dap TECHNOLOGY

1394 Analyzer Operation Manual

Hardware and Software Guide

Doc #: DT-PRO134MAN700E

Table of Contents

Chapter 1. Introduction

Welcome to the 1394 Analyzer Operation Manual . For a quick start, please read the Getting Started chapter. If you need help installing the software, please refer to the Installation chapter.

The analyzer software contains the following functionality, which is explained in the corresponding chapters of the FireDiagnostics Suite Manual. Please be aware that available Analyzer Modules depend on the specific Analyzer model used.

- Monitor
- Recorder
- Symbol Recorder (licensed separately)
- Commander
- Generator
- Scriptor
- Filter/Trigger
- Mil1394 Signal Monitor (licensed separately)

The analyzer is in fact a protocol analyzer, which currently supports the following higher-level protocols (licensed separately):

- Automotive Multimedia Interface Collaboration (AMI-C)
- Serial Bus Protocol (SBP)
- Audio Video / Control Protocol (AV/C)
- Internet Protocol version 4 (IP4)
- Industrial / Instrumentation Digital Camera Protocol (IIDC)
- Mil1394 Protocol
- Custom Protocols

For information on the licensing model, please refer to the License Manager chapter.

Chapter 2. Installation

DapTechnology analyzers are not packaged with any software-installation media. This is because we would like our customers to use the very latest software available and not an old version that happened to be the latest when we packaged the product.

You will be able to download the latest software version from our website: http://www.daptechnology.com

Click on the support button at the top of the web page and after inserting the serial number of your FireSpy, which can be found at the back of the unit, you will get access to your custom download page. On this page you will find the license keys for your analyzer as well as links to the software downloads available for your analyzer. (latest recommended or latest beta version)

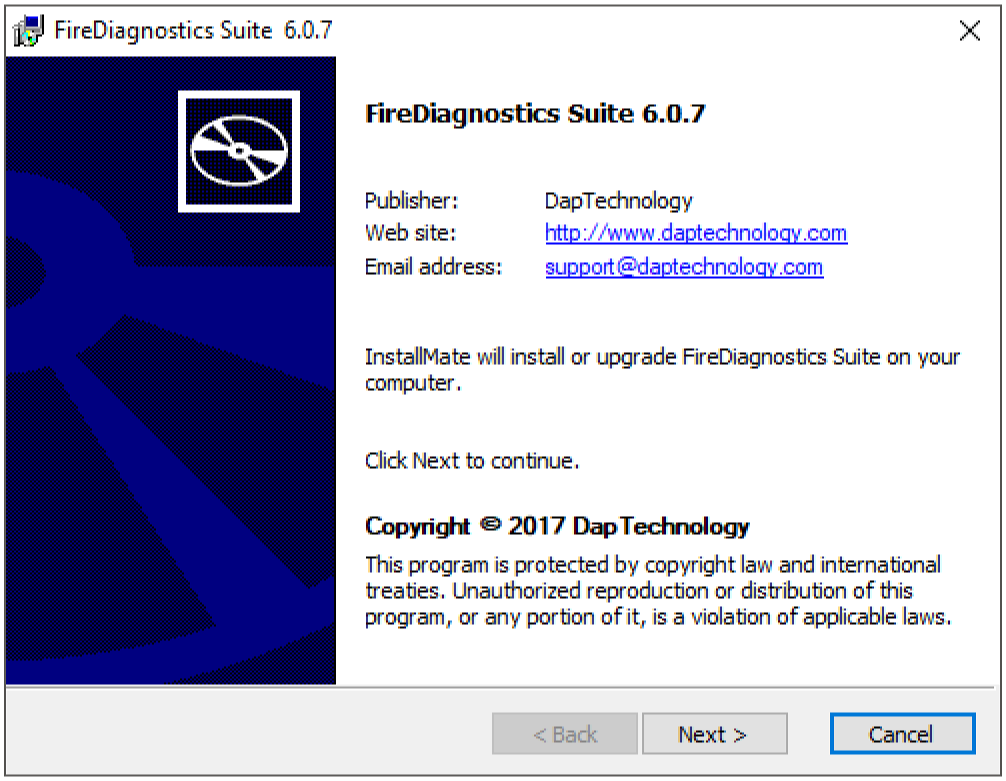
2.1. Windows

2.1.1. Installing the Software

After downloading the FireDiagnostics Suite version of your choice from DapTechnology's website, please open the compressed folder and locate the file setup.exe.
Run the setup.exe file, by one of the following ways:

- double click on the setup.exe file
- select 'Start->Run', browse to the setup.exe file and clock 'OK'
- click with the right mouse button on setup.exe and select' Open'

After starting the setup.exe program, the following window will be displayed. Note that the software version could be different if you are installing another version of the software.



FireDiagnostics Suite 6.0.7

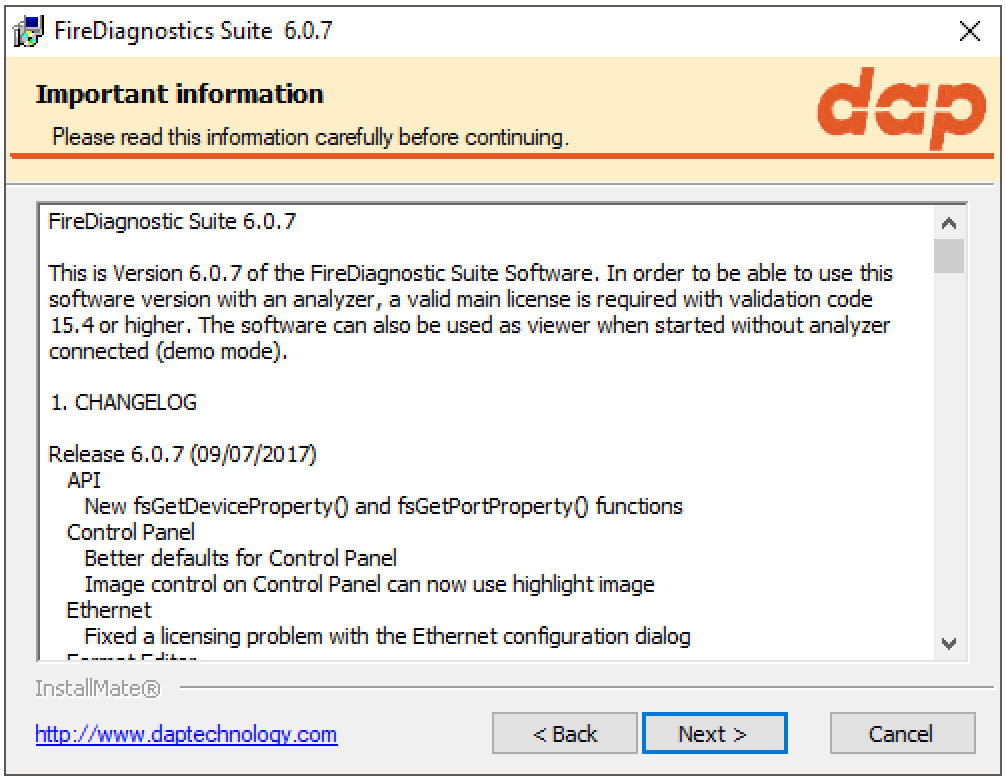
Publisher: DapTechnology
Web site: http://www.daptechnology.com
Email address: support@daptechnology.com

InstallMate will install or upgrade FireDiagnostics Suite on your computer.

Click Next to continue.

Copyright © 2017 DapTechnology
This program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, is a violation of applicable laws.

After clicking on "Next", the following window will be showed. It contains the latest information about the version of the software you are about to install. Please read it carefully before proceeding to the next step. The picture below contains just some example information.



FireDiagnostics Suite 6.0.7

Important information

Please read this information carefully before continuing.

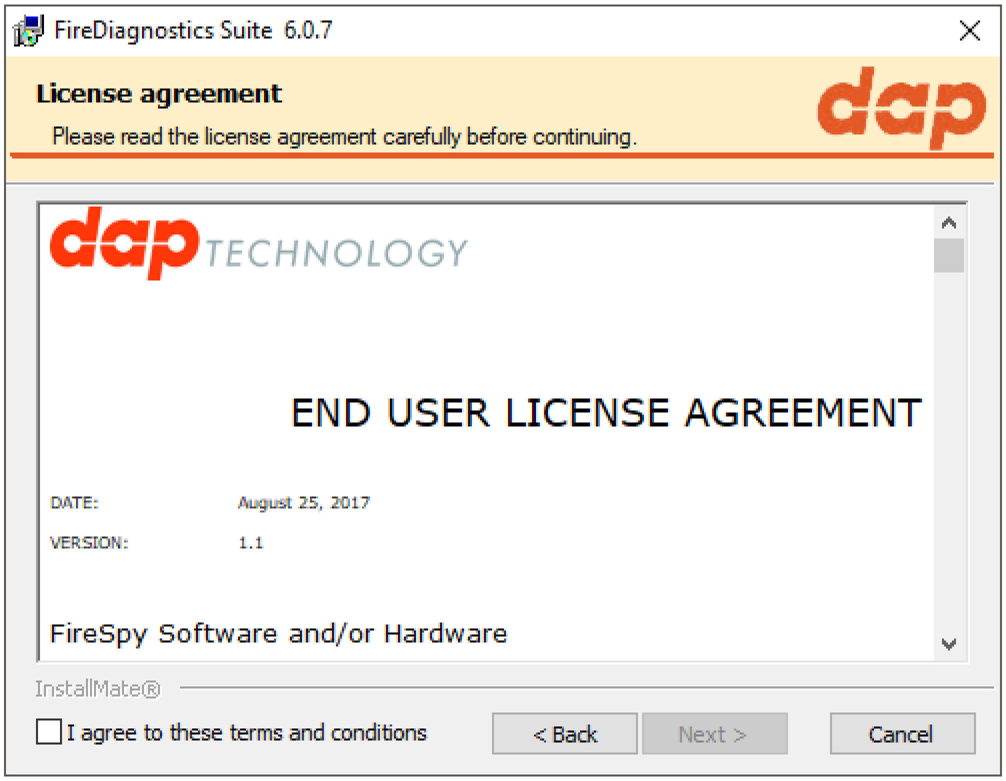
FireDiagnostic Suite 6.0.7

This is Version 6.0.7 of the FireDiagnostic Suite Software. In order to be able to use this software version with an analyzer, a valid main license is required with validation code 15.4 or higher. The software can also be used as viewer when started without analyzer connected (demo mode).

1. CHANGELOG

Release 6.0.7 (09/07/2017)
  API
    New fsGetDeviceProperty() and fsGetPortProperty() functions
  Control Panel
    Better defaults for Control Panel
    Image control on Control Panel can now use highlight image
  Ethernet
    Fixed a licensing problem with the Ethernet configuration dialog

After clicking on "Next", the following window will be showed. It contains the License Information for the software you are about to install. Please read it carefully and check the checkbox to agree to the license agreement before proceeding to the next step.



FireDiagnostics Suite 6.0.7

License agreement

Please read the license agreement carefully before continuing.

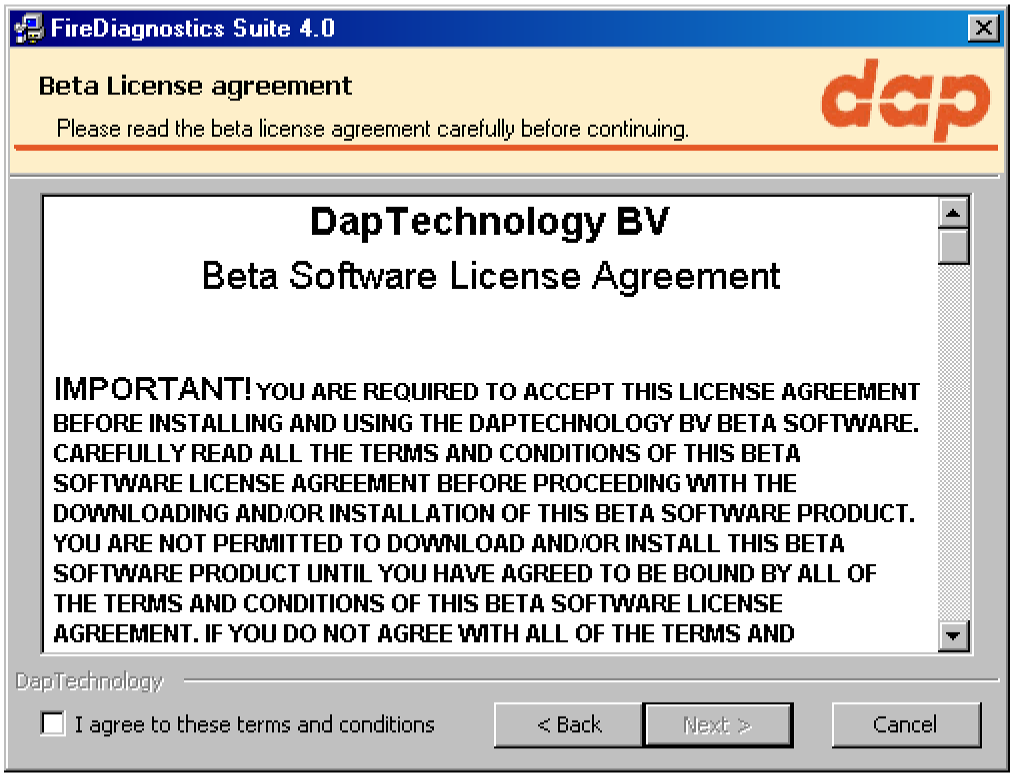
dap TECHNOLOGY

END USER LICENSE AGREEMENT

DATE:	August 25, 2017
VERSION:	1.1

FireSpy Software and/or Hardware

If you are installing a beta software version, an additional license agreement needs to be agreed with before continueing the installation process. Please check the checkbox and click next.



FireDiagnostics Suite 4.0

Beta License agreement

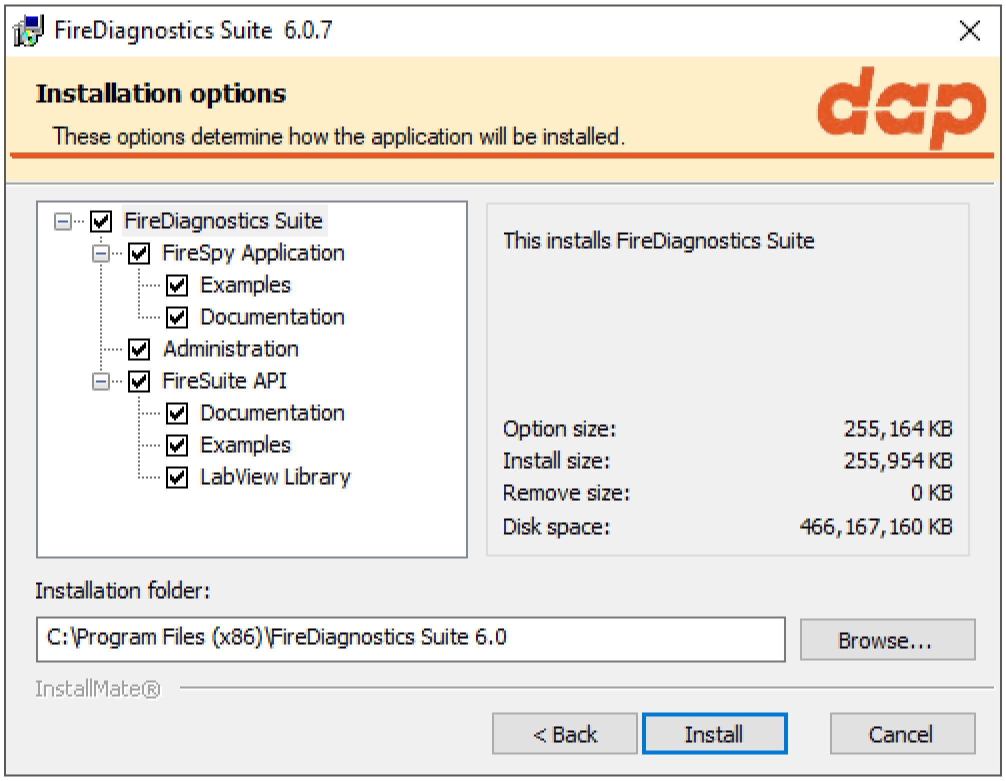
Please read the beta license agreement carefully before continuing.

DapTechnology BV

Beta Software License Agreement

IMPORTANT! YOU ARE REQUIRED TO ACCEPT THIS LICENSE AGREEMENT BEFORE INSTALLING AND USING THE DAPTECHNOLOGY BV BETA SOFTWARE. CAREFULLY READ ALL THE TERMS AND CONDITIONS OF THIS BETA SOFTWARE LICENSE AGREEMENT BEFORE PROCEEDING WITH THE DOWNLOADING AND/OR INSTALLATION OF THIS BETA SOFTWARE PRODUCT. YOU ARE NOT PERMITTED TO DOWNLOAD AND/OR INSTALL THIS BETA SOFTWARE PRODUCT UNTIL YOU HAVE AGREED TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THIS BETA SOFTWARE LICENSE AGREEMENT. IF YOU DO NOT AGREE WITH ALL OF THE TERMS AND

The installer options dialog as shown below allows the user to select which components to install. We recommend installing all components, however it is also possible to leave some components out.



FireDiagnostics Suite 6.0.7

Installation options

These options determine how the application will be installed.

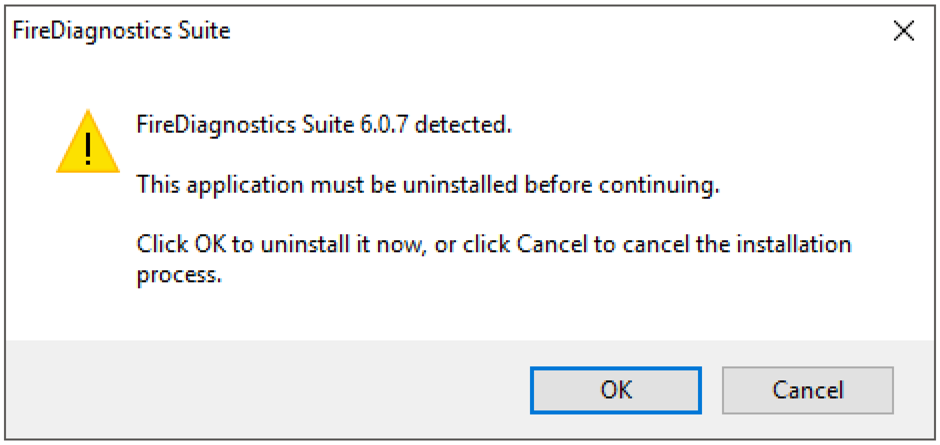
- ☑ FireDiagnostics Suite
  - ☑ FireSpy Application
    - ☑ Examples
    - ☑ Documentation
  - ☑ Administration
  - ☑ FireSuite API
    - ☑ Documentation
    - ☑ Examples
    - ☑ LabView Library

This installs FireDiagnostics Suite

Option size:	255,164 KB
Install size:	255,954 KB
Remove size:	0 KB
Disk space:	466,167,160 KB

Installation folder: C:\Program Files (x86)\FireDiagnostics Suite 6.0

If another FireDiagnostics Suite installed version with the same main version number is found it needs to be reinstalled before the installer is able to continue. Software versions with a different major version number can be used alongside eachother. Please click ok to proceed with the uninstallation process.



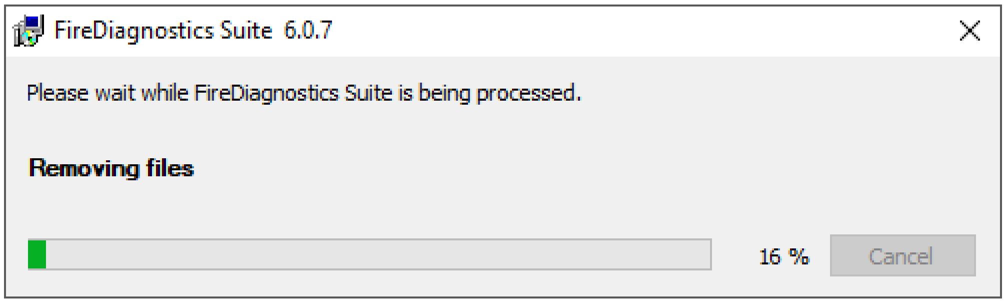
FireDiagnostics Suite

FireDiagnostics Suite 6.0.7 detected.

This application must be uninstalled before continuing.

Click OK to uninstall it now, or click Cancel to cancel the installation process.

The following dialog will show uninstallation progress. Please wait until it finishes.



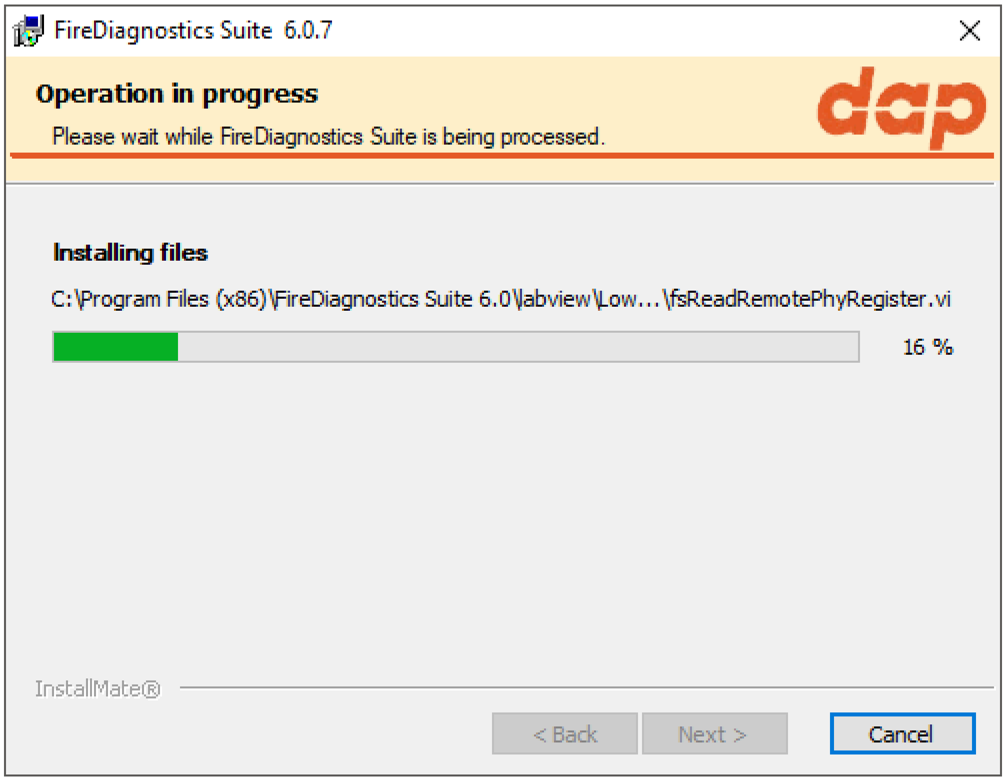
FireDiagnostics Suite 6.0.7

Please wait while FireDiagnostics Suite is being processed.

Removing files

16 %

The following dialog will show installation progress. Please wait until it finishes.



FireDiagnostics Suite 6.0.7

Operation in progress

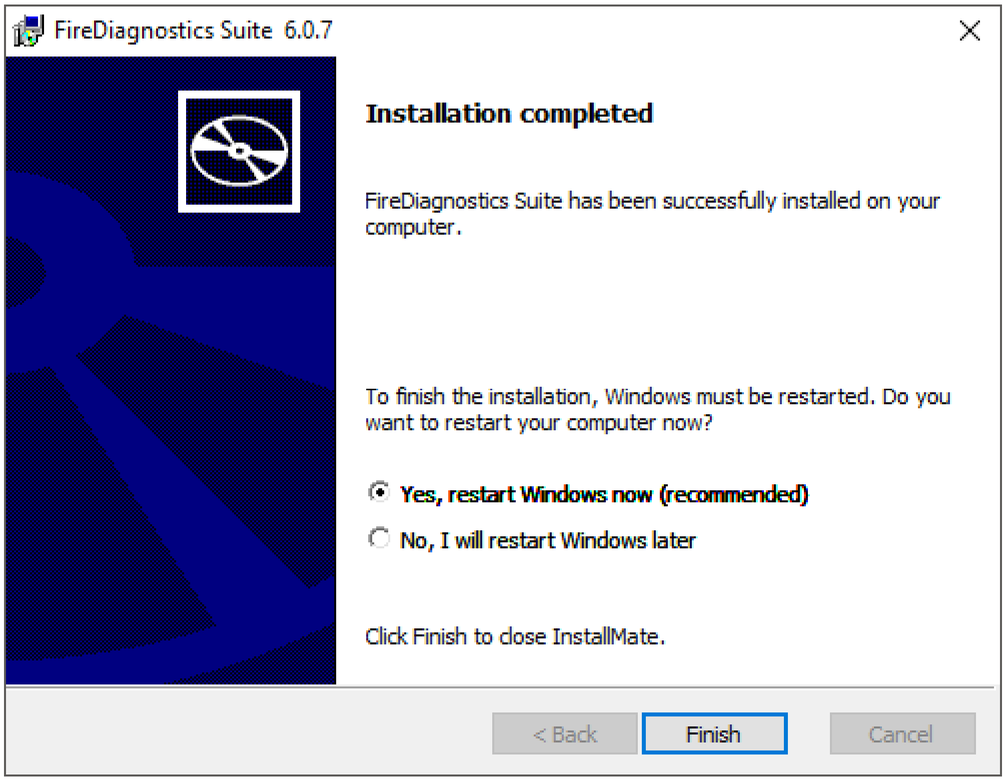
Please wait while FireDiagnostics Suite is being processed.

Installing files

C:\Program Files (x86)\FireDiagnostics Suite 6.0\labview\Low...\fsReadRemotePhyRegister.vi

16 %

After the installation is complete, the following dialog will appear. We recommend to always restart your computer after installing the FireDiagnostics software.


FireDiagnostics Suite 6.0.7

Installation completed

FireDiagnostics Suite has been successfully installed on your computer.

To finish the installation, Windows must be restarted. Do you want to restart your computer now?

[✓ Yes, restart Windows now (recommended)]

No, I will restart Windows later

Click Finish to close InstallMate.

