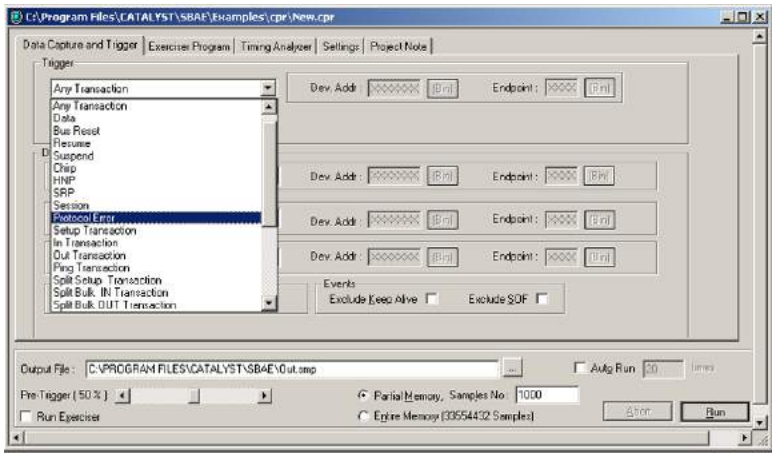


Figure 23 Triggering on Protocol Errors

Exclude Protocol Errors To exclude specific protocol errors from triggering you may mask out the unwanted protocol errors by defining a Protocol Error mask.



To mask the undesired protocol errors, click the **Protocol Errors** Button on the toolbar to open the Protocol Errors Dialog box. Uncheck the protocol errors that you wish to exclude and click **OK**.

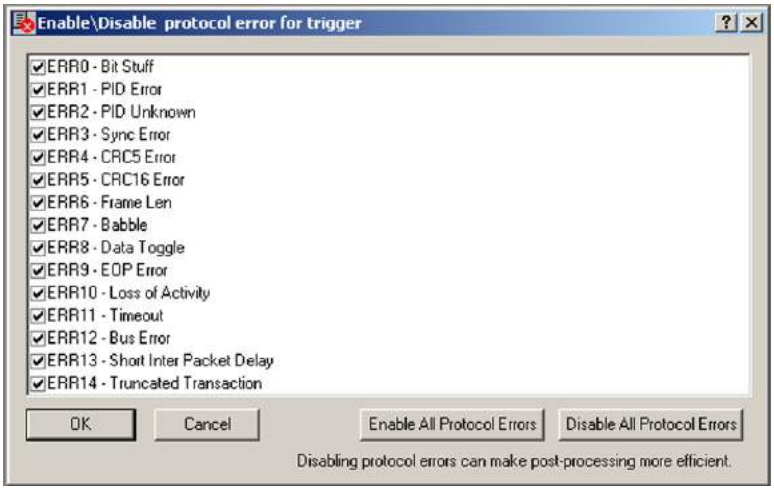


Figure 24 Protocol Error Mask

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

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 2µsec with a tolerance of ±10mV

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

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

* Not available in Full/Low speed.

Full/Low speed only.

Data Capture Options

Memory Display You may limit the captured data display to a specific number of samples by checking Partial Memory and entering the number of Samples to be captured or, you may 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 be captured before and after the triggering event. You may change this percentage by dragging the slider to the desired 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 number pre-trigger event data can be stored. In these cases the data display will show fewer than the specified data points prior to the triggering event. For more detail see **Pre-Trigger** on page 95.

Manual Trigger Select this option when you wish to manually interrupt the data capture based on some external event. Select this option from the Trigger drop down list (see page 37) and start the capture. Data will be continually captured to memory and overwritten as required 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 will 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.

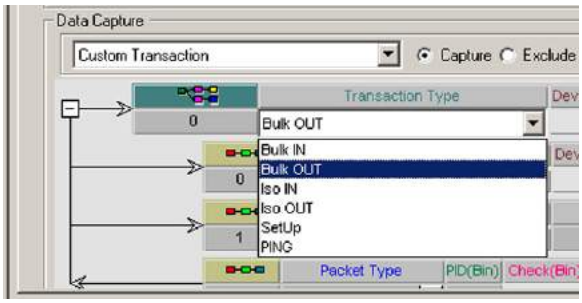
To set up a custom capture:

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



Figure 25 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 may 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 26 shows a functional Exercise and Capture setup for a High Speed Device.

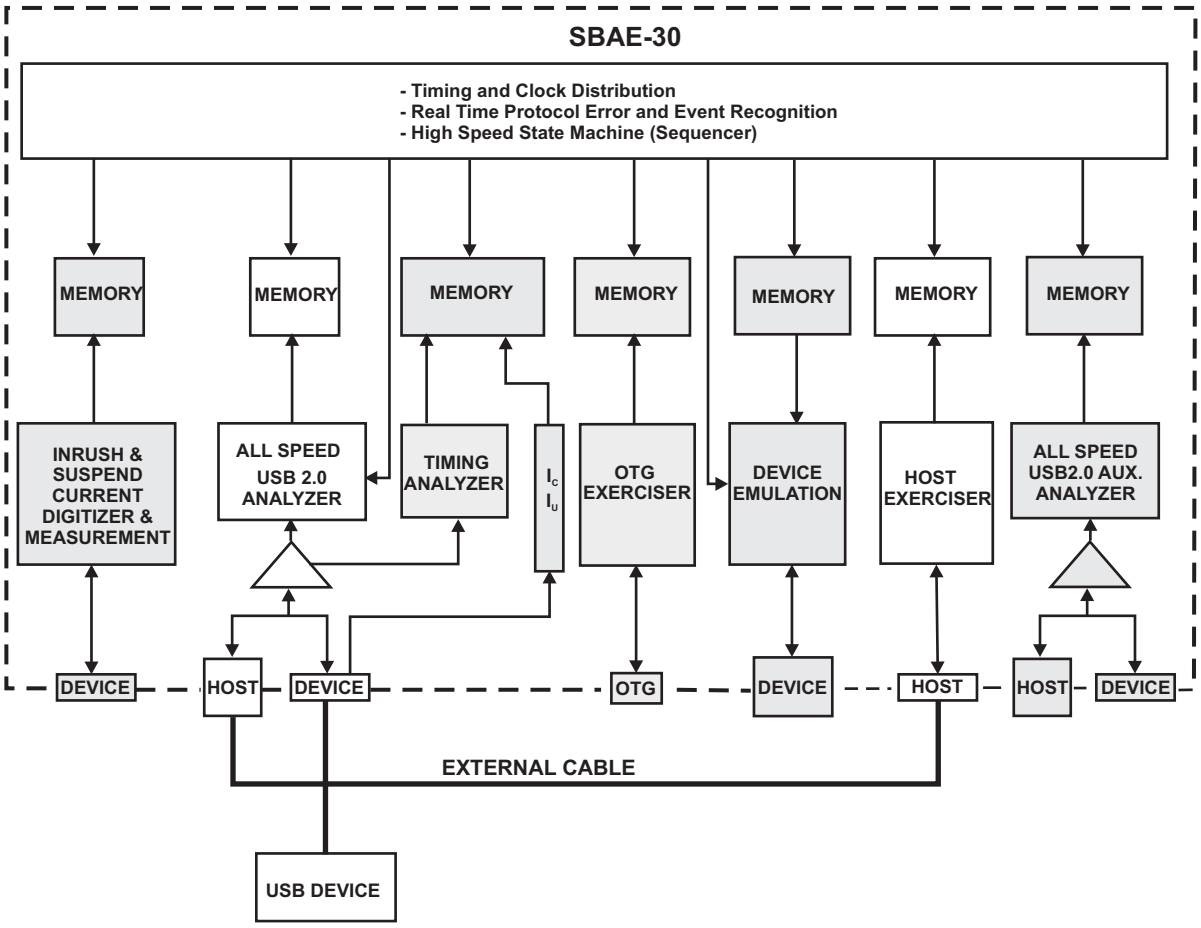


Figure 26 Exercise and Capture Functional Connection

Programming the Exerciser in Easy Mode

Click the **Exerciser Program** tab to open the Exerciser Programming dialog box as shown in Figure 27.

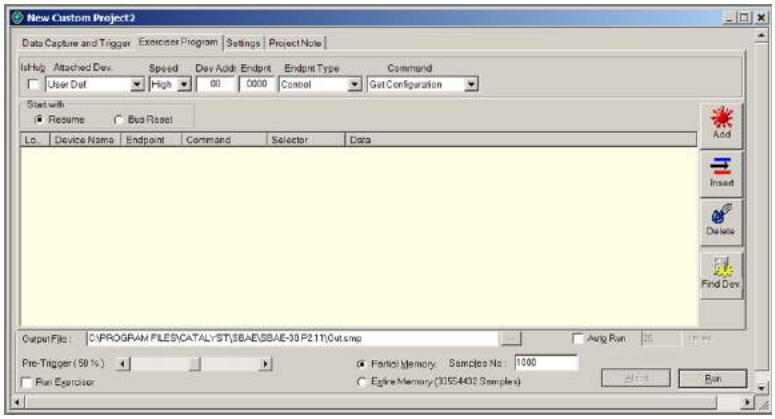


Figure 27 Exerciser Easy Mode Programming Dialog Box

Set up the Device

You must use **Find Device** first in order for the SBAE to enumerate the device automatically. After enumeration is completed, the SBAE will send 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 the SBAE-30 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 28.

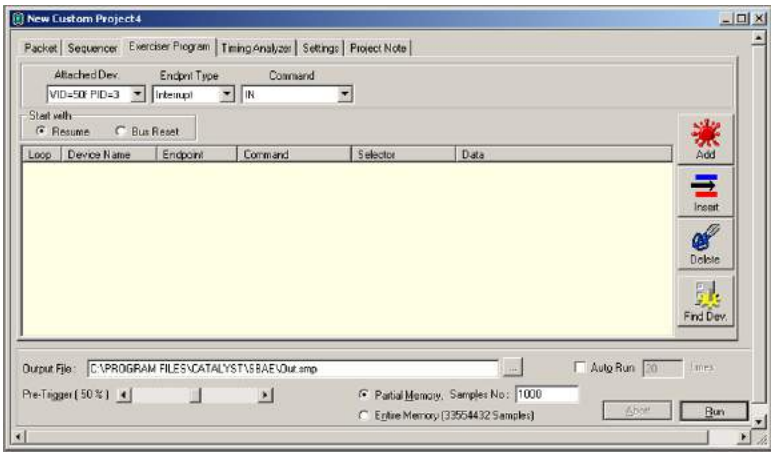


Figure 28 Device Identified

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 as required.

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 46.

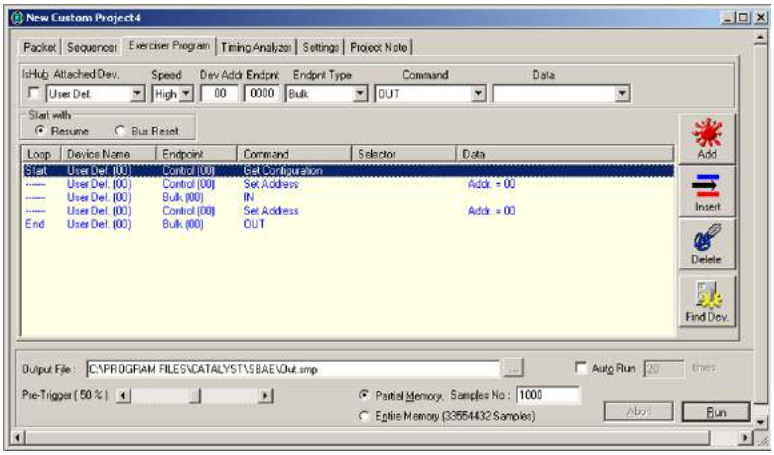


Figure 29 Sample Exerciser Program

Insert Line

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

Define loops To define program loops, click under **Loops** in the program line where you wish for the loop to start and choose **Start**. Similarly click under **Loops** in the program line where you wish the loop to end and choose **End**. See page 46 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, choose a Trigger and Data Capture option or options to analyze the traffic generated by the exerciser.

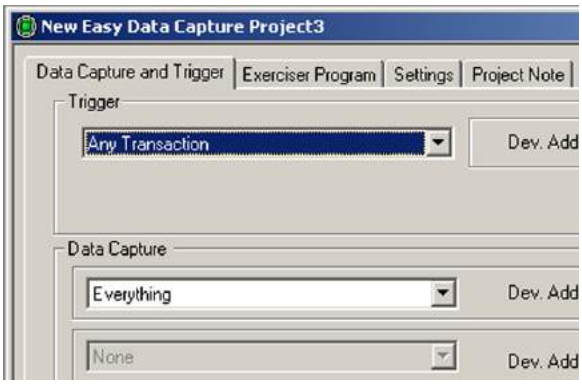


Figure 30 Sample Trigger and Data Capture Selection

Click **Run** to exercise the Device and wait for the captured results to be displayed as shown in Figure 31.



Figure 31 Captured Data Display

Settings

Click the **Settings** tab to open the Project settings dialog box as shown in Figure 32.

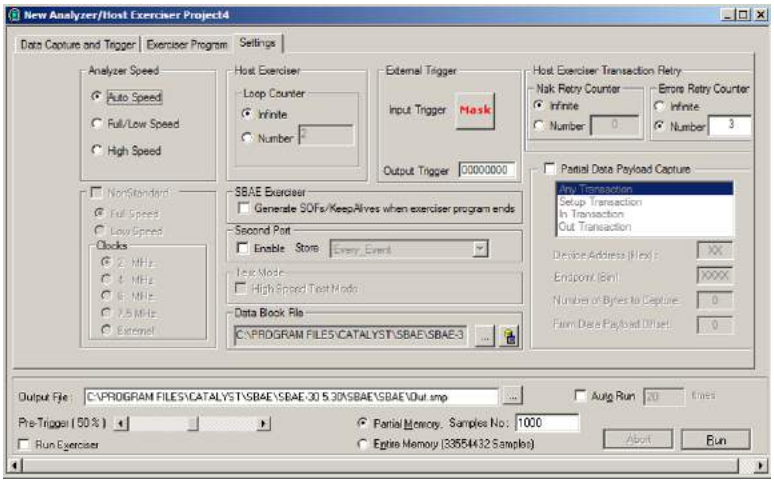


Figure 32 Project Settings Dialog Box

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

Looping You may specify the number of times that you wish the command line(s) to loop or to run continually. You may set the number of times from 2 to a maximum of 1048575 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 the desired **Clocks** choice. The available choices are 4 pre-set clock speeds and an External clock which must be connected to the Analyzer as described on page 9.

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 75.

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 will allow the capture of this packet.

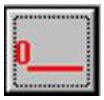
High Speed Test Mode Usage -

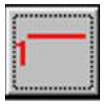
1. Set Test Mode on the Device (This can be performed using the exerciser).

2. On 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 37, click the Trigger Mask button in the Project Settings dialog repeatedly to set the desired 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

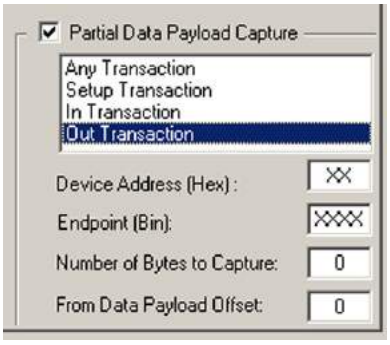
To output a data pattern whenever a trigger occurs, enter the desired pattern in the **External_Bits** text box in Easy Mode and in the Sequencer when operating in the Advanced Mode. To access the data pattern as external signals see "SBAE-30 External I/O Connector Pin Assignment" on page 16.

Transaction Retry

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

Partial Data Payload Capture

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



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

Auxiliary Port

To perform Hub testing using the Auxiliary port the connections must be made as shown in Figure 33.

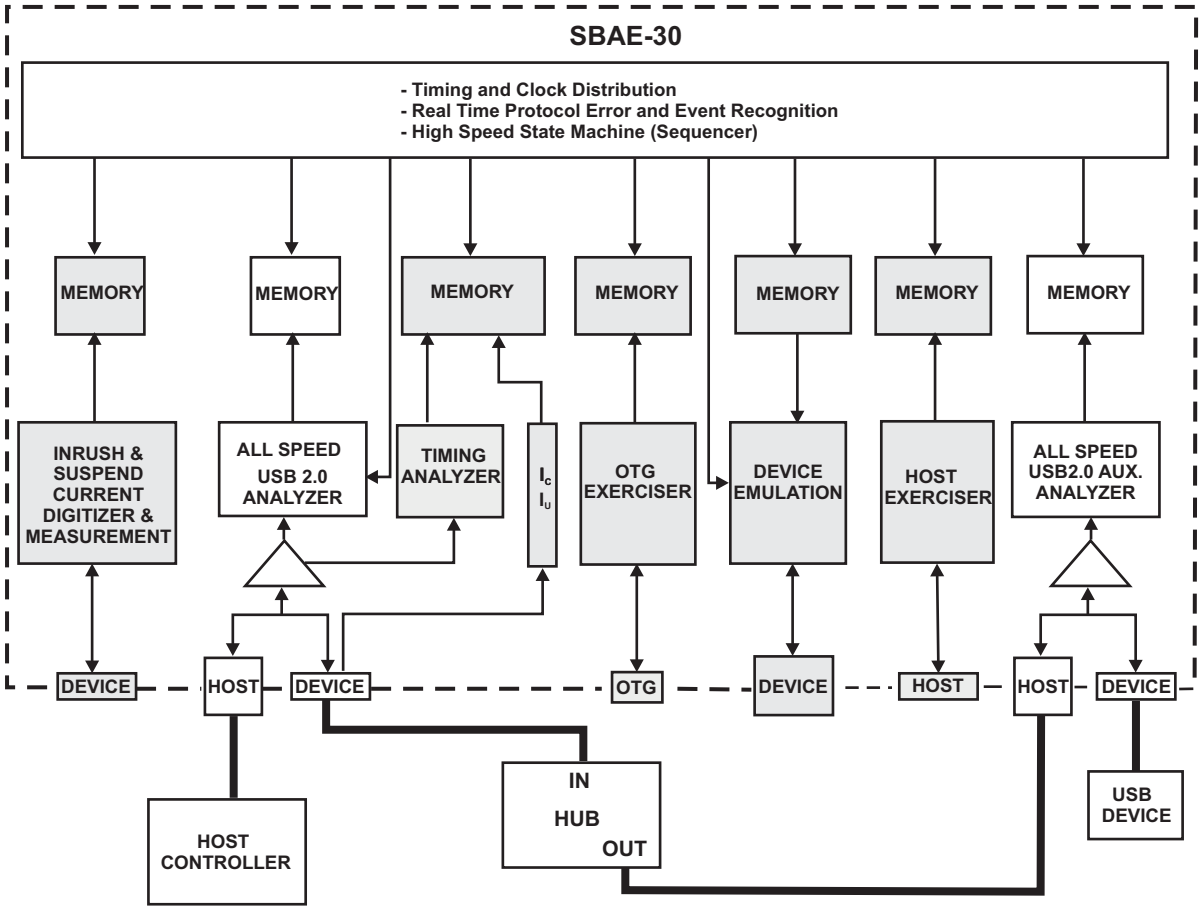


Figure 33 Auxiliary Connector in Hub Testing

Auxiliary Port Data Capture



Figure 34 Auxiliary Port Data Capture Selection

Analyzer Only

When used for Hub testing the SBAE-30 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 an informative note about the current project click on the **Project Note** tab and enter the data that you wish to associate with this project.

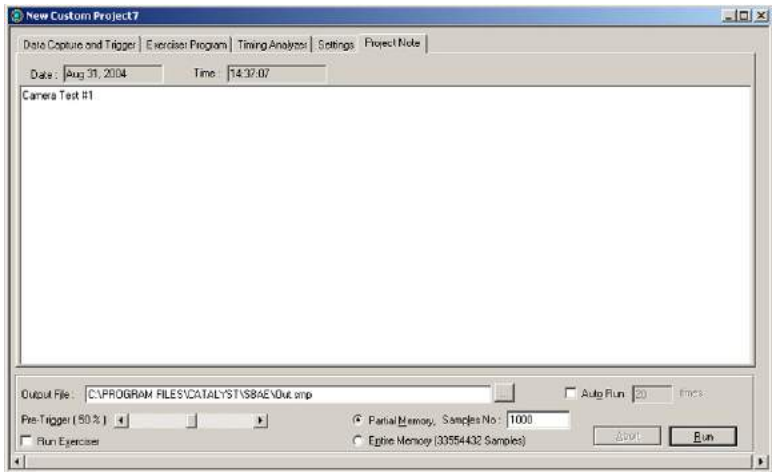


Figure 35 Easy Project Note

Advanced Mode (User Defined)

This mode expands your 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 28.

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

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

For a custom project you must:

- 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 4.

Defining Packets

You may define up to 8 different packets. To define a packet, click on the **Packet** tab to make sure that the packet definition dialog is displayed on top.

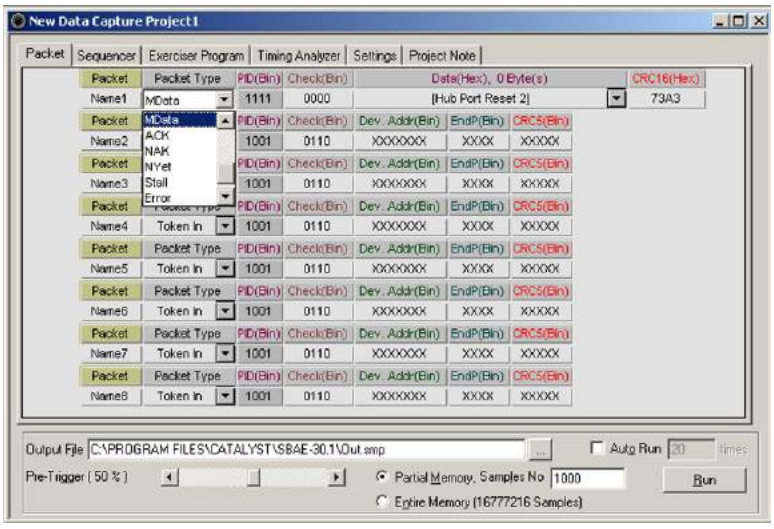


Figure 36 New Project Dialog Box

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

Packet Types[1]

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 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 can be changed 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 can be changed to search for other CRC5 values on the bus. (5 bit CRC used by packets other than data)
CRC16	Default value that can be changed 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 the SBAE-30 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 on 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 75.

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



Figure 37 User Defined Packet Dialog Box

Overlap Warning

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



Figure 38 Overlap Warning

If you proceed with overlapping selections the result will be entries that cannot be edited because of don't care entries. For example, combining 1101 with 1001 results in 1X01.

The Sequencer

The Sequencer is used for data capture manipulation, generating complex triggering and starting the exerciser if required. The 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". Each state can be programmed to go to any other state depending upon the occurrence of certain 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.

A trigger may be set in the Sequencer to occur at any state on:

- Every Packet (Unconditional)
- Every Event (Unconditional)
- The occurrence of any Packet or a boolean expressions of packets.
- The 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 is commanded to run the SBAE-30 starts monitoring data on the bus and starts the sequencer in "State0". "State0" may be user programmed to **start capture immediately** (Default) or **wait for a specific occurrence** on the bus. The choices of what to capture are user selectable and are:

- 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 is programmed to transition to another state selected by the user.

Each state may be programmed to output an external bit pattern, set a trigger and change what is to be captured. If the capture involves the use of the exerciser you may only start it in "State0".

Programming the Sequencer

To program the Sequencer, click the **Sequencer** tab to display the sequencer programming dialog as shown in Figure 39**.**

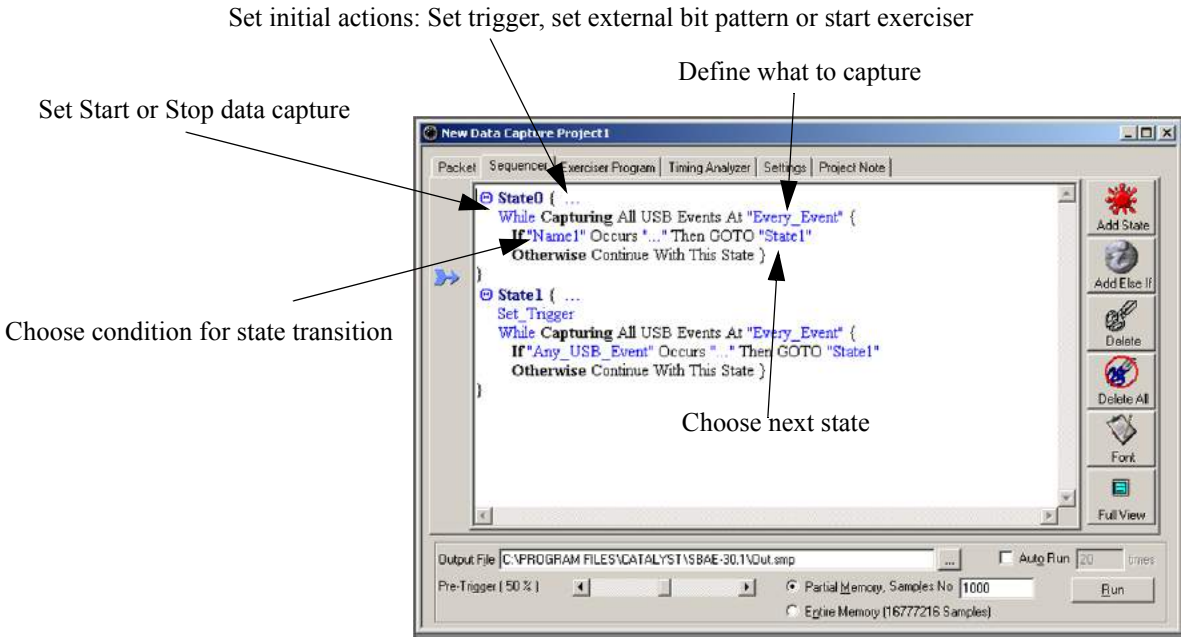


Figure 39 Sequencer Programming Dialog

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

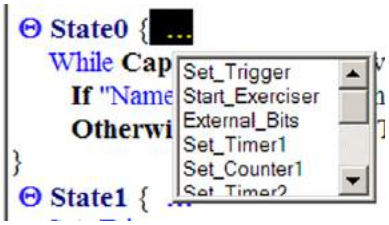


Click the **Add State** button to add additional states as required, up to 32

Program State0

In **"State0"** you choose, if you want to set a trigger, set an external bit pattern, start the exciser, 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**"



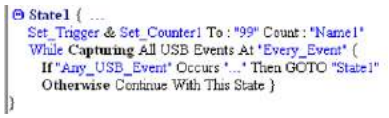
2. Choose one or all of the available choices one at a time.

Counters

The sequencer incorporates two counters each of which may be configured 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

The counters may be set with the Set_Timer and Set_Counter commands with corresponding timer or counter values in any state.



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



Start Exerciser

You may 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 60.

External Outputs

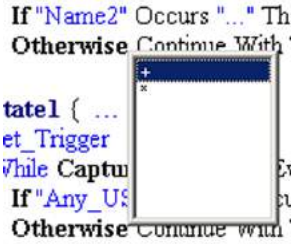
You may set the 8 external output bits on the output connector at any state by clicking the ellipses next to the sate and choosing **External_Bits** from the drop-down list for that state and then entering a desired bit pattern. See page 16 for external output connector pin assignment.

3. To define what is to be captured, click the **"Every_Event"** to display the capture choices.



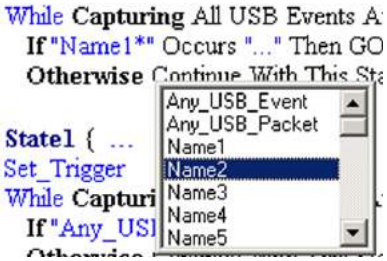
Every_Event is the default, however, you may choose every packet, a specific packet or event or create a boolean expression from the available choices.

Create Boolean Expression You may create a boolean expression by clicking the left mouse button immediately after entering the first parameter to open the boolean list box. Choose the desired boolean operator and then select the next parameter in the expression resulting e.g. Name2+Name3.



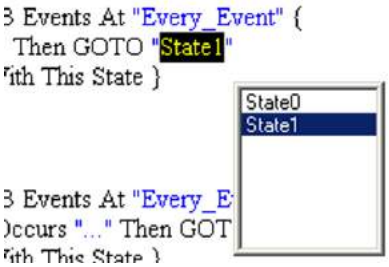
The operator convention used is + for "or" and * for "and".

4. To transition to another state click the blue **"Name1"** to display the transition choices.

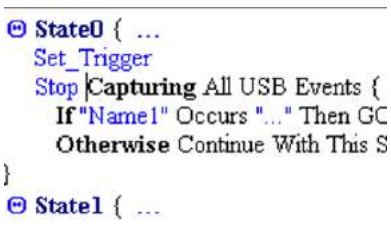


Choose a packet or bus event or create a logical expression. When this choice is detected on the bus, the sequencer will transition to another state.

5. Choose the next state by clicking the blue **"State1"** to display the available states and choose the next state.

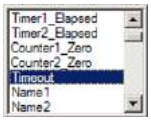


6. The SBAE-30 starts capturing all bus activity at start by default. To delay the start of capture until some occurrence, click the blue **"While"** to toggle it to "**Stop".**



7. 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 may identify a block of data to be discarded. Setting **Tag_Event** will tag the start of a data discard and setting **Discard** will designate the point up to which the data is to be discarded. Data will be discarded between the last Tag_Event set and the Discard set.



Add Else If You may include **Else If** statements in your program to allow the 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 on it and choose <Delete> from the open list box.

To Delete a a Parameter Highlight it and push the delete key on the keyboard.

Optimized Workspace To avoid scrolling the display in long sequencer programs, you may minimize completed states by clicking the symbol next to the state to be minimized. To expand the minimized state, click the state name. Alternatively this can be accomplished by clicking on the state name.



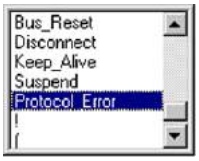
You may maximize your workspace to full screen by clicking the **Full View** button

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

Note: You may not 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 may set the sequencer to trigger on protocol errors at any state by clicking "Packet" and then choosing **Protocol_Error**. If you wish for some of the protocol errors to be excluded then set the protocol mask as described in the following section.

For a description of the detected protocol errors see page 93.

Setting a Protocol Error Mask



To mask undesired protocol errors, click the **Protocol Errors** Button on the toolbar to open the Protocol Errors Dialog box. Uncheck the protocol errors that you wish to exclude and click **OK**.

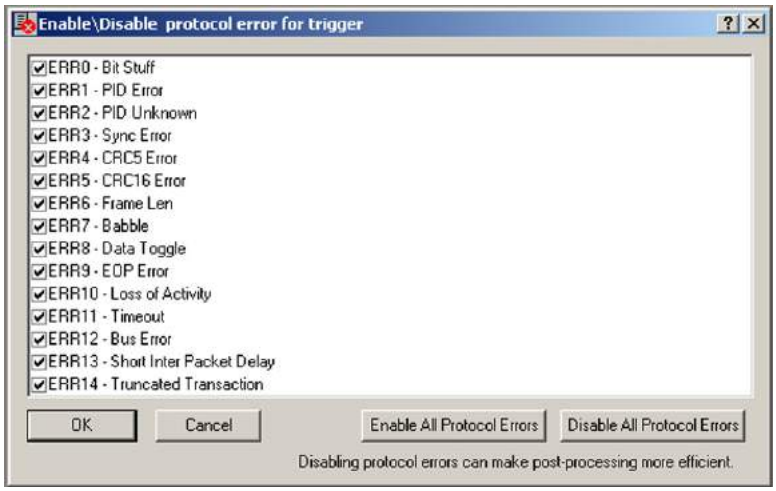


Figure 40 Protocol Error Mask

Host Exerciser (Optional)

Note: To perform data capture and triggering with exerciser generated data you must use the Exerciser Port as shown in Figure 4.