2.1.2. Installing the Driver

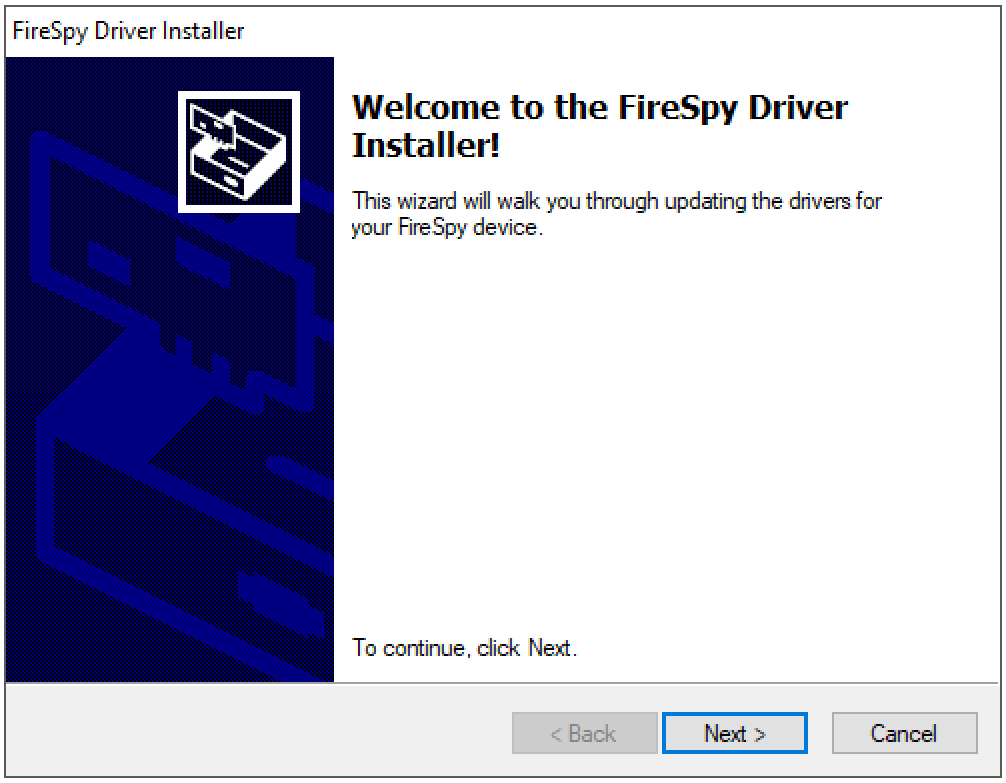
The driver installation will automatically start following the FireDiagnostics Suite installation, after the machine is rebooted. This installer will install the drivers for Windows 7 and Windows 10. Older Windows versions still work with the supplied drivers, but some additional steps may be required during installation. These steps are described in chapter Older Windows versions.

After the device driver is installed the device ready to be used. Please continue reading at the "Getting Started" section.

2.1.2.1. Windows Driver installer

At this point it is required to reboot the machine in order to install the device drivers. The device drivers will pre-allocate some memory that can be used for performing DMA operations and this needs to be done as early as possible after system startup. If done at a later point during system run it can take a very long time to load the driver.

After rebooting the machine, the following window will be displayed. Note that the software version could be different if you are installing another version of the software.

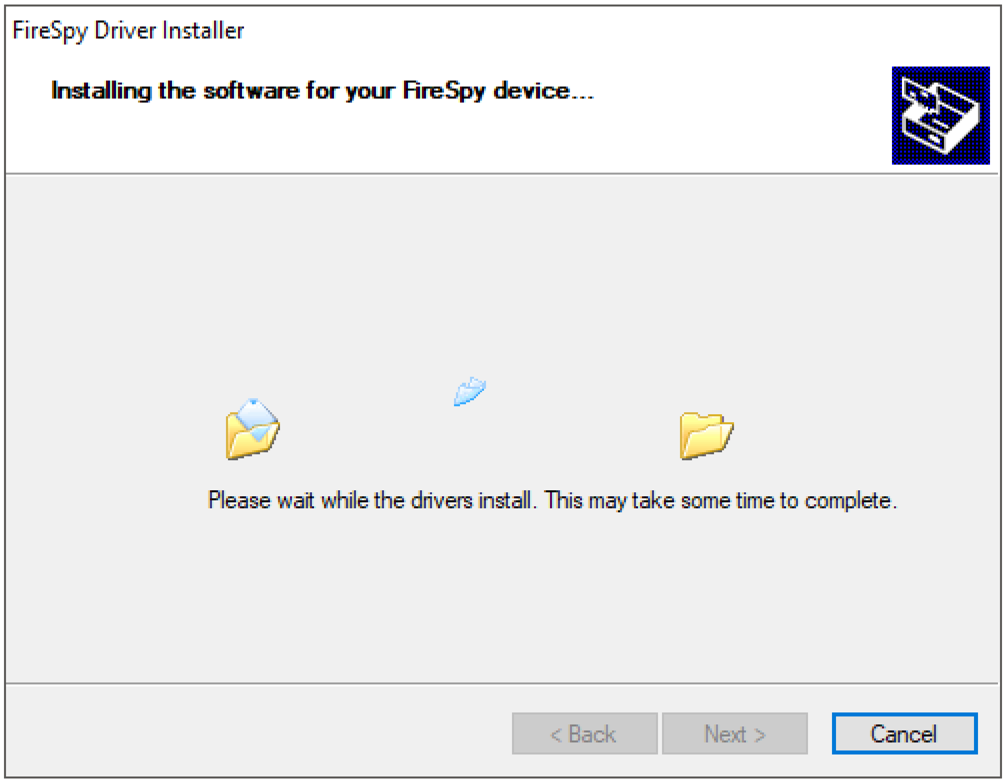

FireSpy Driver Installer

Welcome to the FireSpy Driver Installer!

This wizard will walk you through updating the drivers for your FireSpy device.

To continue, click Next.

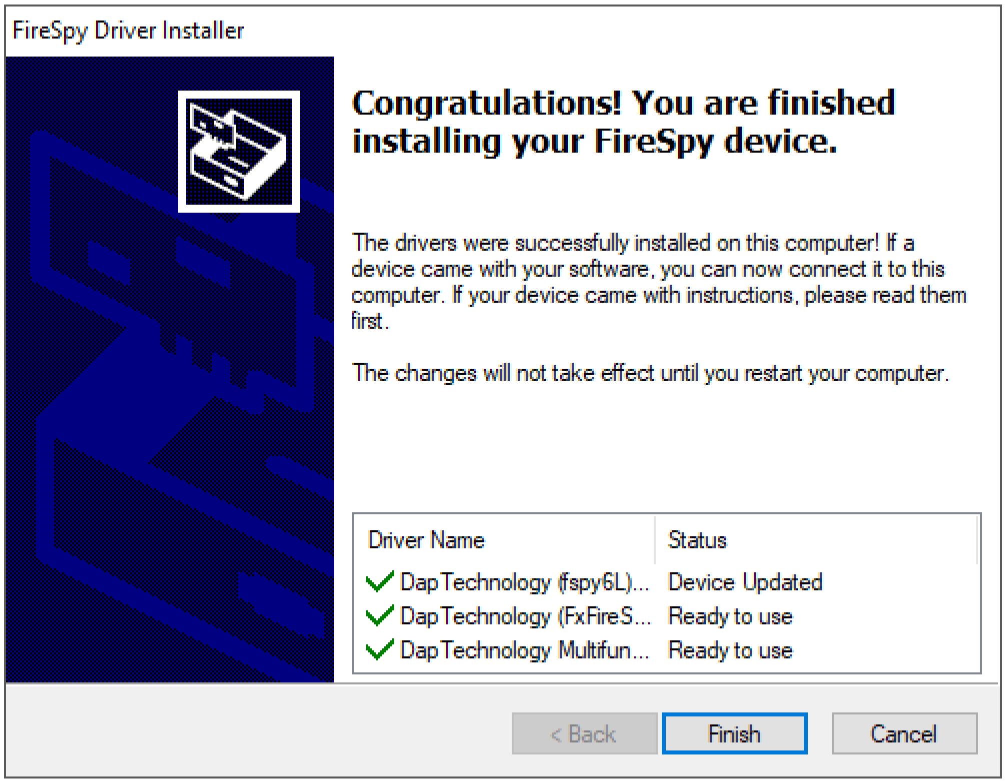
After clicking "Next", the drivers will be installed. Please wait until it finishes.


FireSpy Driver Installer

Installing the software for your FireSpy device...

Please wait while the drivers install. This may take some time to complete.

After the drivers are installed the following window will be showed. Click "Finish" to close the installation.



FireSpy Driver Installer

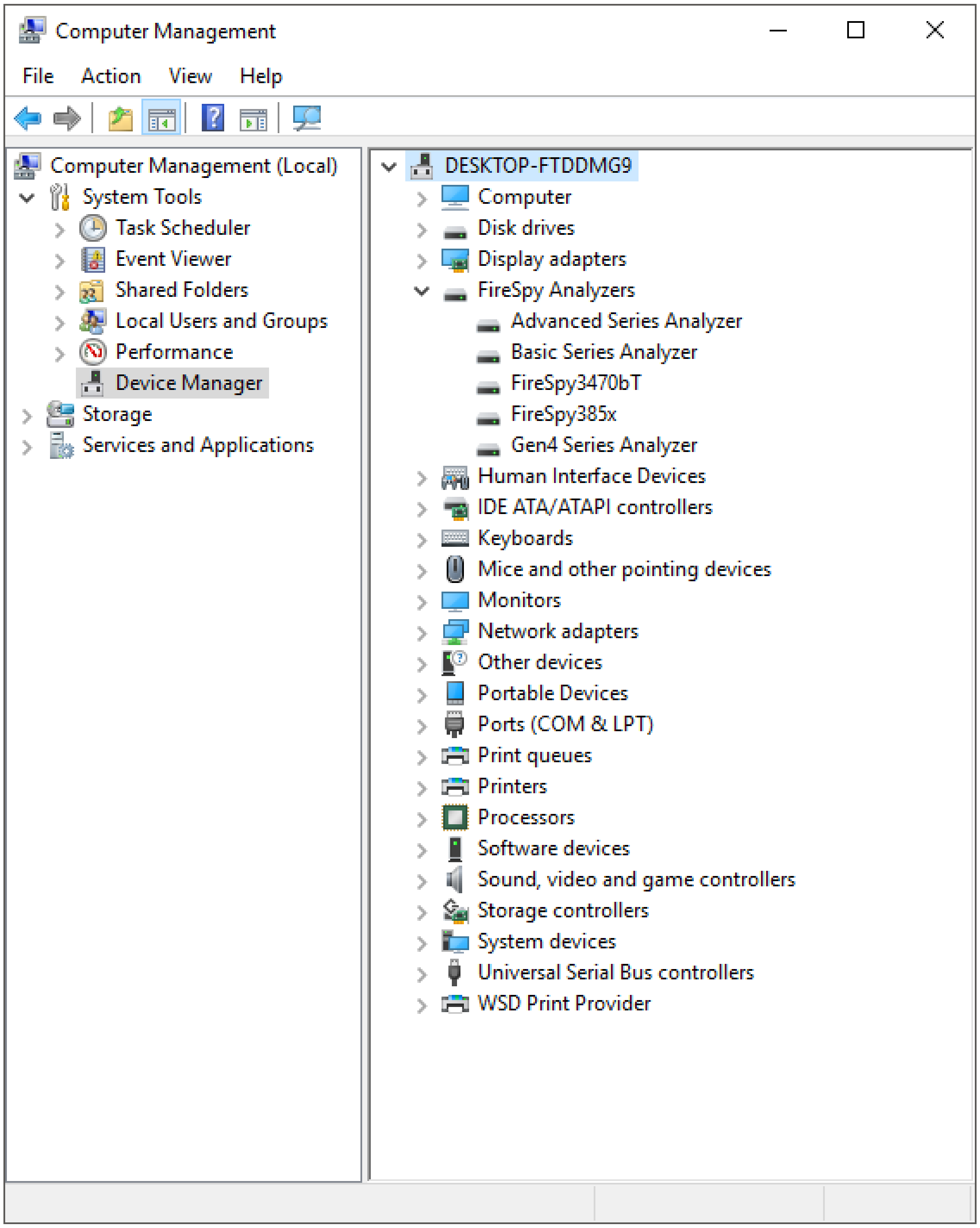
Congratulations! You are finished installing your FireSpy device.

The drivers were successfully installed on this computer! If a device came with your software, you can now connect it to this computer. If your device came with instructions, please read them first.

The changes will not take effect until you restart your computer.

Driver Name	Status
✓ DapTechnology (fspy6L)...	Device Updated
✓ DapTechnology (FxFireS...	Ready to use
✓ DapTechnology Multifun...	Ready to use

After your are booted to the desktop, you may check the device manager if the install has succeeded. The device manager will show a FireSpy under "Multifunction adapters". Here you find a several functions the device has.

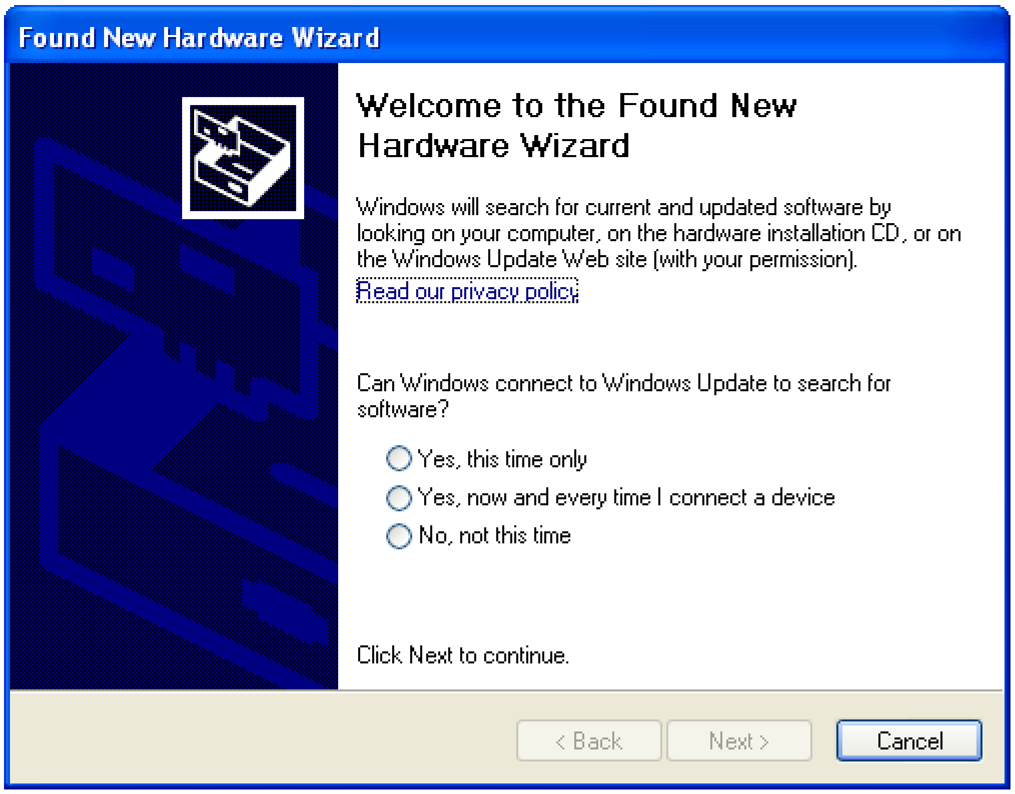


DESKTOP-FTDDMG9
- Computer
- Disk drives
- Display adapters
- FireSpy Analyzers
  - Advanced Series Analyzer
  - Basic Series Analyzer
  - FireSpy3470bT
  - FireSpy385x
  - Gen4 Series Analyzer
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Other devices
- Portable Devices
- Ports (COM & LPT)
- Print queues
- Printers
- Processors
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers
- WSD Print Provider

2.1.2.1.1 Older Windows versions

For Windows versions older than Windows 7 some additional steps during the Driver installation may be required.

After the FireDiagnostics Suite installation, and after rebooting the machine, the Driver installer will start. During the installation the Windows Driver Wizard may appear, as shown in the following image. Please click "Cancel" to close the wizard for now, we need to install the drivers first before Windows can search for them.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

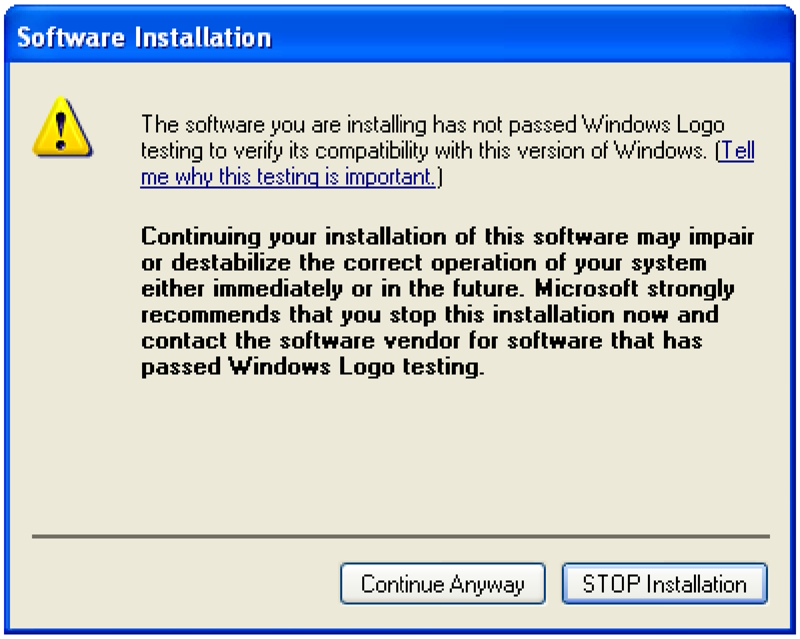
Read our privacy policy

Can Windows connect to Windows Update to search for software?

- Yes, this time only
- Yes, now and every time I connect a device
- No, not this time

Click Next to continue.

At the start of the installation the following message may be displayed, saying the software has not passed Windows Logo testing. The FireSpy Logo testing is not supported for Windows versions older than Windows 7, but function on these Windows version regardless. Click "Continue Anyway" to continue with the installation.

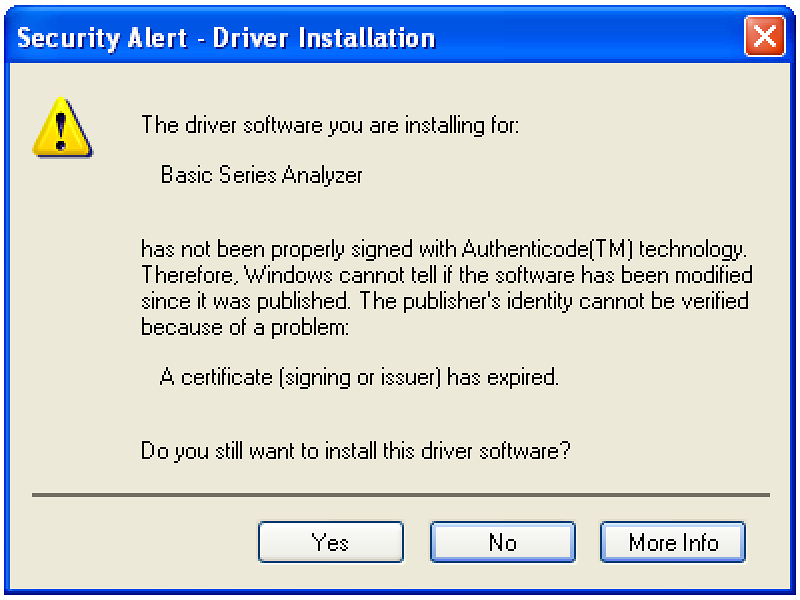


Software Installation

The software you are installing has not passed Windows Logo testing to verify its compatibility with this version of Windows. (Tell me why this testing is important.)

Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the software vendor for software that has passed Windows Logo testing.

During the installation of the Drivers, the following message may be displayed, saying the driver certificate has expired. As with the Windows Logo testing, the FireSpy driver signing is not supported for older Windows versions. Click "Yes" to continue installing the driver. This message may appear once or more during installation, depending on the number of FireSpy devices connected to the machine.



Security Alert - Driver Installation

The driver software you are installing for:

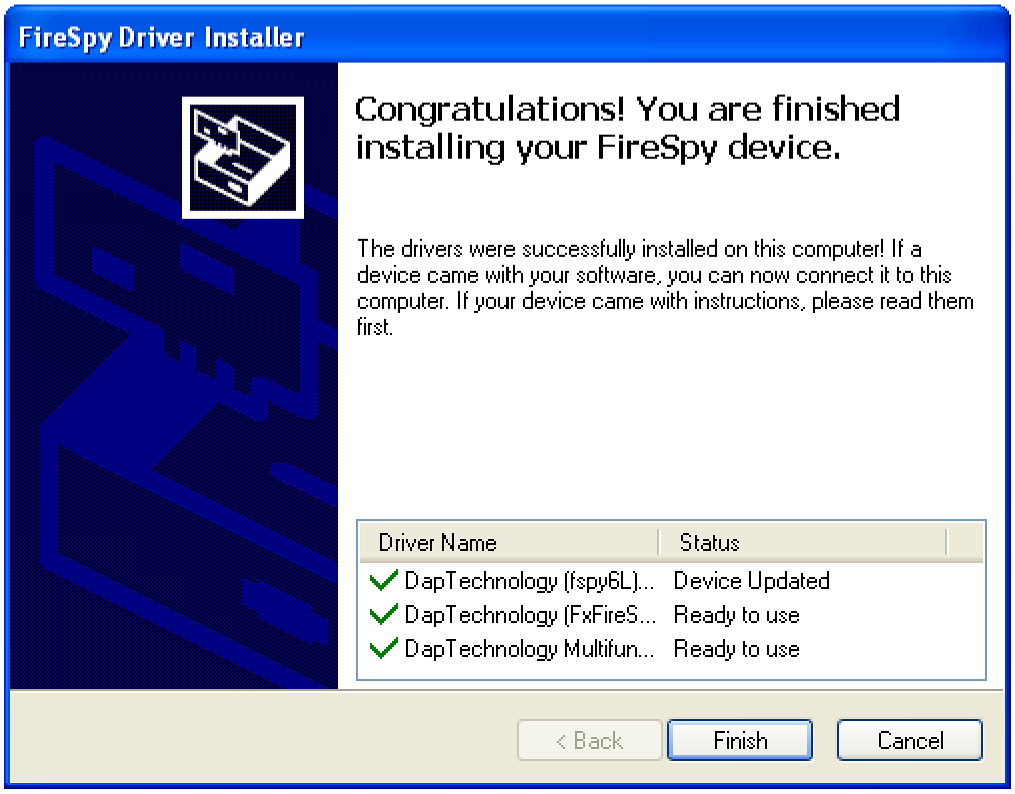
Basic Series Analyzer

has not been properly signed with Authenticode(TM) technology. Therefore, Windows cannot tell if the software has been modified since it was published. The publisher's identity cannot be verified because of a problem:

A certificate (signing or issuer) has expired.

Do you still want to install this driver software?

After the drivers are installed the following window will appear. Click "Finish" to continue starting Windows.



FireSpy Driver Installer

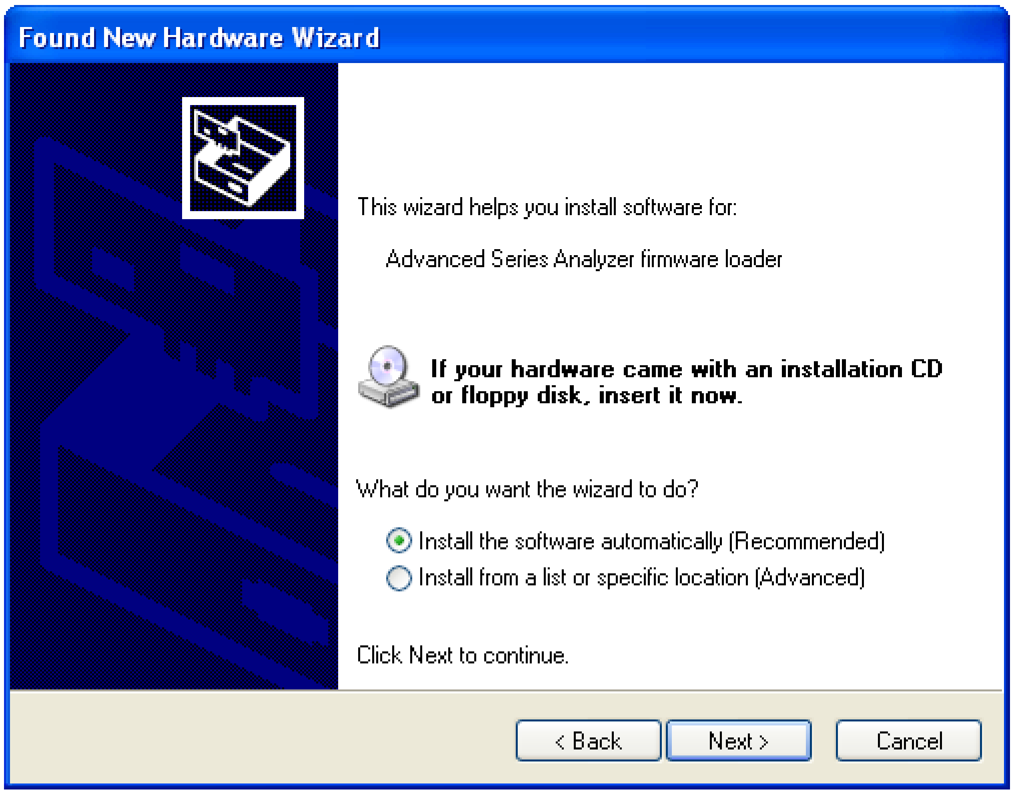
Congratulations! You are finished installing your FireSpy device.

The drivers were successfully installed on this computer! If a device came with your software, you can now connect it to this computer. If your device came with instructions, please read them first.

Driver Name	Status
✓ DapTechnology (fspy6L)...	Device Updated
✓ DapTechnology (FxFireS...	Ready to use
✓ DapTechnology Multifun...	Ready to use

Any devices that were connected to the machine during the installation should work now. If there were any unplugged devices, please connect them to the machine now. The Windows Driver Wizard mentioned earlier should appear now. If it doesn't please reboot the machine. In the Driver Wizard, select the bottom option "No, not this time" to search for the already installed drivers, and click "Next" to continue.

Select the top option "Install the software automatically" from the following dialog, to let Windows search for the installed FireSpy drivers.



Found New Hardware Wizard

This wizard helps you install software for:

Advanced Series Analyzer firmware loader

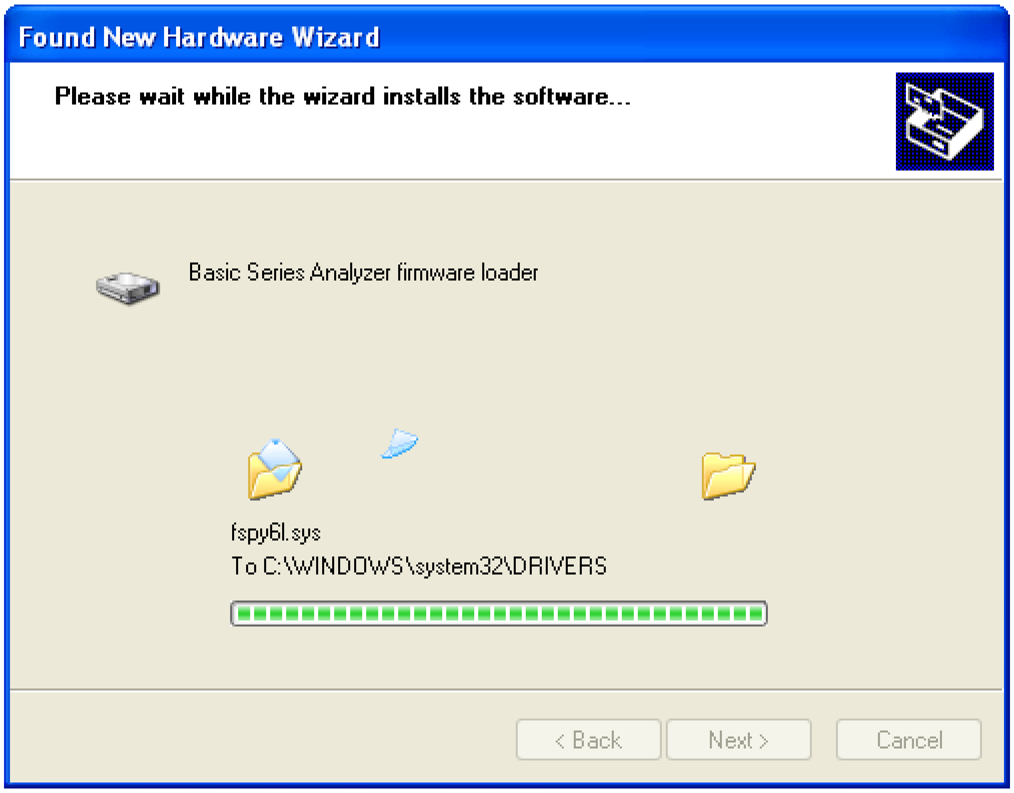
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

[✓ Install the software automatically (Recommended)]
Install from a list or specific location (Advanced)

Click Next to continue.