Programming the Exerciser In Advanced Mode

To program the exerciser click the **Exerciser Program** tab open the exerciser programming dialog box as shown in Figure 41

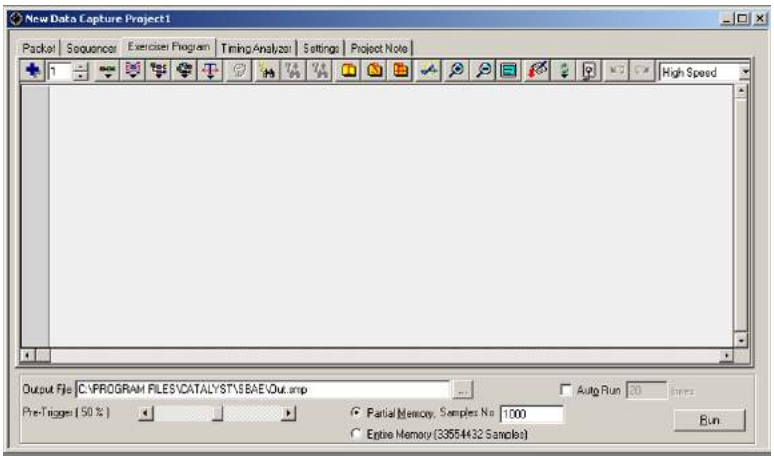
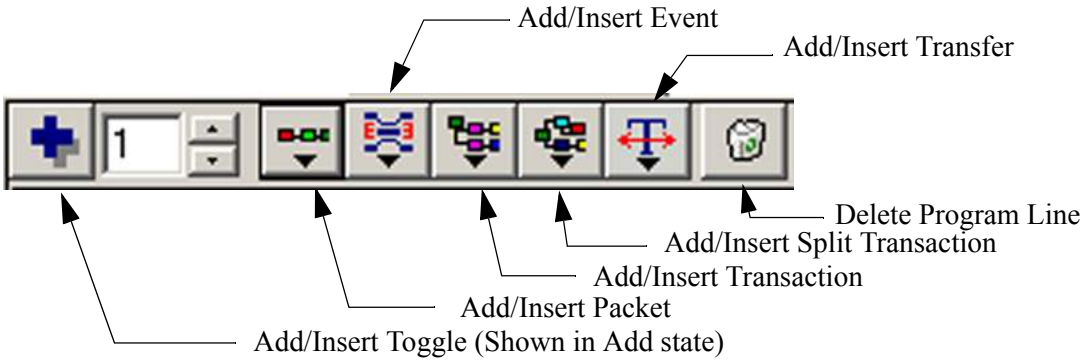


Figure 41 Exerciser Programming Dialog Box

Set the Host Speed First! Click the down arrow for the speed selection drop-down box and choose the desired operating speed.

Maximum Program Lines You may 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 be added and select the type of program lines to be added by clicking the appropriate category button from the choices shown below and then selecting the line type.



Insert Program Lines You may always insert additional program lines as required by positioning the cursor on an existing line. Click the **Add/Insert** toggle to

the insert state, enter the number of lines to be inserted and click the appropriate line type button and select the line type to be inserted. The specified number of lines will be inserted above the line where you positioned the cursor

Delete Program Lines To delete a program line, position the cursor in the line that you wish to remove and click the Delete button. You may 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.

1. Click the Add/Insert Event button and choose **Reset** from the drop-down list to add a Bus Reset Event.

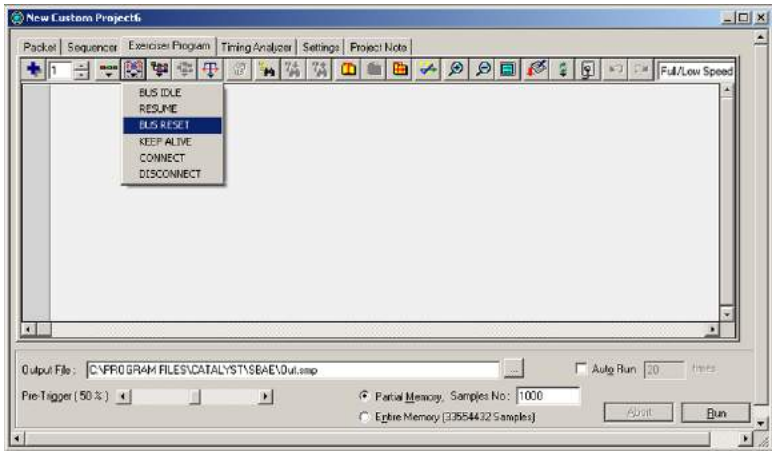


Figure 42 Adding a Bus Reset

Add a Control Transfer(s)

2. Click the Add/Insert Transfer button and choose **Control** from the drop-down list to add a control transfer line.

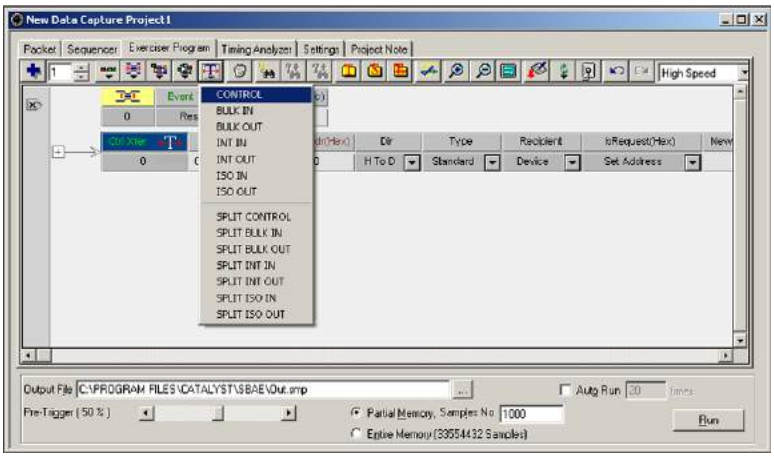


Figure 43 Adding a Control Transfer in High Speed Protocol

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 46.

Transfer type The default transfer type is **Standard**, however you may additionally choose Class, Vendor or user defined by clicking the down arrow under Type and making the appropriate choice from the drop-down list.

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

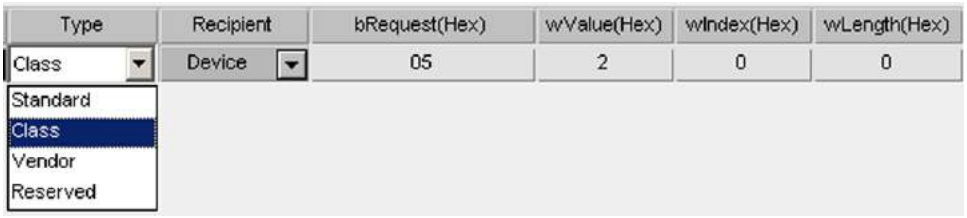


Figure 44 Choosing Class

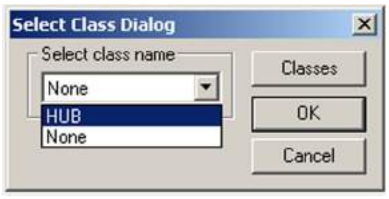


Figure 45 Select Class Dialog

4. Click the down arrow for Select class name, choose a pre-defined class, click OK and click in the blank program entry area to register your choice. **Note** that once you register your choice, all of the parameters for the transfer line are updated automatically for the request or command selected.

Specify a new Class You may specify a new class by clicking **Classes** in the Select Class Dialog and proceed as described on **page 65.**

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

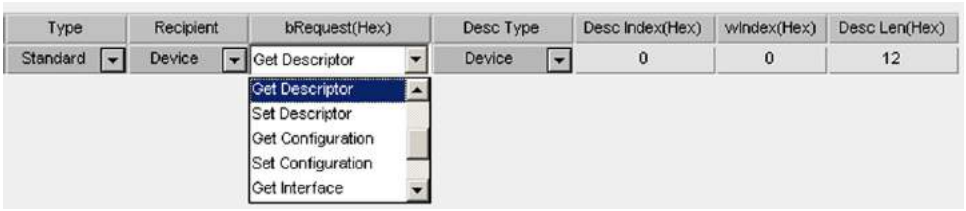


Figure 46 Choosing a Request or Command

6. Enter and set up additional Control Transfers e.g. Set Address as required to set up the exercising session.

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)

7. Click the Add/Insert Transfer button and choose a desired data transfer from the drop-down list e.g Bulk Out.

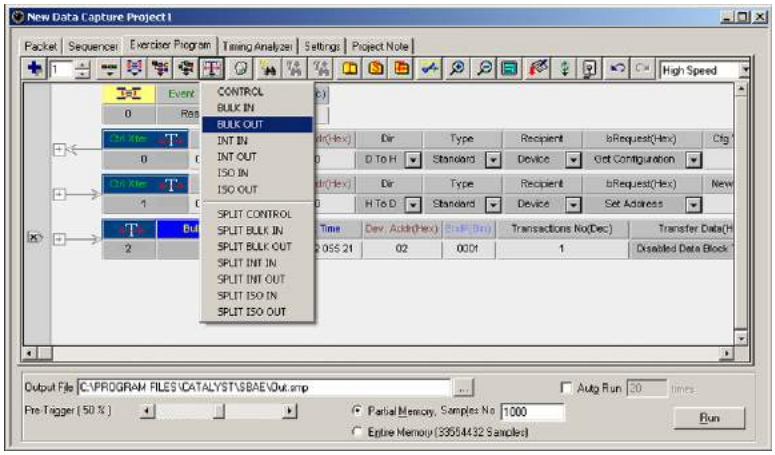


Figure 47 Adding a Data Transfer in High Speed Protocol

Set up Data Transfer

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

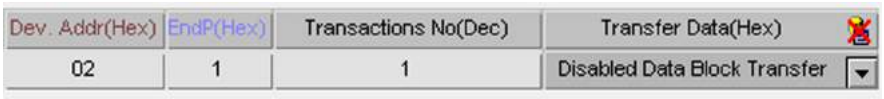


Figure 48 Transfer Setup Fields

Specify Endpoint number

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

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

Define Endpoint Settings



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

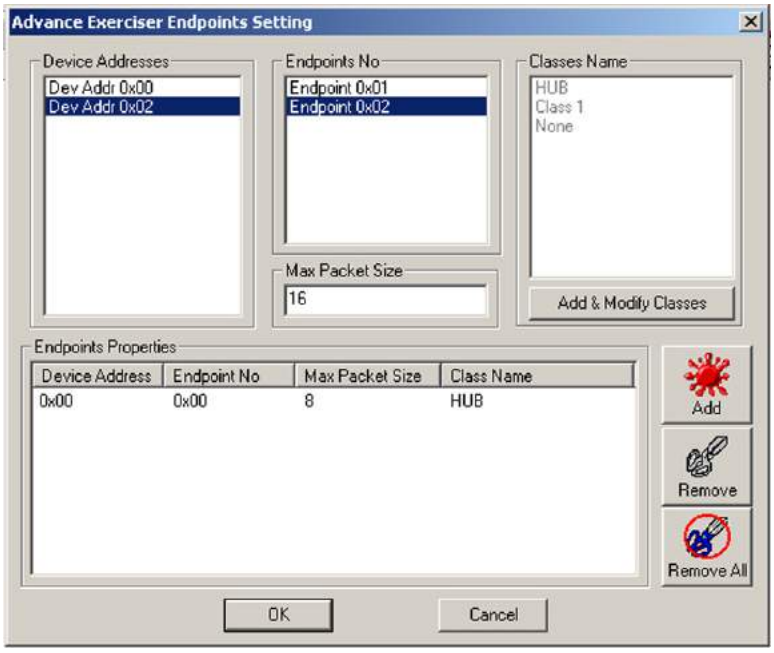


Figure 49 Endpoint Setting Dialog

1. Highlight the Device Address and then highlight each endpoint and enter a packet size value in the **Max Packet Size** text box (only if other than the default of 8 is required) and click **Add** button until all of the endpoints appear in the **Endpoint Properties** box
2. Click OK.

Specifying a New Class

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

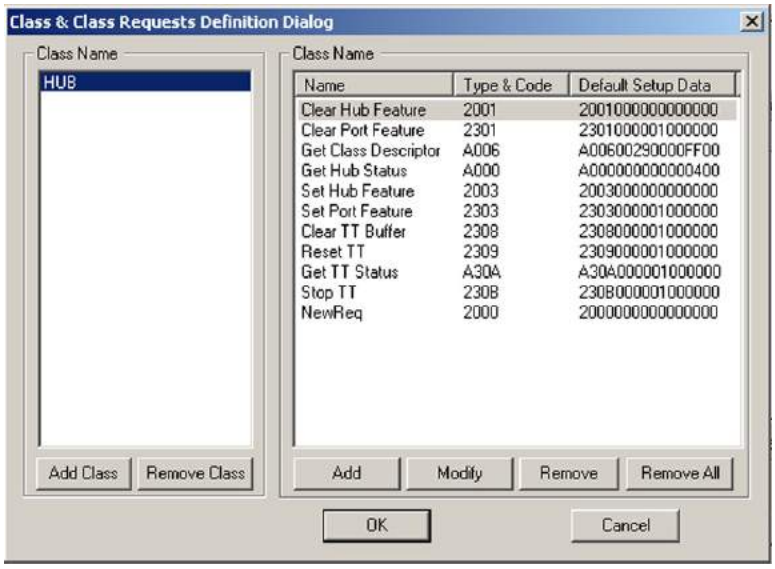


Figure 50 Class & Class Requests Definition Dialog

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



Figure 51 Enter Class Name Dialog

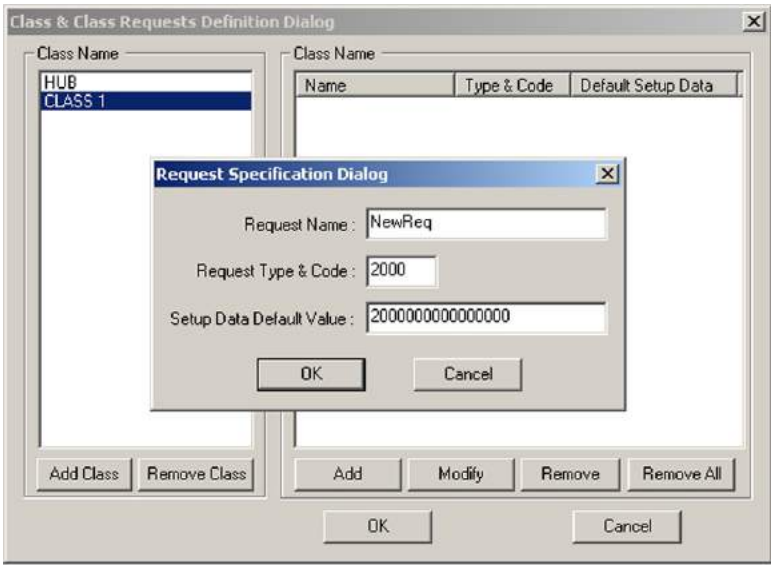


Figure 52 Defining Requests for a New Class

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 required 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 may 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 in specific data that you wish to be used in this transfer.

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



Figure 53 Choosing Data to Transfer

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

7. When the Endpoint number and the data to be transferred has been specified, click in the blank program entry area to register your choice. Once the choice is registered the **Bytes Transferred(Dec)** field will display the number of data out Transactions created.
8. Add additional transfers as required.

Specify data in Transaction layers Transfers specified in Transaction layers offers the user a flexible way to set up a **Transfer** with different data patterns in each transaction for that transfer.

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

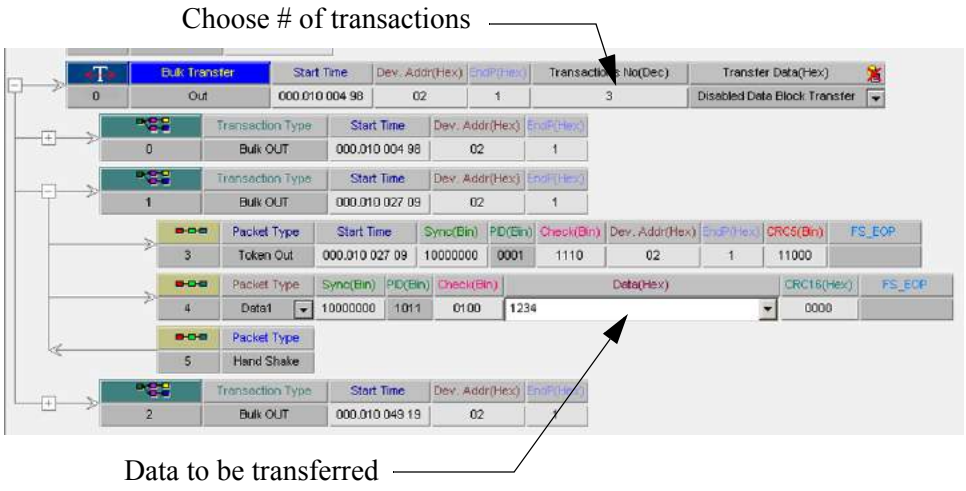
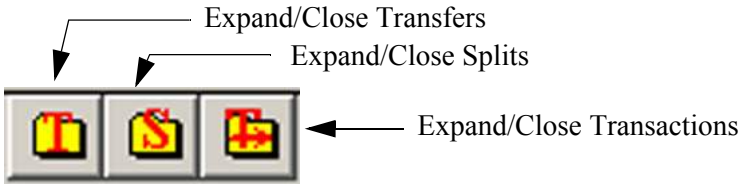


Figure 54 Transfer With 3 Transactions

Create a data block For commands that involve data transfers click the **Data Block** button and define the required data block. See page 75 for procedure on how to create data blocks.

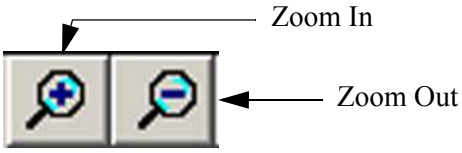
Expand/Close

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



Zoom In/Zoom Out

Clicking the corresponding zoom button on the toolbar allows you to optimize the size of the displayed code lines for your viewing preference.



Forcing Errors

You may introduce errors for any of the fields for a packet by highlighting the field to be changed and making the required change directly in the field. Bit Stuff error (Unchecked has Bit Stuff) and force no EOP (Unchecked has EOP) is available by right clicking on the packet.

Program Loops

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

Set up Loops

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

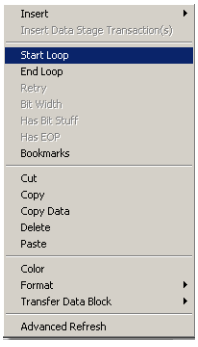


Figure 55 illustrates a programmed loop.

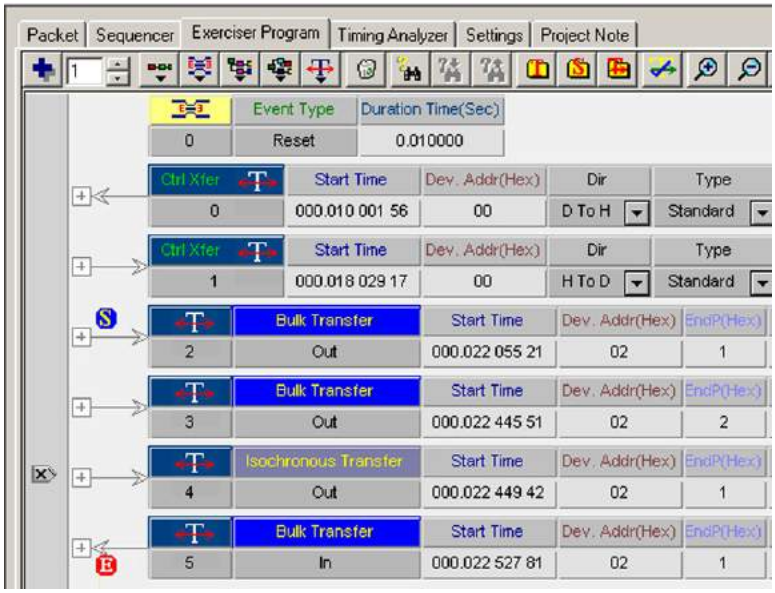


Figure 55 Loop Start and End Set

Deleting Loops

To delete a loop, right click the mouse on the Start Loop or End Loop symbol and click the appropriate **Start Loop** or **End Loop** in the menu to remove the check marks.

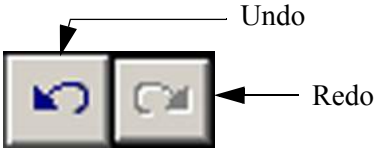


Loop count

Loop count is the number of times that a loop is to be executed. You may set the loop to execute a fixed number of times or to run continually. This choice is set on the **Settings** tab as described on page 81.

Undo/Redo

You may exercise up to 200 levels of Undo/Redo steps in your 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. Bit width may be reduced 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.



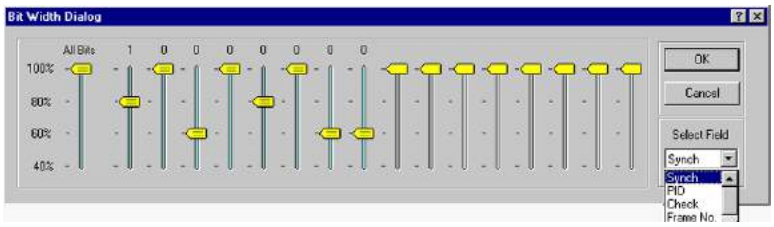


Figure 56 Set Bit Width Dialog

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

Exerciser Programming Shortcuts

You may quickly create an exerciser program from an existing *.smp file. There are two ways to quickly create an exercise program. You may import a complete *.smp file or open a *.smp file and copy the lines that you would like to make your exerciser program.

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.

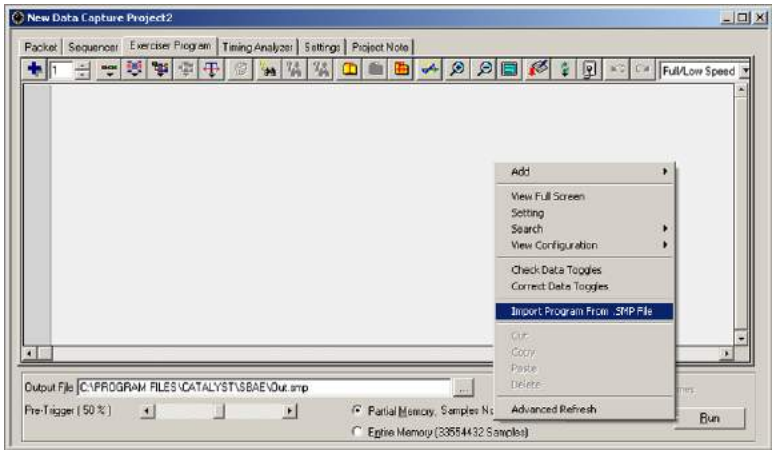


Figure 57 Import Exerciser Program

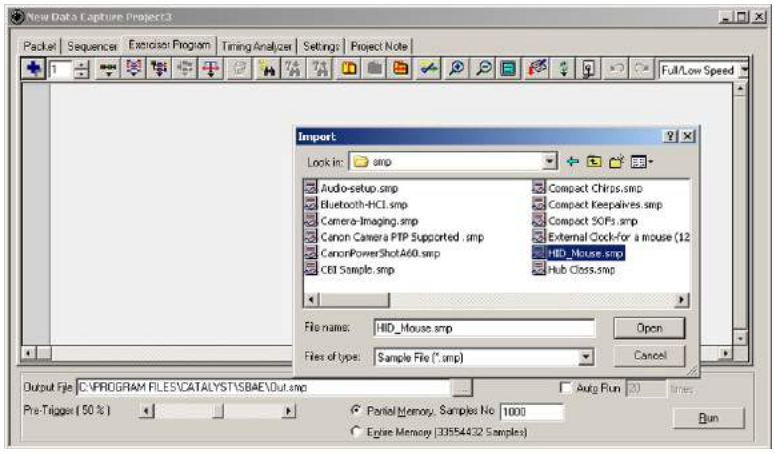


Figure 58 File Import Dialog

Choose an *.smp file that you would like 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.

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.



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

The Advanced Refresh capability allows you to specify the refresh range and to change 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.

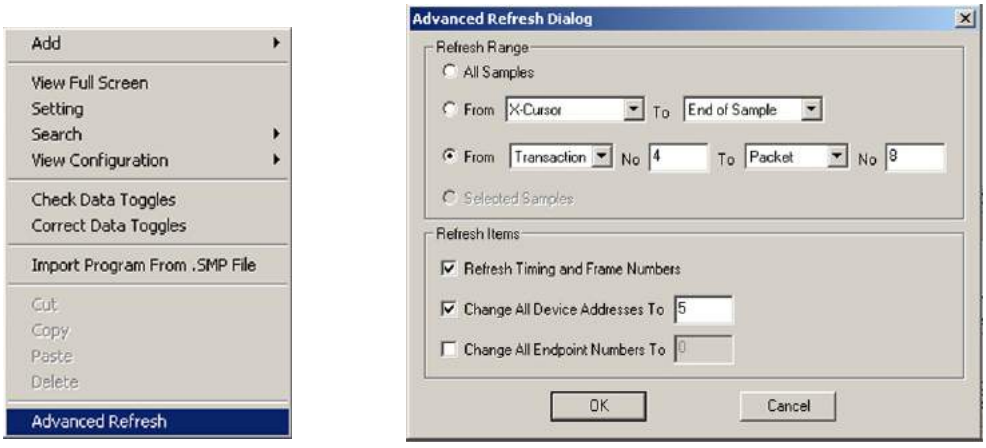


Figure 59 Advanced Refresh Dialog

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 and draw a rectangle around a group of data lines with the mouse cursor and then release the mouse button to select them.



Figure 60 Lines Selected

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



Click the **Copy** Button or right click the mouse cursor in the data area and choose **Copy**.

2. Open an Advanced Project and click on the **Exerciser Program** tab.



Click the **Paste** Button or right click the mouse cursor in the exerciser program area and select **Paste.**

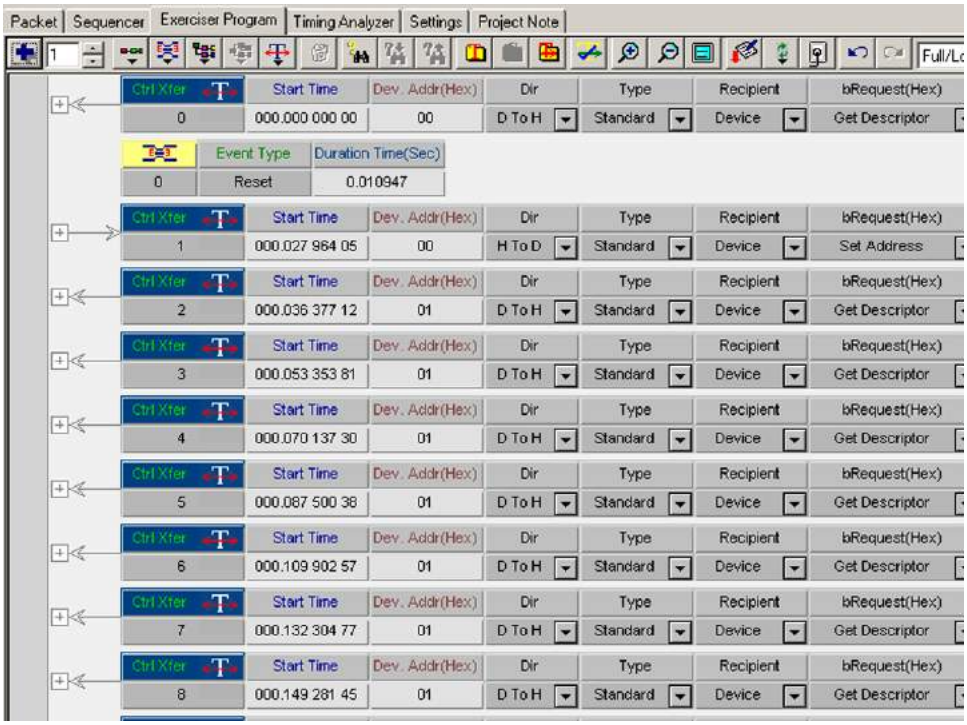


Figure 61 New Exerciser Program

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.

To Refresh the Program

Click the refresh 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.



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 may now use this new exerciser program immediately in your current project or save it for use elsewhere.

Creating a Data Block

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

1. Custom data pattern specifically for your application
2. Random data pattern
3. Counter
4. Walking "1" or "0" pattern.



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 62**.**

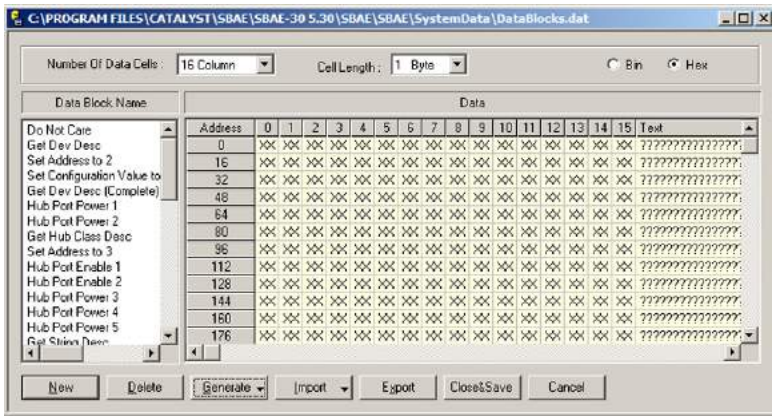


Figure 62 Default Data Block Dialog Box

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

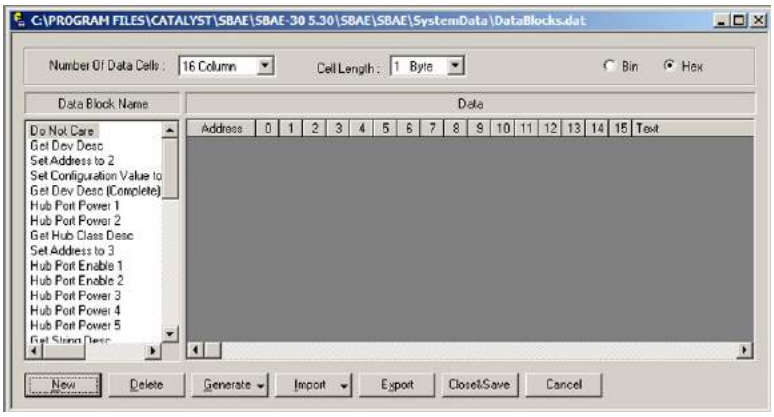


Figure 63 New Data Block Dialog Box

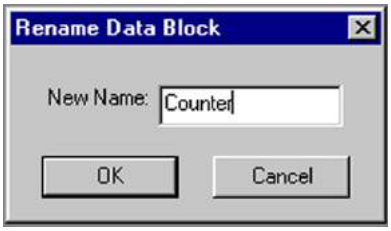
2. Choose the number of data columns (Up to 16 Data Cells/Row) and the Cell length (Up to 16 Bytes/Cell). This is a display function only.
3. Click either the Bin or Hex option button to choose a desired number format.