After clicking "Next" the Wizard will install the driver for the FireSpy device. Repeat this process for any additional FireSpy device that was not yet correctly installed during the installation.



Found New Hardware Wizard

Please wait while the wizard installs the software...

Basic Series Analyzer firmware loader

fspy6l.sys
To C:\WINDOWS\system32\DRIVERS

2.1.2.2. Manual driver (un)installation

Drivers will be uninstalled when uninstalling the FireDiagnostics Suite. The uninstaller can be found in the Windows Start menu, under FireDiagnostics Suite x.x -> Uninstall FireDiagnostics Suite (for Windows 10), or FireDiagnostics Suite x.x -> Administration -> Uninstall FireDiagnostics Suite (for older Windows versions).

For example, when switching between versions of the FireDiagnostics Suite it may be needed to install a specific version of the FireSpy drivers manually without installing/uninstalling the complete FireDiagnosticsSuite software packaged, you can use the Driver Installer. This can be found in the Windows Start menu, under FireDiagnostics Suite x.x -> Install Drivers (for Windows 10) or FireDiagnostics Suite x.x -> Drivers -> Install Drivers (for older Windows versions). A dialog will popup and it will install the drivers.

2.1.3. Ethernet Configuration

This chapter describes the configuration steps to be taken to enable Ethernet communication with a FireSpy x810. To communicate with a FireSpy x810 it should be connected to the company LAN (Local Area Network). Until now only the speed 10Mbit/s full is supported. In the future auto-negotiation and speed 100Mbit/s will also be supported.

The EthernetConfig tool programs the flash memory inside the FireSpy with the same firmware as during the startup phase of the device. Some companies have a policy that all programmable memory must be erased before it can be shipped elsewhere. To do this the Ethernet Configuration Tool has been given an option to erase this flash memory. It can be found under: Extra -> Erase Complete Flash.

2.1.3.1. Configuration steps FireSpy x810

To configure a FireSpy x810 correctly a prerequisite is that you have the following TCP/IP info:

- A free IP-address,
- the subnet mask
- and the IP-address of the default gateway.

Please, contact your system administration if you don’t have this information.

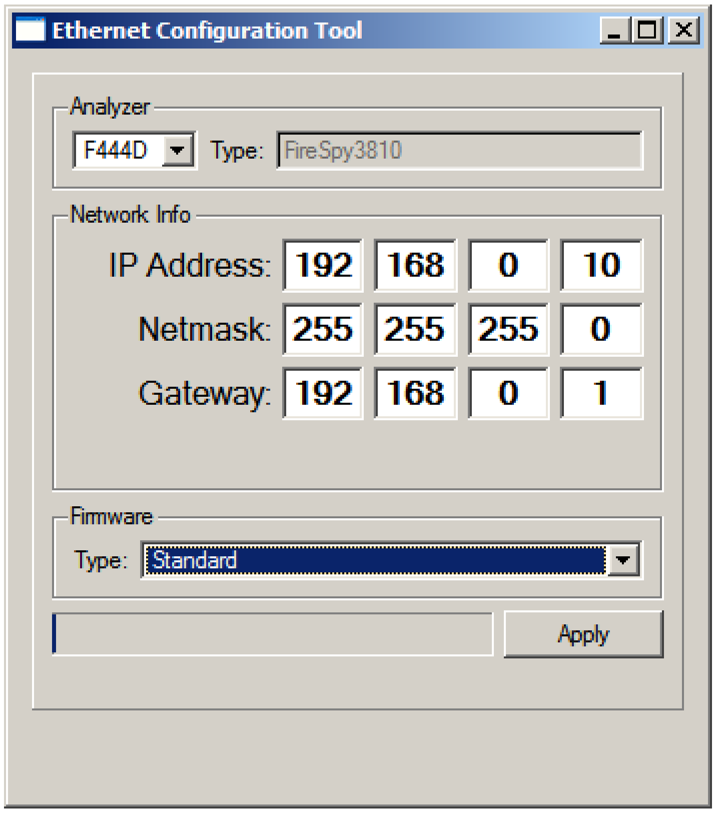
Follow the steps below to configure a FireSpy x810.

Step 1
Connect the FireSpy x810 to a PC via USB on which the FireDiagnostics Suite software is installed. Switch on the power of the FireSpy x810.

Step 2
Start the application ‘FireSpy Configuration Tool’ which is part of the FireDiagnostics Suite Software. You can find this application via the Start menu button under menu FireDiagnostics Suite x.x.

Step 3
Fill in the following TCP/IP info:
- the FireSpy x810 IP-address in field ‘IP-address’,
- the subnet mask in field ‘subnet mask’
- and the IP-address of the default gateway in field ‘Gateway’.

Step 4
Press button ‘Apply’. The filled in TCP/IP info and the firmware will be written respectively to eeprom and flash of the FireSpy x810. The progress of writing firmware is shown by the progress bar, see figure below.



Analyzer: F444D Type: FireSpy3810

IP Address: 192 168 0 10
Netmask: 255 255 255 0
Gateway: 192 168 0 1

Firmware Type: Standard

Step 5
Wait until the firmware is written. Switch off the power and disconnect USB.
Switch on the power and verify that led 'init' is on and turns off in about 8 seconds. If so the FireSpy x810 is configured successfully. Otherwise please repeat the above steps.

The FireSpy x810 is now able to start-up autonomously from flash. The written firmware to flash enables communication with the FireSpy x810 via Ethernet.
Please, note that a FireSpy x810 will not start-up from flash when it's connected via USB.

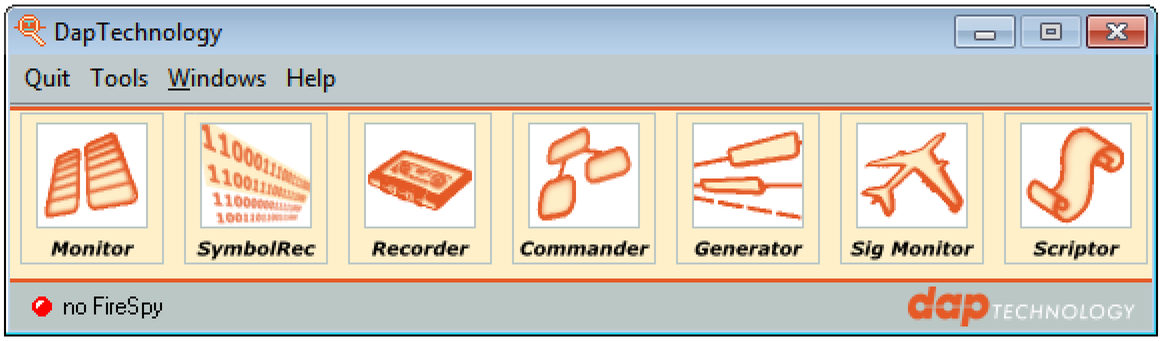
To communicate with the configured FireSpy x810 you should connect it to the company LAN and configure the FireSpy application to search for the configured FireSpy x810. To configure the FireSpy application, see the following chapter.

2.1.3.2. Configuration steps FireSpy application

In order to communicate with a FireSpy x810 connected to the company LAN the FireSpy application must know its IP-address.

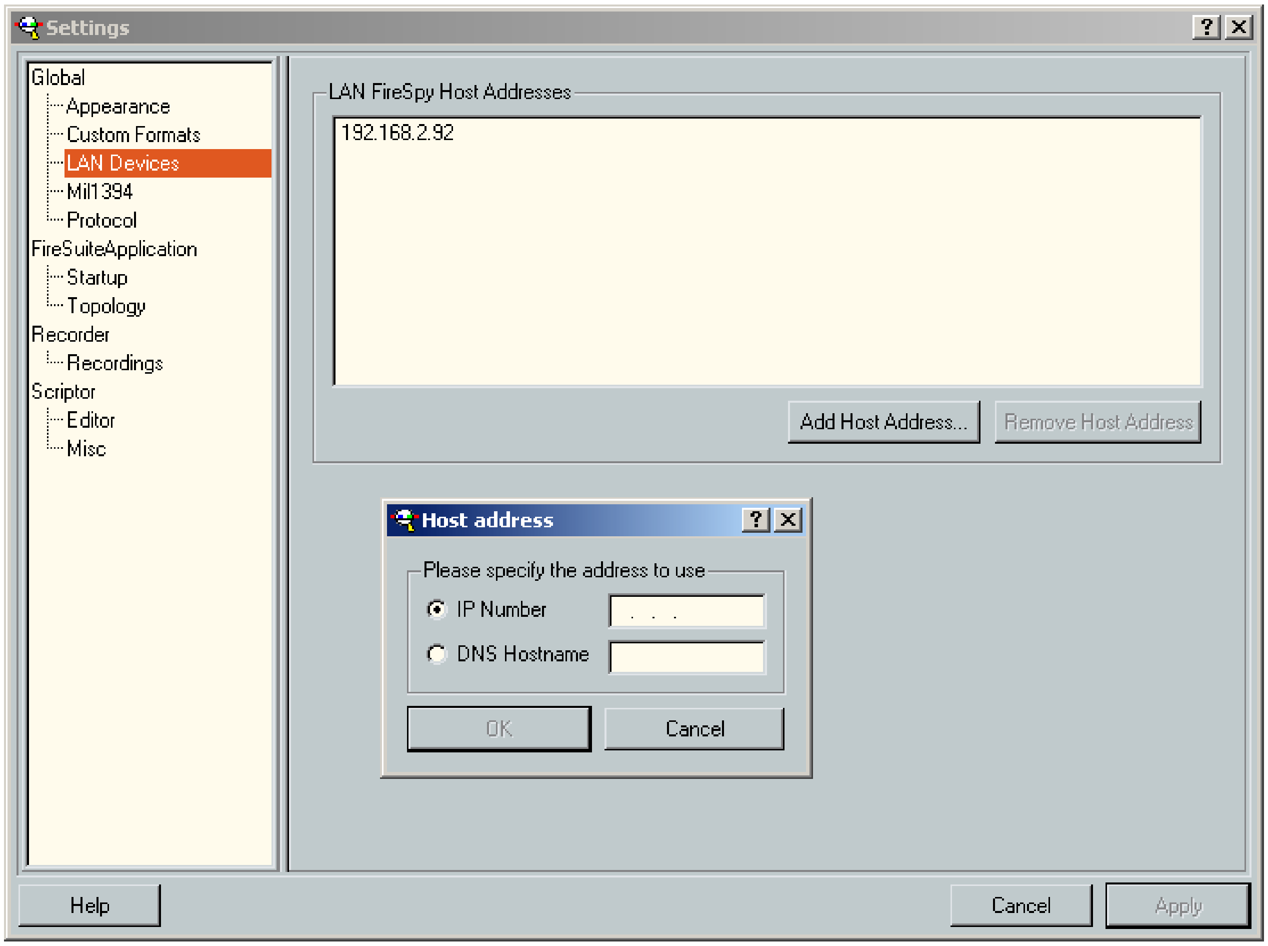
To set-up the FireSpy x810 IP-address please follow the steps below.

Step 1
Start the FireSpy Application. You can find this application via the Start menu button under menu FireDiagnostics Suite x.x. It will either issue a warning or show the FireSpy chooser dialog. This is not important at this time, just continue until the main screen shows up as in the picture below: (the green LED will be red in your case, indicating that no analyzer is connected)



Step 2
From the "Tools" menu open ‘Settings’ and click on item ‘LAN Devices.

Step 3
See figure below. Click ‘Add Host address...’ and fill in either the IP number or the DNS host address that is routed to the FireSpy. Make sure the address is valid and click ‘Ok’ when finished.



192.168.2.92

[✓ IP Number] DNS Hostname

Step 4
Now press the apply button and close the dialog.

Step 5
Please proceed with the Getting Started section to open the newly configured device.

Chapter 3. Getting Started

This section of the manual provides a quick-start guide for your analyzer and software. If you need help installing the application, please take a look at the Installation.

3.1. Starting the Application

If you haven't already done so, install the software as described in Installation. Make sure the devices are properly installed in or connected to the computer, have a matching power source and are turned on (where applicable). Run the FireSpy application from the Windows Start menu -> Programs -> FireDiagnostics Suite.

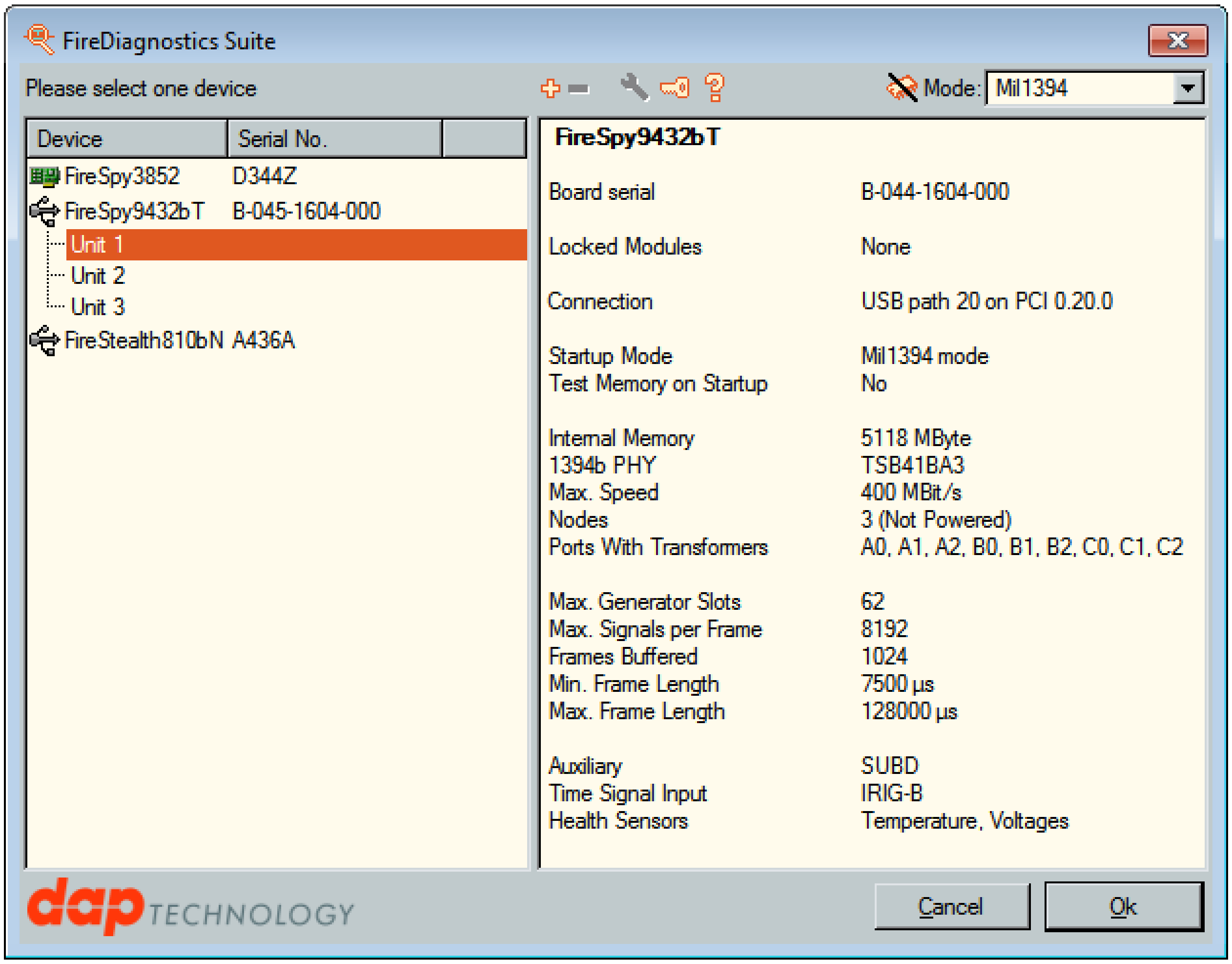
Device Selection
The device selection dialog will pop up, as shown in the picture below. The list on the left side shows the serial numbers of the currently connected devices, as well as their IP addresses or DNS hostnames in case they are connected by ethernet. All registered ethernet addresses are shown, including those of disconnected devices.

Plug and play devices, such as USB and ethernet devices, can be connected or disconnected without restarting the application or resetting the device. They will be automatically added or removed from the list within a few seconds. Ethernet devices first have to be configured, as explained in the next section, and their host address has to be manually added to the list.

The toolbar on the upper right side of the dialog can be used to modify the startup settings of the device. Ethernet addresses can be added or removed using the + and - buttons on the left of the toolbar. Mil1394, Normal or Symbol Recorder mode (when licensed) can be selected by expanding the Mode combo box on the right. Left of the Mode combo box, a tool button exists for enabling or disabling the memory test that is performed after the firmware is uploaded. This tool button is marked by a chip icon, and can be found next to the Mil1394 tool button.

The right side of the device selection dialog has three possible configuration modes, which are explained in further detail below. Apart from the default device information mode, the tool button with the wrench icon can be clicked to enter the ethernet configuration mode and the tool button with the key icon enters or exits the license configuration mode. Finally, the tool button with the question mark can be clicked to open the manual relevant to the current mode. In the default (device information) mode, the device manual of the selected device is opened. In the other two modes, either the ethernet configuration manual or the license configuration manual will be opened.

In some cases, for instance when starting the Monitor application directly, multiple analyzers can be selected at once. Sometimes it is also possible to start the application without selecting any devices (note that clicking on a selected item de-selects it). In that case, you can still select Mil1394 mode, for instance when you wish to use Mil1394 functionality in the Scriptor while no analyzer is available.



Device	Serial No.
FireSpy3852	D344Z
FireSpy9432bT	B-045-1604-000
Unit 1	
Unit 2	
Unit 3	
FireStealth810bN	A436A

FireSpy9432bT

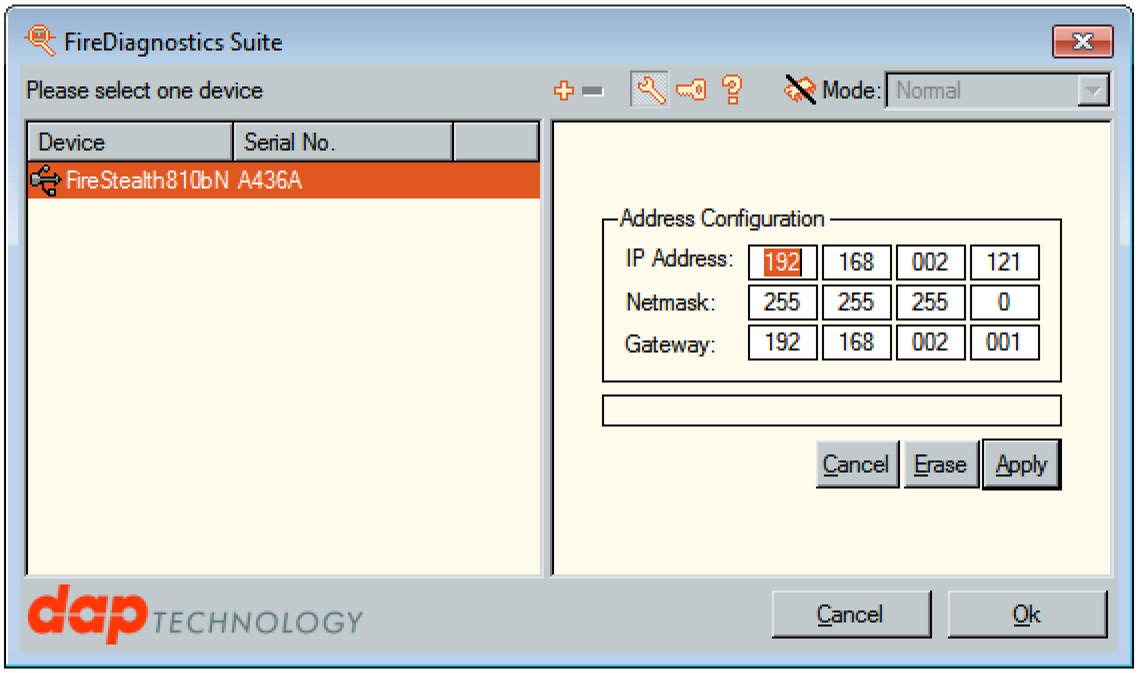
Board serial	B-044-1604-000
Locked Modules	None
Connection	USB path 20 on PCI 0.20.0
Startup Mode	Mil1394 mode
Test Memory on Startup	No
Internal Memory	5118 MByte
1394b PHY	TSB41BA3
Max. Speed	400 MBit/s
Nodes	3 (Not Powered)
Ports With Transformers	A0, A1, A2, B0, B1, B2, C0, C1, C2
Max. Generator Slots	62
Max. Signals per Frame	8192
Frames Buffered	1024
Min. Frame Length	7500 µs
Max. Frame Length	128000 µs
Auxiliary	SUBD
Time Signal Input	IRIG-B
Health Sensors	Temperature, Voltages

No Devices Shown - IMPORTANT NOTICE
In case you are sure a FireSpy is properly connected but it is not shown in the Device Selection dialog, it may be needed to manually install the drivers. This could also be needed when multiple versions of the FireDiagnostics Suite are installed and you want to switch between them. Please refer to the section about Manual Driver Instalation.

Ethernet Configuration
Devices equipped with an ethernet connection can be configured by clicking the tool button with the wrench icon. This enters the ethernet configuration mode, as shown below. When the IP data are loaded, you can change it and press the Apply button to store the new configuration. The tool button with the aircraft icon can be used to configure the ethernet interface in Mil1394 mode or not. The Flash memory can be erased (for secure transport) by clicking the Erase button. The operation can be safely canceled by pressing the 'Cancel' button. However, the ethernet interface will not be operational until it has been fully configured.

Extra options will be shown when a standalone analyzer can enable or disable the power on specific buses on startup.

Note that a device's ethernet interface cannot be opened when it has been loaded by another version of the application. More information about the ethernet configuration can be found in the Ethernet Configuration Manual.



Device	Serial No.
FireStealth810bN	A436A

Address Configuration

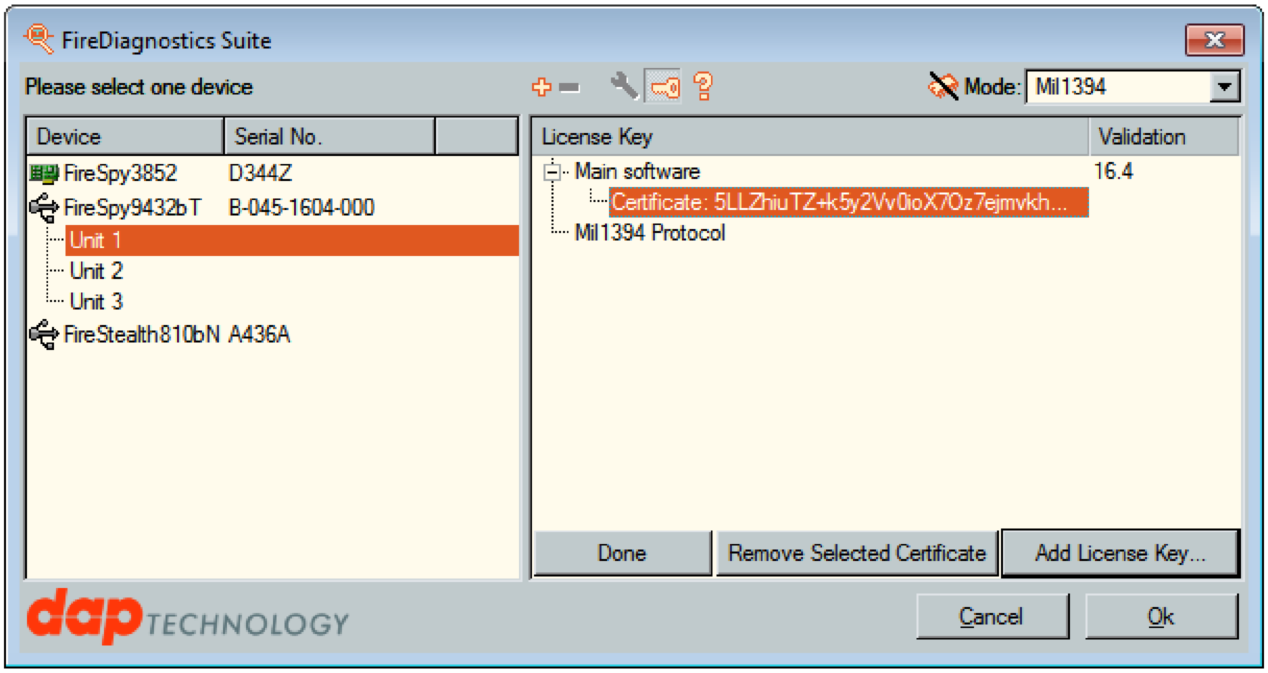
IP Address:	192	168	002	121
Netmask:	255	255	255	0
Gateway:	192	168	002	001

License Management
You can enter the license configuration mode by clicking the toolbutton with the key icon, and go back to the device information mode by clicking it once again (or by clicking the 'Done' button). The license configuration mode shows the main license validation number and the licensed modules and their respective keys. In case of a temporary license, the number of days that the license will still be valid is also shown.

To add a new key, press the 'Add key...' button. This will pop up a dialog asking for a device serial number (which is already filled in when one device is selected) and a license key. To remove a key, expand the corresponding module in the tree view on the right, select the license key and press the 'Remove Selected Key' button.

It is possible to add or remove a license for a device that is not currently connected. In the license configuration mode, the left pane lists the licenses that have been registered by their serial number and a key icon. Disconnected ethernet devices are not shown. A serial can be added by clicking the 'Add key...' button and changing the serial number.

For further information about license configuration, see the License Manager manual.