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.**

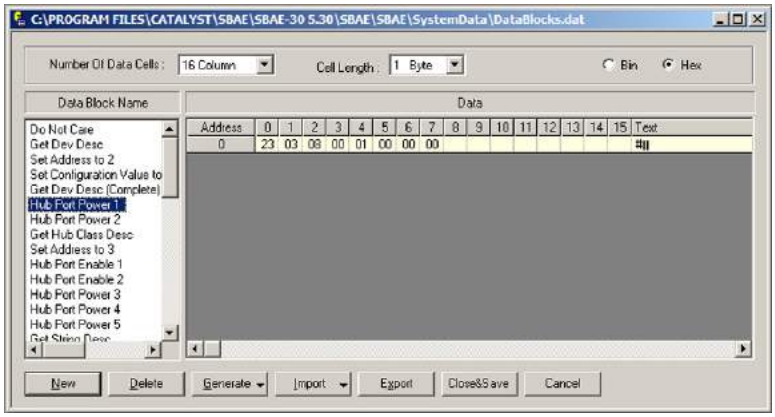


Figure 64 Sample Active Data Block Hub Port Power

You may 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

1. Click the **Generate** button and choose **Pattern** to open the Define Pattern dialog box as shown in Figure 65
2. Enter the desired data pattern in the Data Pattern edit box.
3. Choose the number of times that you would like that pattern to be repeated and click **OK.**



Figure 65 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 may 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 will be overwritten or new data inserted after that cell click the **Insert/Overwrite** button to toggle to the desired operation.

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

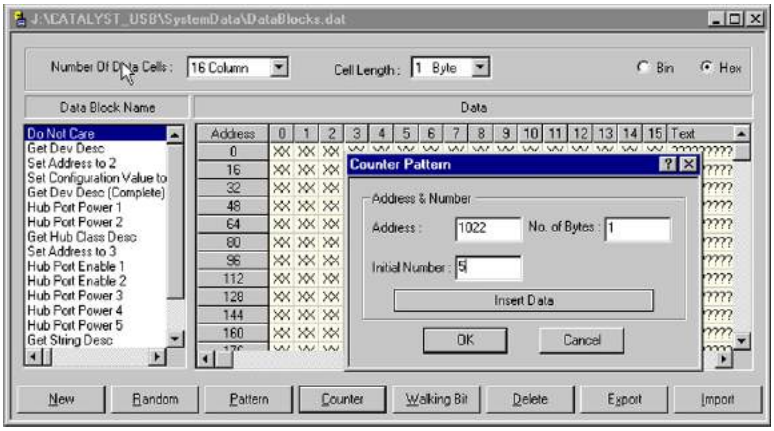


Figure 66 Set Counter as Data

Random Data Pattern To use a random data pattern, click the **Generate** button and enter the number of times that you wish the pattern repeated and click **OK.**

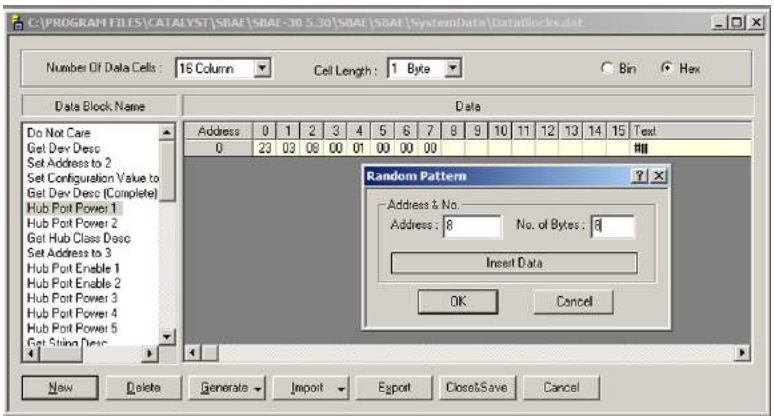


Figure 67 Choose a Random Pattern

Walking Bit Pattern

To use a walking bit pattern, click the **Generate** button and choose either a walking bit of "0" or "1", the walk direction, the start position and the number of times the pattern is to be repeated.

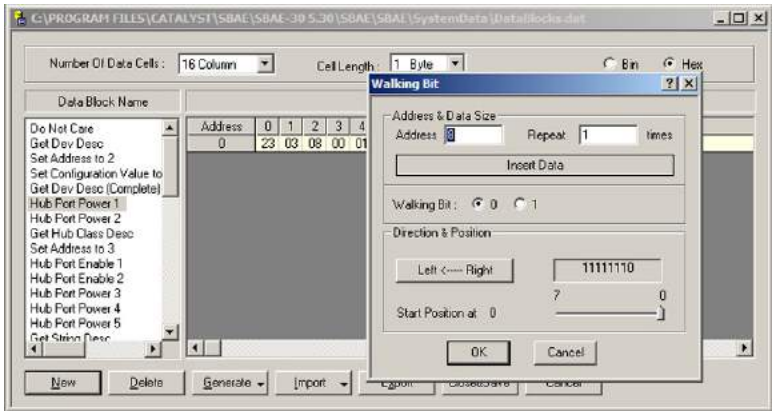


Figure 68 Define a Walking Bit Pattern

Sample Data Block

Figure 69 illustrates a Data block with a defined pattern AA starting at address 0 and repeated 4 times followed by a random pattern starting at address 4 and repeated 4 times and a left to right walking 0 pattern starting at address 8 and repeated 6 times.

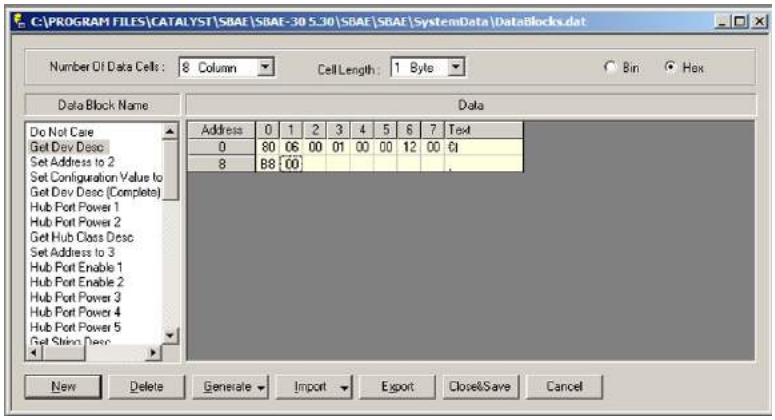


Figure 69 Sample Data Block Definition

Close & Save

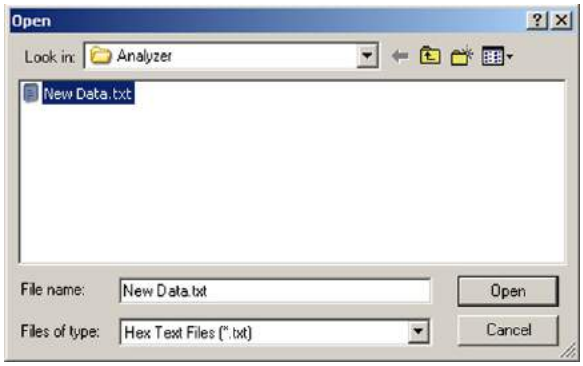
When you have completed a data block definition click the **Close&Save** button to save the newly created data block.

The saved data block will automatically be assigned a sequentially numbered (DataBlock##) title. To rename for easier identification later see "Naming a Data Block" on page 76.

Creating and Editing Data Blocks as Text

You may 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 and save as a *.txt text file.

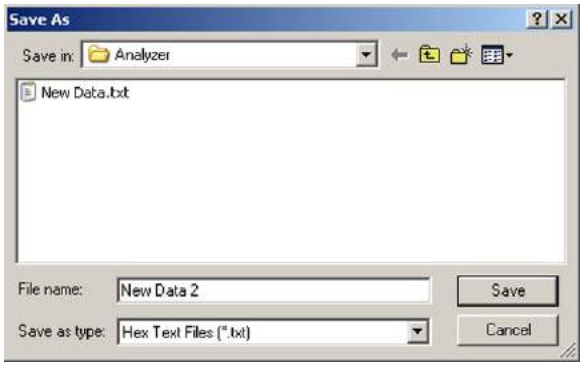
Import from Text File To import Text Editor created data click the **Import** button in the data block definition dialog and choose Import from Text File to open the **Open** dialog.



Choose the desired 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 be edited 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 may 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. See “Settings” on page 46 for description of settings options.

Auxiliary Port

The Auxiliary port is used in Hub testing as shown in Figure 5 on page 12.



Figure 70 Auxiliary Port Data Capture Selection

Analyzer Only When used for Hub testing the SBAE-30 Auxiliary port performs as an analyzer only.

Project Note To enter an informative note about the current project click on the **Project Note** tab and enter the data that you wish to associate with this project.

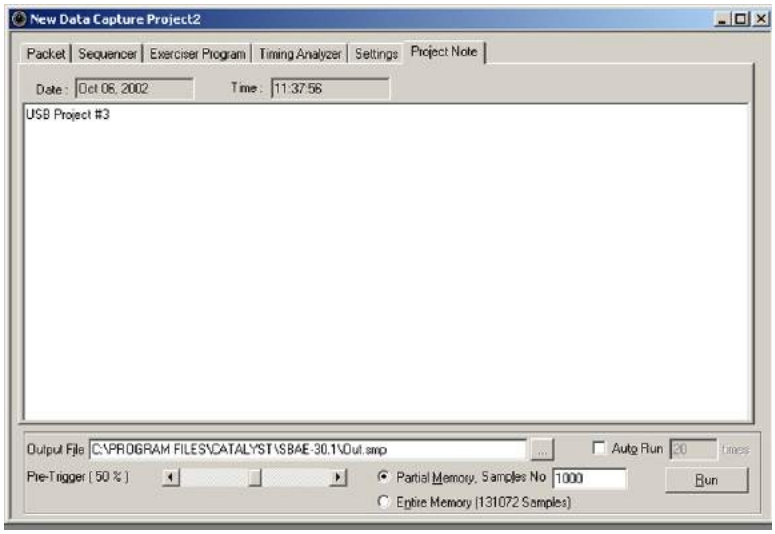


Figure 71 Informative Project Note

Running the Project

Set The Data Capture Options

1. **Capture to Memory -** You may limit the captured data display to a specific number of samples by checking **Partial Memory** and entering the number of Samples to be captured or, you may check **Entire Memory** to allow the capture for the entire memory. Choosing Partial Memory results in less time to display result.
2. **Pre-Trigger -** Pre-Trigger is set by default at 50% which defines the percentage of data to be captured before and after the triggering event. You may change this percentage by dragging the slider to the desired value.

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 in cases where the triggering event occurs before the required number pre-trigger event data can be stored. In these cases the data display will show fewer than the specified data points prior to the triggering event. For more detail see “Pre-Trigger” on page 95.

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

To Run the Project: Click **Run** and wait for the occurrence of a trigger event to open the data capture result.

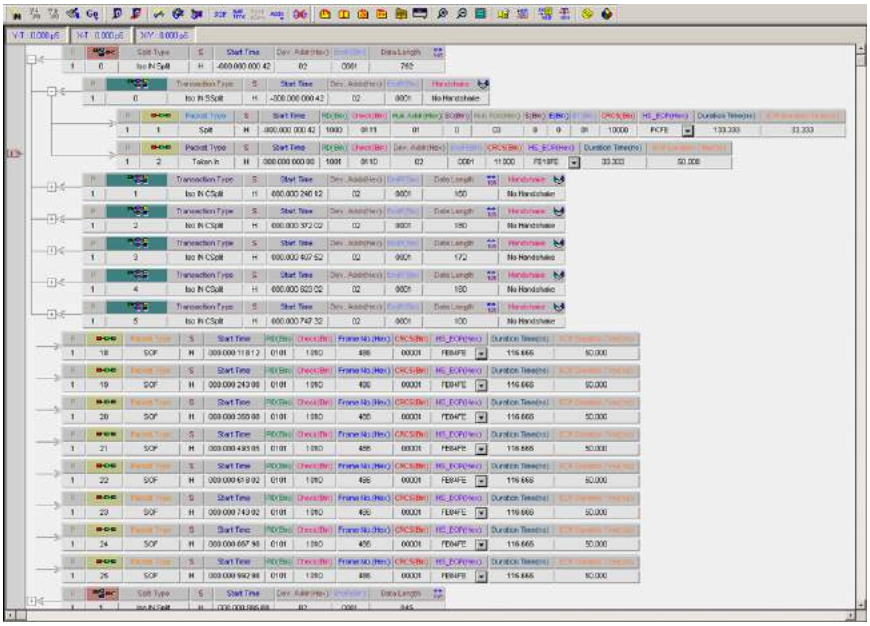


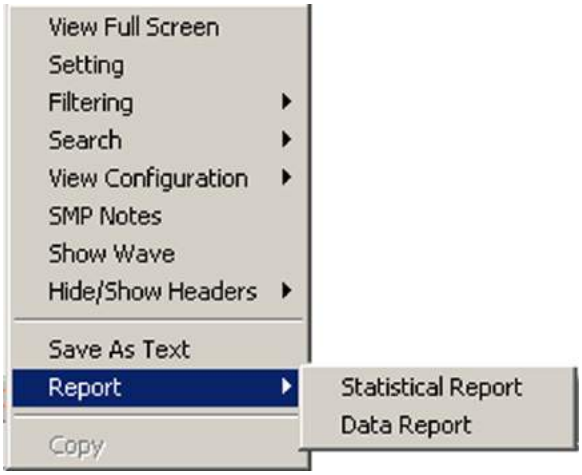
Figure 72 Data Capture Result

Display Manipulation For details on configuring the data capture result display for your specific analysis needs see **“Display Manipulation” on page 155.**

Reports

Whenever a data capture is displayed in the sample viewer, you may 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 may also access the reports from the sample viewer toolbar.



View Statistical Report

Choose **Statistical Report** to open the Select Statistical Report Range dialog. Select the desired report range and click OK to display the result as shown in Figure 73.

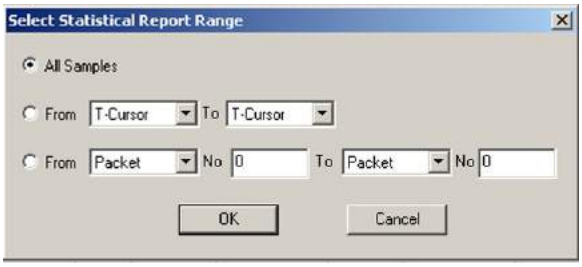




Figure 73 Statistical Report

View Data Report

Choose Data Report to open the Select Report Type dialog.

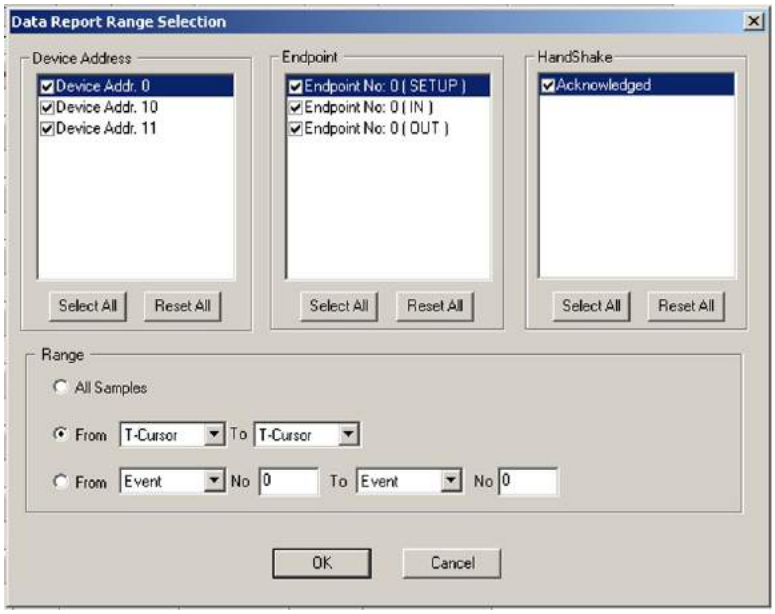


Figure 74 Select Report Type Dialog

Make selections to view the report on Transaction or Endpoint type Device Address(es), Handshake or all. You may specify to report on all of the samples, between specific cursors or between specified Events. Make the appropriate choice and click OK to display the result as shown in Figure 75.

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

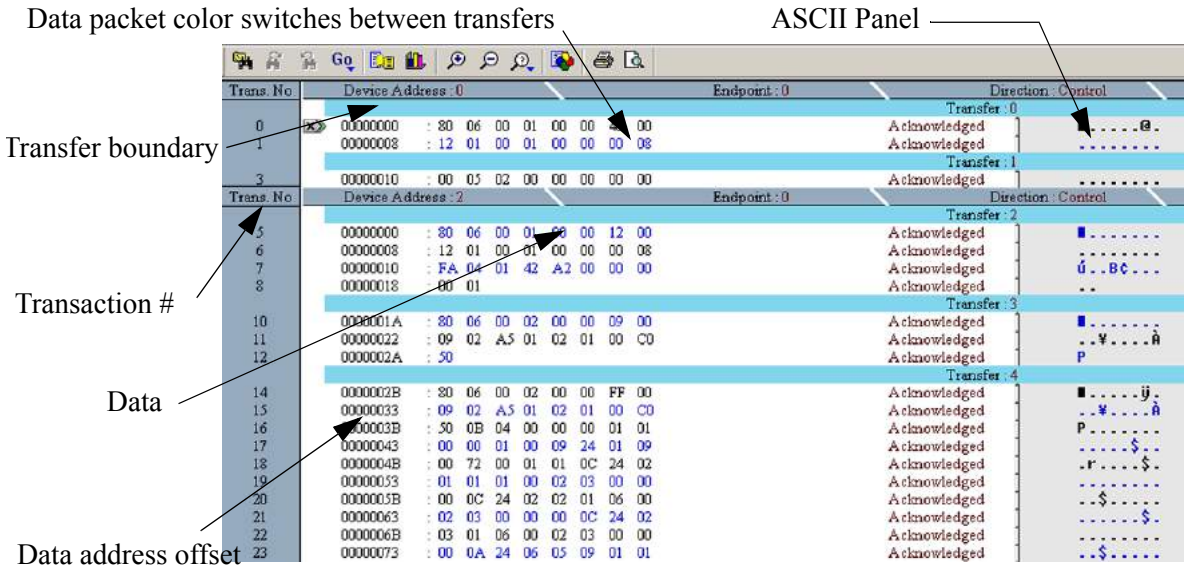


Figure 75 Data Report Hierarchy

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.



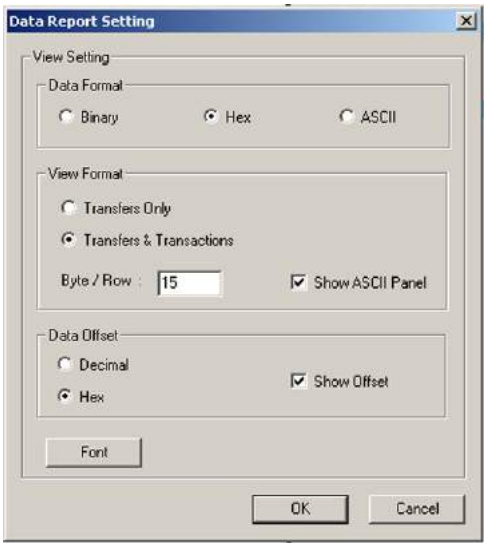


Figure 76 Data Report Setting Dialog

Data Format

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

View Format

You may chose to display Transfers Only by checking the **Transfers Only** button and set the **Byte/Row** to a value that matches the endpoint size of your device to allow a transaction to be on a single line without wrapping. Check the **Show ASCII Panel** in cases where you would like to display an ASCII interpretation of notes.

Data Offset

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

View Data Report Statistics



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

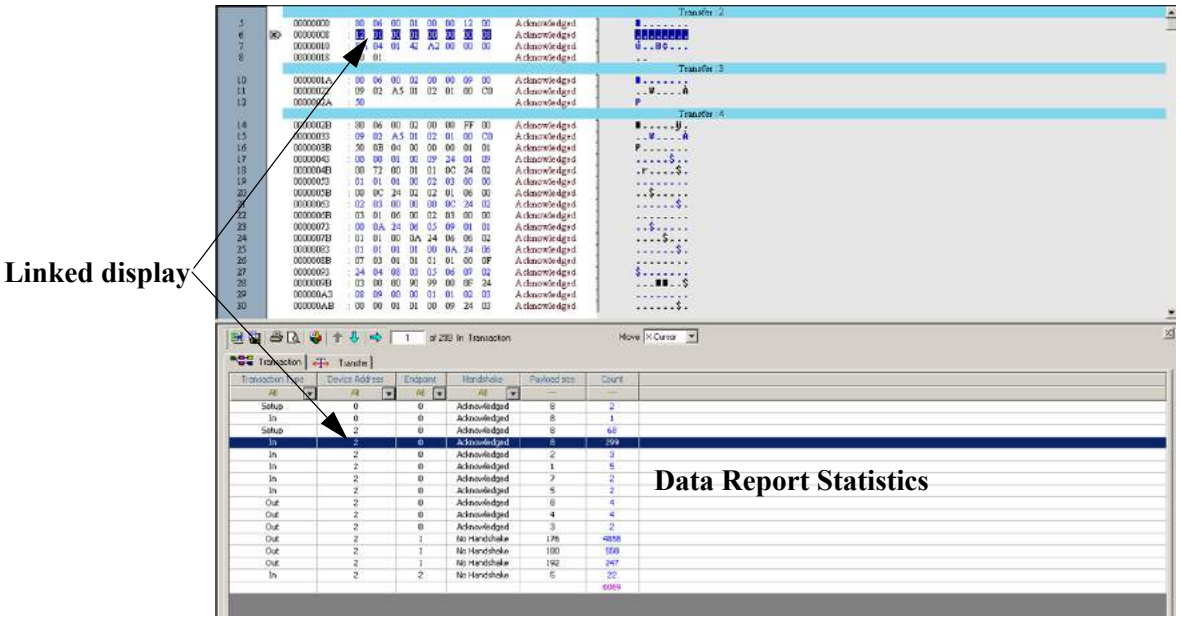


Figure 77 Data Report Statistics Displayed

Linked Displays

Clicking on 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 may 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.

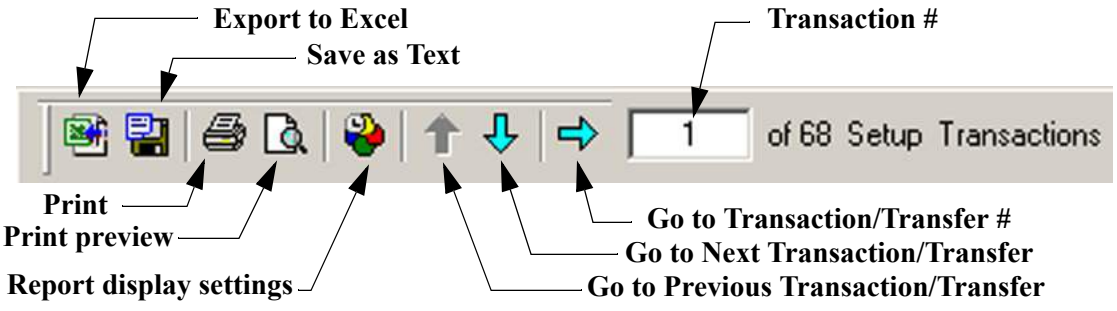


Figure 78 Data Report Statistics Toolbar

Customize the Display

You may select for display, only those items of specific interest for a specific project. To set up the display, click the Report Display settings button on the Data Report Statistics Toolbar.

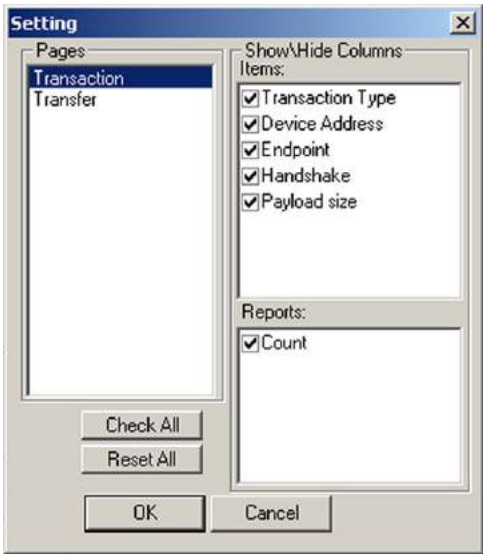


Figure 79 Report Display Settings Dialog

Uncheck the items that you wish to exclude from the display.

Display Histogram



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

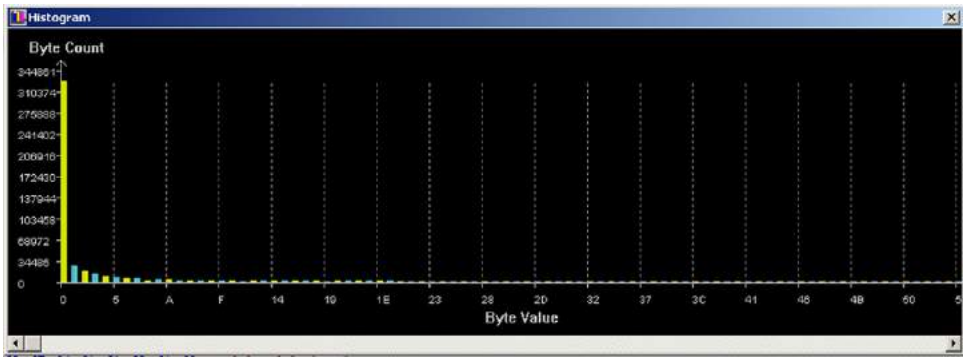


Figure 80 Data Byte Value Histogram

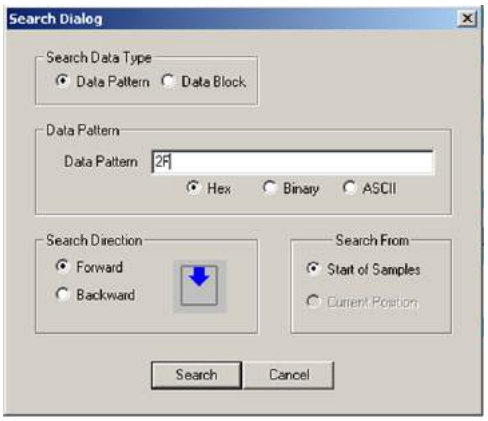
Position the mouse cursor over byte value to display a numeric message for that byte value.

Search a Data Report for a Pattern

Whenever a data report is displayed, you may quickly search for a data pattern in Hexadecimal, Binary or ASCII or a pre-defined data block.

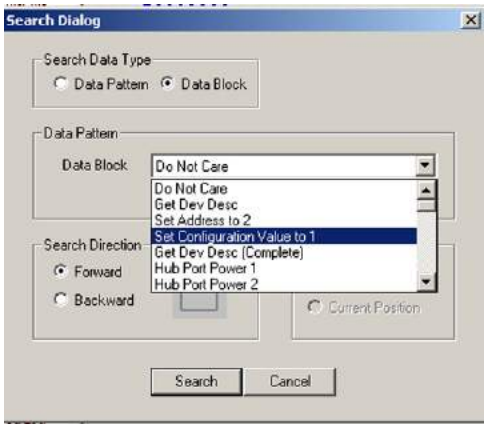


To search for a data pattern or a data block in a data report, click the search button on the toolbar to open the Search for Data Pattern dialog.



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 to search for.



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

Interpret Data

You may 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.

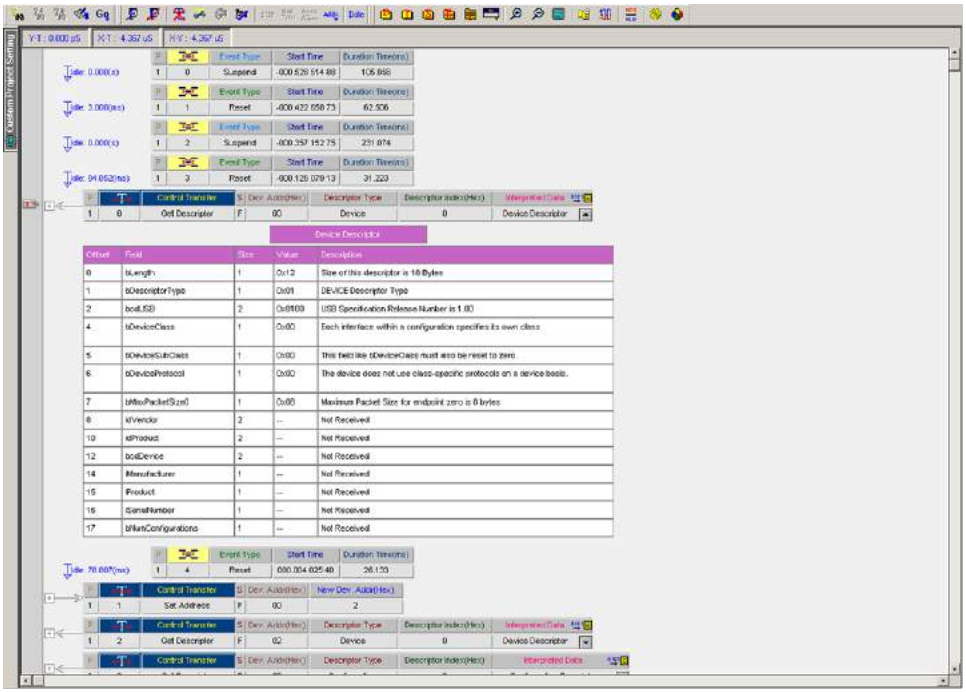


Figure 81 Interpret Data Configuration Display

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 the 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.

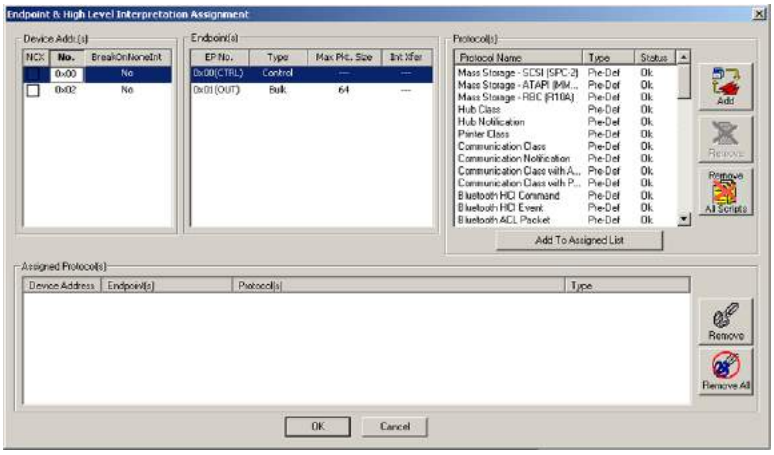


Figure 82 Upper Level Interpretation Assignment Dialog

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

To assign 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.**



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

Note: To Show/Hide Non Control Transfers you must check the NCX box.

NCX Check box



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

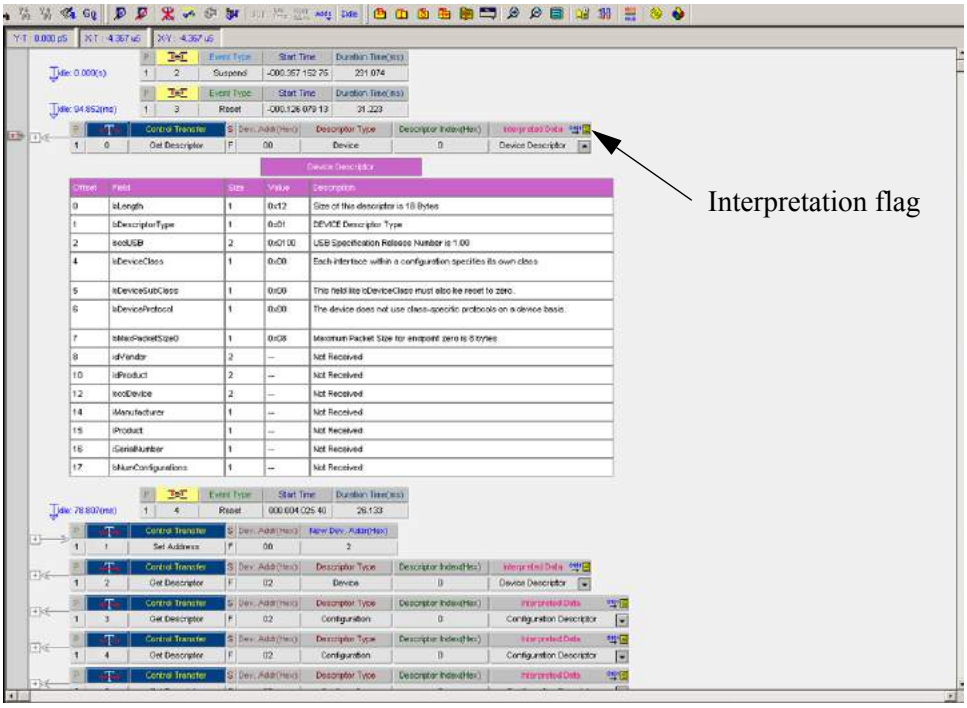


Figure 83 Upper Level Interpretation Result

User Defined Decodes

This feature allows you to create custom Upper Level interpretation tables by using a text based scripting language and save them as *.asl files. For a syntax definition of the scripting language see “Appendix A” on page 181.

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



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

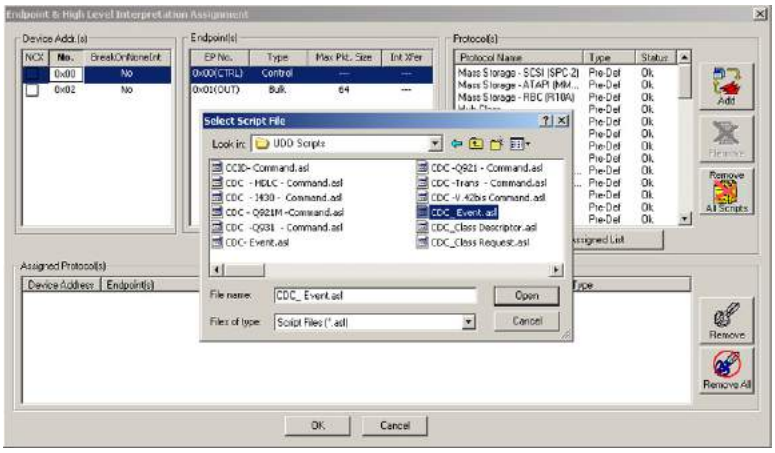


Figure 84 Choosing a Custom Script File

Choose the desired script file from the Select Script File dialog and click **Open** to add this file to the Defined Protocols list. Repeat as necessary for additional protocols.

You may now assign this protocol as described on page 90.

Note: There are 2 types of High Level decodes in the Assigned Protocols window:

1. The pre-defined Protocols that are included with your system.
2. The custom protocols that you create using the ASL script.

Errors in User Defined Scripts Any errors in user defined scripts will be logged as a result of script file parsing the first time that the script is run on the system. An appropriate error message will be displayed and a text file of errors will be generated and saved in the same folder as the source script file for your review.

Protocol Errors

The SBAE-30 monitors and captures a variety of real time protocol errors every time that a data capture is made. Additionally, the SBAE 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

Post Process Protocol Errors Detected

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 keep alive event is expected.

Err17 Packet Size is big: This Error indicates that the total number of bits received for a given packet exceeds the expected number.

Err18 Truncated Packet: This Error 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 in this speed

Err20 - Err31: Reserved for future use.

Transaction Protocol Errors Detected:

Err32 Invalid Idle after Packet: This error indicates that the required Idle time after a packet was less than expected.

Err33 Timeout after Packet: This error indicates that the required Idle time after a packet was more than expected.

Err34 Reserved

Err35 ACK not found: This error indicates that required ACK handshake was not received in this transaction.

Err36 Data Payload Error: This error indicates that expected 8 byte data payload for setup transaction was incorrect.

Err37 DATA0/1 not found: DATA0/1 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. i.e Transaction has not been Completed successfully.

Err40 Complete Split has smash: Complete Split transaction of this Split has smash. i.e Transaction has not been Completed successfully.

Err41 Endpoint Halt has occurred: Endpoint halt occurs when 3 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

The amount of data to be captured before and after the trigger may be set as a percentage of pre-trigger, between 1% and 99%. This may be done by positioning the pre-trigger slider to the desired percentage. This feature allows the 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 85.

Pre-trigger Data is 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 number pre-trigger event data can be stored. In these cases the data display will show fewer than the specified data points prior to the triggering event.

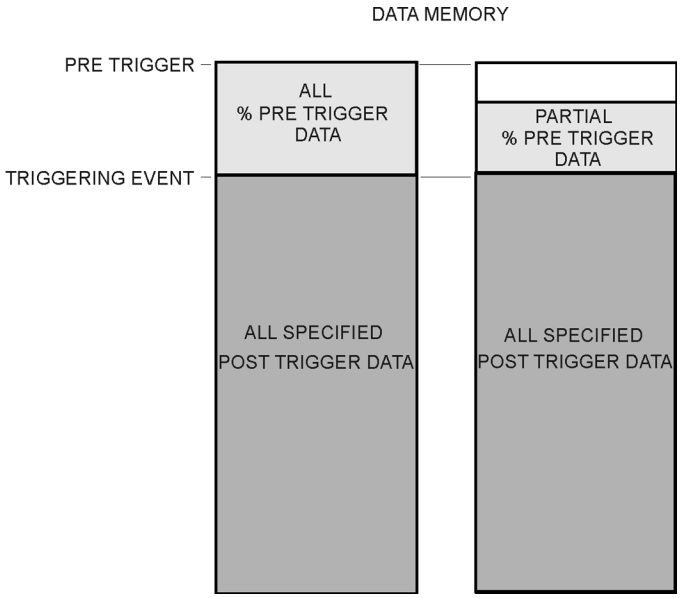


Figure 85 Pre-Trigger Example, 20% Pre-Trigger

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

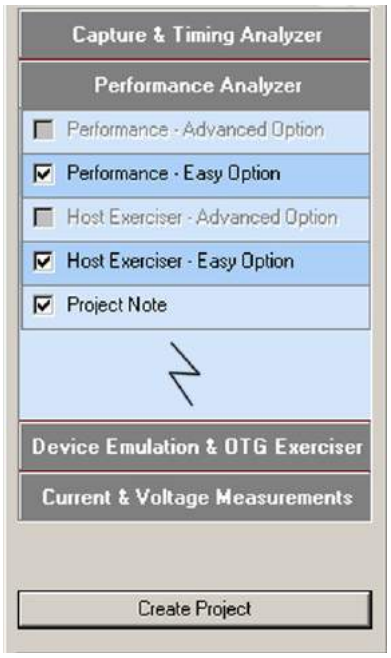
Performance Analysis

Your SBAE-30 offers you 2 modes of operation:

1. An Easy Mode for immediate Performance Analysis with a minimum of setup.
2. An Advanced Mode allowing you to program a custom performance analysis for your specific needs.

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 may perform a performance analysis on normal bus traffic or program the exerciser to measure performance with exerciser generated data.



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

1. Click the Performance Options tab to display the Performance Options Dialog Box.

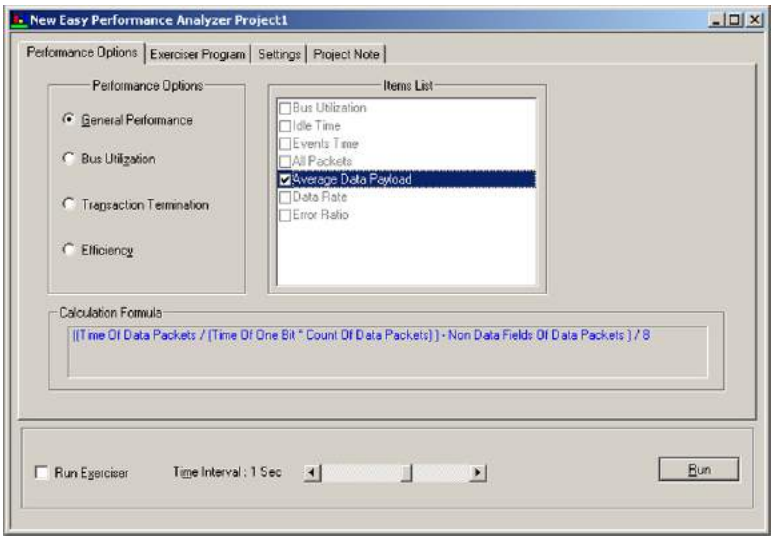


Figure 86 Performance Analysis Options Dialog Box

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

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

3. To perform an immediate Performance Analysis on bus traffic click **Run.**
4. To perform a Performance Analysis using the exerciser, click the Exerciser tab to open the Exerciser program dialog box.
5. Program the Exerciser as described in “Programming the Exerciser in Easy Mode” on page 43 or “Programming the Exerciser In Advanced Mode” on page 60 to match the Exerciser mode selected.

Run Exerciser Make sure to check the Run Exerciser check box.

6. Click the Settings tab to open the settings dialog box and click the Infinite option button to loop continually or the Number option button with a specified number of times that you wish the program to run.

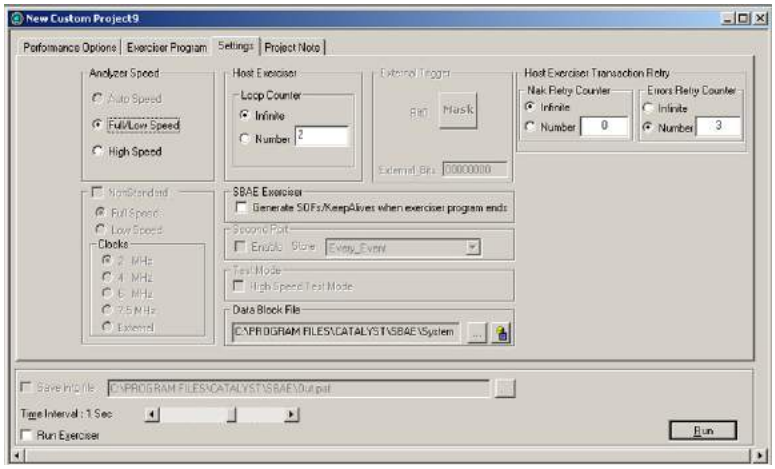


Figure 87 Settings Dialog Box

7. Click **Run**

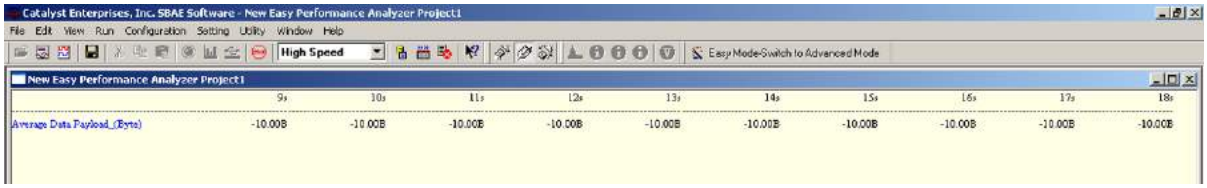


Figure 88 Performance Analysis Result

Performance Analysis (Advanced Mode)

Your SBAE-30 offers you 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 are never stopped to be read.

Advantage	Continuous Real Time measurement.
Disadvantage	Measurement results are based on an average. Minimum and maximum cannot be reported in this mode.

Statistical Trace (not yet available) analysis is done by capturing the data into memory and then post processing it with software. In this mode the SBAE-30 allows capture to occur after an occurrence(s) of events such that the data includes mostly the desired transactions and not unrelated bus activity.

Advantage	Many more measurements are possible than in the Real Time analysis. It is possible to 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 you may open an existing project file or define a new analysis project by defining:

- Packets to be used in the analysis.
- An Exerciser program (if you wish to measure performance with specially generated bus traffic).
- An analysis Expression or expressions if multiple performance parameters are to be measured.
- Results display configuration.
- Performance analysis project options.

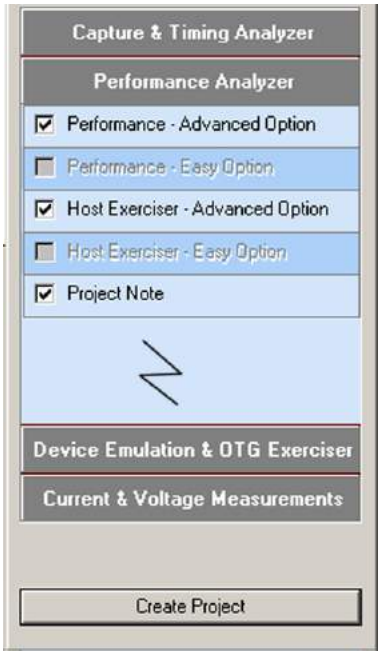
Perform a Pre-defined Analysis

1. Click **File** on the main toolbar and choose **Open.**



2. Select a pre-defined analysis file and click **Open** to load the project.
3. Click **Run** to perform the analysis.

Create a New Analysis



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

Check the **Performance - Advanced Option** and then the **Create Project** button to open a Performance Analysis Project.

Define Packets

Make sure that the Packets tab is selected.

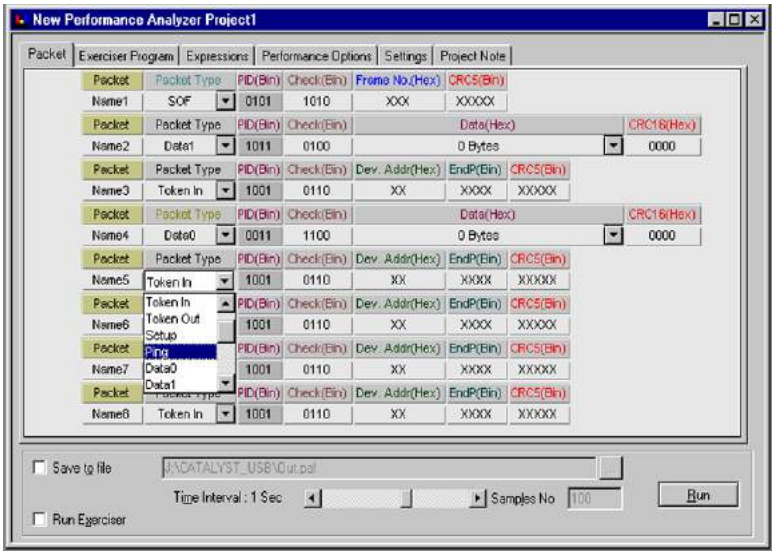


Figure 89 Performance Analysis Dialog Box

Define the packet types to be used in the analysis from the Packet Type list box for each packet to be used. See “Defining Packets” on page 50.

Program The Exerciser

If the exerciser is to be used then click the **Exerciser** Program tab. Program the Exerciser as described in “Programming the Exerciser in Easy Mode” on page 43 or “Programming the Exerciser In Advanced Mode” on page 60 to match the Exerciser mode selected. Be sure to check the **Run Exerciser** check box.

Creating Analysis Expressions

Your Analyzer includes 4 counter and 4 timers that may be used to create Analysis expressions. To create an analysis expression you must first assign a packet, event or a logical expression of them to a counter and/or timer that you wish to include in creating the Analysis expression. To perform the assignment:

1. Click the Performance Options Tab to view the Performance Options setting dialog box.

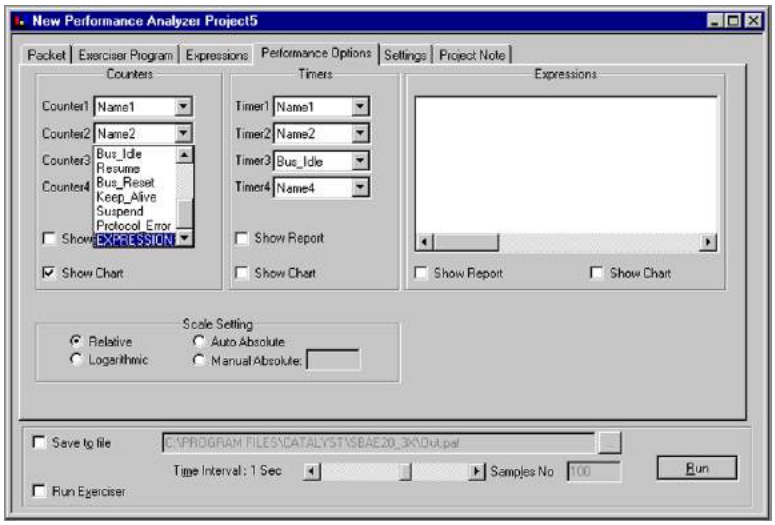
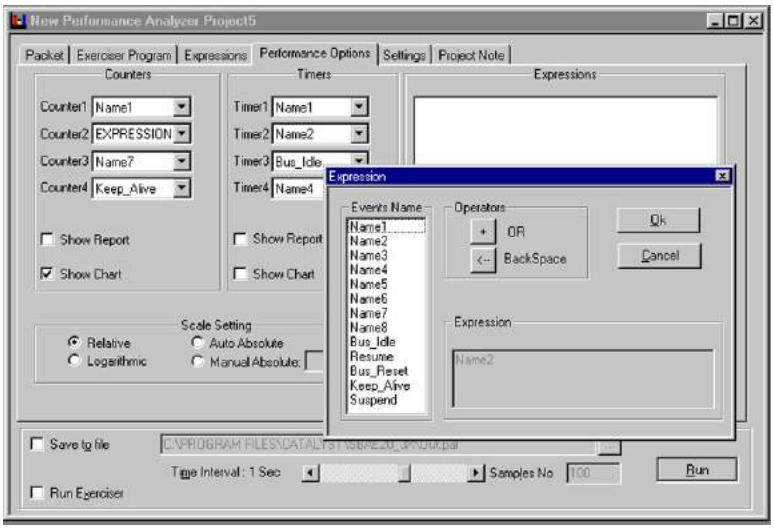


Figure 90 Performance Options Setting Dialog Box

2. Click the down arrow next to each desired 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 on the Name that you wish to include in the logical expression, click the OR operator and then double-click the next name for the expression and when done 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 the desired parameters.

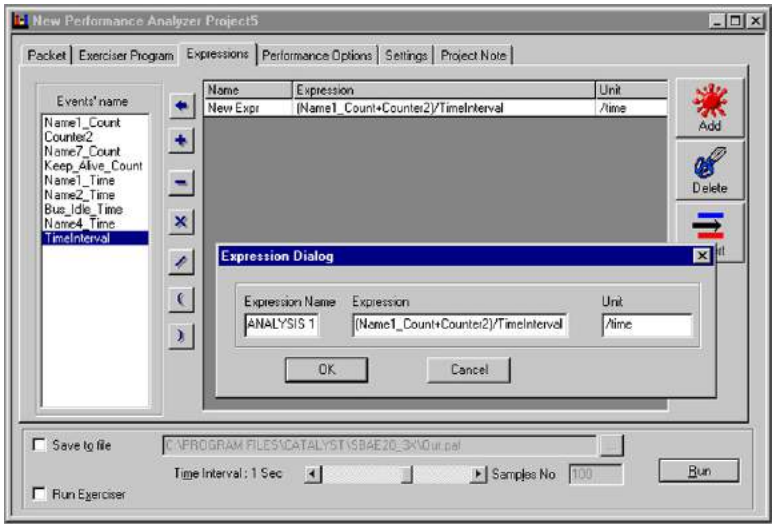


Figure 91 Creating an Expression

Create an expression Click **Add** button to open a blank expression line. You may enter the expression by double clicking on the Event's name and the required operator or by double clicking on the blank expression line to open the Expression Dialog and typing the required expression directly.

Name the Expression

Double click the New Expression name field and enter the desired expression name in the Expression dialog and click OK.



Repeat for additional expressions as required.

Choose Results Display

Click the Performance Options Tab to view the Performance Options setting dialog box.

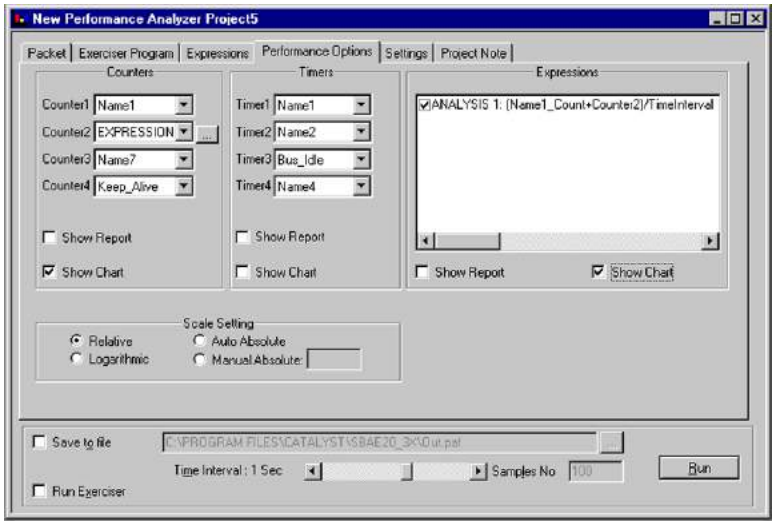


Figure 92 Choosing Display Options

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

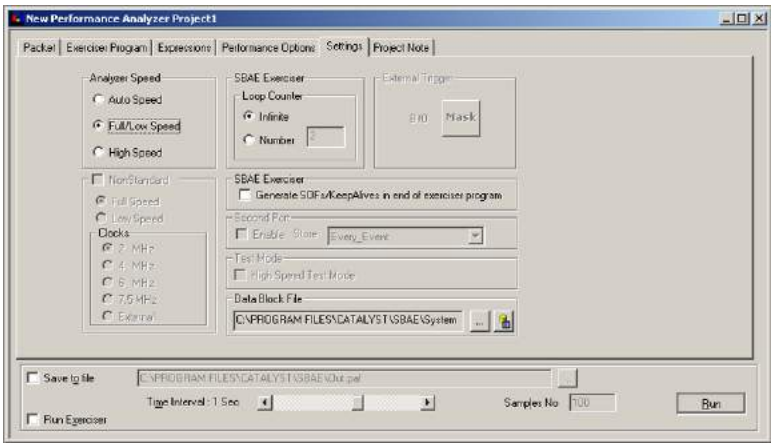


Figure 93 Performance Analysis Project Settings

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

Choose Time Interval Drag the Time interval scroll bar to set the time interval for your 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.

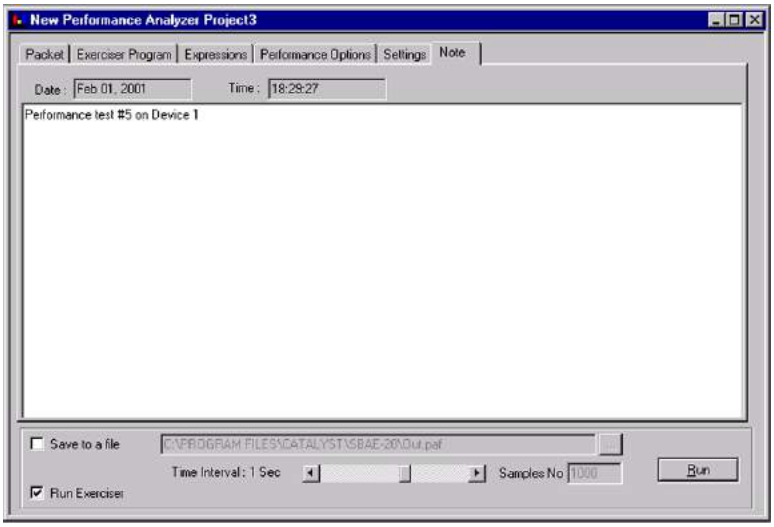


Figure 94 Project Annotation Dialog Box

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 will be 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.

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

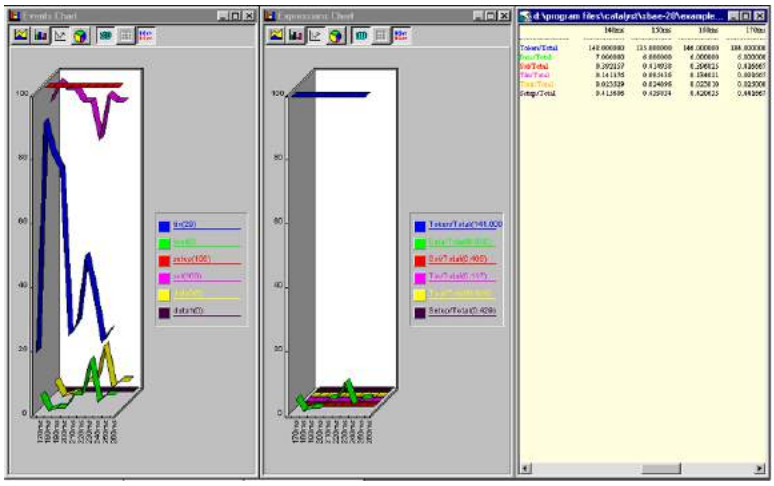


Figure 95 Performance Analysis Real Time Display



To stop the performance analysis, click the **Stop** button on the main toolbar.

Alternate Display Format You may choose to display the result as 2D, 3D, etc. by clicking the corresponding "Graphics Setting" on the Performance Analysis display toolbar.

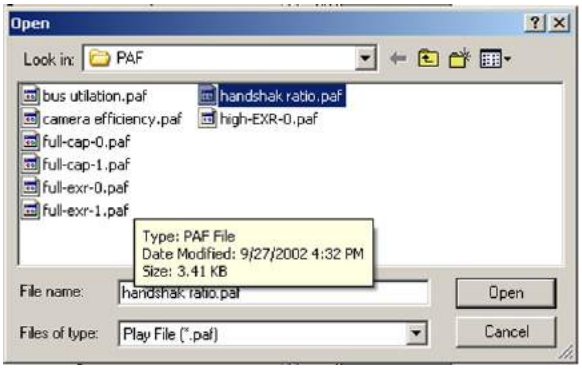


Saved Performance Analysis Review

The SBAE-30 offers you the ability to review the results of previously performed and saved performance analyses.



Click **View** on the toolbar and then select **Performance Analyzer Review** to open the **Open** dialog box.



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

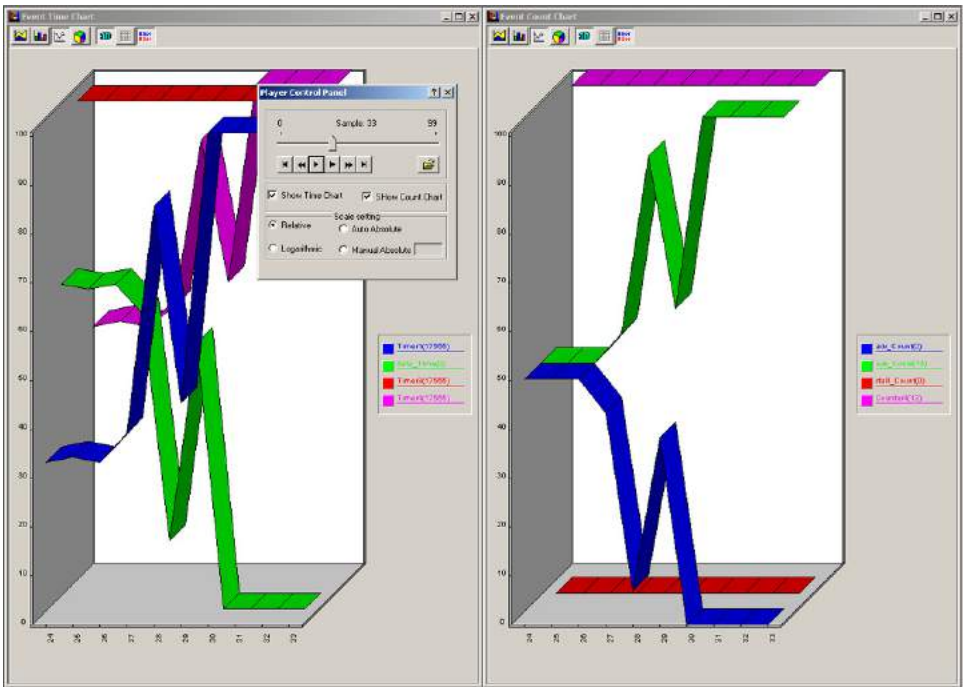


Figure 96 Review of Previously Saved Performance Analysis

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

Timing Analysis (Optional)

High Speed Timing Analysis is not available in SBAE-30. It is available in SBAE-30A. For SBAE-30A differences see "SBAE-30 Versions" on page 1

Easy Mode Timing Analysis

You may 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 97.
3. Click the **Run** button.

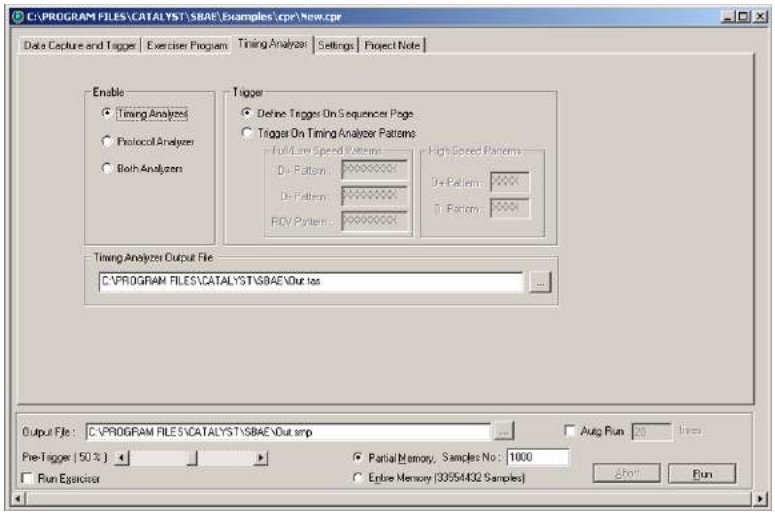


Figure 97 Enabling Timing Analysis

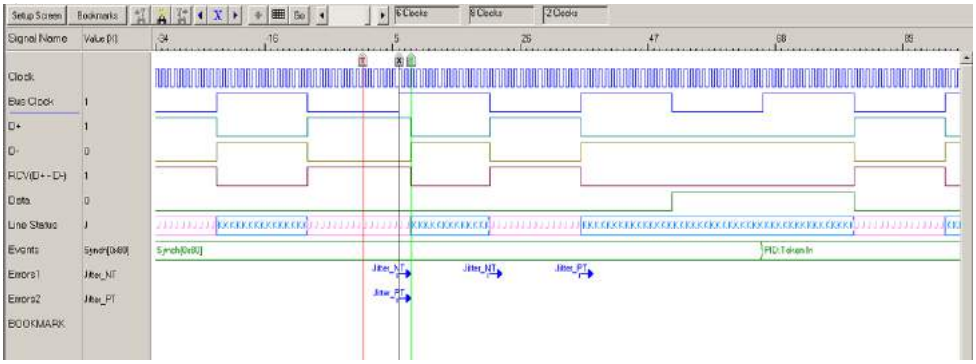


Figure 98 Timing Analysis Result

Using Cursors

You may use the cursors to make measurements within the display. For a detailed description of using the cursors see “Using the Cursors and Bookmarks” on page 165

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 may 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 99.
3. Click the **Run** button.