Device	Serial No.
FireSpy3852	D344Z
FireSpy9432bT	B-045-1604-000
Unit 1	
Unit 2	
Unit 3	
FireStealth810bN	A436A

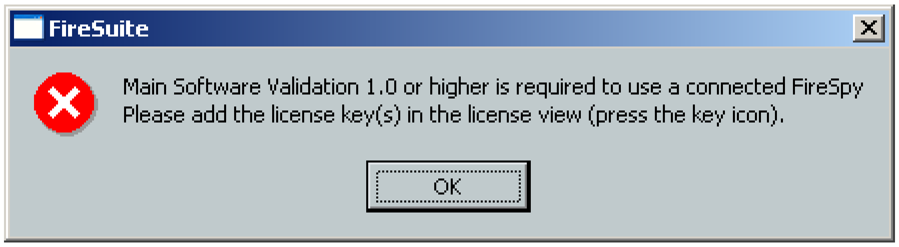
License Key	Validation
Main software	16.4
Certificate: 5LLZhiuTZ+k5y2Vv0ioX7Oz7ejmvkh...	
Mil1394 Protocol	

After Device Selection
If you installed the drivers correctly, the application will recognize the connected analyzer and start configuring it. After configuration, the analyzer is ready for use, the Main Window will be displayed.

Multiple Applications Using the Same Analyzer
It is possible to open the same analyzer from within different applications, however, all main functions like the monitor, recorder, e.o. can only be opened once per analyzer. To prevent main functions from being opened by multiple applications, they are locked as soon as you start the corresponding window.

For example, it is possible to open two instances of the application, which both use the same analyzer. One instance could use the monitor and the other instance could use the recorder. Of course it would be easier to just start one instance of the analyzer application and just open the monitor and the recorder, but other applications using the FireDiagnostics Suite API may access the device as well.

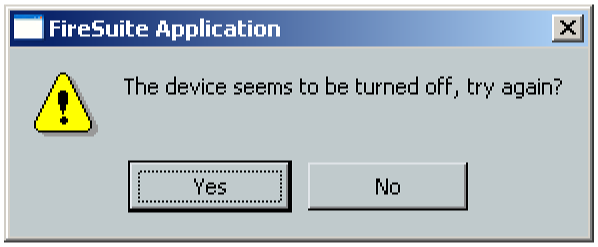
No License
If you connected an analyzer correctly, but you have no license key installed with a validation number high enough to unlock this software version, the following message will be displayed: (validation number depends on software version)



Main Software Validation 1.0 or higher is required to use a connected FireSpy
Please add the license key(s) in the license view (press the key icon).

Please click on the OK button. The application will continue without using an analyzer or return to the device selection dialog. You will need a license key for version 1 or higher to be able to work with a connected device. If you want protocol support, you will need a license key for version 2 or higher and a license key for each protocol you want to have supported. See the license management explanation above on how to install a license.

Analyzer Turned Off
If you connected a analyzer, but it is not connected to the power supply or the power is switched off, the following message will be displayed:

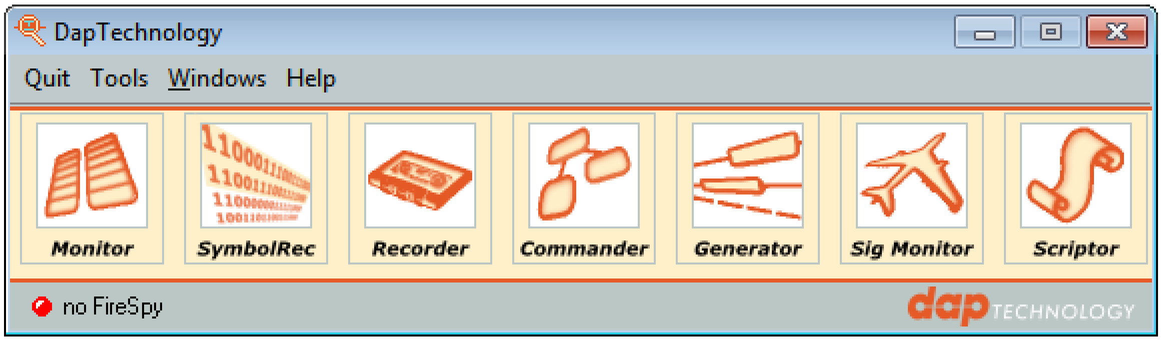

FireSuite Application

The device seems to be turned off, try again?

You may now connect the power supply and/or switch the power on, and try again. Or you may choose to run the application without an analyzer.

3.2. Main Window

The main window consists of the menus and the control area. Below is a screenshot of the main window without a FireSpy connected. This shows all module buttons available. Please note that when used with a specific analyzer model some module may not be available.



Menus

Quit

Exit the application by closing the main window or selecting 'quit' from the menu at the top of the main window. All windows from the same application instance will be closed.

Tools

From here you can open the Settings Dialog and the Filter/Trigger Settings.

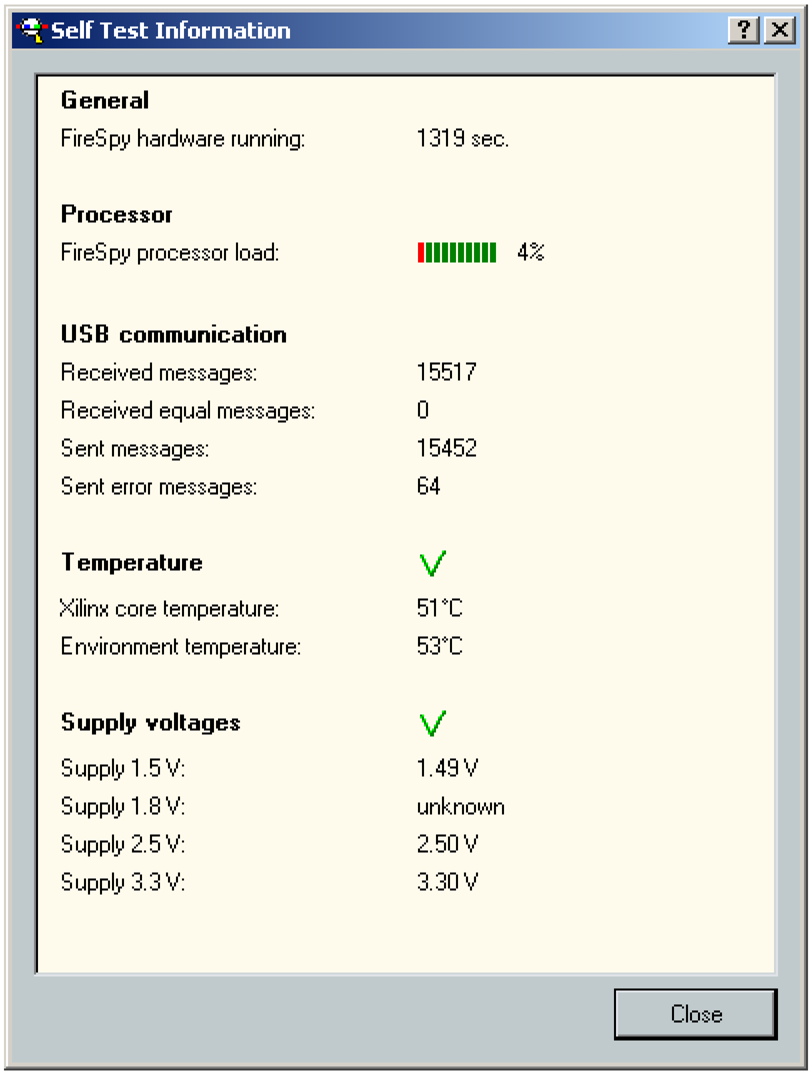
The License Manager is used to manage the possible licenses for the FireDiagnostics Suite. License keys can be added to unlock new features and protocol analyses.

Windows

From the 'Windows' menu or the buttons in the main window area you can open the Monitor, Recorder, Generator, Commander and Scriptor, respectively.

Help

Using the 'About' command in the 'Help' menu you can get information about the hardware status and the current software version. It also displays the web site where to get additional information or to download new software versions and where to get support. To display the results of the Power-On Self Test and the current hardware status, select 'Self test Information' from the menu. The following dialog will be displayed:



General

FireSpy hardware running:	1319 sec.

Processor

FireSpy processor load:	4%

USB communication

Received messages:	15517
Received equal messages:	0
Sent messages:	15452
Sent error messages:	64

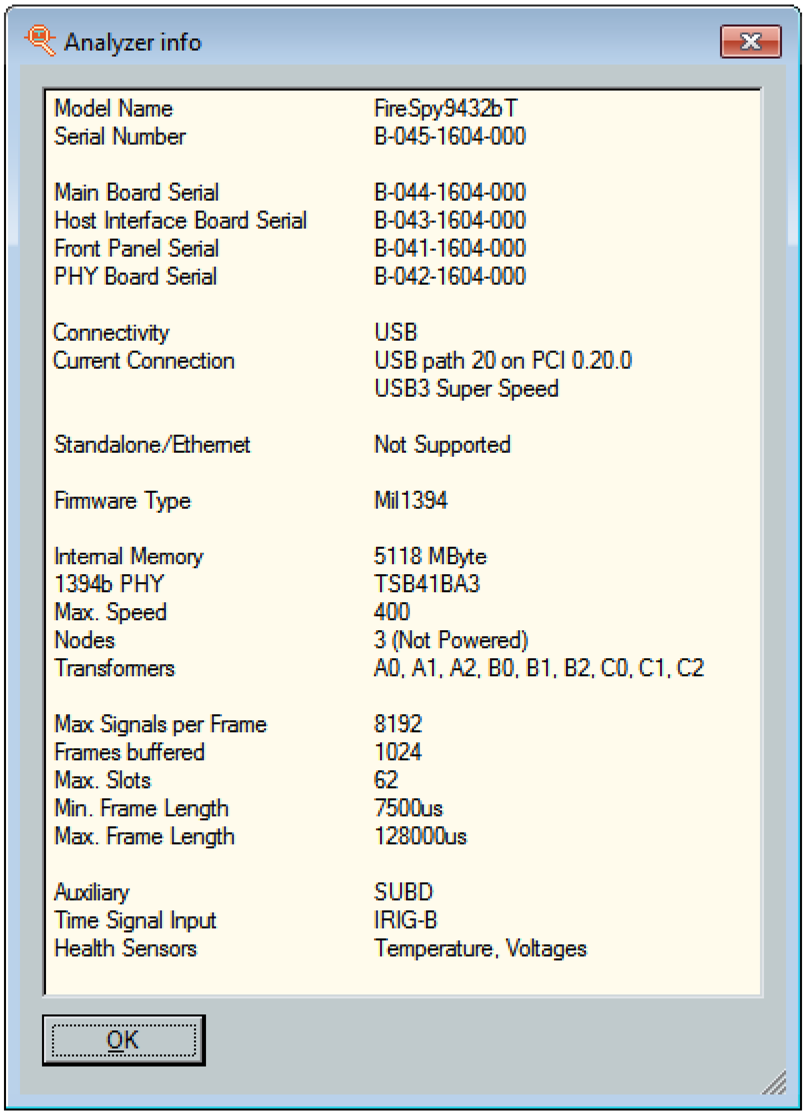
Temperature ✓

Xilinx core temperature:	51°C
Environment temperature:	53°C

Supply voltages ✓

Supply 1.5 V:	1.49 V
Supply 1.8 V:	unknown
Supply 2.5 V:	2.50 V
Supply 3.3 V:	3.30 V

Selecting 'Analyzer Info' from the Help menu opens a window with information about the analyzer(s) currently in use:



Model Name	FireSpy9432bT
Serial Number	B-045-1604-000
Main Board Serial	B-044-1604-000
Host Interface Board Serial	B-043-1604-000
Front Panel Serial	B-041-1604-000
PHY Board Serial	B-042-1604-000
Connectivity	USB
Current Connection	USB path 20 on PCI 0.20.0 USB3 Super Speed
Standalone/Ethernet	Not Supported
Firmware Type	Mil1394
Internal Memory	5118 MByte
1394b PHY	TSB41BA3
Max. Speed	400
Nodes	3 (Not Powered)
Transformers	A0, A1, A2, B0, B1, B2, C0, C1, C2
Max Signals per Frame	8192
Frames buffered	1024
Max. Slots	62
Min. Frame Length	7500us
Max. Frame Length	128000us
Auxiliary	SUBD
Time Signal Input	IRIG-B
Health Sensors	Temperature, Voltages

Controls
The windows for the main functions (Monitor, Recorder, Generator, Commander and Scriptor) can also be opened by clicking on the corresponding button. The button will be highlighted when the mouse is positioned above it.

The led in the left-bottom corner will display the current configuration status. It can be:

- Gray, indicating that the analyzer is not yet completely configured.
- Green, indicating that the analyzer is configured successfully and ready for operation.
- Yellow, indicating a FireStealth analyzer that lost sync.
- Red, indicating that no analyzer was selected or configured successfully. In this mode, you still can open, save and analyze files.

3.3. Connecting to an IEEE1394 Bus

To analyze an IEEE1394 bus, you can connect one of the analyzer's IEEE1394 ports to any free port of an existing bus, or you may disconnect an existing IEEE1394 cable between two ports, and insert the analyzer in-between the two disconnected ports by using another cable.

In both cases you should connect the analyzer to the existing bus such that it can 'see' all nodes at their full speed. If you connect the analyzer in such a way that some node has one or more connections between itself and the analyzer that has a speed lower than the maximum speed this node can handle, then some packet may not be picked up by the analyzer. The analyzer will detect a 'hidden' (or prefix only) packet instead, which are counted by the Monitor. The Recorder will know the duration of the hidden packets, but it does not know at which speed the packet has been sent or how many bytes it consists of.

If you see the Monitor or Recorder detecting these packets, you can take a look at the topology view in the Commander to find the problem.

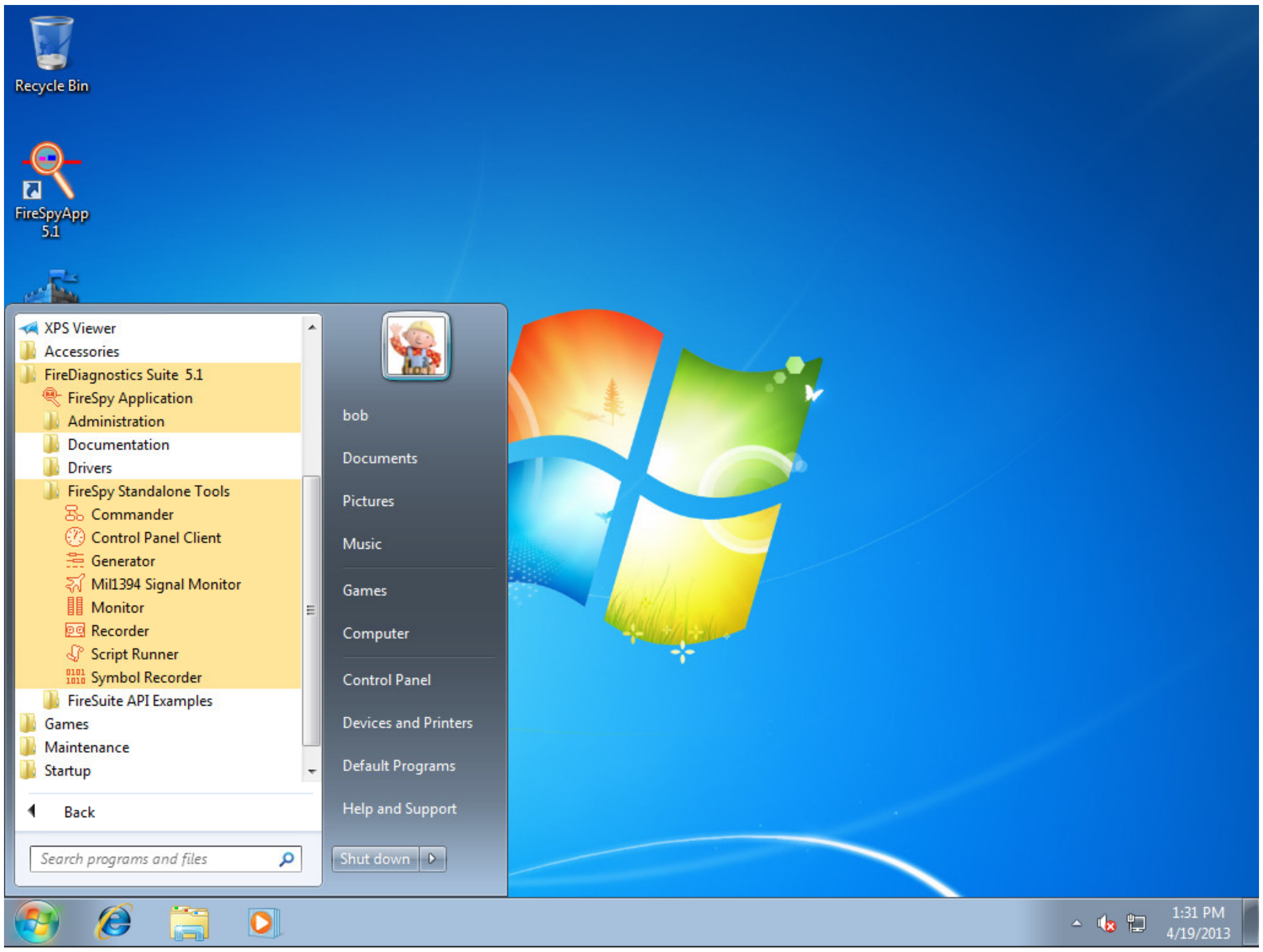
3.4. Command Line Options

Since the 3.3.0 release the FireDiagnostics Suite supports a number of command line options. These command line options make it possible to select an analyzer by its serial number, start the Scriptor with the provided script, open a recording or start the Scriptor directly. To quickly view which options are supported, enter the command: FireSpyApp.exe -h (or use /? for the same result). This will show a message with an overview of the supported arguments:

- '-h' or '/?' shows the command line help.
- '-d <SERIALNUMBER>' is used when multiple analyzers are connected to the computer, and a specific device should be opened.
- filename; when a filename is provided to the application, it recognizes the extension and opens the associated module, For example: FireSpyApp recording.fss starts the Scriptor because the .fss extension is associated with the Scriptor. Multiple files may be supplied, separated by spaces.
- '-r start' or '--recorder start', starts the Recorder.
- '-r export' or '--recorder export', opens up the Recorder's export dialog. If an export script and recording file are specified, the application will export the recording and terminate afterwards.
- '-g start' or '--generator start', starts the Generator. Use this command in combination with a generator file to load and start the Generator.
- '-s start' or '--scriptor start', in combination with a supplied script, will load the script into the scriptor and run instantly.
- '-n' or '--noFireSpy' starts the application without opening a device. This is useful when you only have one analyzer connected and want to view or edit files without interfering with it.

3.5. FireSpy Standalone Tools

In addition to the main FireSpy application with all main FireSpy modules embedded there is also a way to start each module as a standalone application. These standalone versions can be started through the Windows start menu and then navigating to "All Programs", "FireDiagnostics Suite X.X" and then "FireSpy Standalone Tools". The picture below shows where to find the standalone versions.



Multiple standalone FireSpy tools can be started at the same time, but each module can only be started once per FireSpy. The following standalone FireSpy tools allow controlling multiple FireSpy devices from within a single application:

- Recorder
- Monitor
- Symbol Recorder

All other modules only allow controlling one FireSpy device from a single application.

Chapter 4. License Manager

To be able to use the software with a connected analyzer, you will need one or more license keys, at least for the 'Main Software', with the minimal version needed for the application. If the application is started with an analyzer selected, but without the appropriate license key, an error message will be shown and the application will continue without live analyzer support.

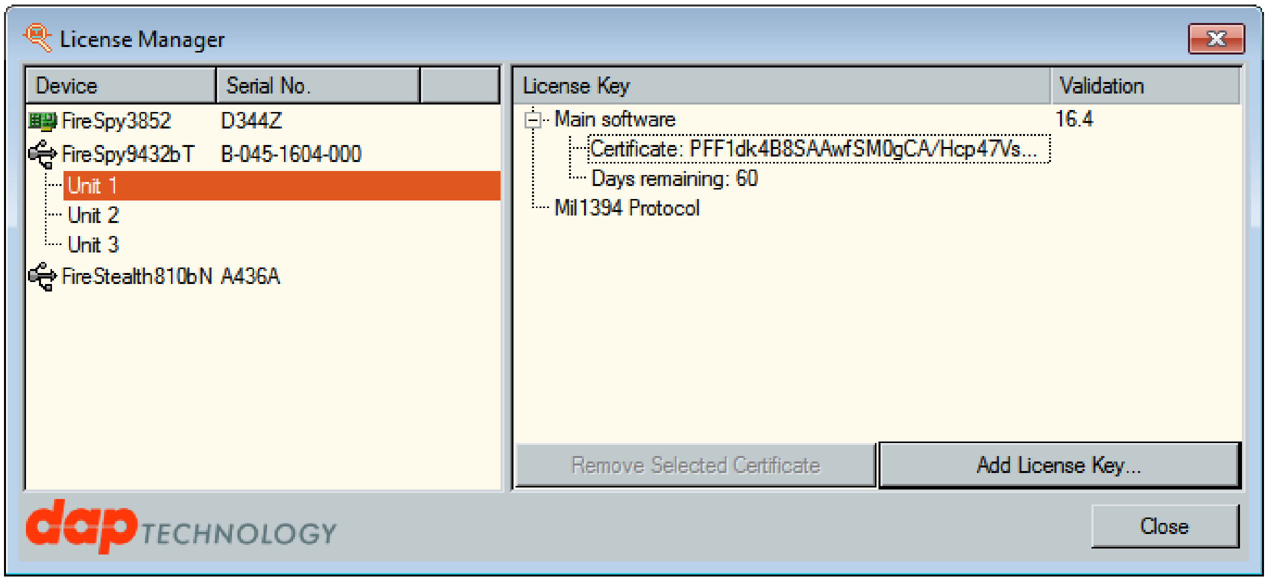
An additional license key is required for each higher-level protocol you want to use. The License Manager is used to install and remove such license keys. You can see which keys are installed and, if applicable, how many days it will remain valid.

License keys are linked to the serial number of the connected analyzer, so if the device is not connected, all license keys will be inactive and the related features will be disabled. However, when you open a recorder file in the Recorder that has been created using an analyzer with some installed license keys, all these license keys become active as long as you have loaded this file. Thus you will be able, for instance, to use the SBP protocol analyzer to analyze a recorder file made by somebody who used an analyzer with SBP license key, without having any license keys yourself.

4.1. How to use it

You can open the License manager dialog by selecting the ‘License Manager' function in the ‘Tools’ menu of the Main Window. Or from the Windows Start menu at FireDiagnostics Suite -> Administration. Licenses can also be configured in the Startup Dialog.

An example of the License Manager dialog window is shown below. In this example the serial number of one of the connected analyzer is B-045-1604-000. A license certificate is installed for this device which contains a 'Main Software (version 16.4)’ license and a license for the Mil1394 protocol. The License certificate is only valid for another 60 days. The icons next to the serial numbers indicate the connection type (if connected) or whether it is a license for a device that is currently offline (key icon).


License Manager

Device	Serial No.
FireSpy3852	D344Z
FireSpy9432bT	B-045-1604-000
Unit 1	
Unit 2	
Unit 3	
FireStealth810bN	A436A

License Key	Validation
Main software	16.4
Certificate: PFF1dk4B8SAAwfSM0gCA/Hcp47Vs...	
Days remaining: 60	
Mil1394 Protocol	

You can add keys with the 'Add License Key...' button or remove them by selecting a key and clicking the ‘ Remove Selected Key’ button. You will need to restart the application to activate the new keys, except in the startup dialog; licenses added there can be immediately used.

Any existing keys for the same serial number and module will be permanently overwritten.

Please contact DAP Technology to obtain additional license keys.

Chapter 5. Monitor

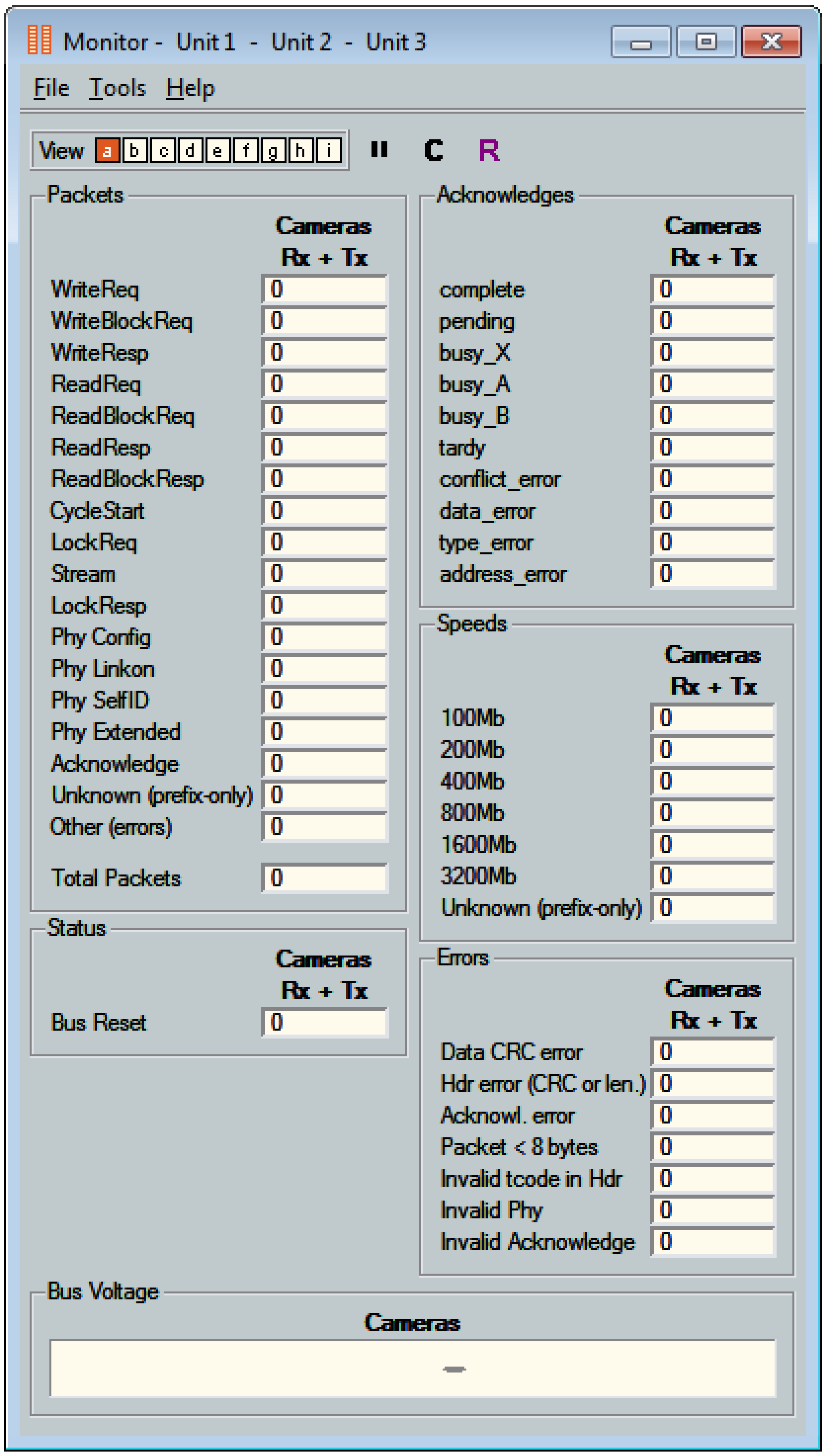
The Monitor gives a quick indication of bus activity. The Monitor displays the number of packets passed on the bus and some additional events. The packets are differentiated to speed and type, including all isochronous, asynchronous, phy packets and acknowledge packets types. Also the number of packets with different kinds of errors and the IEEE1394 bus voltage are monitored and displayed.