RCV in Full/Low Speed only The RCV Pattern specification is not available in High speed mode.

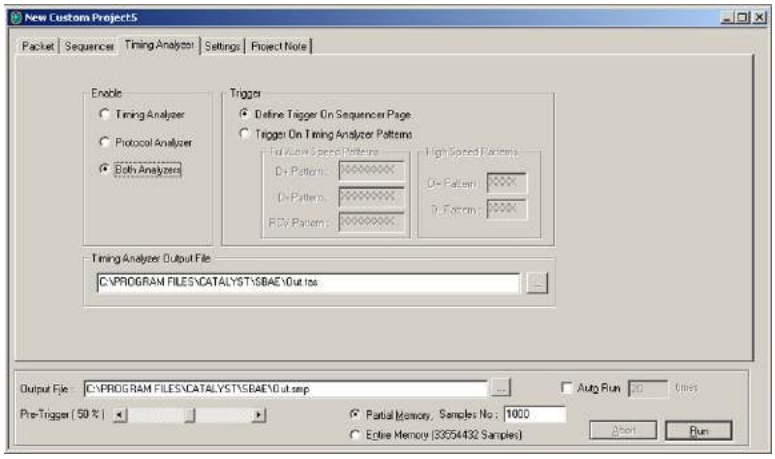


Figure 99 Advanced Mode Timing Analysis Dialog

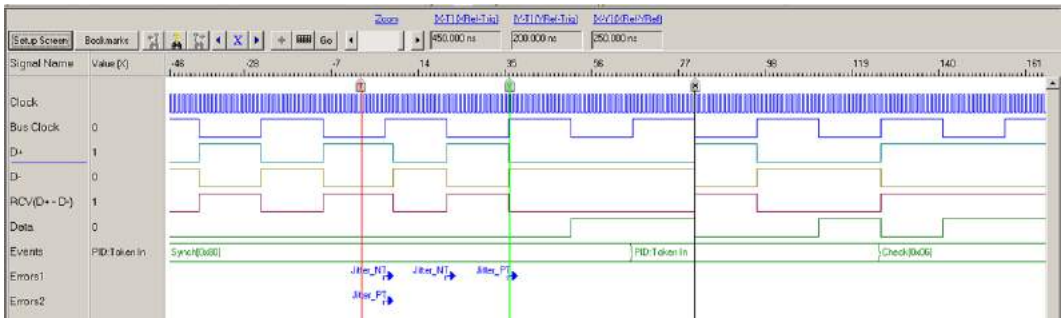


Figure 100 Timing Analysis Result

Results Display

You may customize the results display for your needs by adding or removing signals from the display and use 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 163 for filtering see "Filter" on page 160 and for using cursors and bookmarks see "Using the Cursors and Bookmarks" on page 165.

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.

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 the SBAE-30 as a Device for testing your USB development software. The SBAE-30 may be defined to be a Low, Full, High or a High/Full speed Device. Each Device may include up to 3 configurations with 2 interfaces each. A total of 7 endpoints may be assigned to these interfaces. Figure 101 shows a functional connection for the SBAE-30 operating as a Device.

Your SBAE-30 allows 2 ways to set up as a device:

1. Easy Mode, requiring a minimum of setup.
2. Advanced Mode offering more powerful device emulation by allowing the user to define custom responses to different commands.

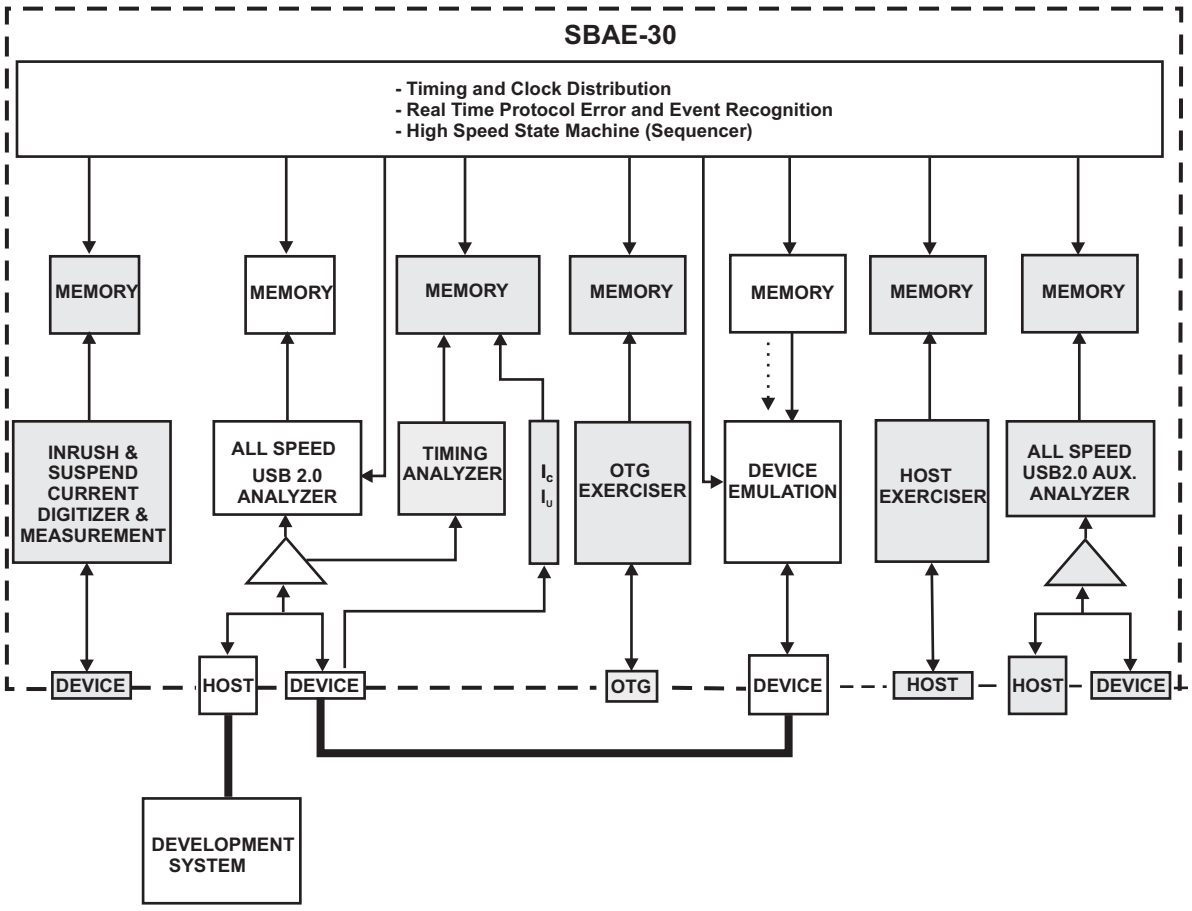
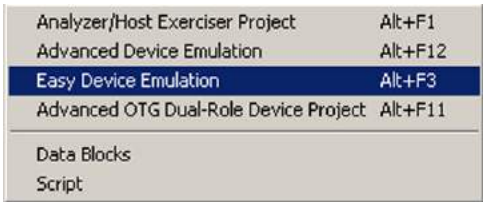


Figure 101 SBAE-30 Operating as a Device Functional Connection

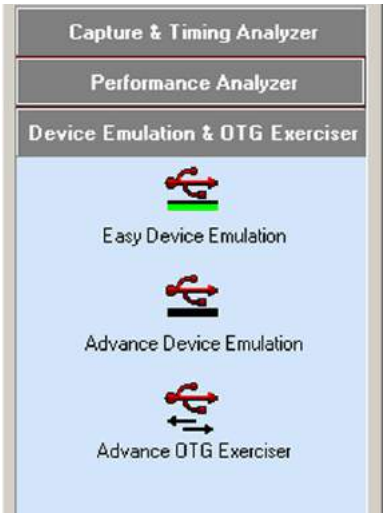
Device Emulation (Easy Mode)

Programming the Device

To open a new Easy Device Emulation project click **File** then **New** and choose **Easy Device Emulation**.



You may also open a default Easy Device Emulation Project.



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

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

The **Device Emulation** dialog opens with a default hierarchy as shown in Figure 102. You may add configurations to total 3. Each configuration by default opens with one interface. You may add interfaces to total 2 per configuration. Each interface by default opens with one Alternative Setting with one endpoint. You may add Alternative Settings to total 4 for each interface, but no more than 7 total for the Device. A Device is limited to a total of 7 endpoints. Descriptor values can be entered at each level of the hierarchy. In addition to support for most standard requests defined for USB, the Device can be configured to respond with up to 4 arbitrary standard request data descriptors and up to 7 String Descriptor strings, and acknowledge up to 8 class requests.

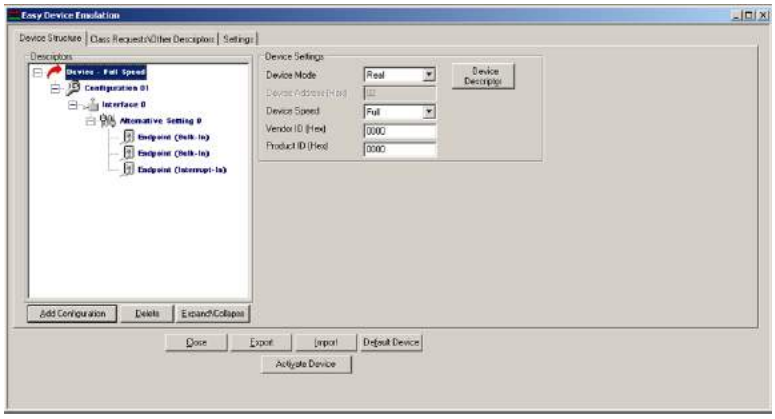


Figure 102 Device Mode Setup Dialog

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 two Devices must be set up independently.
2. Enter Vendor and Product ID in the appropriate text box.
3. Define Device Descriptors. Click the **Device Descriptor** button and enter the parameters appropriate for your Device in the **Descriptor** dialog as shown in Figure 103.

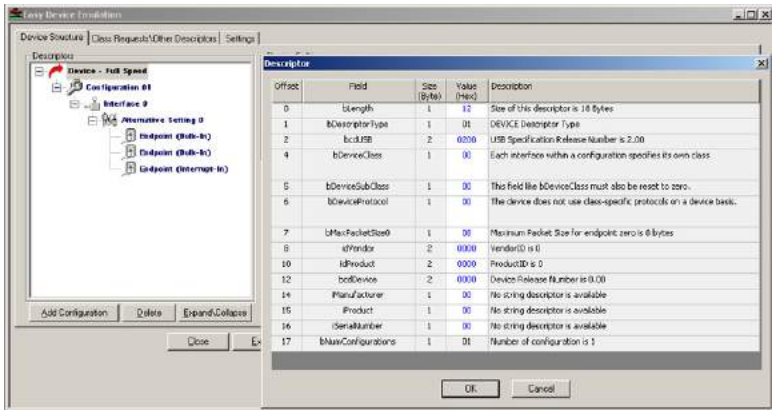


Figure 103 Defining Device Descriptors

Define Configuration A Device initially opens with 1 configuration. You may add additional configurations up to a total of 3 by clicking on a Device and click the **Add Configuration** button.

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

Enable Configuration Class Descriptor Clicking this checkbox will return 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 may add additional interfaces up to 4 per configuration, but not exceeding 7 total for a Device by clicking on a configuration and then the **Add Interface** button.

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

Enable Alternative Class Descriptor Clicking this checkbox will return additional data as part of the Alternative Descriptor. (Up to 128 bytes)

Define Endpoint A Device allows up to a total of 7 endpoints under defined alternative settings. You may add endpoints by clicking on an alternative setting and 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 104.

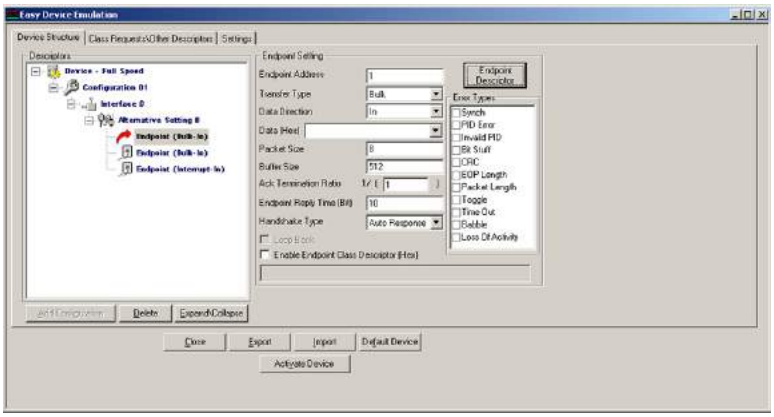


Figure 104 Endpoint Definition Dialog

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 make an appropriate choice.

Packet Size Enter a desired packet size (Bytes/packet) to be returned by the Device. (Default is 64). When data is returned to the Host, it is sent in N bytes/Packet increments.

Buffer Size Enter a desired 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 the desired delay time between the Host packet and the SBAE-30 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 SBAE-30 Device for the selected endpoint.

Error Types Check the error type boxes from the available errors listed in the **Error Types** group box for the errors that you would like to be introduced.

Loopback Checking this option for OUT endpoint will cause the data received from the Host for the OUT endpoint to be returned to the Host on the IN endpoint with the same endpoint number. The IN endpoint is automatically selected when Loopback is checked. There is no need to define another IN Endpoint, 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 will respond with a NAK.

Enable Endpoint Class Descriptor Clicking this checkbox will return additional data as part of the Endpoint Descriptor.

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

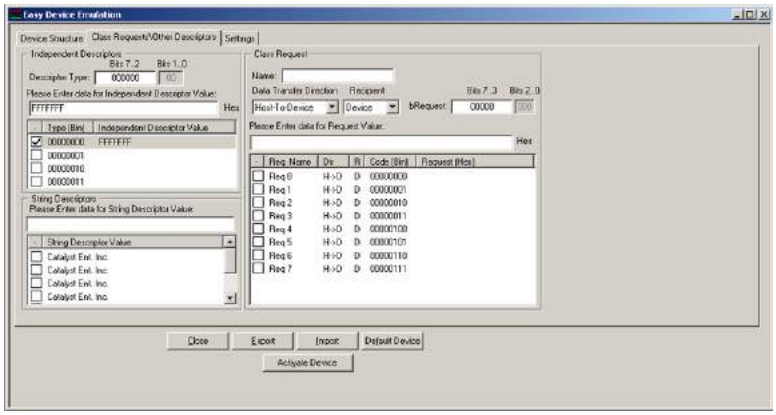


Figure 105 Additional Requests Response Definition

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 be returned in response to a Get Descriptor request. Responses will support requests with index 0.

Note: Only the 6 most significant bits of the Descriptor Type may be set by the user. **Do not**

set the descriptor type to the same value as any standard USB descriptor.

String Descriptors To define String Descriptors enter a desired 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, enter the data for the Request Value. Choose the Data Transfer direction and Recipient.

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

Non Standard Clocks In cases where you would operate the device at a non-standard clock rate, click the **Settings** tab to display the clock selection dialog.

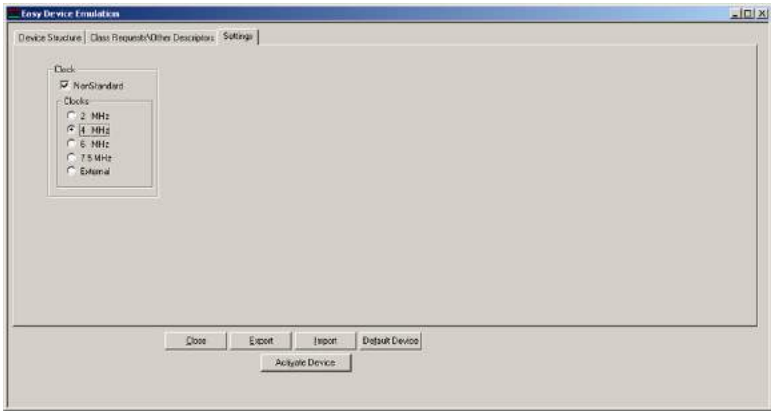


Figure 106 Using Non Standard Clocks

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 if desired check **External** and apply an external clock.

You may 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 may close the dialog, perform other tasks and then activate the device.

Note 2. If you wish 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.

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 31) or Advanced mode (see page 50) and click **Run**.

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

Endpoint errors generated by OTG DRD & Device Emulation

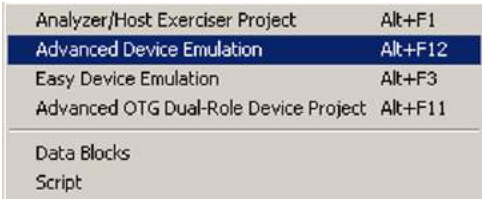
Synch:	Generates the KJKJKJJJ pattern for Synch field
	Not available in HS
Pid:	Check field contains the same bits as PID field
Invalid Pid:	Uses Reserved PID field value of 0000b and Check field value 1111b
Bit stuffing:	Disables bit stuffing
	Not available in HS
CRC:	Incorrect CRC for data packets
	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 returned to idle at end of packet
	Not available in HS

Device Emulation (Advanced Mode)

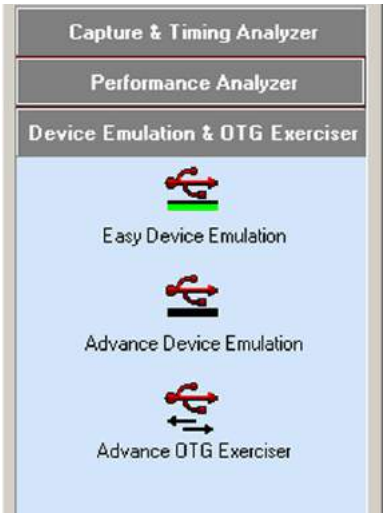
Programming the Device

In order 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 click **File** then **New** and choose **Advanced Device Emulation.**



You may also open a default Advanced Device Emulation Project.



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

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

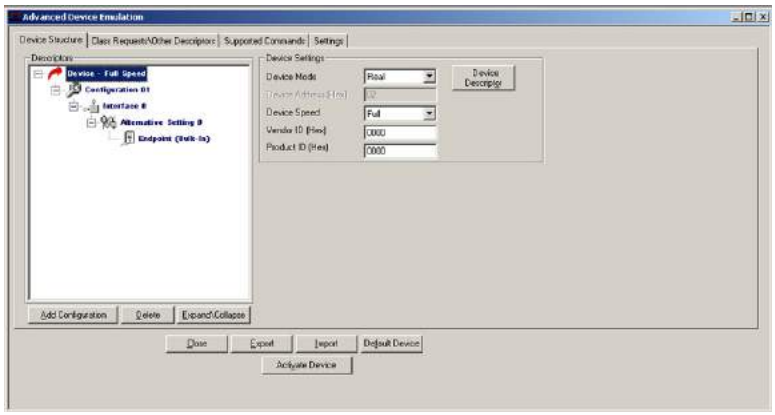


Figure 107 Advanced Device Mode Setup

Device structure and supported class requests are defined in the same manner as Easy Device Emulation as described starting on page 117. **Note**, however that all of the endpoint behavior settings are grayed out because the endpoint responses are programmed in the supported commands page. The Command sets are defined 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, a data stage which 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 will occupy 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 will be 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 108.

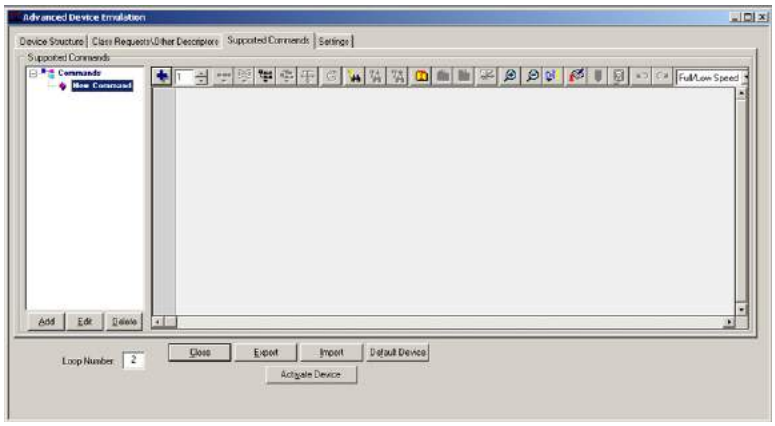


Figure 108 Supported Commands Entry Dialog

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

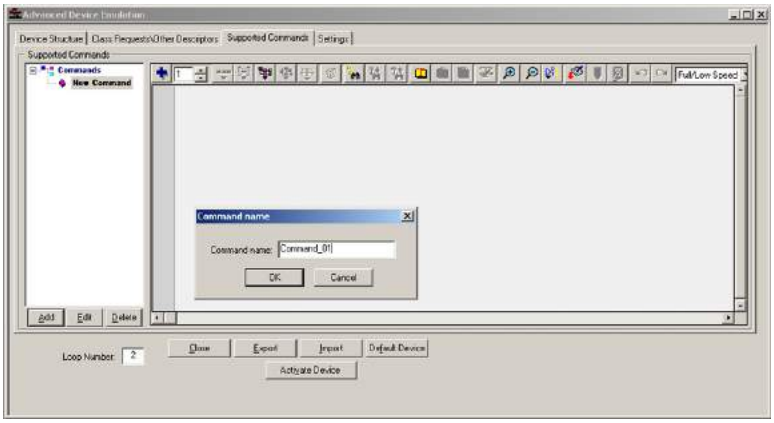


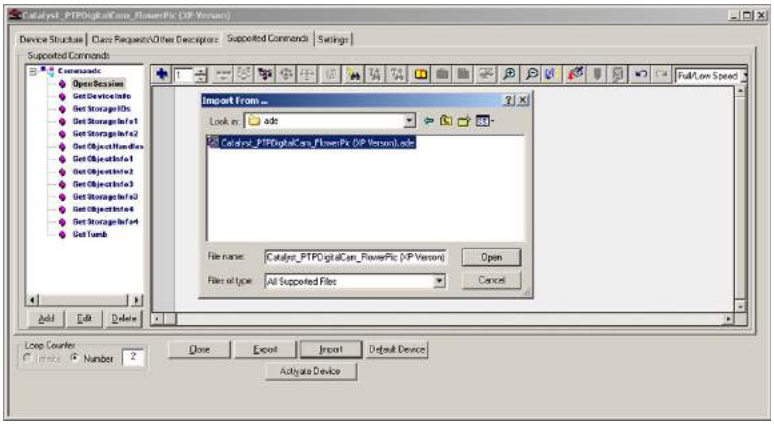
Figure 109 Adding a New Command

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 the desired value.
8. Complete the data and status stage of the command set by adding addition in or out transaction to any endpoint.
9. You can define additional command sets by repeating the above steps.

Note If you wish 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.

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.

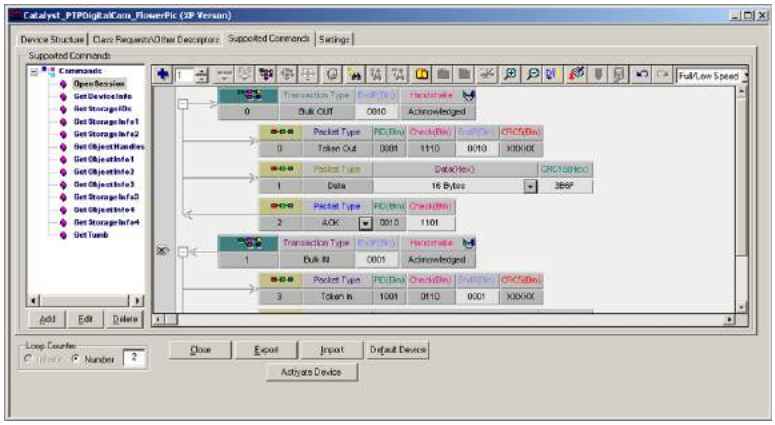


Figure 110 Previously Defined Command Set



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 111. To operate as an A-Device, the user needs to plug in the Mini-A side of the cable into the OTG exerciser port.

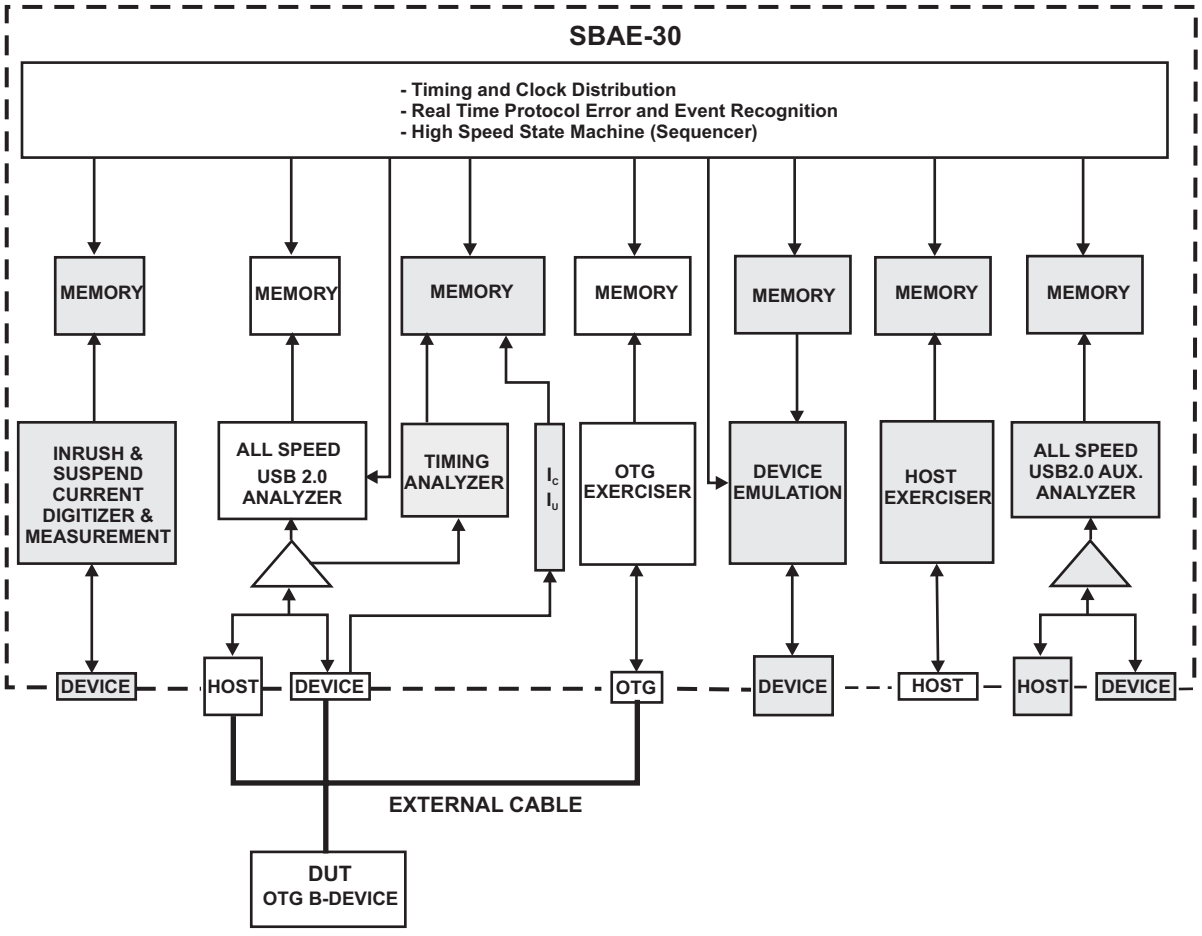
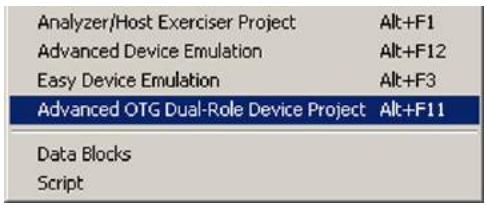


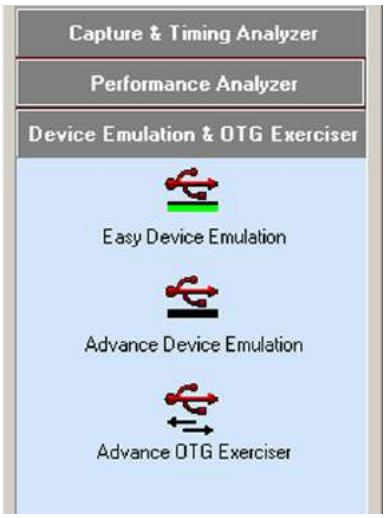
Figure 111 OTG Exerciser as DRD Device A-Device Connection

Programming the Device

To open a new Advanced OTG Dual-Role Device Project click **File** then **New** and choose **Advanced OTG Dual-Role Device Project.**



You may 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 on Device Emulation & OTG Exerciser.

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

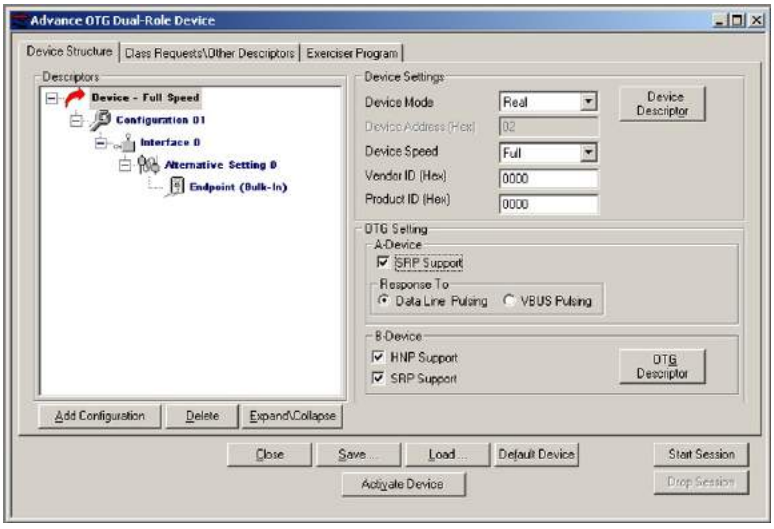


Figure 112 OTG DRD Setup Dialog

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 60. 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 112).
3. The OTG Exerciser loop counter is set in the Exerciser Program dialog.

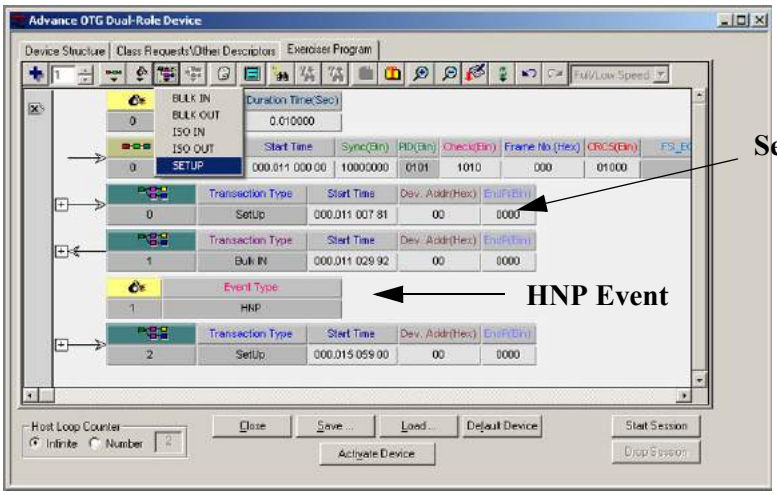


Figure 113 Inserting a Setup Transaction

Note: To enable HNP in the exerciser program, you must incorporate program lines as shown in Figure 113. The exerciser code example shown in Figure 113 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 will become the host and the exerciser will become the peripheral. Additional exerciser code inserted after the lines shown will be executed when the OTG Exerciser switches back to being a host. The SetFeature(b_hnp_enable) request is defined in a Data Block that can be used as data for the Setup Transaction.

Both, Bus Idle and HNP will cause the bus to suspend. However, for HNP, if the OTG exerciser detects a disconnect from the B-Device DUT, it will automatically disconnect, 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 the user may wish to use the Host Exerciser

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 116.

Differences from Device Emulation: The **OTG DRD Setup** dialog opens with a default hierarchy as shown in Figure 112. You may add configurations to total 3. Each configuration by default opens with one interface. You may add interfaces to total 2 per configuration. Each interface by default opens with one Alternative Setting with one endpoint. You may 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. Descriptor values can be entered 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 4 arbitrary standard request data descriptors and up to 7 String Descriptor strings, and acknowledge up to 8 class requests.

Note If you wish 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 the user changes any of the parameters on any of the tabs, the device must be reactivated. This is done 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. A session may be stopped 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 will be enabled in cases where the user has defined 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 114. To operate as a B-Device, the user needs to plug in the Mini-B side of the cable into the OTG exerciser port.

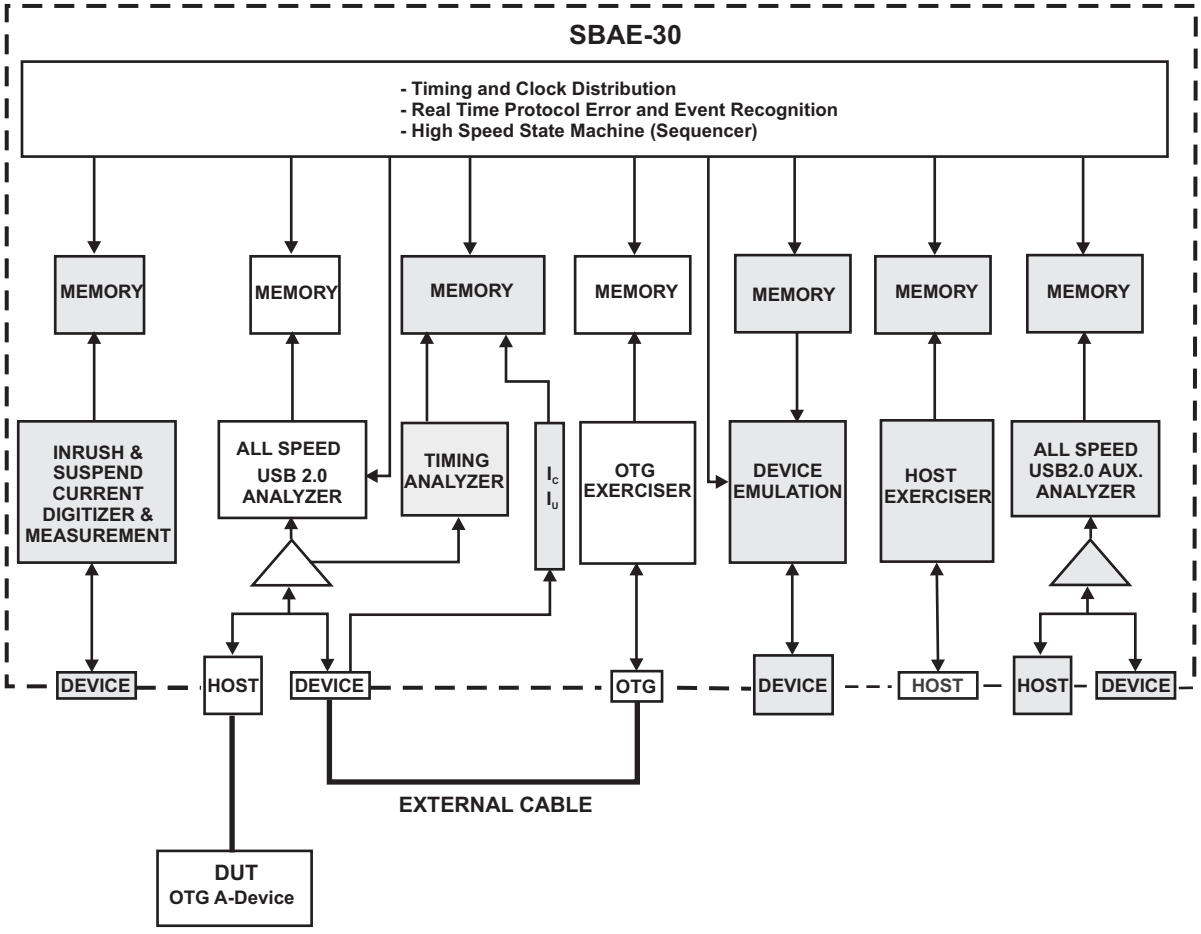
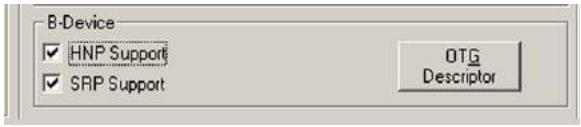


Figure 114 OTG Exerciser as DRD or Peripheral Device B-Device Connection

Define the Device

You may 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 may 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.

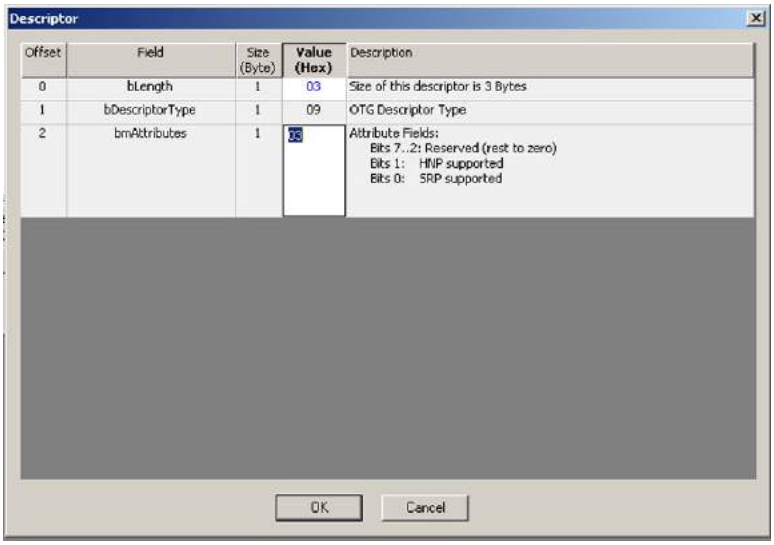


Figure 115 Selecting SRP/HRP in the OTG Descriptor Dialog

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 116.

Differences from Device Emulation: The **OTG DRD Setup** dialog opens with a default hierarchy as shown in Figure 112. You may add configurations to total 3. Each configuration by default opens with one interface. You may add interfaces to total 2 per configuration. Each interface by default opens

with one Alternative Setting with one endpoint. You may 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. Descriptor values can be entered 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 4 arbitrary standard request data descriptors and up to 7 String Descriptor strings, and acknowledge up to 8 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 60. 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 112).
3. 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: The user does 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 will automatically return control back to the A-Device.

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 will be enabled. You may now:

1. 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 the user changes any of the parameters on any of the tabs, the device must be reactivated. This is done 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, the user may 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 will be enabled in cases where the user has defined long program loops.

Current Measurement (Optional)

Current measurements are performed using the USB 2.0/1.x Analyzer, Exerciser and Inrush Current connector and require the unplugging and reconnecting the Device cable in accordance with on screen commands Figure 116 shows the functional connection used for measuring Unconfigured and Suspend current.

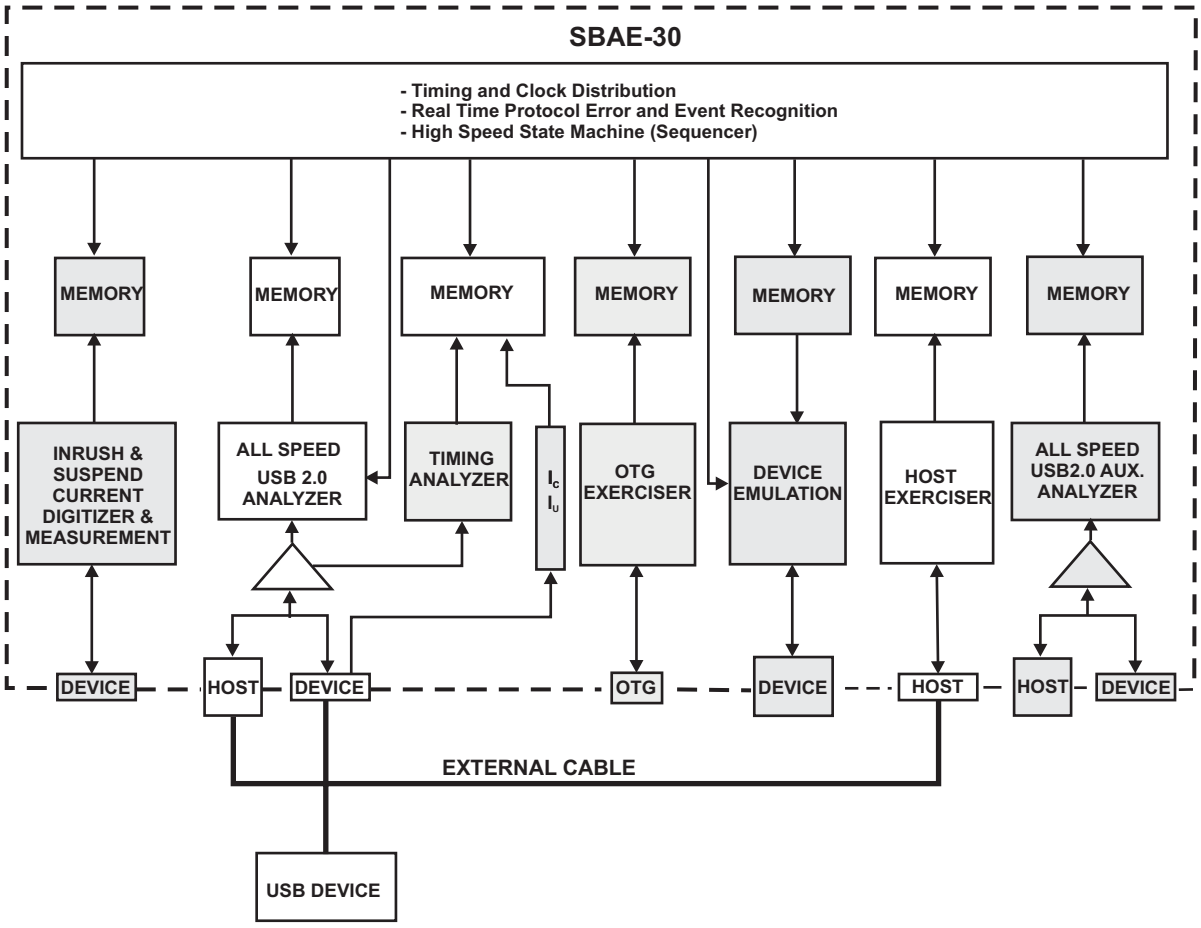


Figure 116 Unconfigured and Suspend Current Measurement Functional Connection

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 116 on page 135.

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

.



Figure 117 Unconfigured Current Measurement Dialog

If you would like a graphic display of the unconfigured current as a function of time check the **Show Chart** check box.

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



Perform the required connection and click **OK**.

The required measurement is now performed and the result displayed.

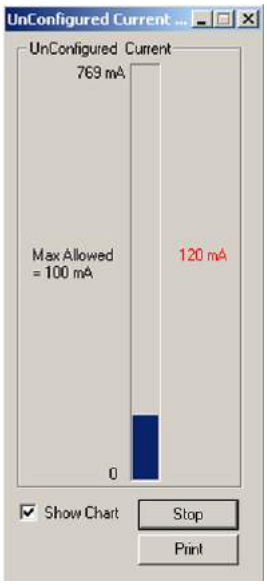


Figure 118 Unconfigured Current Measurement Results Display

Stop the Measurement If you are viewing the graphical display and would like to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

.

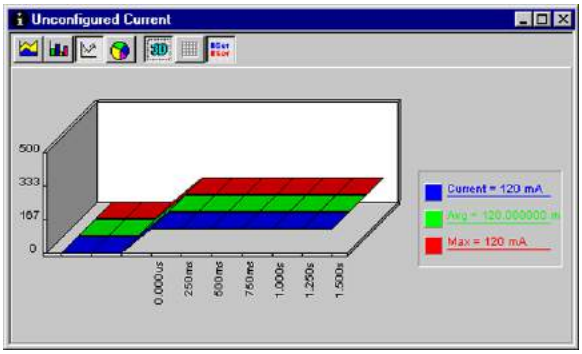


Figure 119 Unconfigured Current Chart Display

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

Operating Current Measurement

To perform this measurement the USB Device and a PC must be connected to the USB 2.0/1.x Analyzer port

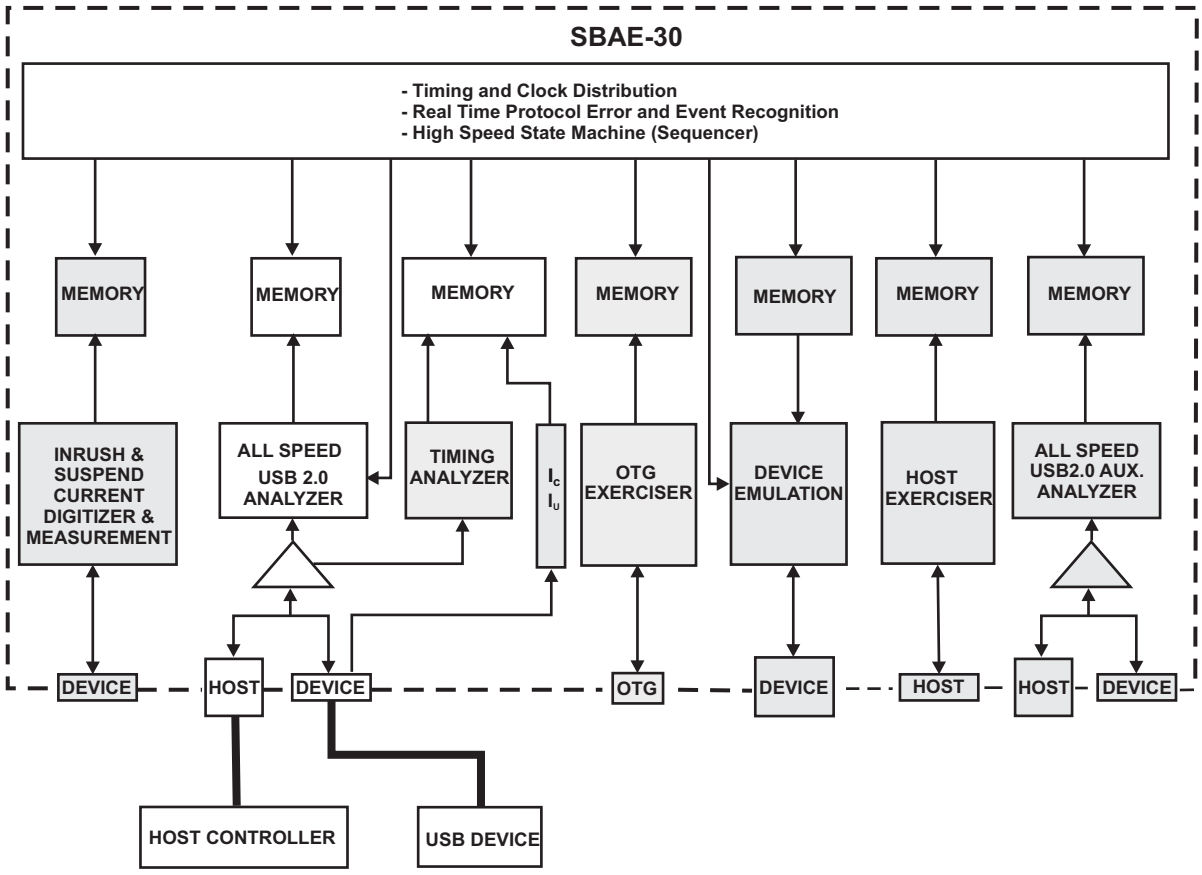


Figure 120 Operating Current Measurement Functional Connection

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



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



Figure 121 Operating Current Measurement Dialog Box

If you would like a graphic display of the suspend current as a function of time check the **Show Chart** check box.

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



Perform the required connection, click OK and wait for the measurement result to appear.

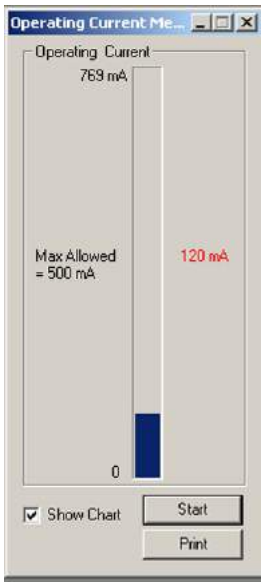


Figure 122 Operating Current Measurement Results Display

Stop the Measurement If you are viewing the graphical display and would like to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

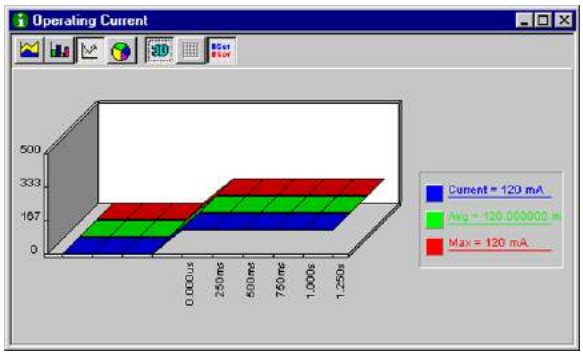


Figure 123 Operating Current Chart Display

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

Making the VBus Measurement

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



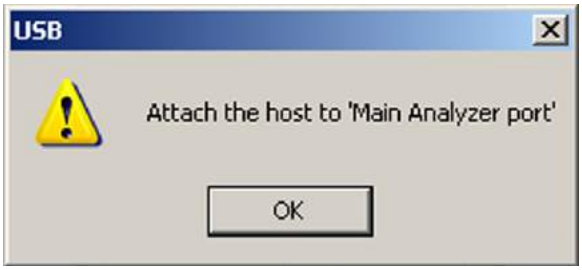
Click the blue VBus button in the **Current & Voltage Measurements** project selector to open the **VBus Measurement** dialog.



Figure 124 VBus Measurement Dialog Box

If you would like a graphic display of the VBus measurement as a function of time, check the **Show Chart** check box.

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



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

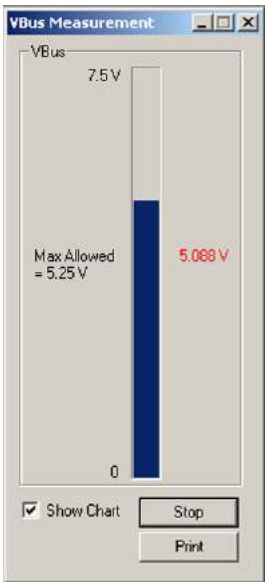


Figure 125 VBus Measurement Results Display

Stop the Measurement If you are viewing the graphical display and would like to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

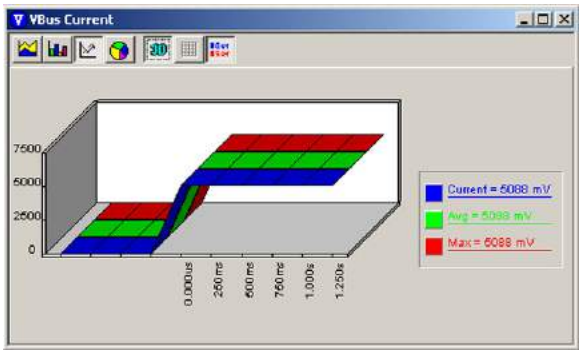


Figure 126 VBus Measurement Chart Display

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

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 127.

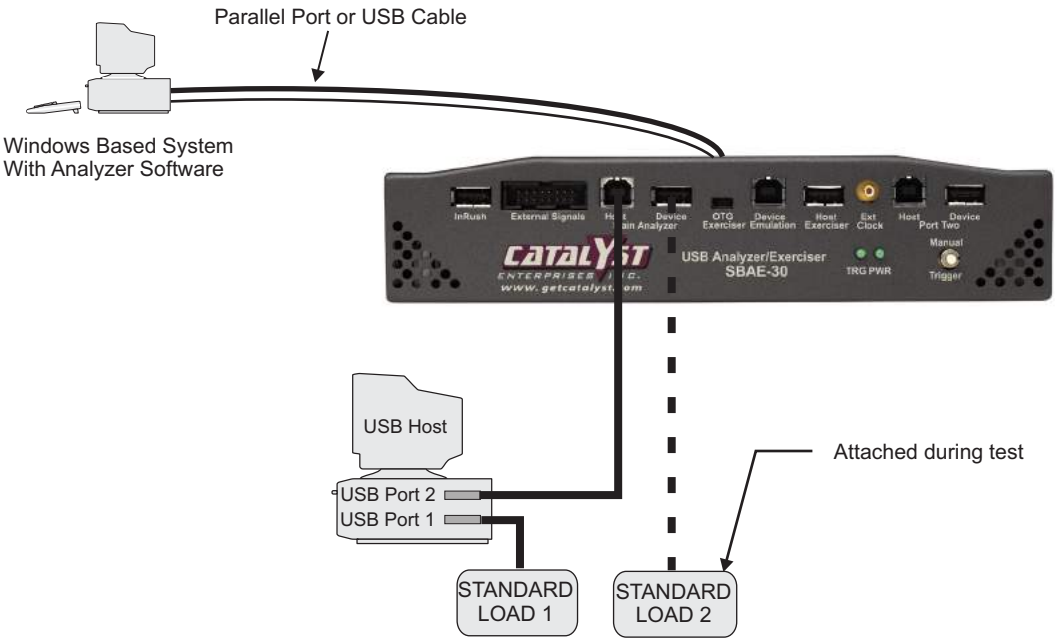
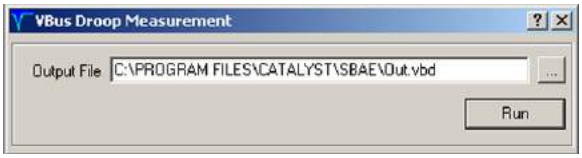


Figure 127 VBus Droop Test Setup

Performing the Measurement



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

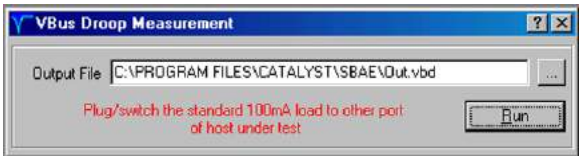


Each time this measurement is performed the result will be written to the default Out.vsb file overwriting the previously saved test. If you wish to retain the current result, click the ellipses next to the output file name and enter a new file name.

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 will ask you to connect the Standard Load #2 to the Analyzer.



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

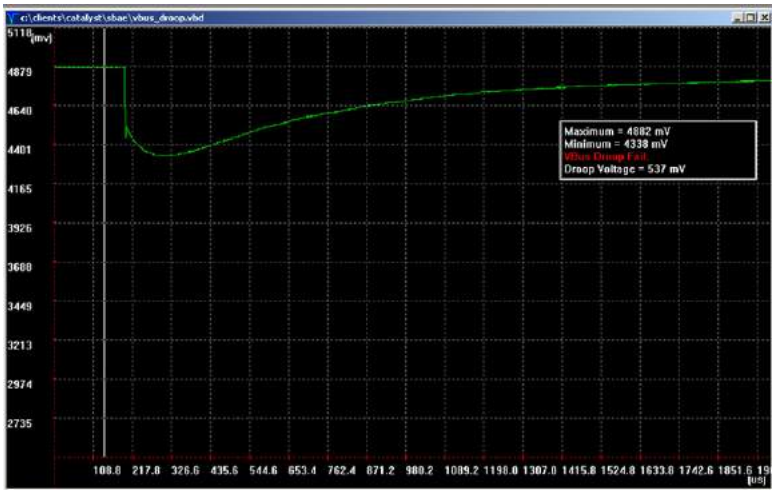


Figure 128 VBus Droop Test Result

Inrush Current Measurement

This test measures inrush current for the connected Device over the first 10 milliseconds of Device activation. The current measurement result will be 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 129 shows the functional connection for inrush current measurement.

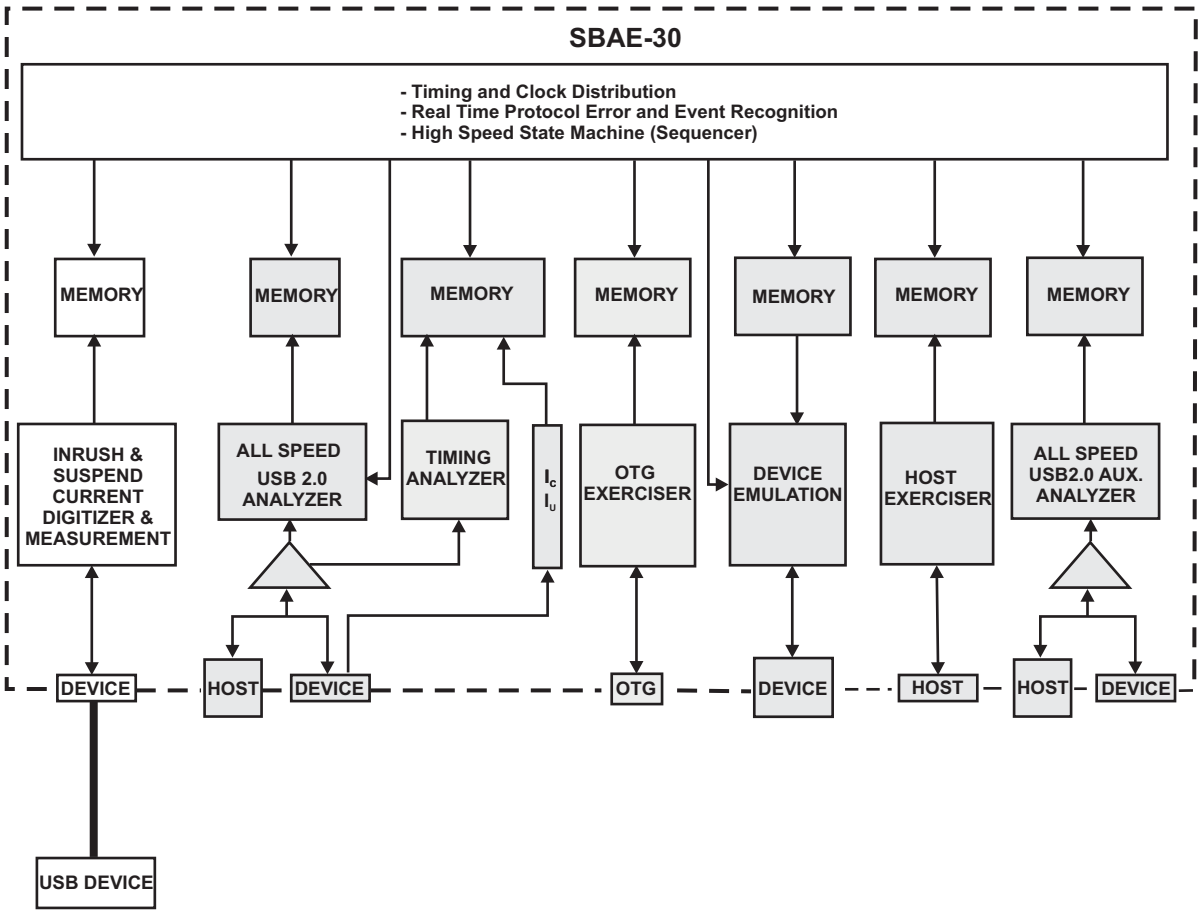


Figure 129 Inrush Current Measurement Functional Connection



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 130**.**

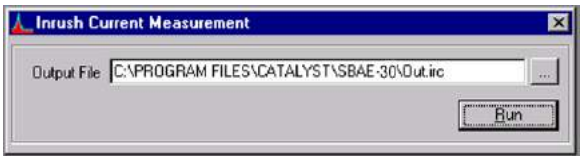


Figure 130 Inrush Current Measurement Dialog Box

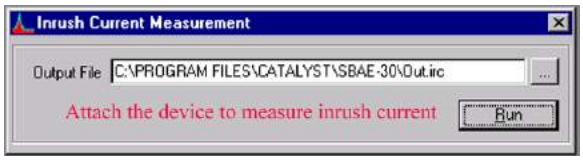
Click the **Run** button and follow the on-screen prompts.



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



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



Plug the desired 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 the SBAE-30. In this case you may perform the measurement again by clicking **Run**.