5.1. How to use it

You can start the Monitor by clicking on the Monitor button in the main window, or select the Monitor from the 'FireSpy' menu at the top of the main window or one of the other open windows. It is also possible to start a standalone monitor from the start menu, this monitor will have the ability of showing the busses from multiple analyzers in one window.

When you start the Monitor, the window below will appear.

If you connected the Analyzer to an IEEE1394 bus, some counters may start running indicating bus activity. The 'Bus Voltage' indicates the voltage between the Cable power and the Cable ground pins. They may also indicate a value other then 0.0 volt when connected to a powered bus.



Packets

	Cameras Rx + Tx
WriteReq	0
WriteBlockReq	0
WriteResp	0
ReadReq	0
ReadBlockReq	0
ReadResp	0
ReadBlockResp	0
CycleStart	0
LockReq	0
Stream	0
LockResp	0
Phy Config	0
Phy Linkon	0
Phy SelfID	0
Phy Extended	0
Acknowledge	0
Unknown (prefix-only)	0
Other (errors)	0
Total Packets	0

Acknowledges

	Cameras Rx + Tx
complete	0
pending	0
busy_X	0
busy_A	0
busy_B	0
tardy	0
conflict_error	0
data_error	0
type_error	0
address_error	0

Speeds

	Cameras Rx + Tx
100Mb	0
200Mb	0
400Mb	0
800Mb	0
1600Mb	0
3200Mb	0
Unknown (prefix-only)	0

Status

	Cameras Rx + Tx
Bus Reset	0

Errors

	Cameras Rx + Tx
Data CRC error	0
Hdr error (CRC or len.)	0
Acknowl. error	0
Packet < 8 bytes	0
Invalid tcode in Hdr	0
Invalid Phy	0
Invalid Acknowledge	0

Bus Voltage

Cameras
-

File Menu

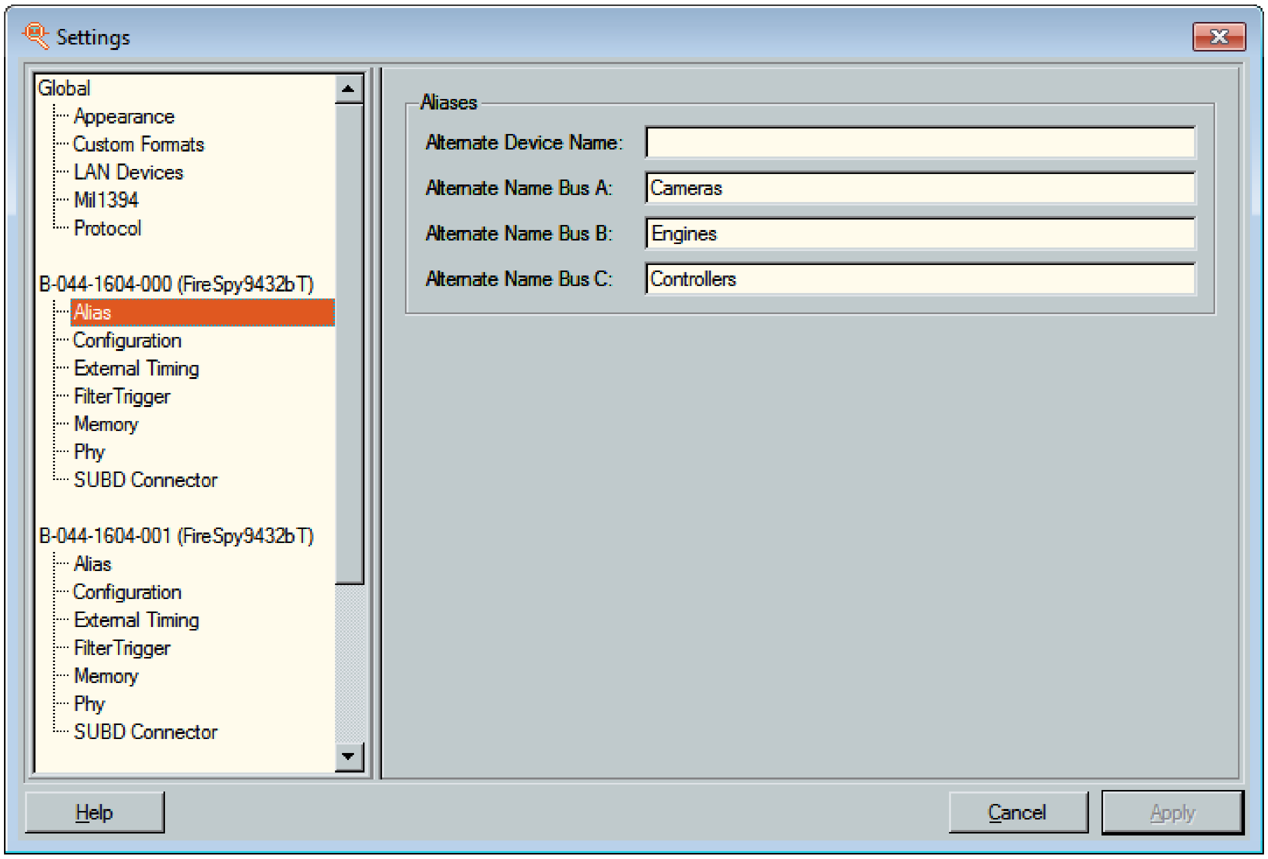
Export...
The export menu allows for saving current counter values to an external file. The file will be stored as a comma seperated spreadsheet (csv) including column headers.

Tools menu

License Manager
The license manager can be opened when running the monitor in the standalone version. The Monitor that is integrated within the application will not have this option.

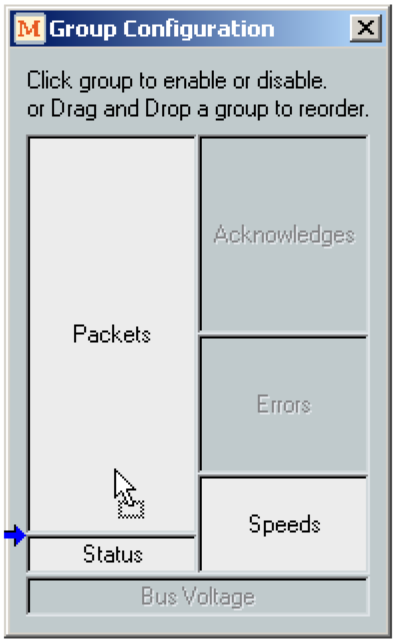
Settings

This option will open the settings dialog. The Monitor can use alternate device and/or bus names, they can be changed on the alias tab. (See screenshot)

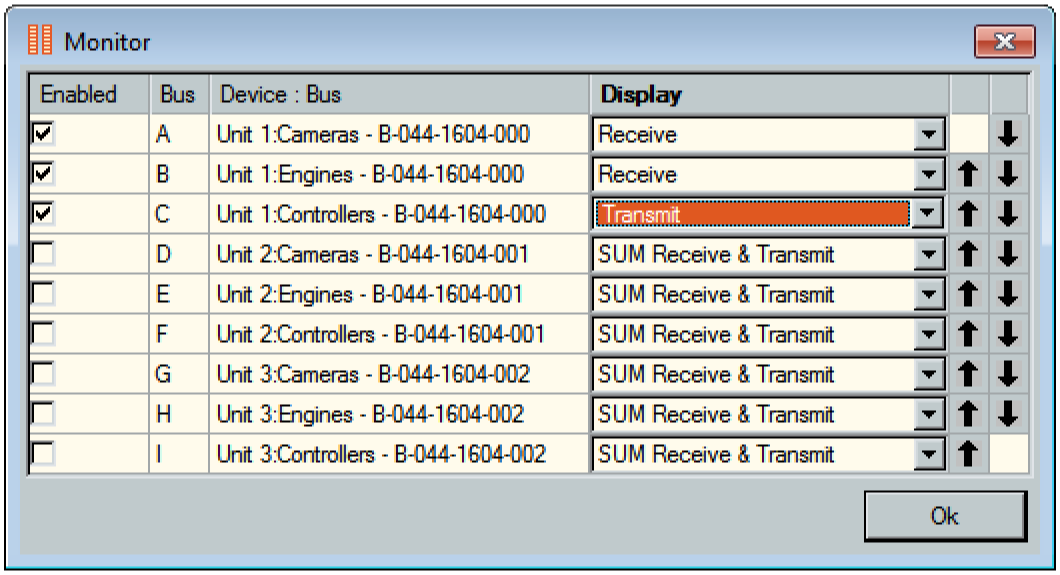


Alternate Device Name:
Alternate Name Bus A: Cameras
Alternate Name Bus B: Engines
Alternate Name Bus C: Controllers

Window Configuration
The Monitor can use quite some desktop space, to reduce this space you can make a selection of the monitor groups you want to monitor. To disable a section just click its group in the group selection box. To reorganize the Monitor groups you can drag a group by selecting it and then move it to the location where you want it. In the picture below, the Speeds group is dragged on the lower part of the packets group. This results in Speeds being placed between Packets and Status.
A Configuration can be saved as a preset in the tools menu by using Load or Save Preset.

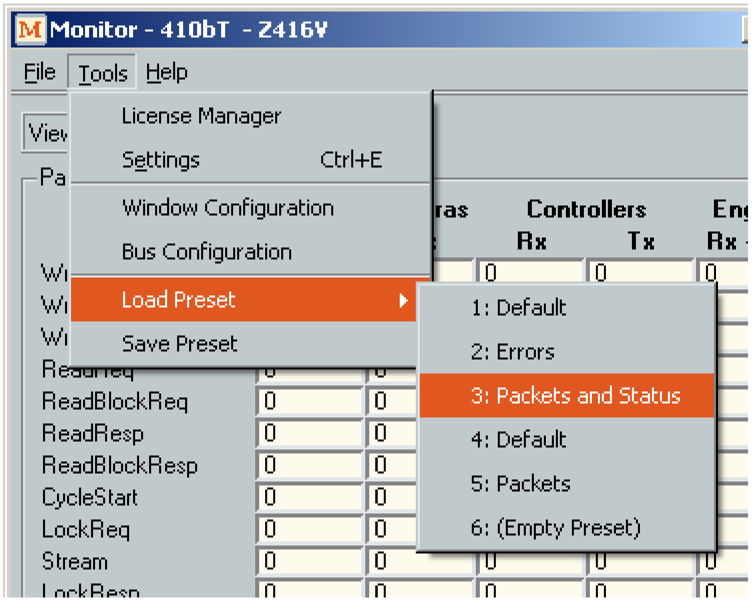

Group Configuration
Click group to enable or disable.
or Drag and Drop a group to reorder.
Packets
Acknowledges
Errors
Speeds
Status
Bus Voltage

Bus Configuration
The Bus Configuration dialog has two functions; reorder the busses and most important, select how to display incoming and/or outgoing packets. For stealth analyzers it indicates packets going from 'pipe A to B' or from 'pipe B to A'. As a user you have four options:
-Receive. Only show packets that are received by the link layer in the Analyzer.
-Transmit. Only show packets that are transmitted by the link layer in the Analyzer.
-SPLIT Receive & Transmit. Use two columns for displaying monitored data
-SUM Receive & Transmit. The default, use one column and add received to the transmitted packet.



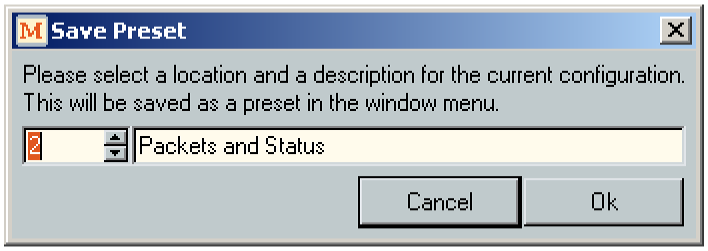
Enabled	Bus	Device : Bus	Display		
✓	A	Unit 1:Cameras - B-044-1604-000	Receive		↓
✓	B	Unit 1:Engines - B-044-1604-000	Receive	↑	↓
✓	C	Unit 1:Controllers - B-044-1604-000	Transmit	↑	↓
☐	D	Unit 2:Cameras - B-044-1604-001	SUM Receive & Transmit	↑	↓
☐	E	Unit 2:Engines - B-044-1604-001	SUM Receive & Transmit	↑	↓
☐	F	Unit 2:Controllers - B-044-1604-001	SUM Receive & Transmit	↑	↓
☐	G	Unit 3:Cameras - B-044-1604-002	SUM Receive & Transmit	↑	↓
☐	H	Unit 3:Engines - B-044-1604-002	SUM Receive & Transmit	↑	↓
☐	I	Unit 3:Controllers - B-044-1604-002	SUM Receive & Transmit	↑	

Load Preset
To store a defined group layout use the "Save Preset" menu option. These presets can be loaded in this Load Preset menu.



Save Preset

To save a preset, select this menu option. The following dialog will let you select a name and location for the preset.


Save Preset

Please select a location and a description for the current configuration. This will be saved as a preset in the window menu.

2 Packets and Status

Buttons

Pause/Resume display
Pressing this button will hold the updates to the monitor. When pressed again the updates will resume. Note that only the display is paused, not the monitoring itself.

Clear
When this button is clicked all packet and event counters described above will be cleared and start counting from 0 (zero) again. Note that only the visible nodes will be cleared.

Initiate Bus Reset
Pressing this button will generate a bus reset on all visible nodes. Nodes that are not selected will not initiate a bus reset.

5.2. Details

Packets
The 'Packets' box counts all the possible packets the Analyzer detects on the bus, divided into a number of groups.
It includes all primary packets: Write Request, Write Block Request, Write Response, Read Request, Read Response, Read Block Request, Read Block Response, Cycle Start, Lock Request, Stream and Lock Response.

It also includes all possible phy packets: Phy Configuration, Phy Link On, Phy SelfID and all the Extended Phy Packets.

It also includes all possible acknowledge packets and prefix-only packets. Prefix-only packets are packets that cannot be seen by the Analyzer because one or more connections between the node sending the packet and the Analyzer cannot handle the speed of the packet. The Analyzer can detect the existence of such a packet but it cannot see the data and thus the packet type is 'Unknown' for the Analyzer.

Note that on a IEEE1394b network prefix-only packets can also occur around a bus status. Sometimes a bus status event is send through the network using a null-packet. This packet will be seen by the Analyzer as a prefix-only packet. Normally this are very short prefix-only packets.
All packets not belonging to one of the groups above are considered error packets and are counted by the 'other (errors)' field.

Acknowledges
There is also a separate box for the acknowledges. All possible acknowledges that can be returned are counted separately. You can get a quick indication how the transactions are performing.

Speeds
The Speeds box counts again all detected packets, but this time differentiated to their speed. Because the speed of prefix-only packets is unknown, there is a separate field for prefix_only packets in the Speeds section.

Errors
The Errors box gives the user a quick indication of the existence of erroneous packets. It handles the following kind of errors:

Data CRC error
All packets that have a data block and with an error detected in the data CRC are counted here. Note that only packets that have a valid header (correct header CRC and valid tcode) can have a data block, and thus packets with data CRC error are not counted as 'error' packets in the Packets section.

Hdr error (CRC or len.)
All packets that have a valid tcode, but a header CRC error are counted here. Also when the CRC could not be calculated because the packet was to small, it will be counted here. These packets will always be bigger than or equal to 8 bytes and will not be phy packets. They are counted as 'error' packets in the Packets section.
Note that packets with invalid tcode have unknown header size, and thus the header CRC cannot be calculated. Therefore packets with invalid tcode are counted separately (see below).

Acknowledge error
All packets of exactly one byte, but with an error in the inverse check, are counted here.

Packets < 8 bytes
All packets smaller than 8 bytes, but not exactly one byte (acknowledge packets) are counted here.

Invalid tcode in Hdr
Packets with invalid tcode (a tcode that is not allowed) are counted here. These packets will always be bigger than or equal to 8 bytes and will not be phy packets. They are counted as 'error' packets in the Packets section.
Note that packets with invalid tcode have unknown header length, and thus the header CRC cannot be calculated.

Invalid Phy
All packets exactly 8 bytes (2 quadlets) long and with the second quadlet the inverse of the first quadlet are considered phy packets. If the Analyzer detects such a packet with invalid information such that the phy type is unknown, this count is incremented.

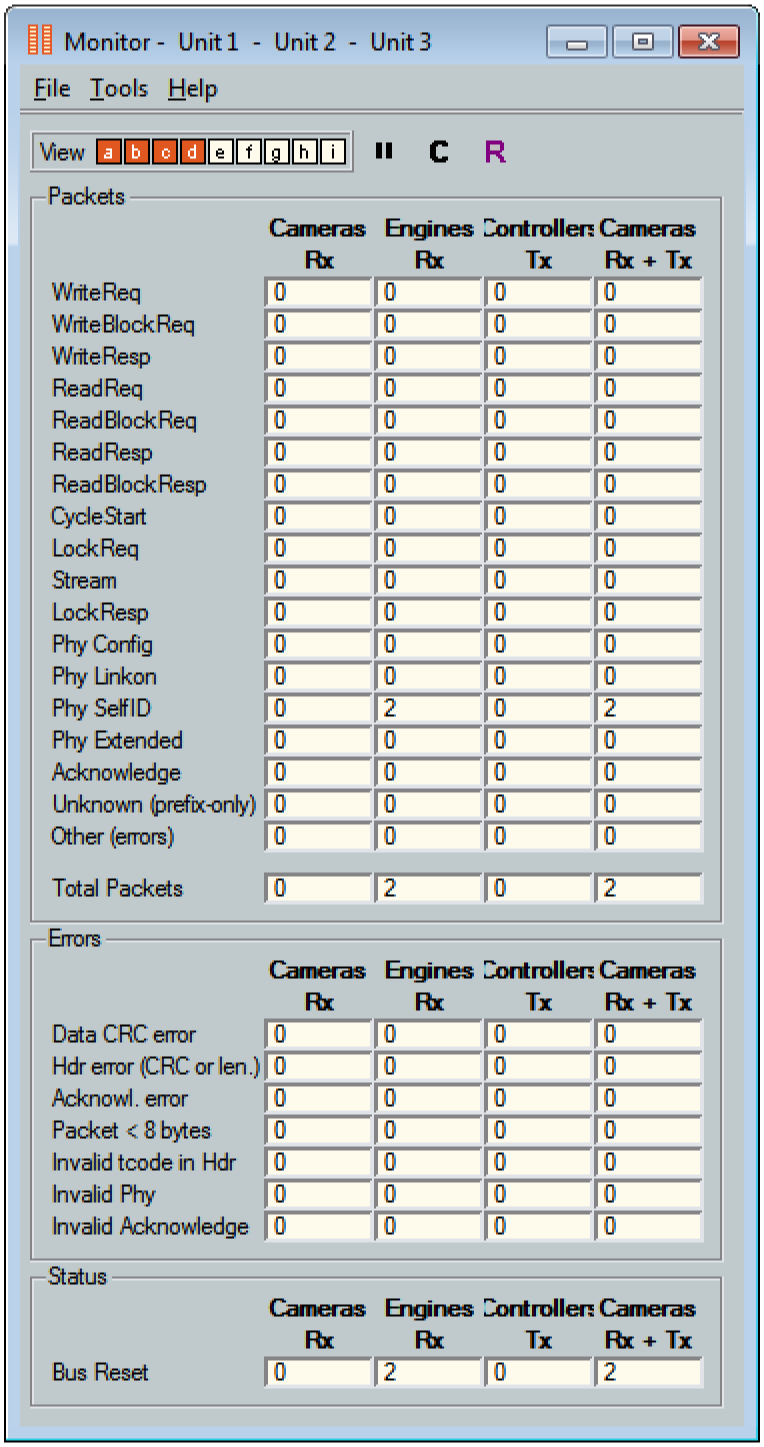
Invalid Acknowledge
All packets of exactly one byte with correct inverse check are considered correct acknowledge packets. If the Analyzer detects such a packet with invalid (not allowed) acknowledge code, this count is incremented.

Status
Here the number of bus resets is counted.

Bus Voltage
The Analyzer will measure the IEEE1394 bus voltage and display the value here in volts. The resolution is 0.1 volt.

5.3. Multi-bus Version

For multi bus analyzers, or when using the standalone Monitor (found in the tools menu), the monitor is capable of showing counter values for multiple buses. A multi bus Monitor can look as follows:



Packets

	Cameras Rx	Engines Rx	Controllers Tx	Cameras Rx + Tx
WriteReq	0	0	0	0
WriteBlockReq	0	0	0	0
WriteResp	0	0	0	0
ReadReq	0	0	0	0
ReadBlockReq	0	0	0	0
ReadResp	0	0	0	0
ReadBlockResp	0	0	0	0
CycleStart	0	0	0	0
LockReq	0	0	0	0
Stream	0	0	0	0
LockResp	0	0	0	0
Phy Config	0	0	0	0
Phy Linkon	0	0	0	0
Phy SelfID	0	2	0	2
Phy Extended	0	0	0	0
Acknowledge	0	0	0	0
Unknown (prefix-only)	0	0	0	0
Other (errors)	0	0	0	0
Total Packets	0	2	0	2

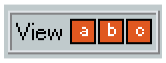
Errors

	Cameras Rx	Engines Rx	Controllers Tx	Cameras Rx + Tx
Data CRC error	0	0	0	0
Hdr error (CRC or len.)	0	0	0	0
Acknowl. error	0	0	0	0
Packet < 8 bytes	0	0	0	0
Invalid tcode in Hdr	0	0	0	0
Invalid Phy	0	0	0	0
Invalid Acknowledge	0	0	0	0

Status

	Cameras Rx	Engines Rx	Controllers Tx	Cameras Rx + Tx
Bus Reset	0	2	0	2

Node Selection
If the Monitor is used in multi-bus setup, it is possible to show the counters for just one node or several nodes at once. The toolbar contains the following control to select which nodes to display.



By clicking on the "a", "b" or "c" button, the corresponding node can be hidden or shown. A red background color means it is visible, a white background color means it is hidden.

Note that the bus reset button will cause a bus reset to be initiated on the selected nodes only.

Chapter 6. Commander

The Commander has four separate functions:

The 'Topology' is used to display the current bus topology with, optionally, more-detailed information about the nodes. Depending on node capabilities, one can suspend, resume, disable and enable a node. One can also generate bus resets here and display the new bus topology.

With the 'Phy Register R/W' function, one can read and write the registers of the local and read the registers of a remote phy interface. The register layout and a list of all register fields are shown.

The 'Memory R/W' function is used to read and/or write the memory of remote nodes. Read, Write and Lock operations can be performed on a selected set of predefined addresses (like the CSR registers) or on user-definable addresses.

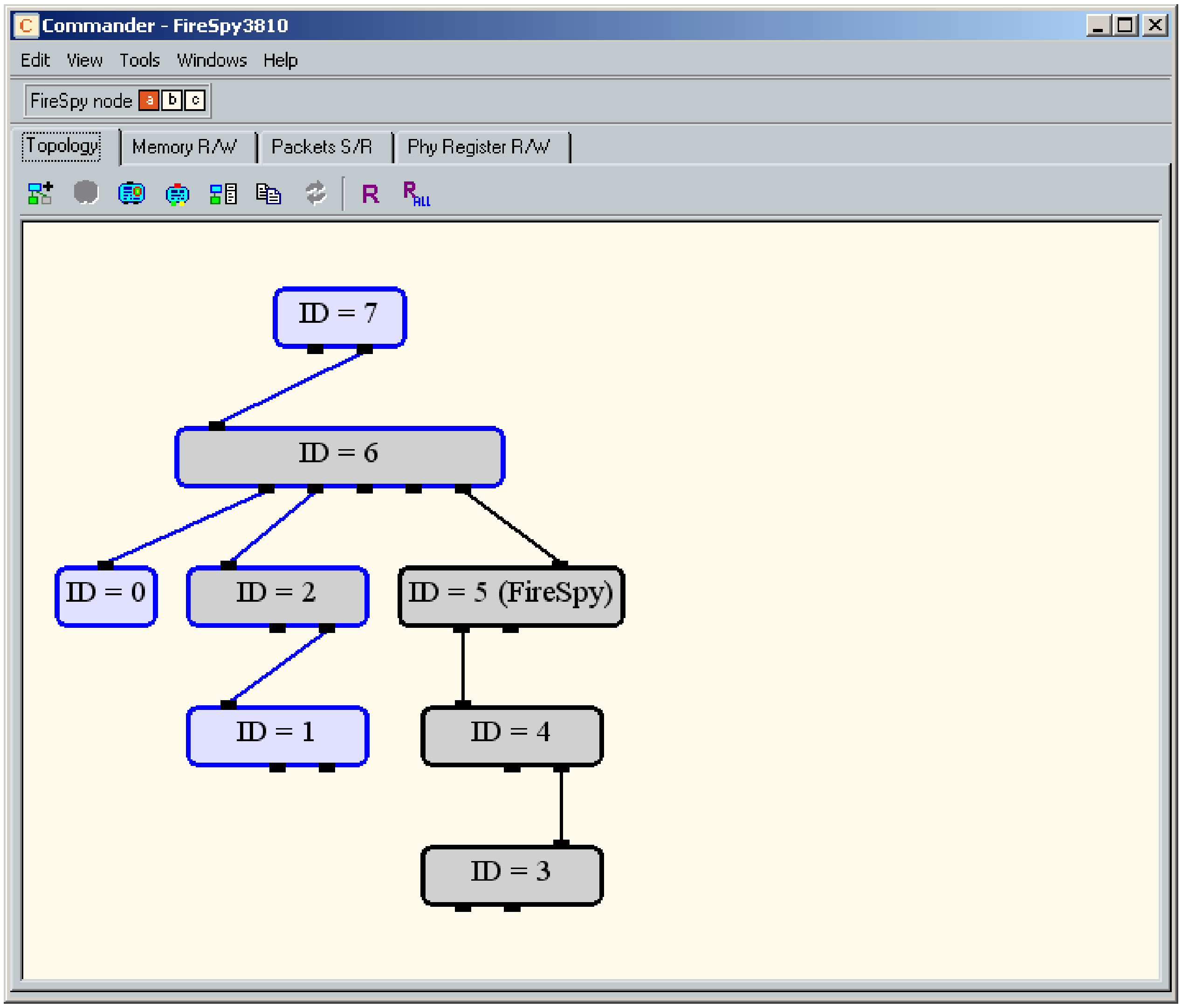
With the 'Packet S/R' function, one can send all kinds of packets including phy packets, unformatted packets and erroneous packets to a selected node. Optionally, one can wait for and display the response. Both the packet sent and the packet received can be displayed as layout or field list.

6.1. How to use it

You can open the Commander by clicking on the Commander button in the main window, or select the Commander from the 'Analyzer' menu at the top of the main window or one of the other open windows.