You may receive the following message:



This is not an error, but an indication that the SBAE-30 needs to switch range. To continue with the measurement, click **OK** followed by **Run.**

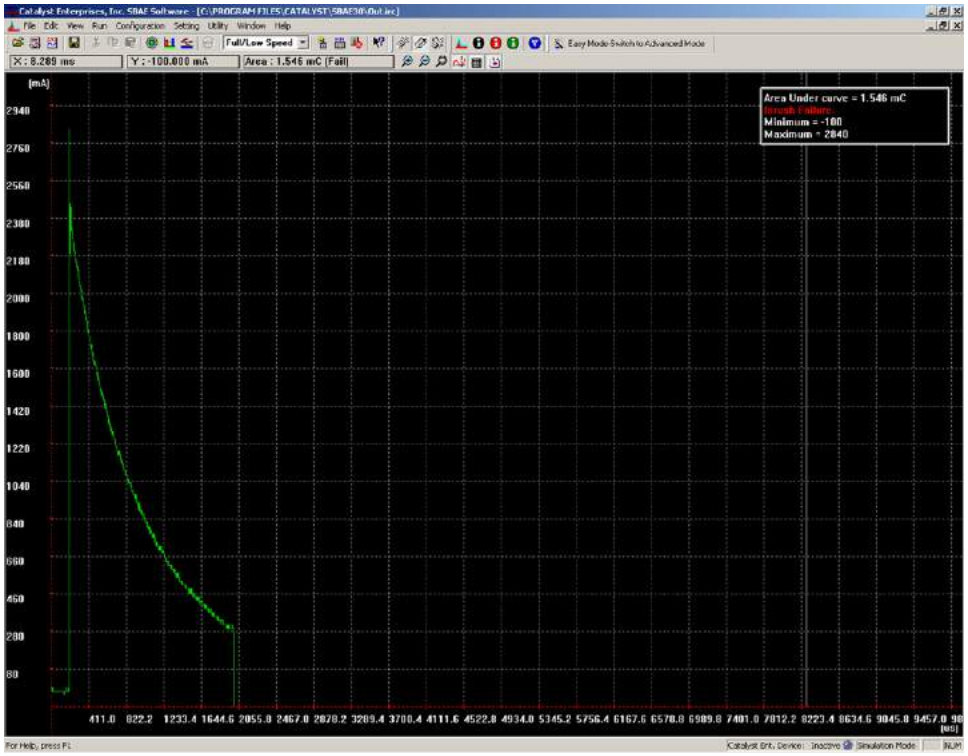


Figure 131 Inrush Current Measurement Result Display

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 uC, the test will be reported as a pass.

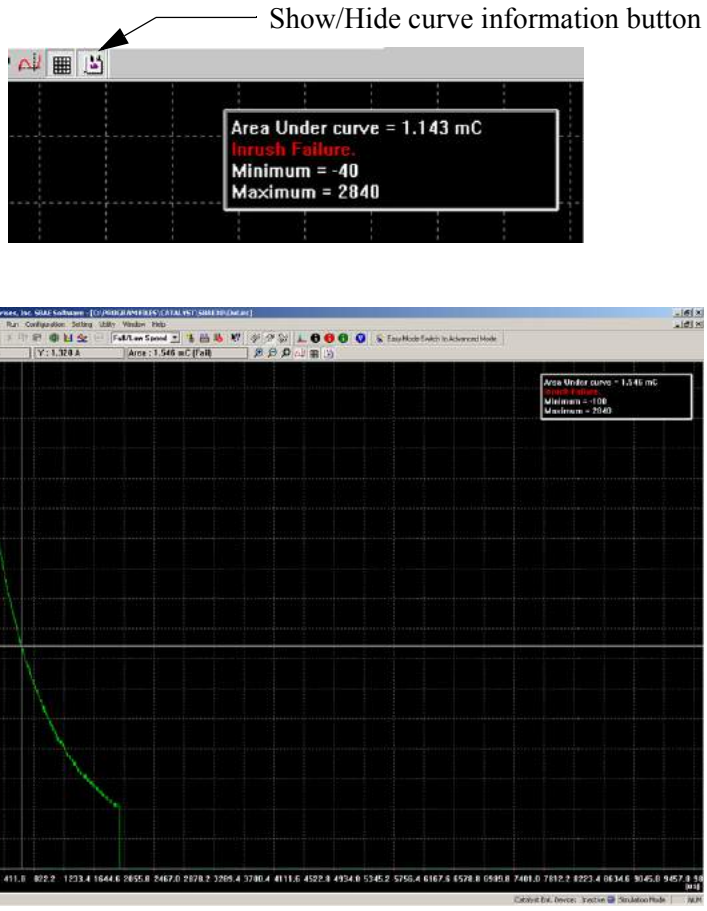


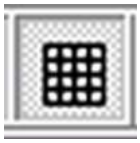
Figure 132 Inrush Current Display With Cursors

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 where you would like the cursors.

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** buttons.



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 116 on page 135

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 133 Suspend Current Measurement Dialog Box

Check the **CCAL** checkbox if using the Current Measurement Calibration Board as described on page 152. If you would like a graphic display of the suspend current as a function of time check the **Show Chart** check box.

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



Perform the required connection and click **OK**.

The required measurement is now performed and the result displayed.



Figure 134 Suspend Current Measurement Results Display

Stop the Measurement If you are viewing the graphical display and would like to stop the display from scrolling, click the **Stop** button in the Measurement Results dialog.

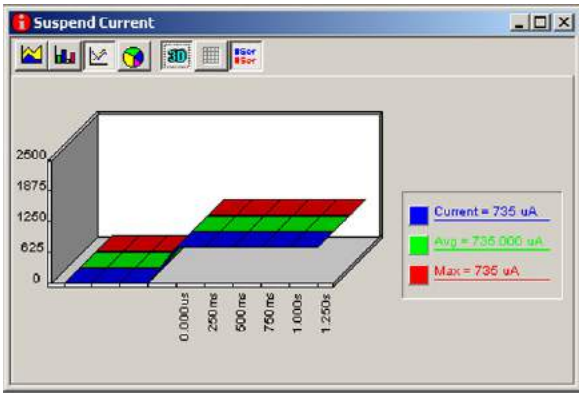


Figure 135 Suspend Current Chart Display

Print Result You may 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 the SBAE-30.

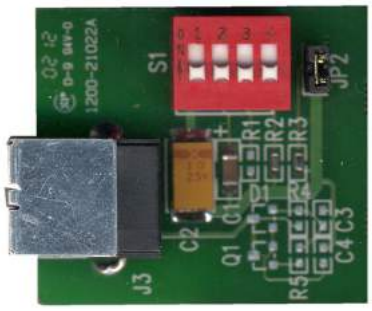


Figure 136 Current Measurement Calibration Board

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	SBAE-30 Measurement
Inrush 1 (5A range)	50µC	Inrush	ON	OFF	OFF	OFF	40µC***
Inrush 2* (10A Range)	50µC	Inrush	ON	ON	OFF	OFF	50µC***
Unconfigured	100mA	Analyzer/Exerciser	OFF	OFF	ON	OFF	66mA
Suspend	500µA	Analyzer/Exerciser	OFF	OFF	OFF	ON	107µA
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 is required to be run twice to change from 5A to 10A range

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

*** There is a ±3µC tolerance on the measurements limited by the components on the calibration board.

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 137.

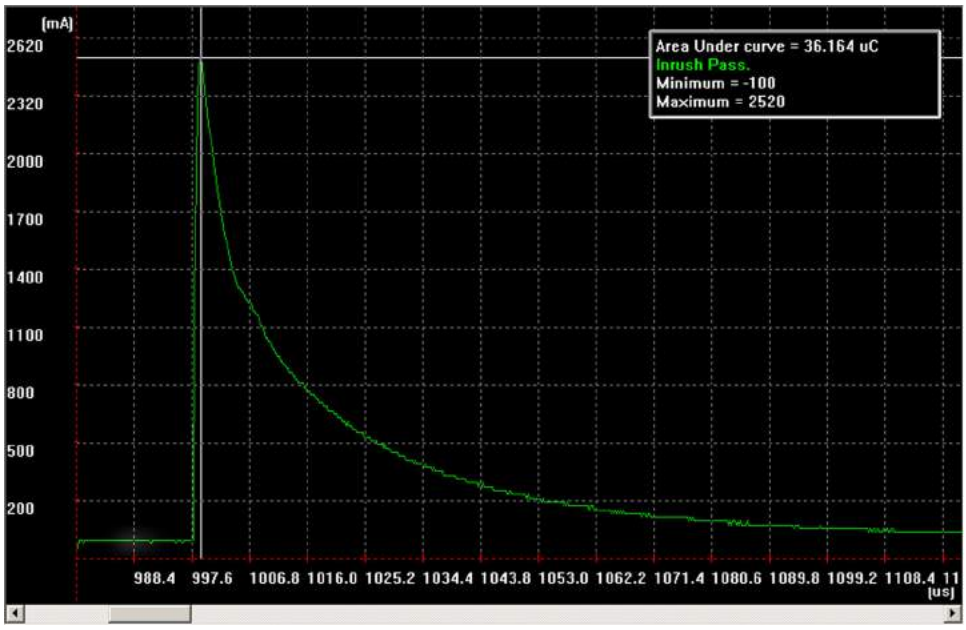


Figure 137 Clean Inrush Measurement Result

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

Display Manipulation

The captured data display may be configured to meet your individual test and viewing preference needs.

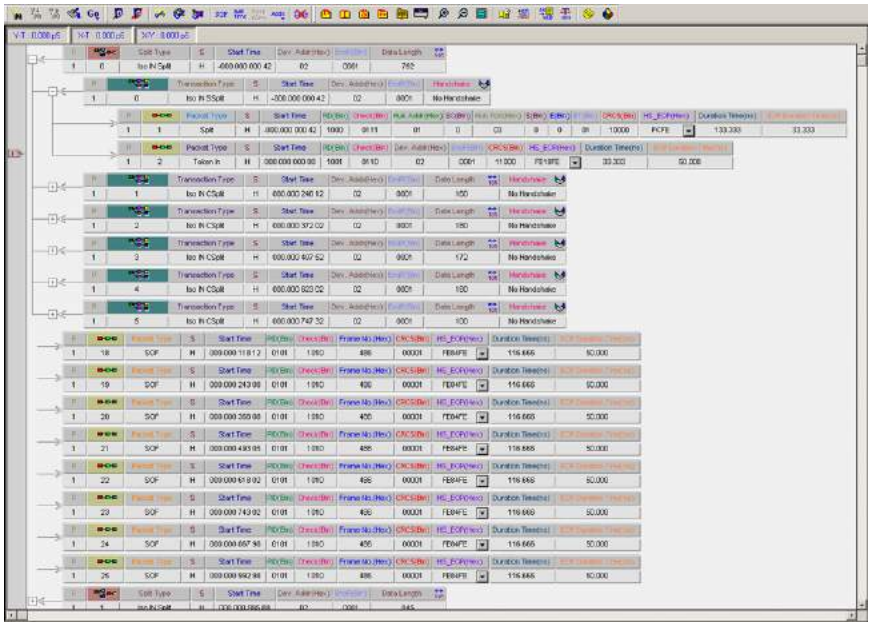
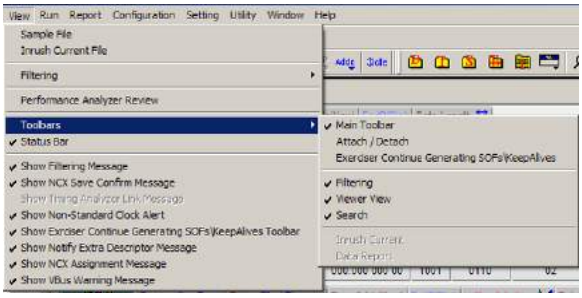


Figure 138 Captured Data Display

Simplify Tool Bar

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



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

Results Display Viewing Preferences

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



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



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



Clicking the **Show/Hide transfers** button hides or displays transfers.



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



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



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



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



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

Compact View

You may 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 choose to Show all Headers, Hide All Headers or to Hide Repeated Headers.

Figure 139 shows a results display with all headers hidden.

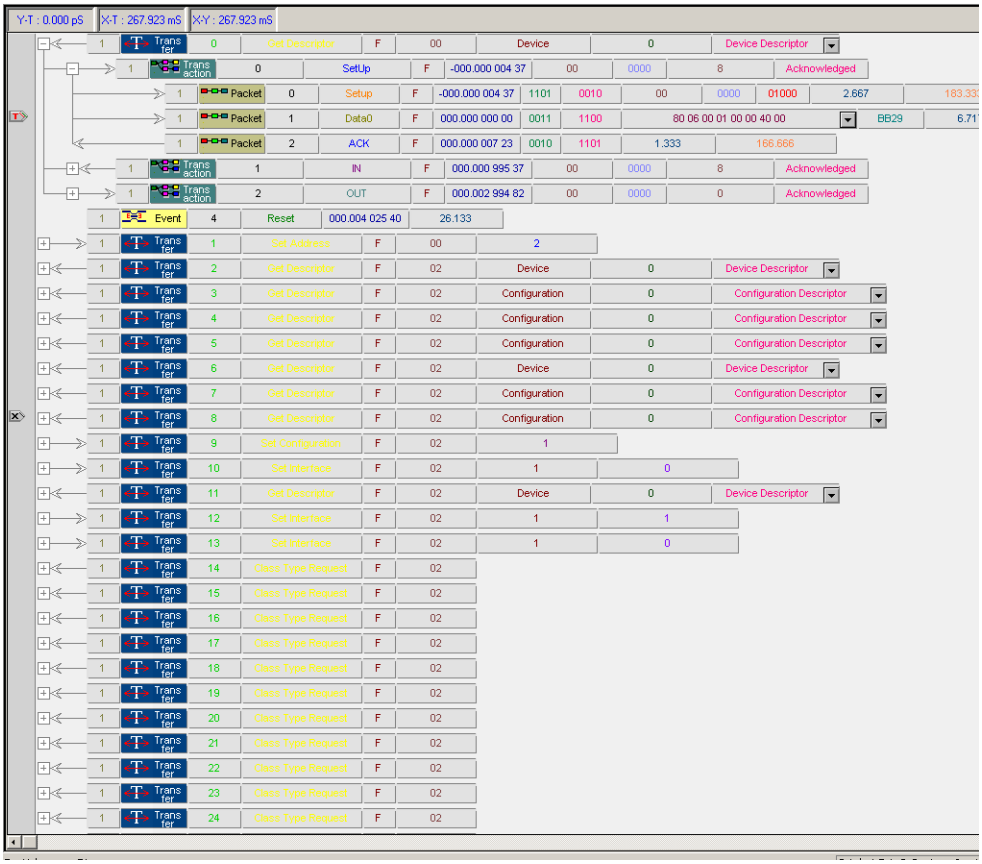


Figure 139 Results Display With All Headers Hidden

Display Idle Time



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

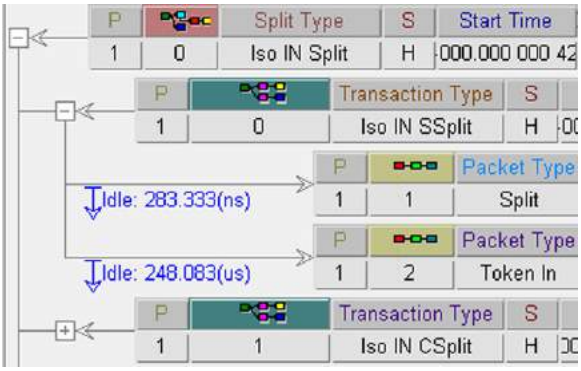


Figure 140 Time Between Events Display Enabled



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

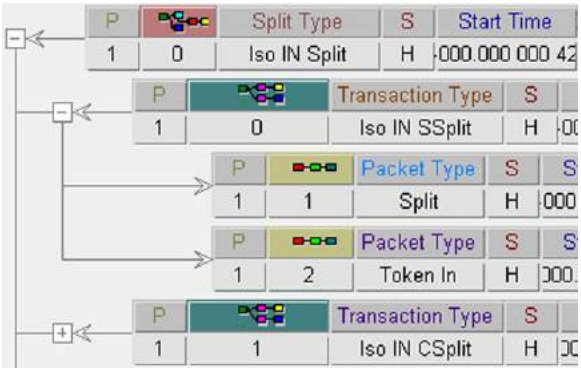


Figure 141 Time Between Events Hidden

Waveform Display

You may 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.**

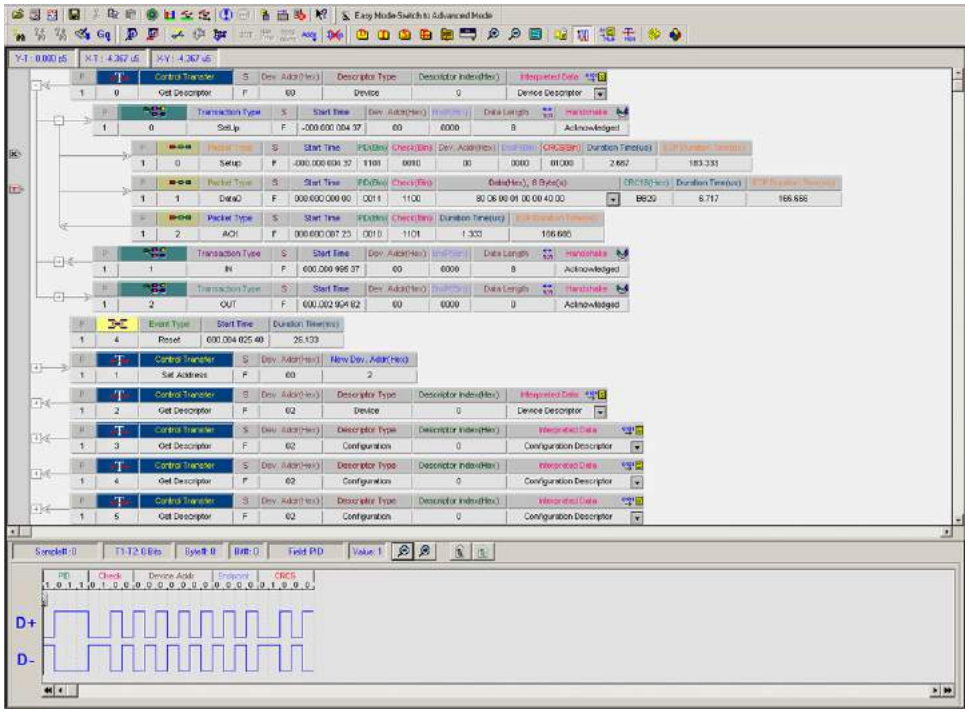


Figure 142 Wave Display Selected



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, T2 pointers. To rapidly return to the desired pointer, click either the T1 or the T2 **Scroll to Pointer Page** button on the wave display tool bar.



Scroll to Pointer Page Buttons

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.

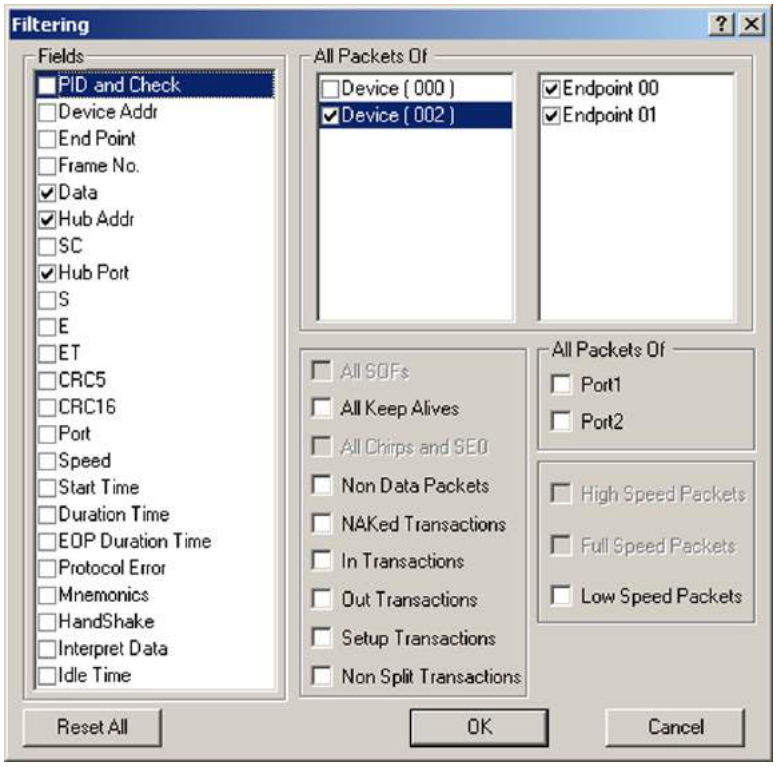


Figure 143 Filter Setup Dialog

Click **OK** and note that the selected filtering patterns have been removed from the data display.

Smart on Screen Filtering

In addition to the normal filtering function, your SBAE-30 offers a quick way to filter the result. You may quickly filter out Device related transactions, all Keep Alives/ SOF and all NAK transactions. To perform Smart filtering right click in the results area and choose the desired filtering action.

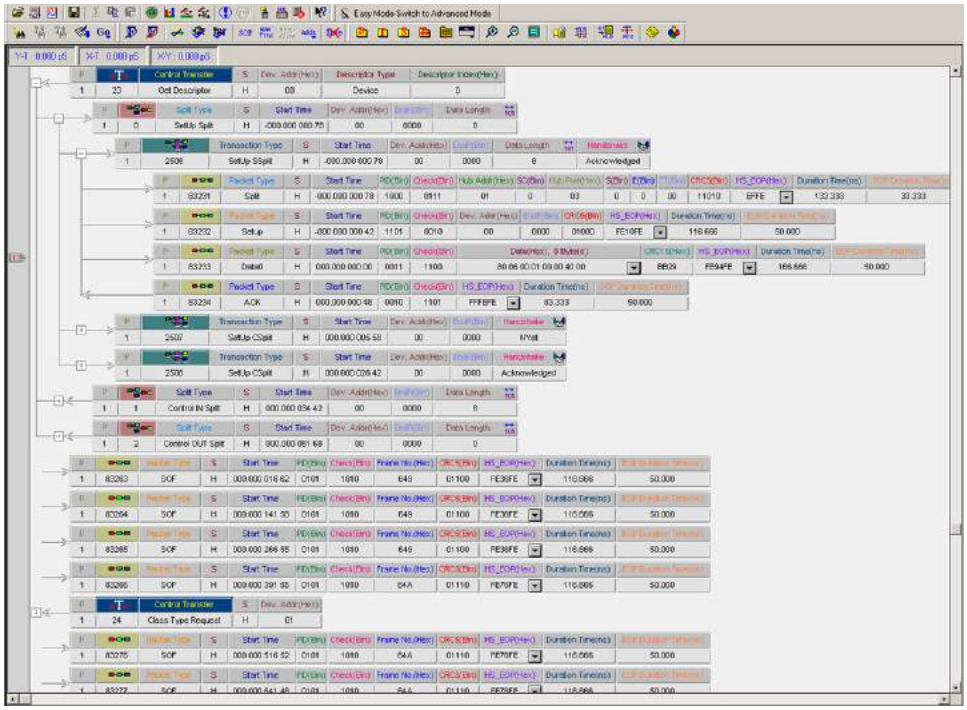


Figure 144 Filtering out All SOF Transactions

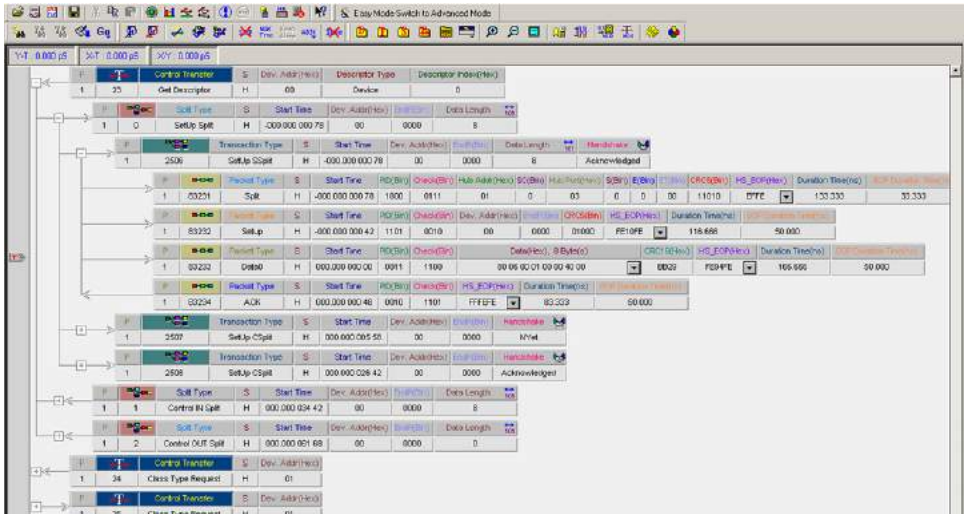


Figure 145 All SOF Transactions Filtered

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

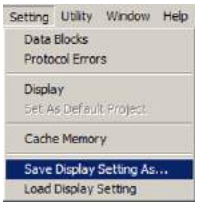


Only Transactions Present Filtering buttons are enabled only for the transactions that occur in the results display.

Save Display Settings

Once you have made all of the display settings for your application you may 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** and open the saved settings file**.**

Timing Analysis Display

All USB signals are always captured and stored on disk, but may not necessarily be displayed in the results display unless specially selected for display.



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

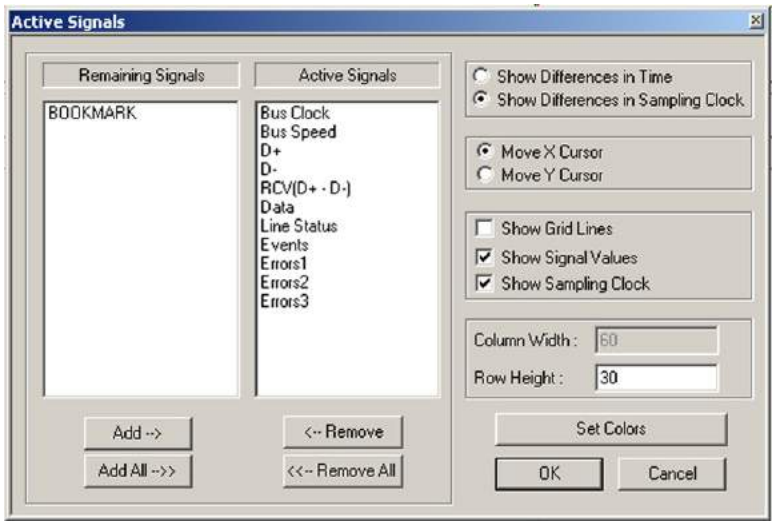


Figure 146 Active Signals Dialog Box

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 may insert a blank line between displayed signals by right clicking in the timing display area between signal lines.



Choose Insert Blank Line

Similarly, you may 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 SBAE-30 sampling clock.

View Timing Details

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

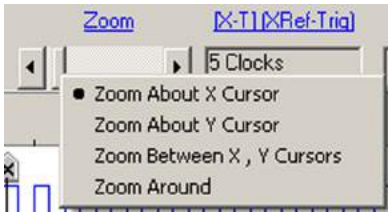
Move X/Y Cursor Click the **X** or **Y** toggle button to enable positioning either the X or Y cursor one sample at a time to the right or left by clicking the right or left arrow.



Zoom around cursors To zoom around either the X or Y cursor click on the cursor to make it active and then slide the zoom slider right or left for the desired 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 138**).** 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.

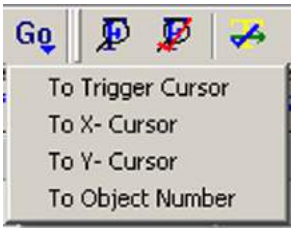
Positioning the X Cursor To position the X-Cursor within the captured data display, click the left mouse button in the line where you wish to see the cursor.

Positioning the Y Cursor To position the Y-cursor within the captured data display, click the right mouse button where you wish to see the cursor.

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

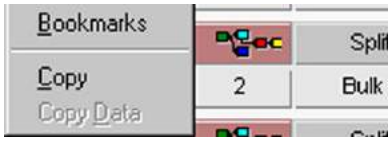


Locate Cursors To quickly locate any of the cursors within the result, click the **Go** button on the toolbar and choose the cursor to locate.



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 where you wish to place the bookmark.



Click **Bookmarks** 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.

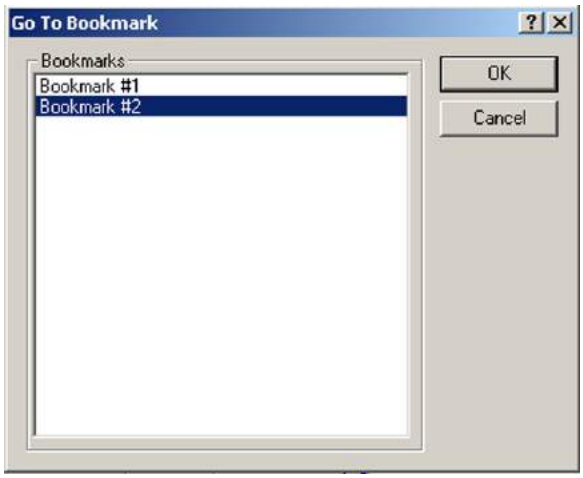


Figure 147 Go To Bookmark Dialog Box

Highlight the bookmark that you wish to go to and click **OK.**



Figure 148 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.

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 149.

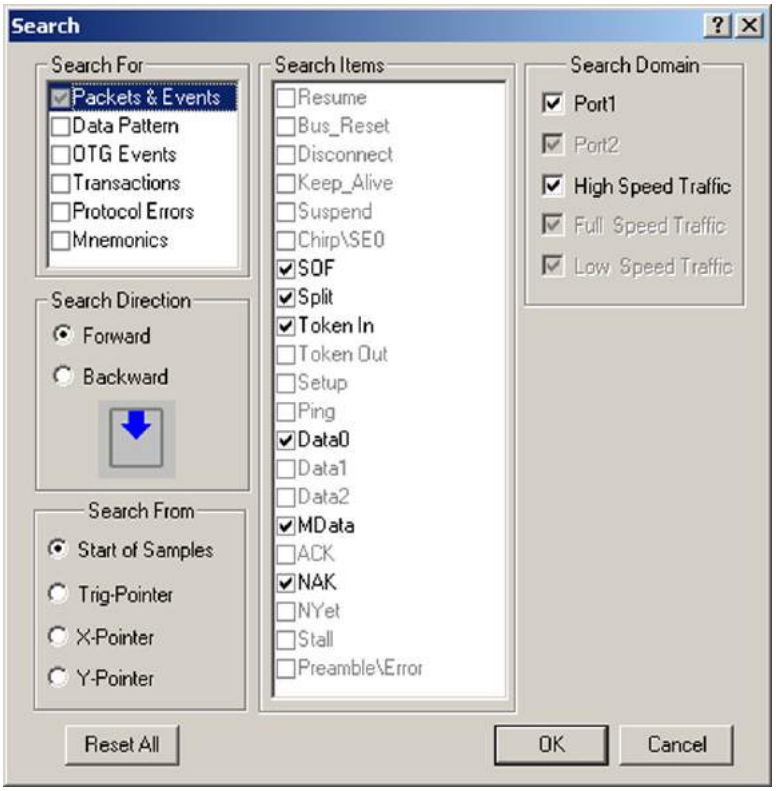


Figure 149 Search Parameter Definition Dialog Box

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

You may 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 as required.