When you open the Commander, a window as shown below will appear.

The displayed topology will of course depend on the actual IEEE1394 bus configuration the Analyzer is connected to.
You can select one of the pages depending on the commands you want to perform. We will describe each page below.



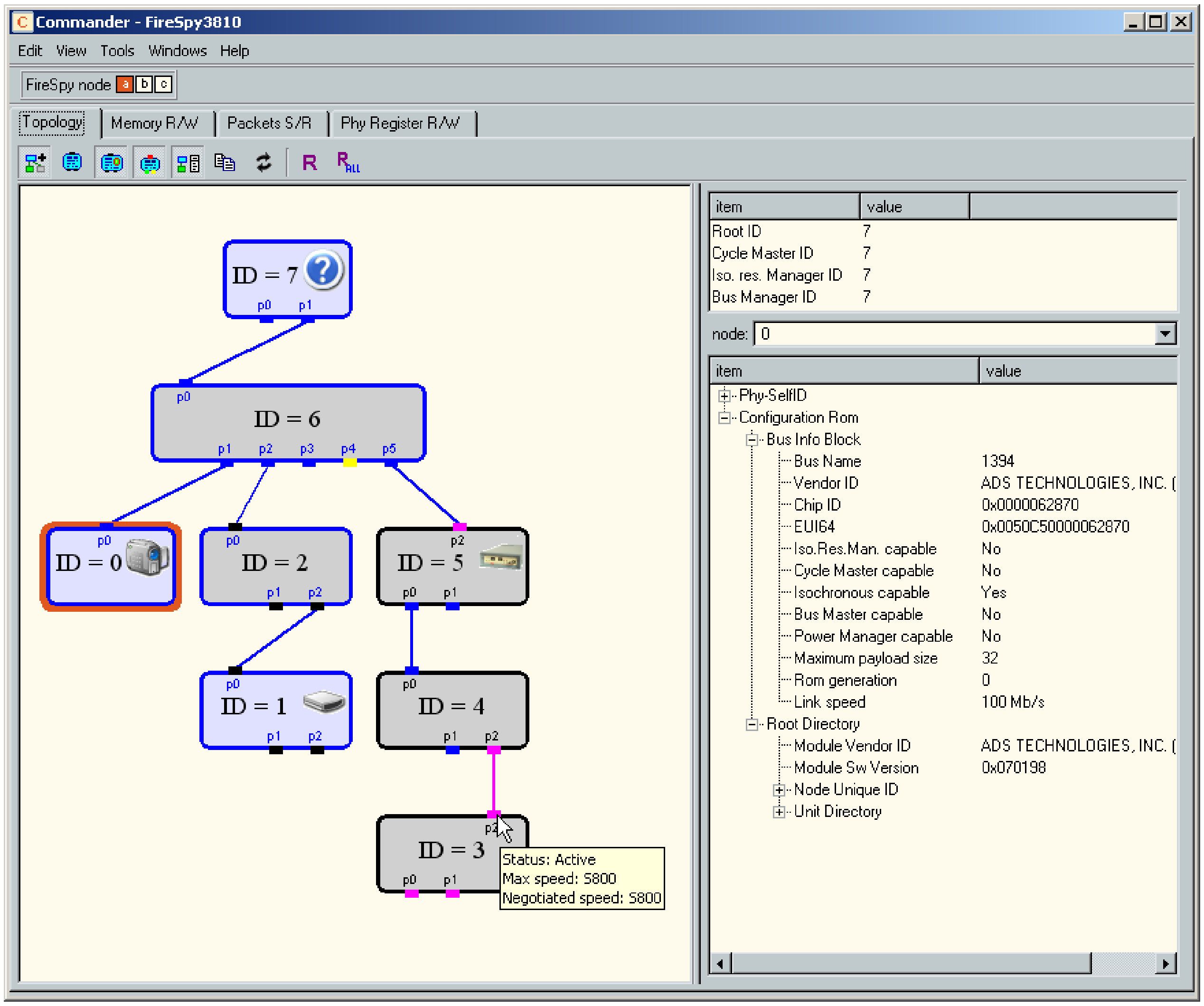
6.1.1. Topology

Select the topology page if you want to display the current bus topology. When connecting or disconnecting nodes, you will see the topology picture change, reflecting the changed bus topology. See above for an example of the Topology page.

With the 'Detailed view mode' button you can toggle between detailed and compact view mode. In detailed view mode, you can select the options to be displayed inside the node rectangles in the picture with the 'Select node display options' button, just like in the Topology View of the Recorder. The 'icons in nodes' button enables the display of icons in the nodes. With the 'Port information' button, visualization of port status and capabilities can be toggled. The ‘Bus and node information’ enables extra bus and node information of the selected node. The 'Refresh view' button can be used to redraw the topology view in case of changes the don't cause a bus reset. Finally, you can force a Bus Reset by clicking the 'Initiate Bus Reset' button. The topology picture will be updated too if the bus topology changes.

Beware that the options "View icons in nodes" and "Port information" cause the application to read the configuration ROM and PHY registers of other nodes on the bus. Such traffic initiated by the FireSpy is visualized with the red led in the toolbar.

When the right mouse button is clicked on a node or port in the topology picture, a context menu pops up. Depending on the node's capabilities, the context menu shows options including 'try to become the root node' for a node, and 'suspend', 'resume', 'disable' and 'enable' for a port.



item	value
Root ID	7
Cycle Master ID	7
Iso. res. Manager ID	7
Bus Manager ID	7

item	value
Phy-SelfID	
Configuration Rom	
Bus Info Block	
Bus Name	1394
Vendor ID	ADS TECHNOLOGIES, INC. (
Chip ID	0x0000062870
EUI64	0x0050C50000062870
Iso.Res.Man. capable	No
Cycle Master capable	No
Isochronous capable	Yes
Bus Master capable	No
Power Manager capable	No
Maximum payload size	32
Rom generation	0
Link speed	100 Mb/s
Root Directory	
Module Vendor ID	ADS TECHNOLOGIES, INC. (
Module Sw Version	0x070198
Node Unique ID	
Unit Directory	

Status: Active
Max speed: S800
Negotiated speed: S800

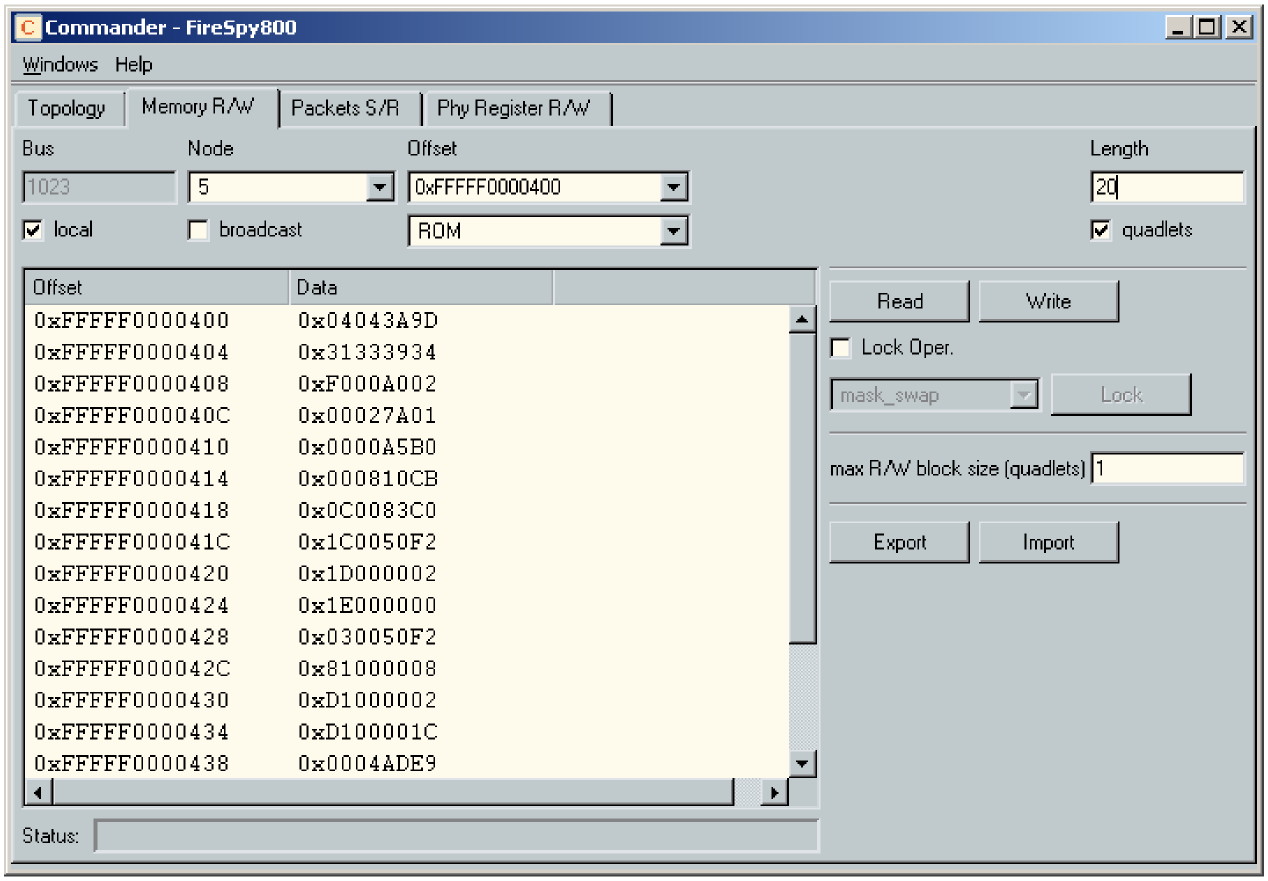
If the software encounters violations of the CSR specification while parsing the Configuration ROM, these entries will be displayed in orange.

Note: The 'Root Directory' will not be displayed if there is no license key available for the version 2 software (see License Manager). The context menu, 'Detailed view mode', 'Icons in nodes', 'Port information' and 'Refresh view' require a version 5.11 (or newer) license key.

If you select a node, this node will be the default target node for the Memory R/W, Packets S/R and Phy Register R/W functions of the commander.

6.1.2. Memory Read/Write

If you want to read and or write memory locations of remote nodes, you can select the 'Memory R/W' page. Below is an example of the 'Memory R/W' page.
You can specify the address to be read or written by specifying the bus, node and offset. You can select an offset from the addresses popup box or type any decimal or hexadecimal address in the offset box.
You can specify the length of the data to be read or written. It can be specified in bytes or quadlets. After specifying the address and length, the current data in the read/write buffer is displayed in the data table, but this is not yet the data of the specified addresses. To read the data, press the 'Read' button. You can also change the data values and press the 'Write' button to write the data.



Bus: 1023 — Node: 5 — Offset: 0xFFFFF0000400 — Length: 20

[✓] local — [ ] broadcast — ROM — [✓] quadlets

[ ] Lock Oper. — mask_swap

max R/W block size (quadlets): 1

Offset	Data
0xFFFFF0000400	0x04043A9D
0xFFFFF0000404	0x31333934
0xFFFFF0000408	0xF000A002
0xFFFFF000040C	0x00027A01
0xFFFFF0000410	0x0000A5B0
0xFFFFF0000414	0x000810CB
0xFFFFF0000418	0x0C0083C0
0xFFFFF000041C	0x1C0050F2
0xFFFFF0000420	0x1D000002
0xFFFFF0000424	0x1E000000
0xFFFFF0000428	0x030050F2
0xFFFFF000042C	0x81000008
0xFFFFF0000430	0xD1000002
0xFFFFF0000434	0xD100001C
0xFFFFF0000438	0x0004ADE9

Status:

To perform a lock operation, check the 'Lock Oper.' check box. Because lock operations can only be performed on one-quadlet or two-quadlet data, the length changes to 1 or 2 quadlets. The data table now has a data and an argument column, because some lock operations need data and arguments. You can specify the kind of lock operation to be performed using the popup box with the possible lock operations. Pressing the 'Lock' will result in the lock operation specified and the data column in the read/write buffer box will be updated according to the lock response packet.

You can specify the maximum data-block size the Analyzer may use to do the read or write operation.

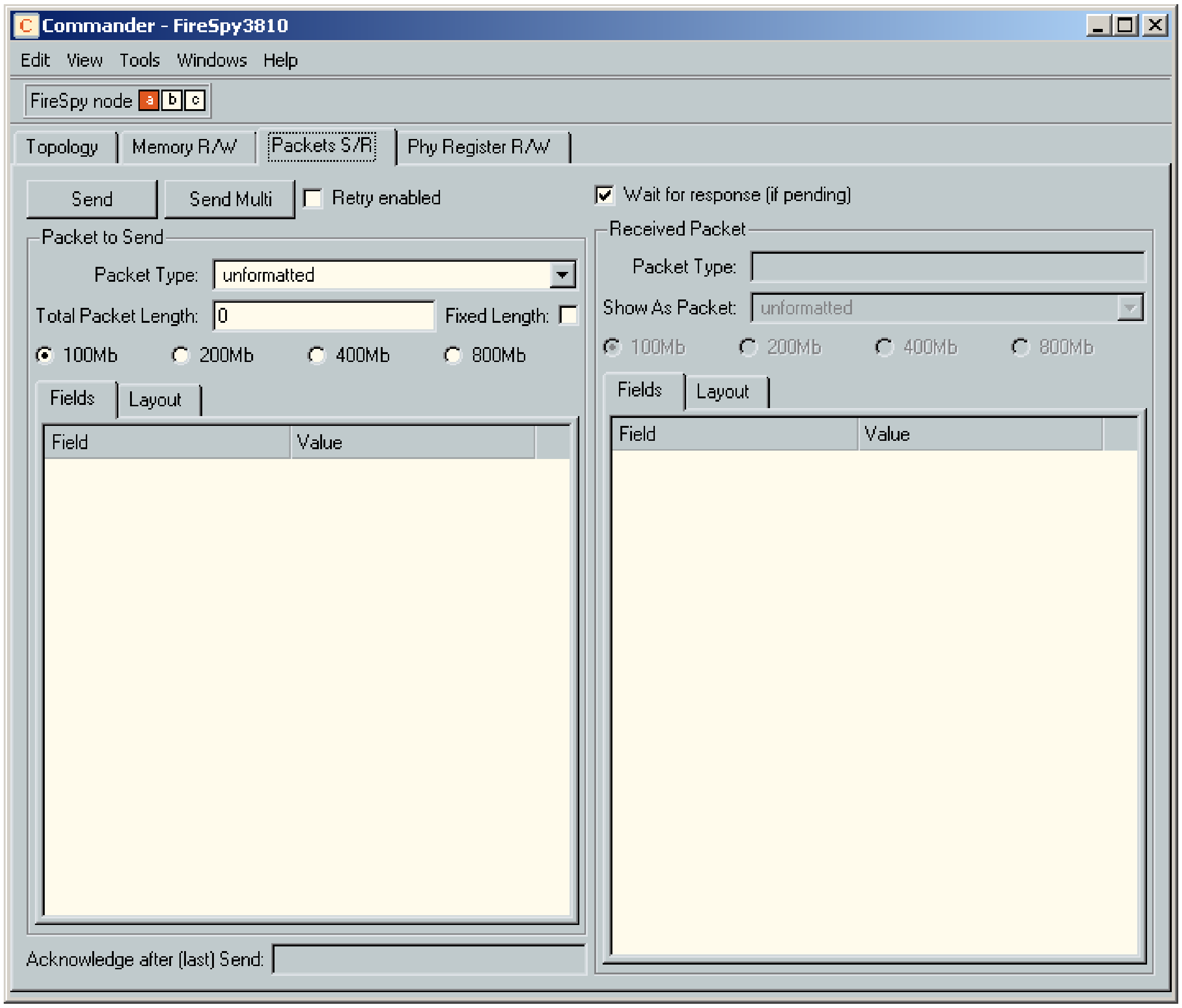
If a read, write or lock could not be executed successfully; an error message will be displayed.

6.1.3. Packets Send/Receive

If you want to send packets (and optionally wait for their response), you can select the Packets S/R' page. Below is an example of the Packets S/R page.

The 'Packets S/R' page has a ‘send packet’ side (left) and a ‘receive packet’ side (right). Using the ‘send packet’ side, you can specify the packet to be sent by selecting the packet type from the 'Packet Type' popup box. Then you can change the packet fields in the 'Fields' page. The total packet length is displayed in the 'Total Packet Length' box. You can also select the packet speed and enable or disable the retry mechanism with the 'Retry enabled' checkbox.

Using the ‘receive packet’ side, you can specify if the Analyzer should wait for a response packet if the sent packet is acknowledged with an ack_pending code, by checking the 'Wait for response (if pending)' box.



Packet Type: unformatted

Total Packet Length: 0

Fixed Length: ☐

(● 100Mb ○ 200Mb ○ 400Mb ○ 800Mb)

Acknowledge after (last) Send:

Show As Packet: unformatted

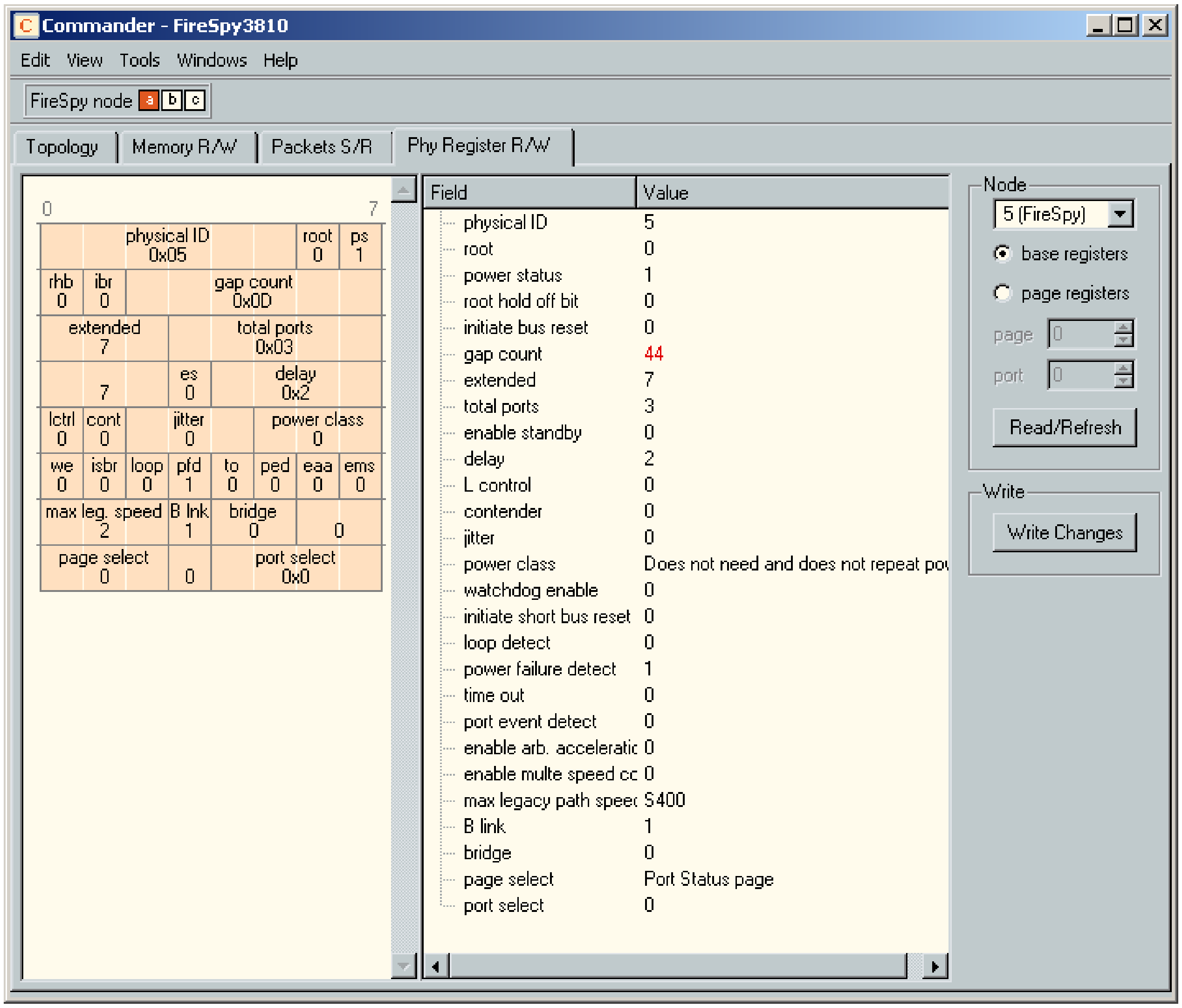
The send packet will be sent when you press the 'Send' button. If an acknowledge has been received for the packet, it will be displayed left at the bottom of the page.
If retry is enabled, the packet will be retried up to 15 times if needed. The packet is retried if the target node did not respond with an acknowledge, or if it is acknowledged with an ack_busy or ack_tardy code.

If the packet was sent successfully and the packet is acknowledged with an ack_pending code and the waiting for response is enabled, the Analyzer will wait for the response. If received, it will be displayed at the right side (‘receive packet’). If not received immediately, a progress bar will be displayed, indicating the wait process. The user can cancel this wait.
Like in the 'Packet View; of the Recorder, the received packet can be displayed as any type, and possible errors are displayed (if present) in an extra 'Errors' page.

6.1.4. Phy Register Read/Write

If you want to read registers of local or remote nodes, or write phy register of local nodes, you can select the 'Phy-Registers R/W' page. Below is an example of the 'Phy-Registers R/W' page.

After you specify the node, or press the 'Read/Refresh' button, the base or page registers of that node are read (depending on which one was selected), and displayed in the left and middle part of the page. The left part displays the layout of the registers, which are arranged in bytes. The middle part lists all the fields that are specified in the selected registers by name and value.
For the page registers, you can select a page number and for page 0 (zero) also a port number. Pressing the 'Read/Refresh' button will read the specified registers.


Commander - FireSpy3810

Edit View Tools Windows Help

FireSpy node a b c

Topology | Memory R/W | Packets S/R | Phy Register R/W

0							7
physical ID 0x05						root 0	ps 1
rhb 0	ibr 0	gap count 0x0D					
extended 7			total ports 0x03				
7			es 0	delay 0x2			
lctrl 0	cont 0	jitter 0			power class 0		
we 0	isbr 0	loop 0	pfd 1	to 0	ped 0	eaa 0	ems 0
max leg. speed 2			B lnk 1	bridge 0		0	
page select 0			0	port select 0x0			

Field	Value
physical ID	5
root	0
power status	1
root hold off bit	0
initiate bus reset	0
gap count	44
extended	7
total ports	3
enable standby	0
delay	2
L control	0
contender	0
jitter	0
power class	Does not need and does not repeat pov
watchdog enable	0
initiate short bus reset	0
loop detect	0
power failure detect	1
time out	0
port event detect	0
enable arb. acceleratic	0
enable multe speed cc	0
max legacy path speec	S400
B link	1
bridge	0
page select	Port Status page
port select	0

Node: 5 (FireSpy) [✓ base registers] page registers; page 0; port 0; Read/Refresh

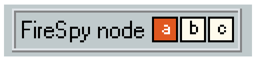
Write: Write Changes

If you want to change a value, change it in the field list and press the 'Write Changes' button. As soon as a value has been changed by the user, but has not been written yet, the changed field will be displayed in red and the 'Write Changes' button will be enabled (see example above).

Note that the phy registers of the local node can be read and written. Phy registers of remote nodes however, can only be read. And they can only be read if the node complies with the IEEE1394a standard, because the Remote-Access Packet and Remote-Reply Packet type of the phy packets defined in IEEE1394a are used for this operation.
If a node could not be read, the left and middle displays are empty.

6.2. Details

Multi-bus Analyzers
For multi-bus analyzers only, the toolbar contains the following control to select the active Analyzer node.



This control can be used to select the Analyzer node to view information for. By clicking on the "a", "b" or "c", the corresponding node can be hidden or shown. A red background indicates the node is shown. A white background indicates a node is hidden. In the Topology View, only one node can be shown at the same time.

For multi-bus analyzers only, pressing this button initiates a bus reset on all busses of the FireSpy.

The following sections describe the details per tab page of the Commander.

6.2.1. Topology

The 'Topology' page consists of a toolbar, the topology picture and optionally extra bus and node information.

Toolbar
The toolbar has the following buttons:

Detailed view mode
This is a toggle button. When pressed, it enables the traditional, larger view mode. It also enables additional information to be displayed with 'Select node display options'.

Select node display options
Pressing this button will display a dialog to select the node display options. It works the same as the equivalent button in the toolbar of the Topology View of the Recorder.

Icons in nodes
This is a toggle button. When pressed, icons are displayed in the nodes. The Analyzer will use the file icons.txt in the Analyzer\bin folder to look up the image using the EUI64 found in the node's bus info block. The icons must be 32x32 pixel images in PNG, JPG or BMP format.

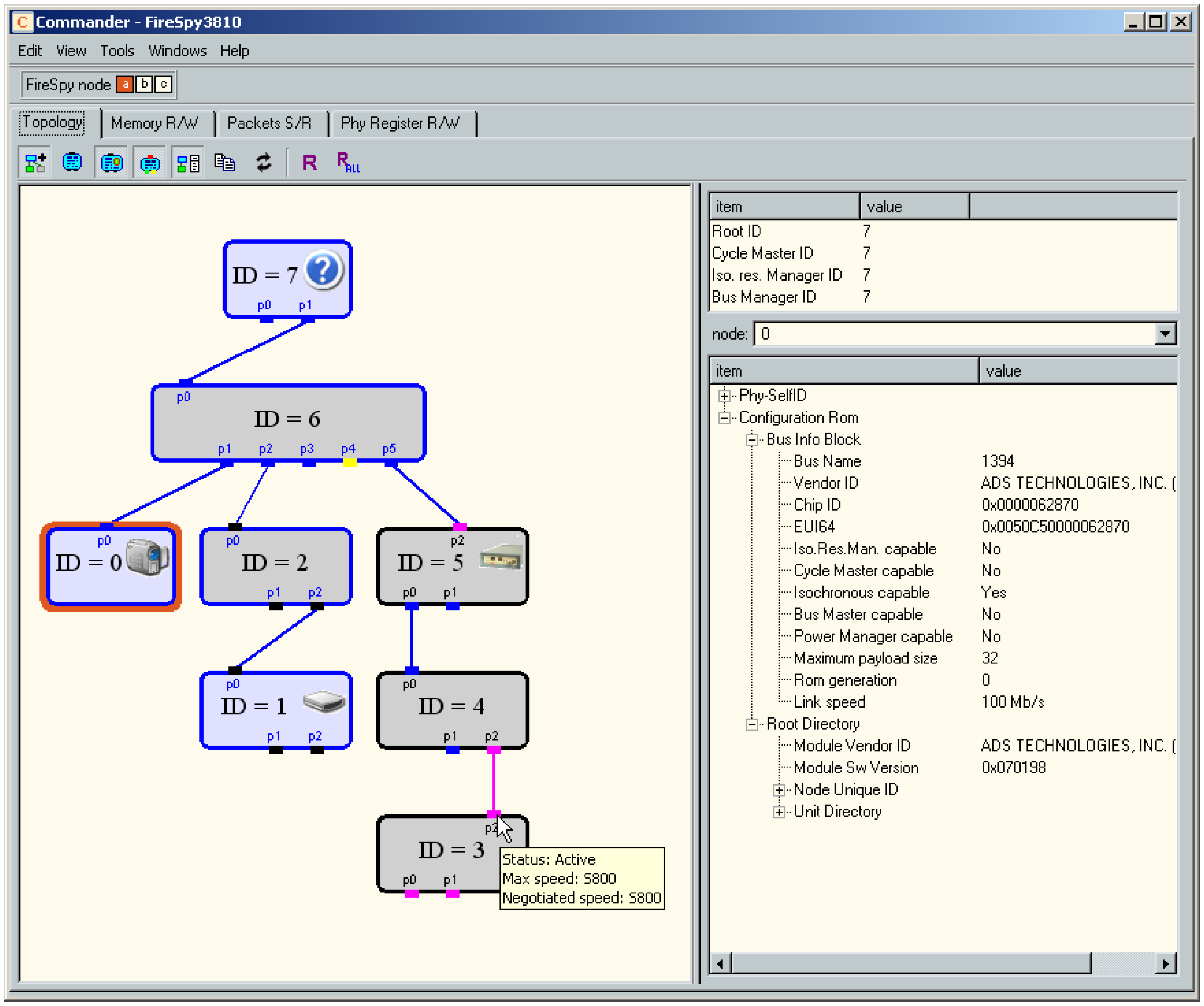
Port information
When this toggle button pressed, extra port capabilities and status information are displayed. If the mouse is positioned over a port a tooltip with more detailed information pops up. The information is collected by reading the PHY registers of the nodes.