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

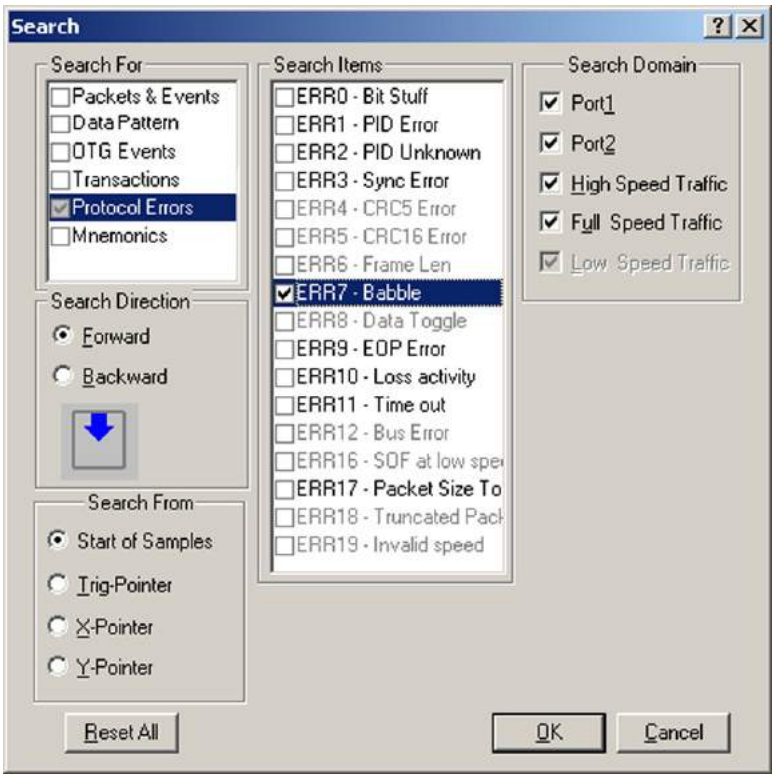
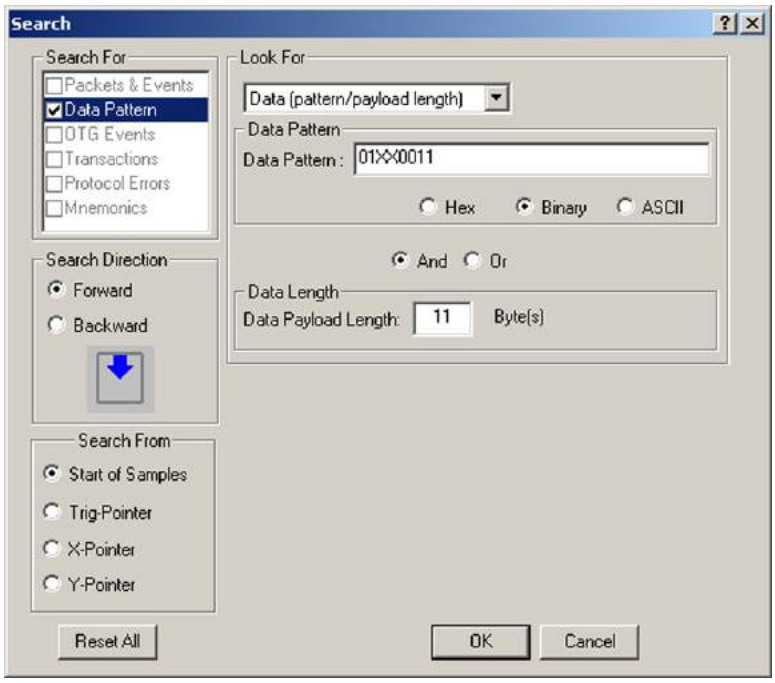


Figure 150 Expanded Search Parameter Definition Box

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

Note: A data pattern will only be found if it is within a data packet boundary, however, if the data is spread across multiple Data packets you should perform the search in the Data Report as described on page 88.



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 175. Once Mnemonics are defined proceed similarly as for searching for Protocol Errors.

Search a Timing Analysis Result You may 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 151.

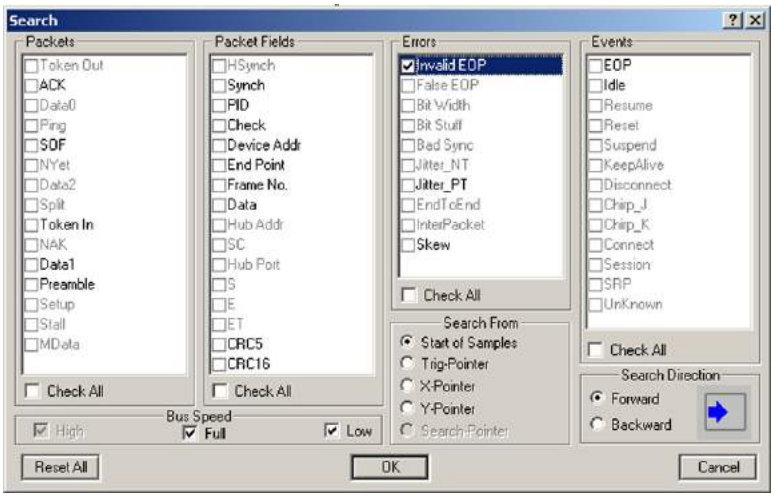
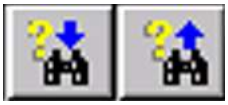


Figure 151 Timing Result Search

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

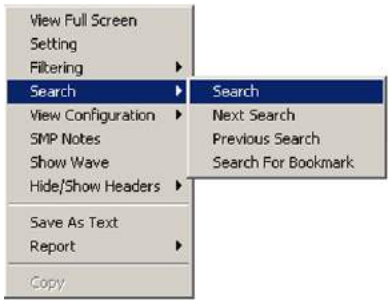
Choose Item(s) for search Select all of the items that you would like to search for and click OK.

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



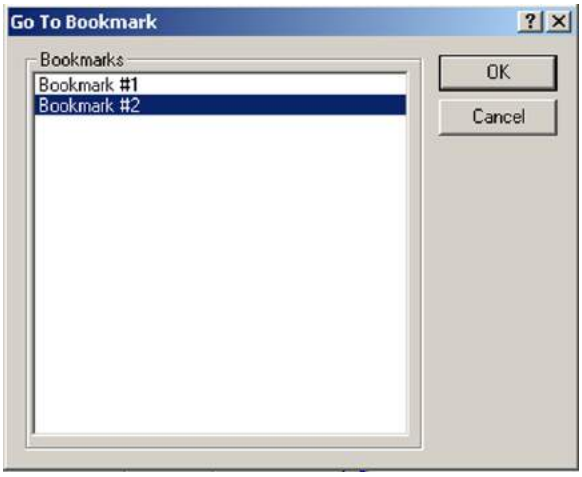
To continue the search for the same set of items, Click the **Next Match** or **Previous Match** buttons on the results display toolbar as required.

Additional Search Options You may access additional search options by right clicking in the right side of the results display for additional choices.



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



Display Configuration

The SBAE-30 ships with a default display font and color configuration. You may, however, define your own fonts and color scheme for a particular testing scenario. You may 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.

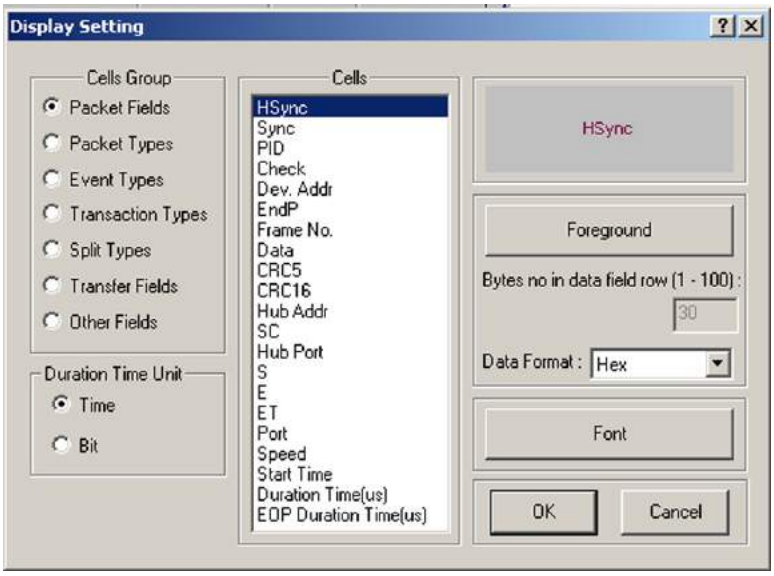
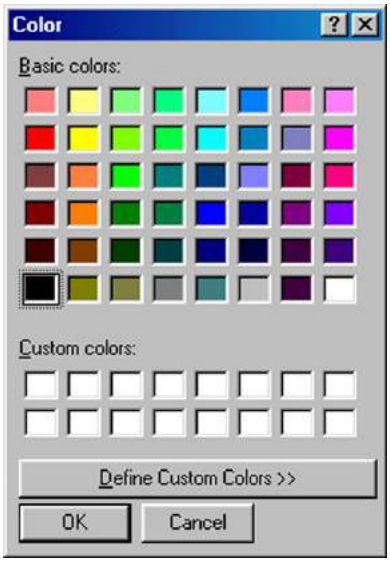


Figure 152 Display Configuration Dialog Box

To change Cell Color

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



3. Choose an appropriate color and click OK.

To change Display Fonts Click the **Font** button to open the Font dialog box.

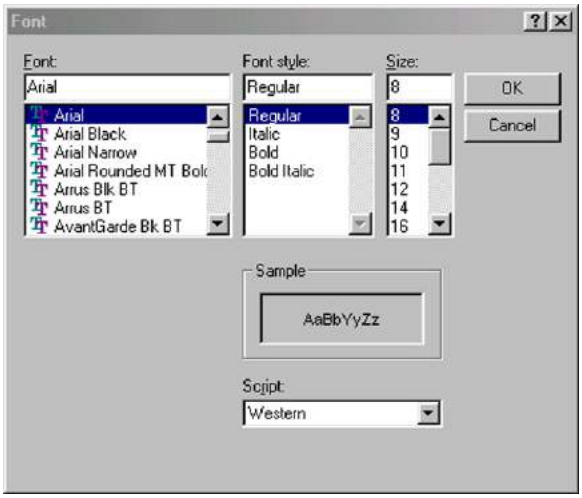


Figure 153 Display Font Definition Dialog Box

4. Choose the desired font, font style, size and click **OK.**
5. 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. These names may be used:

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



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



Figure 154 Mnemonics Definition Dialog Box

1. To assign a unique name to a mnemonic, highlight the Packet "Name" field and type in the desired name.
2. 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 50.

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 may perform two levels of self test on the SBAE-30 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.

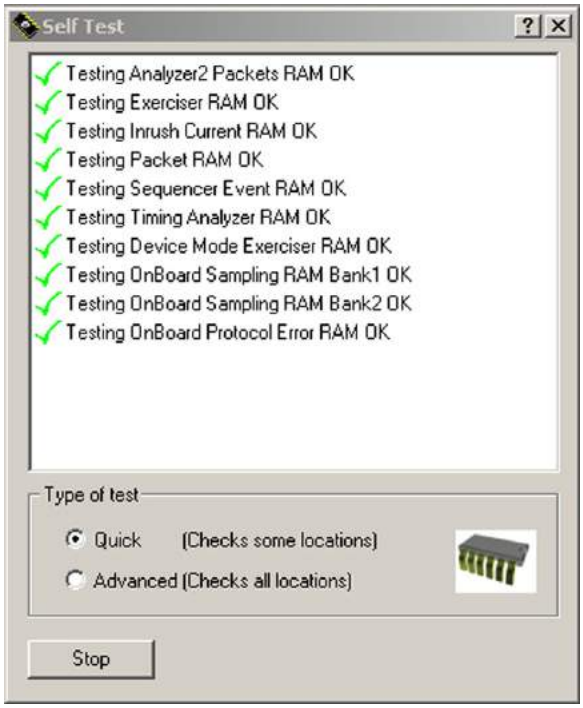


Figure 155 Self Test Dialog With Results Shown

View Scan Descriptors

You may view all the Device Descriptors of all the Devices that are connected to the Exerciser port of the SBAE-30.



Click **Utility** and then choose **View Scan Descriptor** to open the View Scan Descriptors dialog.

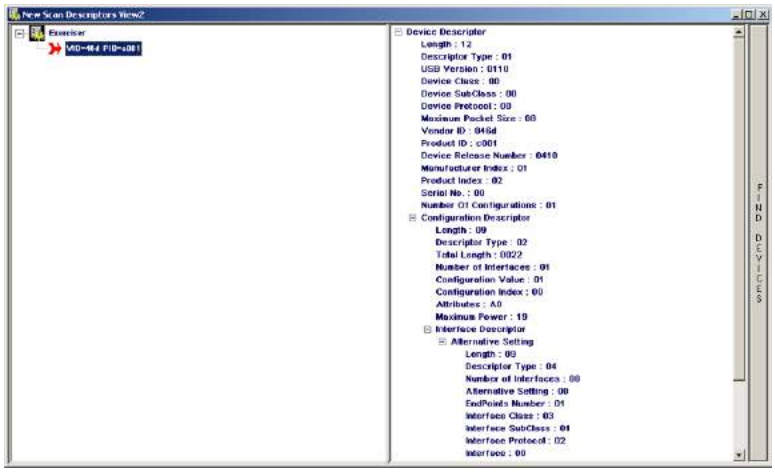


Figure 156 View Scan Descriptor Display

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

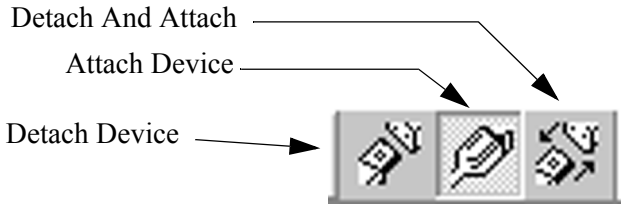
Attach/Detach Device

This feature allows the user to effectively attach and detach a Device from the Analyzer without physically disconnecting the cable.

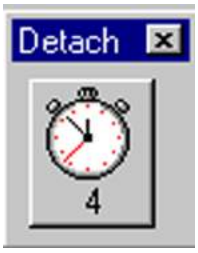


Click **Utility** and then choose **Attach Device**, **Detach Device** or **Detach And Attach** to perform the desired action.

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



When **Detach And Attach** is chosen the Device is detached for 5 seconds with the remaining detached time countdown displayed.



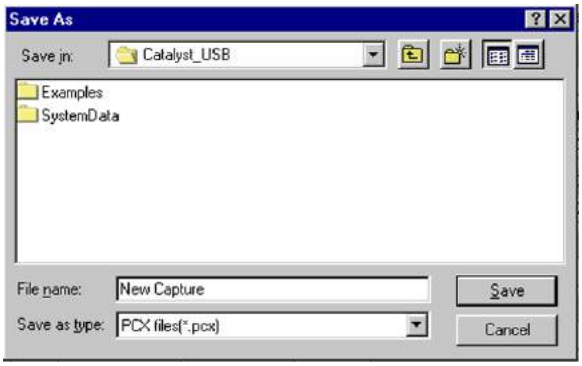
Capture Screen

You may 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 you would like 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 be saved and click **Save.** The capture progress bar will display and when complete, you may open the *.pcx file for viewing using a graphics program or import it in a word processing program such as Microsoft Word™ as an illustration.

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, as defined, enables the user to extract and interpret arbitrary data from the USB stream by creating custom scripts.

The ASL protocol decodes may be used to decode any class or vendor-specific requests and descriptors, protocols or any structured data transferred over the USB bus, or can be used to complement pre-defined protocols.

This appendix is the essential reference for ASL users including definitions, code snippets, and complete examples. The decode script files can be written in any text editor or using the Catalyst script editor (See example on page 212) and must be 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
italics	Identifiers
underscored	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 are permitted to 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)"
```

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 202.

Examples:

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

Comments:

Comments are enclosed in '/*' and '*/' and may 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 may 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;
}
```

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 = USBControlTrasfer | DataStageUSBOfControlTransfer |
EndpointId <**MCS =** number>
<**ProtocolData =** *EndpointId1* <, *EndpointId2*> **<MDS =** number> <**IsOptional**>>
<**ProtocolStatus =** *EndpointId* <**MSS =** number> <**IsOptional**>>
<**ToggleEndpoints =** *EndpointId1, …, EndpointIdN* >
} >
End

ProductName The product name for SBAE series ASL files must be set to **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 sourced from either a control endpoint zero or 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 machanism (i.e 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. The described endpoint is assigned an identifier *EndpointId* that may be referenced 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 specified as two distinct endpoints using two *Endpoint* statement groups.

ProtocolTransferDefinition In general, the mechanism by which high-level protocols are transferred over USB may 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, the Data and Status the stages can be omitted. Note that the function of this block should not be confused 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 *EndpointId*s. To optimize the decoding the user may 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 *EndpointId*s. To optimize the decoding the user may 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 *EndpointId*s. To optimize the decoding the user may 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.

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 will assist 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 USB Specification for more details about these mechanisms.
- If Data or Status stages are independent from the Command stages, they must be implement as separate protocols. For instance, Hub Class (carries hub class requests) and Hub Class Notification (indicates change on any of hub ports) are implemented as separate pre-defined protocols. This rule often applies to any device with an Interface that includes an Interrupt endpoint. It is up to the user to 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
```

Example 3:

```
[ProductName = SBAE]
ProtocolName = "Blutooth 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 the user to assign meaningful interpretation of the values in each field. This assist the user in the interpretation of 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. User may 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 a description is to be assigned. 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 *AddField* 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 descripion 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 will be 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 a description is to be assigned.

Input Parameters*:*

fieldId*:*	Specifies the field. This identifier is the output of a *AddField* statement.
Value:	Specifies the field values for which to apply the given description string. User 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
```

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 the user may decode "dynamic" protocols, the fields for which, 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 202.

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 on an expression.
Length:	Specifies the length of the field, in bits. This parameter can be an integer on 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 will be used instead. This parameter is currently unused 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 will be read from Lowest Bit first. */

F1 = AddField(2, 3, "Flags bitmap",
              "This field contains the flags bitmap.",
              "Flgs ", LSBLEFT);
```

BitStuff:

BitStuff (*fieldId***,** bit**);**

Remark:

This statement preforms a bit-unstuffing algorithm on the field before the field is interpreted. This statement is only required for high-level data of some protocols, and should not be confused with the low-level bit-unstuffing of USB. USB traffic is bit-unstuffed by SBAE hardware before the data stream is presented to the decoder.

Input Parameters:

fieldId: This is the identifier that specifies the field. This identifier is the output of a *AddField* 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","");

BitStaff(Data,5);
```

GetData:

DataId = **GetData(**StartBit**,** Length**);**

Remark:

This statement assigns an identifier to a block of data from the data stream. The complete data block can then be referenced 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);
```

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 will repeat until current position >= 100 */
Repeat (CURPOS < 100)
{
  SubF1 = AddField(CURPOS, 23, "SubField of field F1",
                   "subfield of F1", "SubF1");
}
```

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 will be run 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 user does not include the *Separated* keyword, then the subfield(s) will be shown in the tables as a subset of the parent field that can be expanded or collapsed with an arrow. If user includes the *Separated* keyword a separate table will be 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");
}
```

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 be shown 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 *AddField* 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 the user 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 *AddField* statement.
Title:	Specifies the string to be shown in the header of the table.
OptionalfieldId:	Specifies the field from which to get the header string. This identifier is the output of a *AddField* 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 the user to assign meaningful interpretation of the values in each field. This assists the user in the interpretation of 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
```

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 will become 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 C*URPOS* has reached the end of the data stream, this function returns a TRUE. Othewise, a FALSE is returned. This function may be used to detect if there is 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 the desired 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 to be searched. 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");
}
```

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
{
      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]
{

/*
  Here would be the script of other protocols.
*/


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");
```

```
            If (ValueOf(TypeLength) > 32) then
            {
            Tag = AddField(CURPOS, ValueOf(TypeLength)-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;
                    "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 will define 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)
      {
            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

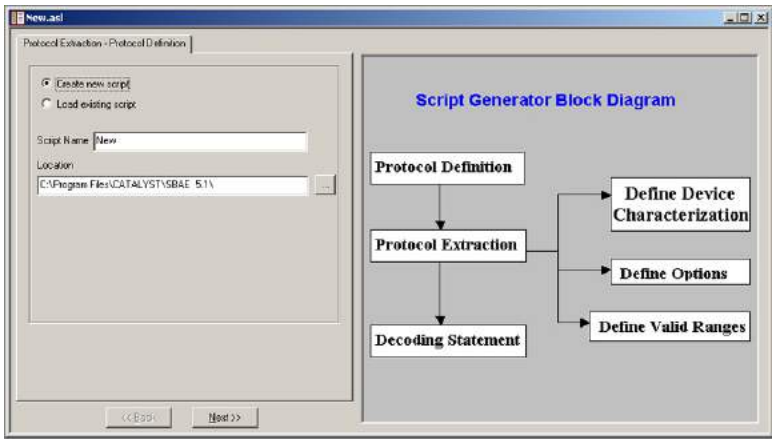
0	bmRequesType	1	Bits 0…4 (Recipient)	Interpretation
			0b00000(0x00)	"Device"
			0b00001(0x01)	"Interface"
			0b00010(0x02)	"Endpoint"
			0b00011(0x03)	"Other"
			0x04-0x1F	"Reserved"
			Bits 6…5 (Type)	
			0x0	"Standard"
			0x1	"Class"
			0x2	"Vendor"
			0x3	"Reserved"
			Bit 7 (Direction)	
			0x0	"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 will appear)
3	Windex	2		No field option needed. (Same value will appear)
4	wLength	2		No field option needed. (Same value will appear)

Note: An option name must be assigned to all fields that have multiple interpretations based on their values.

Creating the Script

1. Click File, New and choose Script to open the script generator dialog.





2. Choose and enter a Name for the Script and enter a path to a location where you would like to save it and click Next

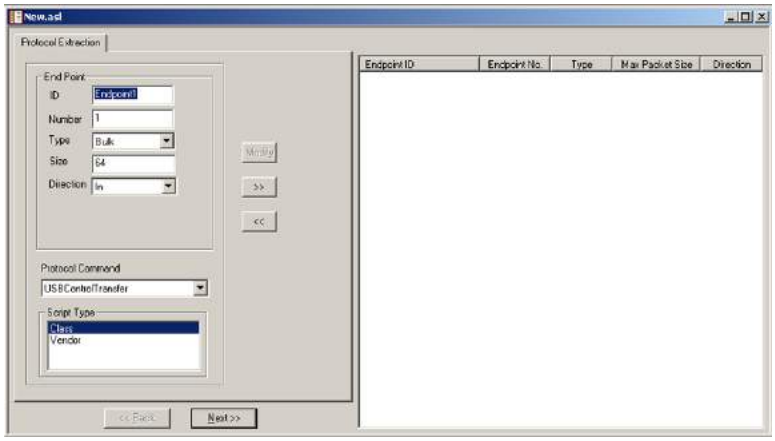


Figure 157 Choosing Command and Class

3. Chose **USBControlTransfer** as protocol command and **Class** as Script Type and click Next.

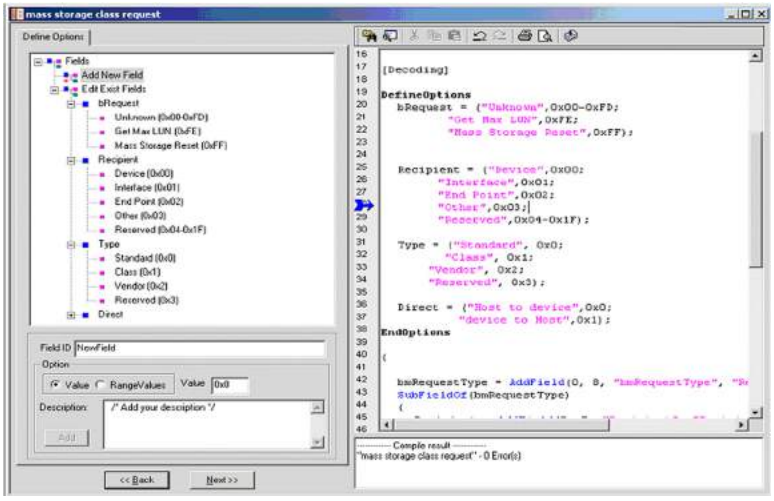


Figure 158 Defining Field Options

4. 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 editor directly)
5. When done click **Next**.
6. Add a Valid Range to your script if needed. (You do not need to define any valid range for this example script). (Press **Next** Button to go to Next page)
7. Add your decoding statements to decoding area by defining your command and adding it as a decode command.

Note: The command will be placed at the cursor location in the decoding area. If the cursor is above the decoding area the command will be placed at the top and if it is under decoding area it will be added at the end of decoding area

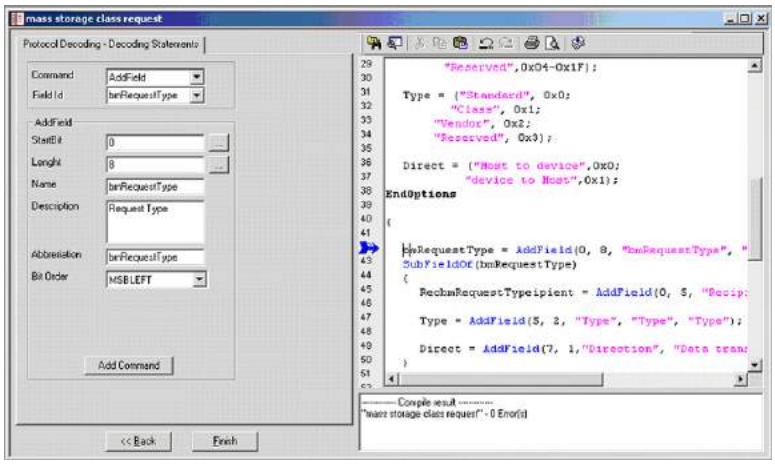


Figure 159 Command Location

Index

A
activate
device mode 120
additional requests
in device emulation 119
advanced refresh
in exerciser program 72
Advanced Script Language 181
analysis
new 101
pre-defined 101
real time 100
analysis Expressions
creating 102
analyzer options 7
analyzer speed 46
ASL
functions 202
language elements 182
protocol decoding 191
protocol extraction 186
script structure 184
USB descriptor section 200
authorization code
for upgrades 23
auxiliary Port 48
B
Bi-directional
parallel port 22
bit width
adjustable 70
bookmarks 165
finding 166
boolean expression
in sequencer 57
C
capture
partial data payload 47
capture screen 179
class
specifying new 63
clock
non standard 46
Conquest
as analyzer 15
as exerciser 15
included components 10
control transfers 62
counter
as data 78
counters
in sequencer 56
create a new analysis 101
current measurement 135
calibration 152
inrush compliance 145
suspend 150
unconfigured 136
cursors
inrush current display 149
locating 165
positioning 165
D
data
decoded 89
in transaction layer 67
in transfer layer 66
interpreting 89
specifying for transfer 66
data block
counter data 78
custom pattern 77
defining 75
editing as text 80
naming 76
random pattern 78
walking bit 79
data report 84
settings 85
DC Compliance 7
decodes
user defined 91
default capture 27
descriptors
independent 119
string 120
device
attach/detach 178

configuration define 117
definition 117
emulation 115
mode 115
device emulation 7
advanced mode 122
connections 14
device interface
define 118
display
cell color 172
fonts 173
inrush current 148
viewing preferences 156
display Configuration 172
display manipulation 155
E
easy mode
pre-defined setups 31
enable exerciser
easy mode 45
endpoint
define 118
settings 64
endpoint errors
definition of 121
error message
startup 17
errors
forcing 68
in user defined scripts 92
protocol 93
ethernet
connecting with 20
example
files 24
folder 24
exclude
KeepAlive events 33
NAK transactions 33
NYet transactions 33
protocol errors 38
SOF events 33
exercise and capture
setup 42
exerciser
configuration 12
programming in advanced mode 60
programming in easy mode 43
exerciser program
copying and pasting 73
importing 71
shortcuts 71
terminating in OTG 134
expand/close
splits and transactions 68
expressions
for analysis 102
external
signals 8
external I/O Connector
pin assignment 16
external outputs 56
programming 56
external Trigger
triggering options 47
external trigger
enable 37
F
file type
definition 24
filter 160
access from menu bar 162
filtering 160
smart, on screen 161
forcing errors 68
H
hardware setup 11
headers
show/hide 157
high level imterpretation 90
high level interpretation
protocol 90
Host Exerciser 7
host exerciser
programming with transfers 61
hub testing 12
I
idle time
show/hide in display 158
inrush current compliance 145
insert
blank line in timing display 163
insert line
in exerciser program 44
installing your analyzer 11

interface
establishing 19
interpretation
high level assignment 90
interpretstion assignment 90
L
loops
in exerciser program 45, 68
LPT port
not configured 22
LPT port setting 22
M
Manual Driver
installing 17
manual Trigger
advanced mode 58
manual trigger 8, 37
with sequencer 58
mnemonics 175
multiple analyzers 20
N
new analysis
advanced mode 101
new class
specifying 65
new project
advanced mode 50
easy mode 32
O
operating speed
for refresh 74
options 7
OTG
analysis connection 13
OTG device
definition 129, 132
OTG Exerciser 7
as DRD device-A 13
as DRD or peripheral only device-B 14
OTG VBUS
response type 129
outputs
external 56
overview 5
P
packet
defining 50
payload data 52
type 51
user defined 52
packet types 51
parallel port
set mode 22
parameters
deleting in sequencer 58
performance analysis 100
advanced mode 100
easy mode 97
pointers
in waveform display 159
lost 159
ports 8
pre-defined analysis 101
pre-defined data capture selections 40
pre-defined trigger selections 39
pre-trigger 95
program lines
maximum in exerciser 60
program loops
in exerciser program 68
programming
exerciser 60
project
file type definition 24
importing 124
protocol
user defined decodes 91
protocol error mask 59
protocol errors 58
detected 93
exclude 38
post process 93
triggering on 93
R
random data pattern 78
reduced bit width 70
refresh
advanced 72
auto 74
exerciser program 74
manual 74
reports 83
results display
viewing preferences 156
run the project 82

S
saving
performance analysis result 107
SBAE
versions 1
SBAE-30
included components 10
SBAE-30 external I/O Connector
pin assignment 16
SBAE-30A 1
SBAE-30B 1
scan descriptors
viewing 177
search
access from menu bar 171
data report 88
for bookmark 171
for data pattern 169
for protocol errors 168
using mnemonic 169
search data report
for data pattern 88
select
components for installation 17
self test 176
separate systems 11
sequencer
description 54
operation overview 54
programming 55
serach data report
for data block 88
shortcuts
exerciser program 71
simulation mode 22
software installation 17
start exerciser
advanced mode 56
easy mode 45
statistical report 83
status LED 8
suspend current
measurement 150
T
test mode
high speed 46
high speed usage 46
timing analysis
easy mode 111, 113
Timing Analyzer 7
timing analyzer
errors 114
timing details
viewing 164
timing display
cursor positioning 164
viewing options 164
tool bar
simplifying 155
transaction retry 47
transfers
control 62
non control 63
trigger
external enable 37
manual 37
on data packet 35
on data pattern 35
on protocol error 37
over multiple transactions 35
trigger mask 47
U
unconfigured current
measurement 136
Undo/Redo
in exerciser program 69
unpacking 10
USB Driver
manual install 17
update 18
User Defined
packet 52
using the cursors 165
V
VBus droop 143
VBus measurement 141
view as report 83
view scan descriptors 177
viewing preferences
results display 156
W
walking bit pattern 79
waveform
display 159
workspace
optimizing 58

Z

Zoom in exerciser program 68

Catalyst Enterprises, Inc.
6680 Via Del Oro
San Jose, CA 95119
Phone 408.365.3846
Fax 408.365.3847
www.getcatalyst.com

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

Gold-standard solutions

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

We buy equipment

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

Learn more!

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

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

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

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

ARTISAN® TECHNOLOGY GROUP