Bus and node information
This is a toggle button. When this button is pressed, extra bus information and node information of the selected node will be displayed. See the description of the same button in the Topology View of the Recorder for more information.

Copy topology picture to clipboard
When this button is pressed, the current topology picture is copied to the clipboard.

Refresh view
When pressed, the topology picture will be refreshed. This is useful when topology changes that do not cause a bus reset happen, such as a port suspending.

Initiate Bus Reset
Pressing this button will generate a bus reset and the topology picture will be refreshed if needed.



Status: Active
Max speed: S800
Negotiated speed: S800

item	value
Root ID	7
Cycle Master ID	7
Iso. res. Manager ID	7
Bus Manager ID	7

node: 0

item	value
Phy-SelfID	
Configuration Rom	
Bus Info Block	
Bus Name	1394
Vendor ID	ADS TECHNOLOGIES, INC. (
Chip ID	0x0000062870
EUI64	0x0050C50000062870
Iso.Res.Man. capable	No
Cycle Master capable	No
Isochronous capable	Yes
Bus Master capable	No
Power Manager capable	No
Maximum payload size	32
Rom generation	0
Link speed	100 Mb/s
Root Directory	
Module Vendor ID	ADS TECHNOLOGIES, INC. (
Module Sw Version	0x070198
Node Unique ID	
Unit Directory	

In the example above, various buttons are switched on. The node with ID 5 is selected, so extra information about this node is displayed. The node information is split in two parts: information extracted from the PhySelfID of that node and the Configuration Rom information of the node. The ports in blue are S400 capable, the ports on the Analyzer are pink to show they are capable of S800. On the 6-port hub (ID 4), one port has been disabled, recognizable by it's yellow color, and a tooltip is show for another port. The ports of the harddisk are black: it's PHY registers cannot be read because it is a 1394-1995 device.

Note: The 'Root Directory' will not be displayed if there is no license key available for the version 2 software (see License Manager). The 'Compact view mode', 'Icons in nodes', 'Port information' and 'refresh view' require a version 5.11 (or newer) license key.

Topology picture

There is not much to say about the topology picture because it is almost equivalent to the topology picture described in the Topology View of the Recorder. The color of the node rectangles and connection indicate their maximum speed and the rectangle is colored gray inside if the link layer is not active. Extra node information can be displayed inside the node rectangles using the 'Select node display options' toolbar button.
The only differences are that the node corresponding to the Analyzer is indicated with the 'Analyzer' word right to the ID number (unless 'Icons in nodes' is selected), and the selected node is also used as default for the Memory R/W, Packet S/R and Phy Register R/W pages of the Commander.

Extra bus and node information

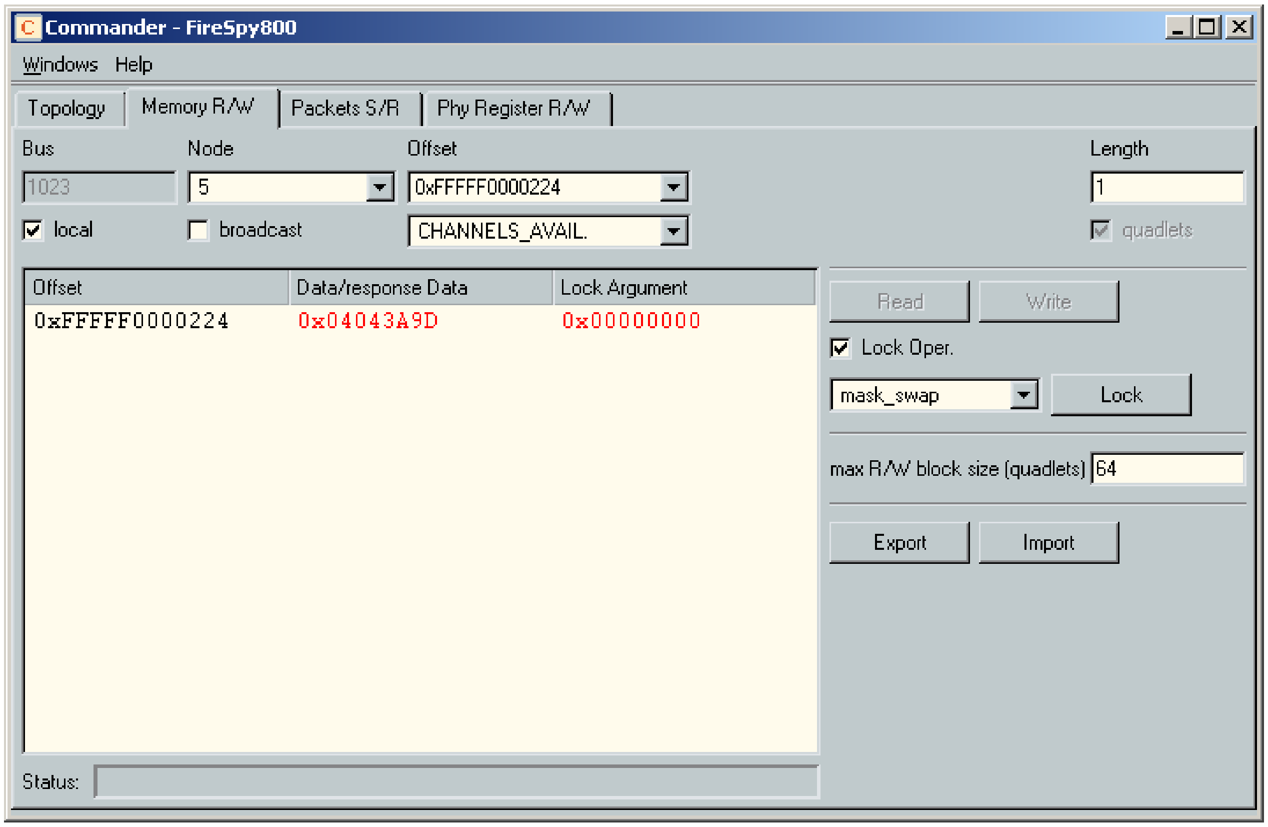
Here extra bus information and information about the selected node will be displayed if the option is switched on. It is equivalent to the corresponding button in the Topology View of the Recorder. Only the way some extra information is found differs from the Recorder. See below for more information about this.

Active Configuration Rom and Bus_Manager_ID reads
If Configuration-Rom information of some node or the Bus_Manager_ID register value is needed, the Analyzer will read it actively. It will only use quadlet read transactions.
If the 'node display options' that are retrieved from the Bus_Info_Block are all disabled and if the 'Bus and node information' button is not pressed, the Analyzer will not generate any active transactions.
If one or more 'node display options' are enabled that need to be retrieved from the Bus_Info_Block or if the 'Bus and node information' button is pressed and a node is selected, then the Configuration Rom of one or more nodes will be read when needed.
If the 'Bus and node information' button is pressed, the Bus_Manager_ID register will be read too when needed.

6.2.2. Memory R/W

The 'Memory R/W' page consists of the following parts:

- address specification
- length specification
- read/write/lock control
- data table



Bus: 1023
☑ local
Node: 5
☐ broadcast
Offset: 0xFFFFF0000224
CHANNELS_AVAIL.
Length: 1
☑ quadlets
☑ Lock Oper.
mask_swap
max R/W block size (quadlets): 64
Status:

Offset	Data/response Data	Lock Argument
0xFFFFF0000224	0x04043A9D	0x00000000

address specification
The read/write and lock start address can be specified by specifying the bus, node and offset.

local
If the 'local' checkbox below the 'Bus' box is checked, the bus number will be fixed to 1023, which is the number used to identify local busses. Otherwise, you can fill in a decimal number in the 'Bus' box.

Bus
If 'local' is not checked, you can fill in a decimal bus number here. The range is from 0 to 1022.

broadcast
If the 'broadcast' checkbox below the 'Node' box is checked, the node number will be fixed to 63, which is the number used to identify broadcast transmissions.

In the case of broadcasts, only write actions will be possible.

Node
If 'broadcast' is not checked, you can select a node from this popup list. The nodes in this list will consist of all nodes connected to the bus if the local bus is specified (local checked), otherwise it will be a list of all possible node Ids (0 to 62).

Offset
To complete the address, you can specify the offset, by putting a decimal or hexadecimal value in the 'Offset' box. If you select an offset from the popup list beneath it, this 'Offset' box will be filled in automatically. When the offset is changed, the offset values in the data table will change accordingly.

offset popup list
Alternatively, you can select one of the pre-programmed addresses from the popup list below the 'Offset' box. Selecting one of those addresses will automatically fill in the 'Offset' value and the 'Length' box will be initialized with a default value. Both the 'Offset' and 'Length' can be changed manually afterwards.

length specification
Length specification consists of the 'quadlets' checkbox and 'Length' box:

quadlets
Depending on the state of this checkbox, the length indicated in the 'Length' box is specified in bytes (not checked) or quadlets (checked), and the data in the data table will be represented in bytes or quadlets accordingly.

Length
You can put a decimal or hexadecimal number in the 'Length' box. When the length is changed, the length of the data displayed in the data table changes accordingly.
The length can be specified in bytes or quadlets, depending on the 'quadlets' checkbox.

data table
The data table displays the data currently in the read/write buffer. The date will be displayed with a red color if the read, write or lock transaction isn’t performed yet. After a read, write or lock transaction, the correct read, written or lock-data will be displayed with black color. Depending on the 'quadlets' checkbox state, it will display the data as bytes or quadlets.
For read and writes, the data table has two columns: the offset and the data. If the 'Lock Oper.' checkbox is checked however, it contains an additional 'Lock argument' column (like the example above). This is because some lock operations need both data and arguments. After a correct lock operation, the returned data in the lock response packet is put in the data column. Note that lock operations can only be performed on data sizes of 1 or 2 quadlets.

read/write/lock control
The read/write/lock control consists of:

Read
When the 'Read' button is clicked, the specified addresses will be read and the data will be displayed in the data table. The read is disabled if the address does not specify a valid read address (e.g. a broadcast address).
If a read could not be performed successfully, a message will display the reason for the failure, and the data table will be unchanged. All data that could be read successfully will be displayed with black color in the data table.

Write
When the 'Write' button is clicked, the data specified in the data table will be written to the specified addresses. The write is disabled if the address does not specify a valid write address.
If a write could not be performed successfully, a message will display the reason for the failure. All data that could be written successfully will be displayed with black color in the data table.

Lock Oper.
You must check this checkbox to perform a lock operation. When checked, the 'Read' and 'Write' buttons will be disabled, and the 'Lock' button will be enabled if the address specifies a valid read/write address (e.g. not a broadcast address).

Because lock operations can only be performed on a data size of 1 or 2 quadlets, the length will automatically change to 1 or 2 quadlets. The data table will get an extra 'Lock Argument' column, because some lock operations need a data value and an argument value for each address location.
Finally, when checking this box the popup list to specify a lock operation will be enabled.

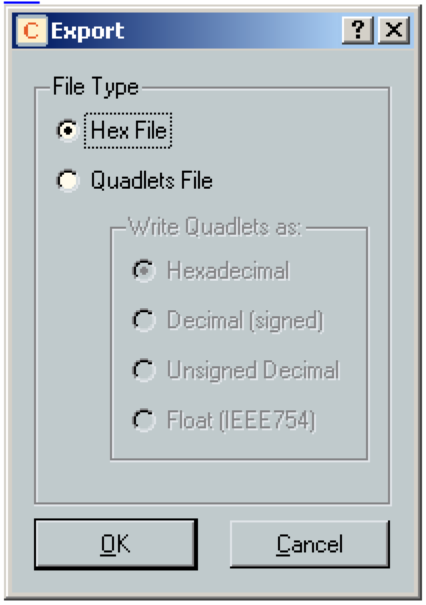
lock operation popup list
You can select the lock operation here, by selecting it from a popup list.

Lock
When the 'Lock' button is clicked, the specified lock operation will be performed using the data and if needed arguments from the data table. The resulting data (data returned by the lock response packet) will be displayed in the 'Data' column of the data table. The lock is disabled if the address does not specify a valid read/write address (e.g. a broadcast address).
If a lock could not be performed successfully, a message will display the reason for the failure. If a lock operation was successfull, the data will be displayed in black in the data table.

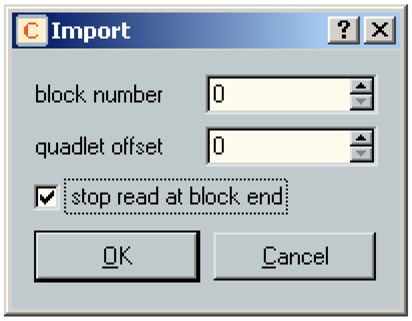
max R/W block size (quadlets)
Here you can fill in the maximum block size (in quadlets) that the Analyzer should use to do the read or write operation. If you specify 1 quadlet, only quadlet read or write transactions will be used if the total size is a multiple of 4 bytes. Otherwise, block read or write transactions will be used. If the specified maximum value is bigger than 1 quadlet, and the block transactions fails, the Analyzer will automatically fall back to a maximum size of 1 quadlet (thus only using quadlet read or write transactions if a multiple of 4 bytes need to be read or written).

Export/Import
The export/import control consists of:

Export
Here you can export the data table to a text file. Data can be stored as a Hex Data file, or a Quadlets Data file.



Import
Here you can import data from file. The import dialog allows you to import only part of the file by specifying a block number (a block is defined as one or more lines of data followed by one or more blank lines) or an offset in the file. See also Hex Data file.

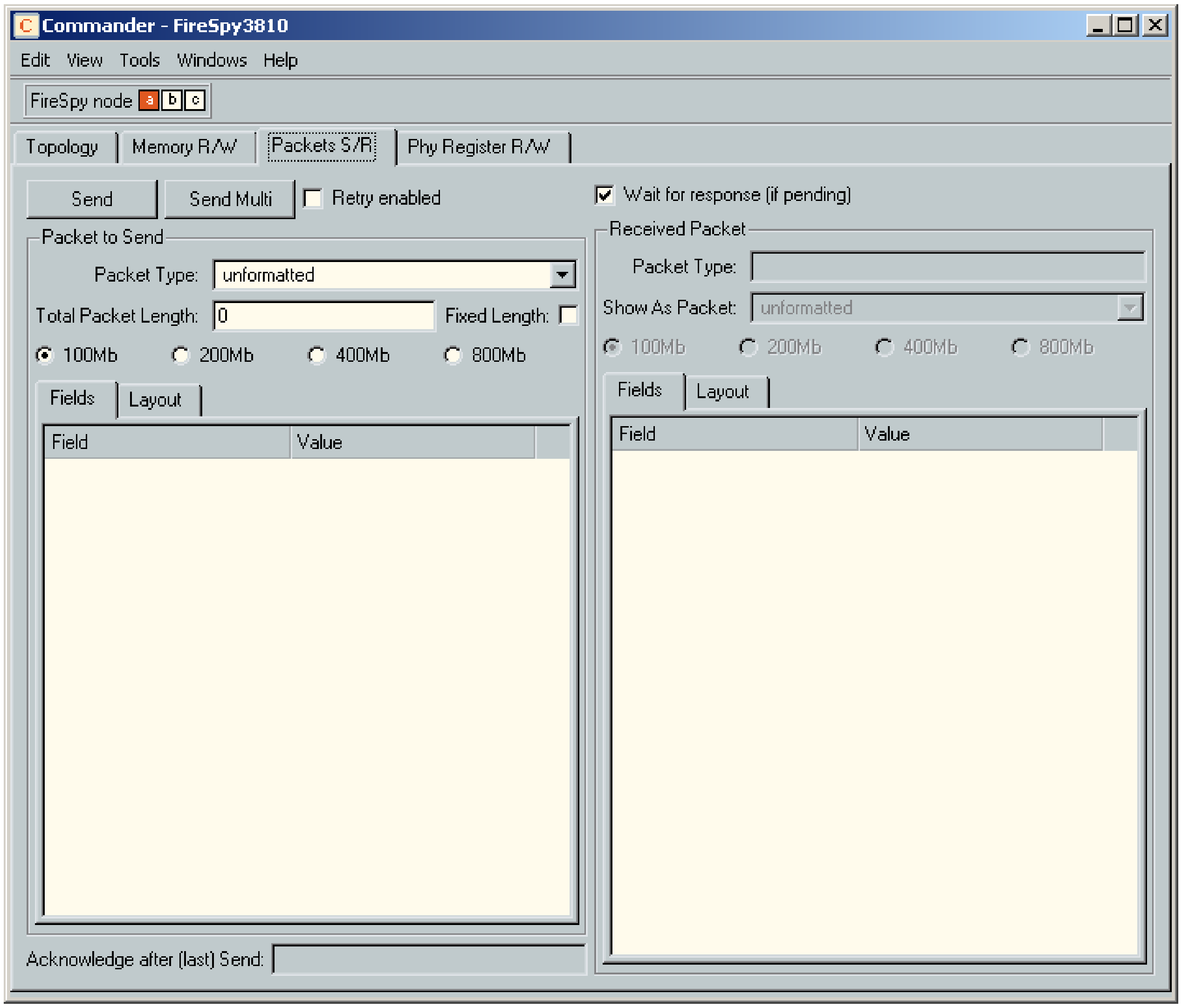


block number: 0
quadlet offset: 0
☑ stop read at block end

6.2.3. Packets S/R

The 'Packets S/R' page consists of the following parts:

- send packet (everything left from the middle)
- receive packet (everything right from the middle)



☐ Retry enabled
☑ Wait for response (if pending)

Packet to Send
Packet Type: unformatted
Total Packet Length: 0
Fixed Length: ☐
● 100Mb ○ 200Mb ○ 400Mb ○ 800Mb
Acknowledge after (last) Send:

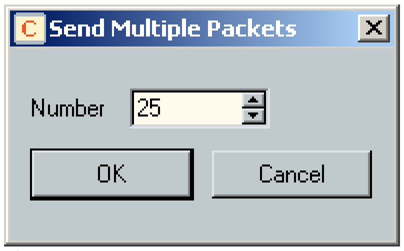
Received Packet
Packet Type:
Show As Packet: unformatted
● 100Mb ○ 200Mb ○ 400Mb ○ 800Mb

send packet
This part consists of the 'Send' button, 'Retry enable' check box, the 'Packet to Send' box that defines the packet to be sent, and the 'Acknowledge after Send' box.

Send
If this button is pressed, the packet defined in the 'Packet to Send' box is sent. If the packet is acknowledged, the acknowledge code is displayed in the 'Acknowledge after Send' box. If 'Retry enable' is checked and the packet is not acknowledged or the acknowledge code is one of the ack_busy codes or

the ack_tardy code, the packet will be retransmitted, up to 15 times. If the send fails a message is displayed with information about the failure.

Send Multi
If this button is pressed, a dialog pops up prompting the user to enter the number of times the packet will be sent:



If the user enters a number equal to, or bigger than 1, and presses 'OK', the application proceeds to send the packet the selected number of times in rapid succession. If the packet(s) sent result in response packet(s), only the last one received will be displayed in the Packet S/R window.

Retry enable
If the packet to be sent is sent and this checkbox is checked and the packet is not acknowledged or the acknowledge code is one of the ack_busy codes or the ack_tardy code, the packet will be retransmitted, up to 15 times.
If there is still no acceptable acknowledge packet received after 16 transmits, a message will be displayed which informs the user about this failure. Acceptable acknowledge packets are one of ack_complete, ack_pending, ack_conflict_error, ack_data_error, ack_type_error or ack_address_error codes.

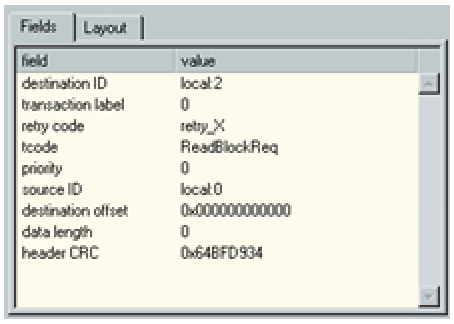
Packet to Send
The packet to be sent can be defined in this box. The box includes:

Packet Type
You can select the packet type from the popup list. The packet data will be initialized to a correct default packet of the selected type. If the 'unformatted' type is selected, the current packet data is unchanged and the packet will be displayed in unformatted form. You can use the unformatted type to change the packet data without having automatically recalculated CRC values or inverse check values (phy packets). This way, packets with erroneous fields (e.g. wrong CRC value) can be defined.

Total Packet Length
The total packet length expressed in bytes will be displayed here. When you change the data-length field of the packet (if it has one), the 'Total Packet Length' box, will be updated accordingly. You can also change the packet length directly from this box, but this will create an erroneous packet because the real packet length does not match the packet contents. For unformatted packets on the other hand, you can use this box to change the packet length.

packet speed
The speed of the packet can be selected here, using these speed selection buttons.

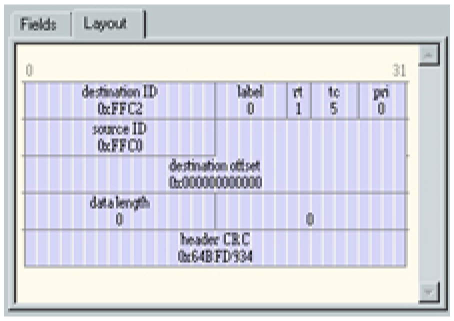
Fields
For primary packets the 'Fields' page shows all the fields the header of the packet is made of, plus the quadlet data fields if they are present. For Phy packets it displays all the fields the phy packet is made of. And for unformatted packets, it just shows all the packet quadlets. The 'source ID' and 'destination ID' fields are displayed in a special format. The bus number and node number are displayed separately, with a ':' character in-between. If the bus number is 1023, it is substituted by the text 'local'. Below is an example of the 'Fields' page for some read block response packet. You can change most fields in the 'Value' column by using the mouse to click on the value. A popup list may appear to choose a value, or a value edit box is displayed to enter a new value. The tcode field can never be changed because then the packet type wouldn't match the data anymore. To change the tcode, select another packet type or change it using the unformatted packet type. When changing the data-length field (if the packet has one), the 'Total Packet Length' above will be updated accordingly. For any field that has been changed, the corresponding CRC or inverse value check (phy packets) will be automatically recalculated. Below is an example of the 'Fields' page of a read block request packet.



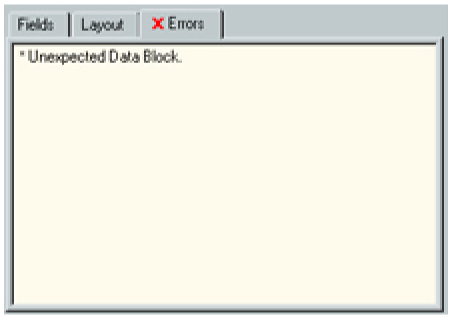
Fields | Layout

field	value
destination ID	local:2
transaction label	0
retry code	retry_X
tcode	ReadBlockReq
priority	0
source ID	local:0
destination offset	0x000000000000
data length	0
header CRC	0x64BFD934

Layout
The same fields displayed in the 'Fields' page are displayed in the 'Layout' page. But now the layout of the packet is shown. Each line shows 32 bits, or one quadlet. The exact bit position of each field is shown, including the field name (mostly some abbreviation because of space limitation) and the value. Reserved fields and fields with fixed values are shown too. They have no field name, but only a value. It can be handy to find out which bits a field occupies, if you want to change these bits in the unformatted packet type. Below is an example of the 'Layout' page for the same packet as above.

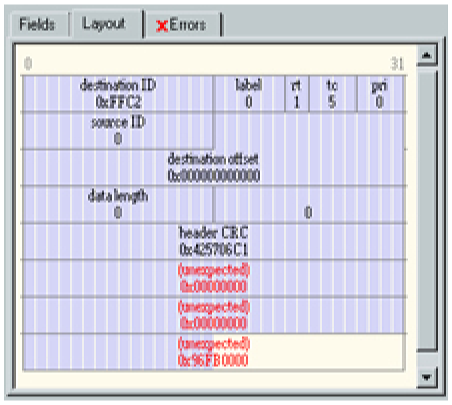


Errors
If the defined packet has one or more errors, an error page is added to the 'Fields' and 'Layout' pages. Selecting this page will list all detected errors. If one or more fields or data is missing the 'Field' and 'Layout' pages will also reflect the error by displaying the missing parts in red. Likewise, when there is too much packet data, the parts that are too much are also displayed red in the 'Fields' and 'Layouts' pages. Below is an example of the errors page for the ReadBlockReq packet above where the packet size is increased with 10 bytes using the 'Total Packet Length' box without changing the packet type, which will create an erroneous packet.



* Unexpected Data Block.

The corresponding Layout page looks like below.



Acknowledge after Send
If the transmitted packet is acknowledged, the acknowledge code is displayed here. If the packet is retransmitted, and acknowledged again, the last acknowledge code will be displayed.

receive packet
This part consists of the 'Wait for response (if pending)' checkbox and the 'Received Packet' box that displays the received packet.

Wait for response (if pending)
When this checkbox is checked, and the sent packet is acknowledged with an ack_pending code, the Analyzer will wait for a response. If the response is received, it will be displayed in the 'Received Packet' box. If it takes a while before the response is received, a progress dialog is displayed, indicating the wait process. The user will be able to cancel the wait.

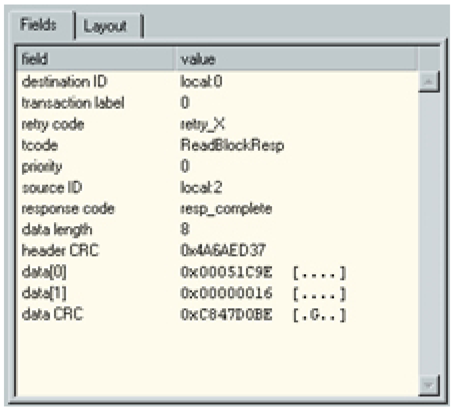
Received Packet
If a packet is received, the details of the packet can be seen in this box. The box includes:

Packet Type
In this box the type of packet that is received can be seen.

Show As Packet
This box has a popup list from which you can select a packet type. The packet will be displayed in the format of the packet type selected here. Initially it will be the same as the 'Packet Type' box. For unknown packets, it will be 'unformatted'.

packet speed
These selection buttons will indicate the speed of the received packet. They are not selectable by the user; they just function as an indicator.

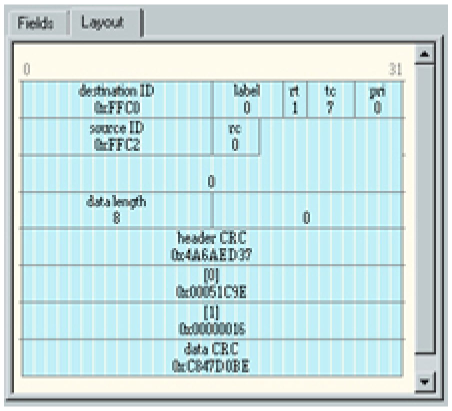
Fields
For primary packets the 'Fields' page shows all the fields the header of the packet is made of, plus the quadlet data fields if they are present. For Phy packets it displays all the fields the phy packet is made of. And for unformatted packets, it just shows all the packet quadlets. The 'source ID' and 'destination ID' fields are displayed in a special format. The bus number and node number are displayed separately, with a ':' character in between. If the bus number is 1023, it is substituted by the text 'local'. Below is an example of the 'Fields' page for the ReadBlockResp packet that was the result of the send ReadBlockReq packet above.



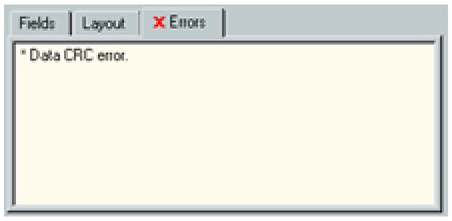
Fields | Layout

field	value
destination ID	local:0
transaction label	0
retry code	retry_X
tcode	ReadBlockResp
priority	0
source ID	local:2
response code	resp_complete
data length	8
header CRC	0x4A6AED37
data[0]	0x00051C9E [....]
data[1]	0x00000016 [....]
data CRC	0xC847D0BE [.G..]

Layout
The same fields displayed in the 'Fields' page are displayed in the 'Layout' page. But now the layout of the packet is shown. Each line shows 32 bits or one quadlet. The exact bit position of each field is shown, including the field name (mostly some abbreviation because of space limitation) and the value. Reserved fields and fields with fixed values are shown too. They have no field name, but only a value. Below is an example of the 'Layout' page for the same response packet as above.



Errors
If the selected packet has one or more errors, or if errors are detected because the selected 'Show As Packet' type does not correspond to the real packet type, an error page is added to the 'Fields' and 'Layout' pages. Selecting this page will list all detected errors. If one or more fields or data is missing, the 'Field' and 'Layout' pages will also reflect the error by displaying the missing parts in red. Likewise, when there is too much packet data, the parts that are too much are also displayed red in the 'Fields' and 'Layouts' pages. Below is an example of the errors page for a packet with a data CRC error.

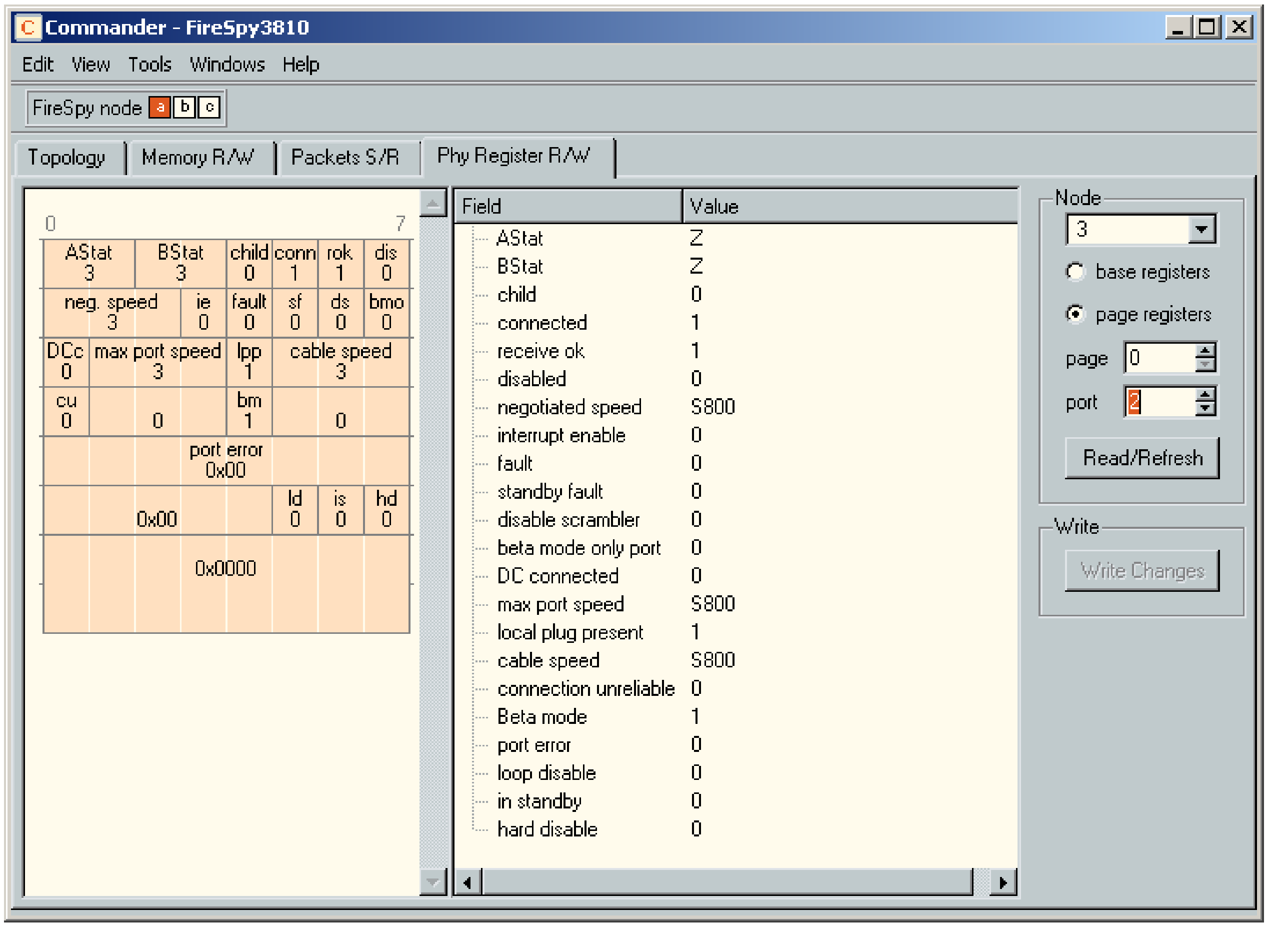


* Data CRC error.

6.2.4. Phy Register R/W

The 'Phy Registers R/W' page consists of the following parts:

- Register layout
- Register-fields table
- Node box
- Write box



Field	Value
AStat	Z
BStat	Z
child	0
connected	1
receive ok	1
disabled	0
negotiated speed	S800
interrupt enable	0
fault	0
standby fault	0
disable scrambler	0
beta mode only port	0
DC connected	0
max port speed	S800
local plug present	1
cable speed	S800
connection unreliable	0
Beta mode	1
port error	0
loop disable	0
in standby	0
hard disable	0

Node: 3
base registers [✓ page registers]
page: 0
port: 2

register layout
This part (most left) displays the layout of the selected register set. All phy registers are arranged in bytes. One byte can hold several fields and one field may occupy several bytes. The field names and values are included in the layout picture. Reserved bits and bits with fixed values have no label, but only a value.

register fields table
In the register fields table right to the layout, all fields of the selected register set are displayed with name and its value. The value can be changed by the user. If a value is changed, the field will be displayed in red until the change has been written to the register(s).

Node box
The node information consists of:

node popup list
Here you can select a node from which you want to read the register.
When you select a node, the register layout and register-fields table will be cleared.

base register - page register
You can select if the Analyzer should read the 'base registers' or 'page registers' by selecting the corresponding selection button.

page
If you have selected the 'page register' set, you will be able to select a page number (from 0 to 7) here. If page 0 is selected, you will also be able to select the port number below it (from 0 to 15).

port
If you have selected the 'page register' set, and you have selected 'page' number 0, you will be able to select the port number here (from 0 to 15).

Read/refresh
Pressing this button will read the specified registers from the selected node.

The Analyzer can only read the local phy registers or the phy registers from remote nodes that comply with the IEEE1394a standard. If the read could not be performed, an error message will be displayed, and no data will be displayed.

Write box
The write box has only one button:

Write Changes
When you press this button, the fields that have been changed by the user (which are displayed red) will be written. This will be minimal one byte, but could be more. The button is only enabled if the selected phy registers can be written and the user has changed one or more fields. Note that only local phy registers can be written.

Chapter 7. Recorder

Packets and events can be stored in the internal memory and viewed later, using a number of possible Views, using the Recorder. All Views can be switched on or off individually.

The 'Time View' enables the user to view the timing of events and packets (including acknowledge packets and the packet prefix) at a resolution of about 20nS for a FireSpy400 and 10nS for a FireSpy800. A time cursor is used to make time measurements.

The 'Topology View' shows the bus topology at the cursor position of the Timing View in a graphical way. Note that the topology may change during recording because of node connection or disconnection. Node details like Configuration Rom content can be viewed too if that information is recorded.

The 'Packet View' displays all packets in a list. Each packet can be viewed in detail, by showing its packet fields, packet layout or packet errors. Packets can be displayed as any possible packet type, so the user can find out what kind of packet an erroneous packet may be.

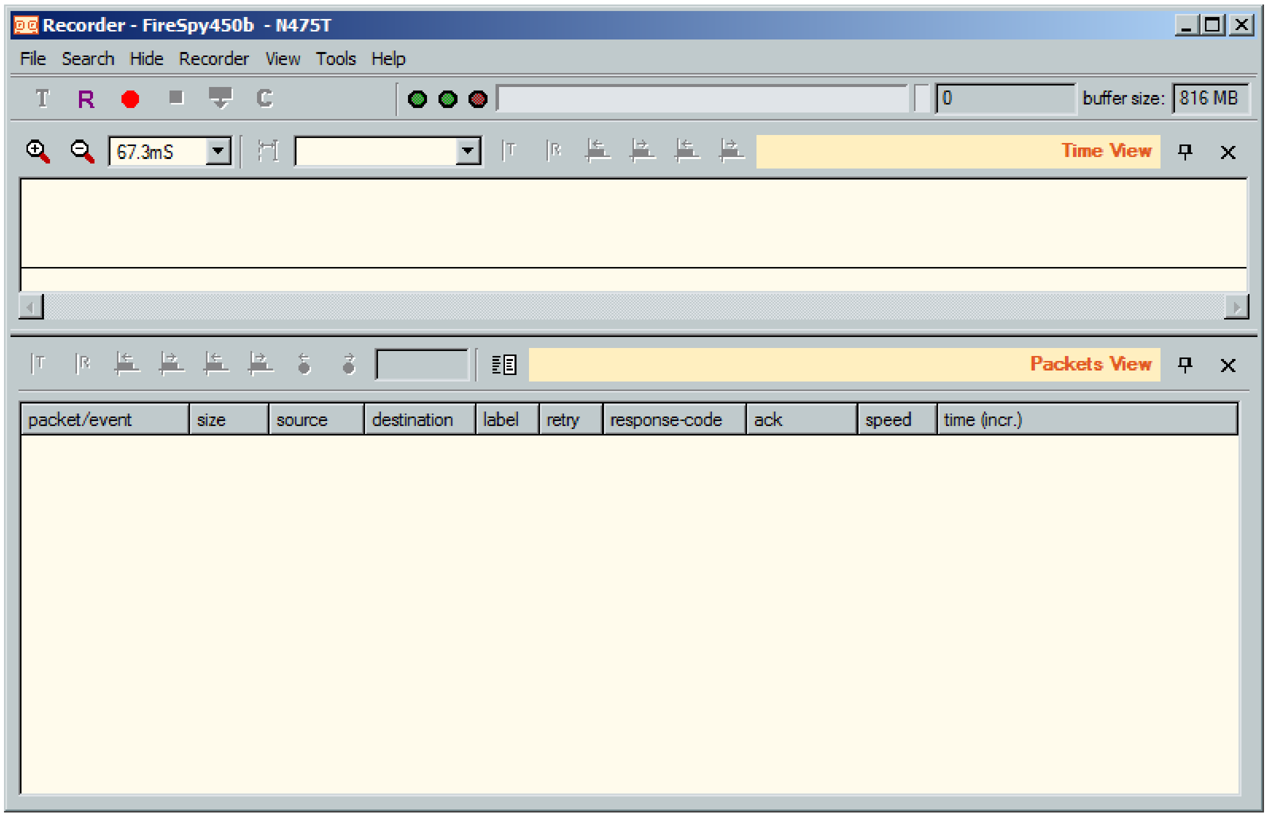
The 'Transaction View' displays a list of all complete and incomplete transactions. The details of each transaction can be displayed as a packet list or as a graphical flow presentation.

The ‘Protocol View' displays one or more of the supported higher protocols. The number of supported protocols will increase as new software versions are released. For each protocol you want to use, a license key will be needed. See the License Manager for more details.

7.1. Main Window

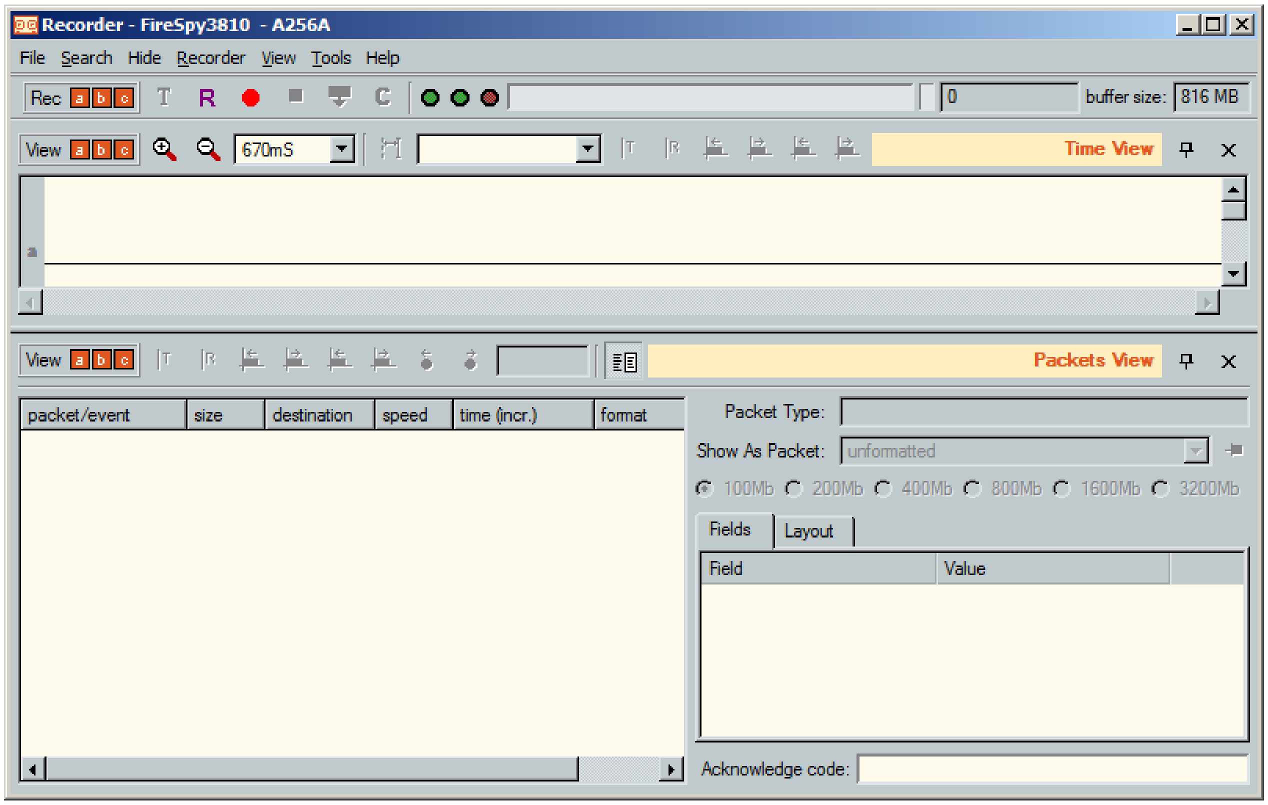
You can open the Recorder by clicking on the Recorder button in the main window, or select the Recorder from the 'FireSpy' menu at the top of the main window or one of the other open windows.
Alternatively, a dedicated Standalone Recorder application with built-in support for controlling multiple analyzers in a single user interface can be started through the Windows start menu as is explained in the getting started section.

When you open the Recorder, the window below will appear.



The window is filled with different kinds of views for the recorded data. You can switch them on and off or dock/undock them. The Time View is always displayed (if enabled) at the top. If a protocol view is switched on, it is showed at the bottom of the window. The remaining views share the area between them . You can enable and disable the different kinds of views using the 'View' menu. In the picture above, the Time View and the Packets View are visible. Using the 'pin' icon on the left of the close button will allow you to detach the view from the main window. This allows you to move it to any position on your screen. Pressing the 'pin' icon on a detached (undocked) view will attach it again in the main window. Using the menu "view->dock all views" item, all the undocked views are re-attached to the main window.

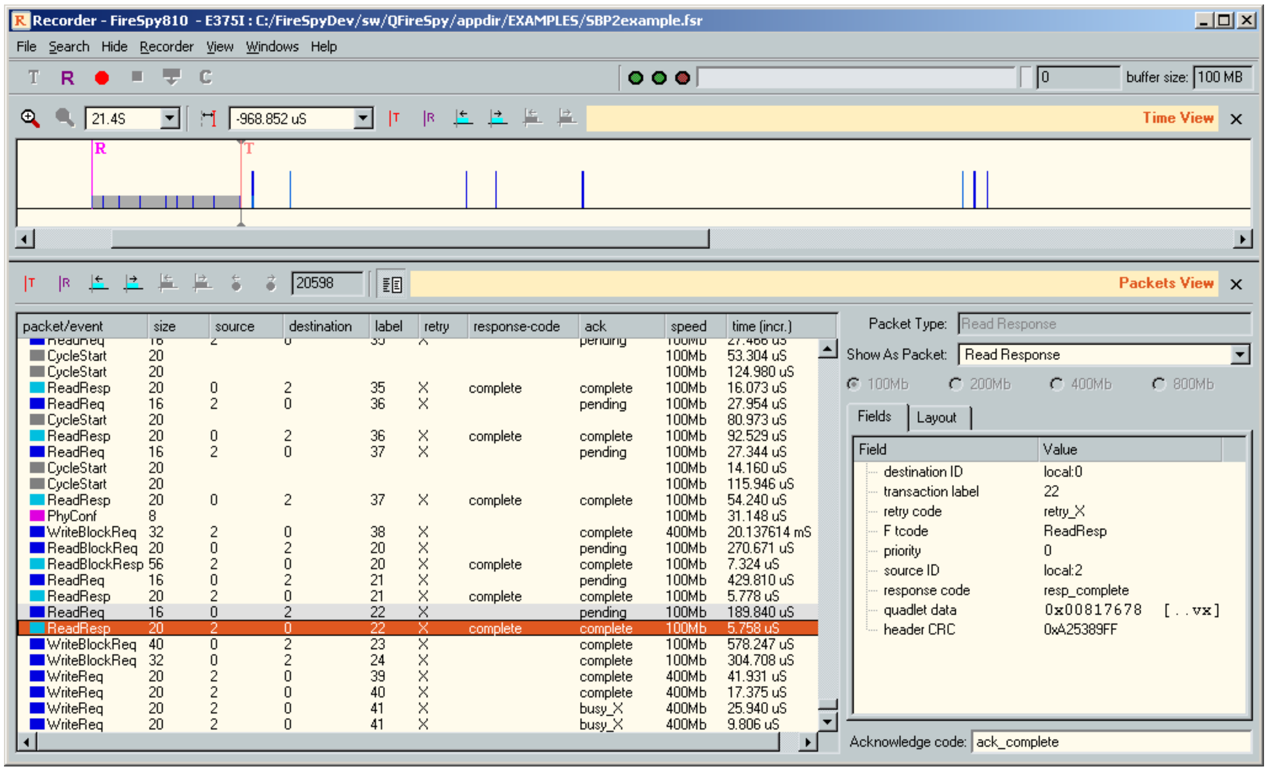
Triple Recorder
For triple node Analyzers, the main window looks as in the following picture.



7.1.1. How to use it

7.1.1.1. Display Recorder Files

Initially, there is no recorded data and the views are empty. You can load a Analyzer recorder file into the Recorder using the 'Open' command of the 'File' menu. You can select a recorder file (extension .fsr) and it will be loaded into the Recorder. In the window example below, the recorder file 'SBP2example.fsr' is opened, and the packet details option is switched on (see below about this option).


Recorder - FireSpy810 - E375I : C:/FireSpyDev/sw/QFireSpy/appdir/EXAMPLES/SBP2example.fsr
File Search Hide Recorder View Windows Help
buffer size: 100 MB
21.4S
-968.852 uS
Time View
Packets View

20598

packet/event	size	source	destination	label	retry	response-code	ack	speed	time (incr.)
CycleStart	20							100Mb	53.304 uS
CycleStart	20							100Mb	124.980 uS
ReadResp	20	0	2	35	X	complete	complete	100Mb	16.073 uS
ReadReq	16	2	0	36	X		pending	100Mb	27.954 uS
CycleStart	20							100Mb	80.973 uS
ReadResp	20	0	2	36	X	complete	complete	100Mb	92.529 uS
ReadReq	16	2	0	37	X		pending	100Mb	27.344 uS
CycleStart	20							100Mb	14.160 uS
CycleStart	20							100Mb	115.946 uS
ReadResp	20	0	2	37	X	complete	complete	100Mb	54.240 uS
PhyConf	8							100Mb	31.148 uS
WriteBlockReq	32	2	0	38	X		complete	400Mb	20.137614 mS
ReadBlockReq	20	0	2	20	X		pending	100Mb	270.671 uS
ReadBlockResp	56	2	0	20	X	complete	complete	100Mb	7.324 uS
ReadReq	16	0	2	21	X		pending	100Mb	429.810 uS
ReadResp	20	2	0	21	X	complete	complete	100Mb	5.778 uS
ReadReq	16	0	2	22	X		pending	100Mb	189.840 uS
ReadResp	20	2	0	22	X	complete	complete	100Mb	5.758 uS
WriteBlockReq	40	0	2	23	X		complete	100Mb	578.247 uS
WriteBlockReq	32	0	2	24	X		complete	100Mb	304.708 uS
WriteReq	20	2	0	39	X		complete	400Mb	41.931 uS
WriteReq	20	2	0	40	X		complete	400Mb	17.375 uS
WriteReq	20	2	0	41	X		busy_X	400Mb	25.940 uS
WriteReq	20	2	0	41	X		busy_X	400Mb	9.806 uS

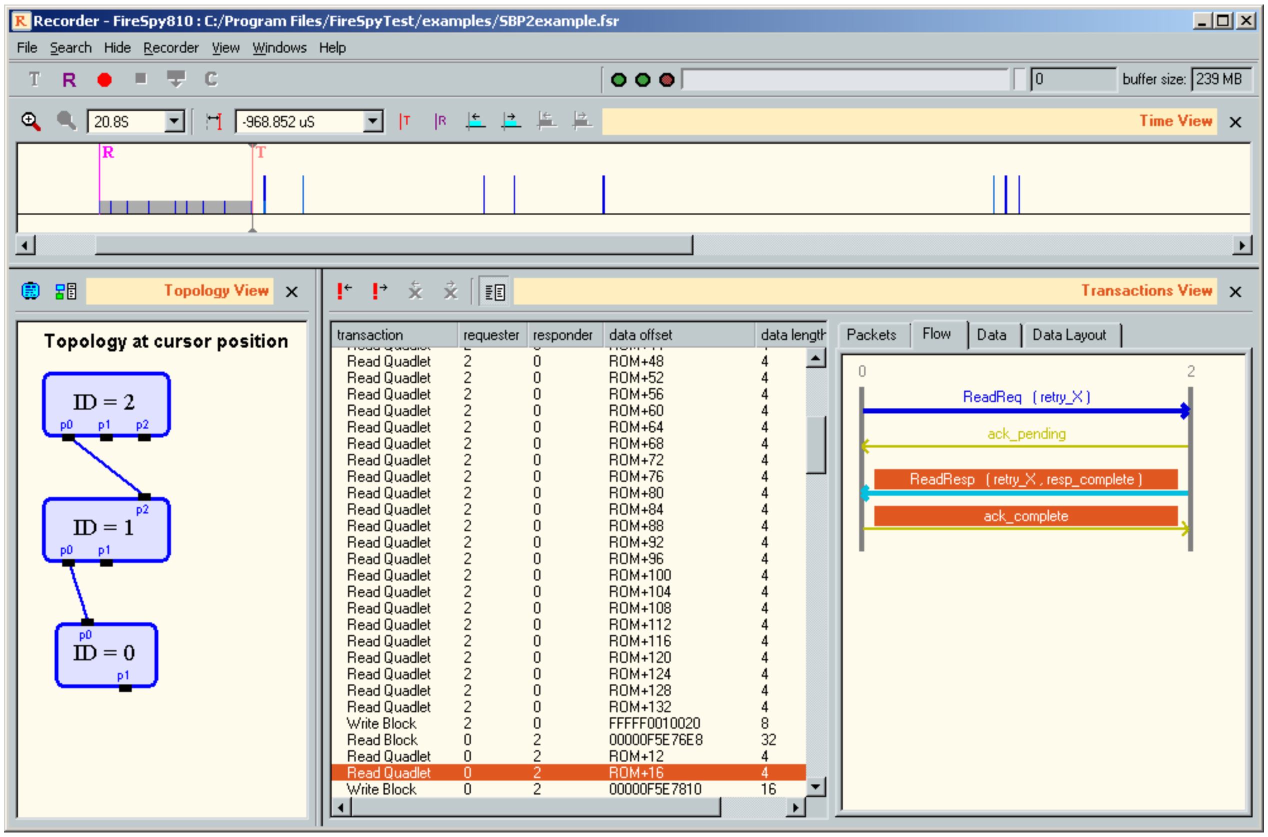
Packet Type: Read Response
Show As Packet: Read Response
[✓ 100Mb] 200Mb 400Mb 800Mb

Fields | Layout

Field	Value
destination ID	local:0
transaction label	22
retry code	retry_X
F tcode	ReadResp
priority	0
source ID	local:2
response code	resp_complete
quadlet data	0x00817678 [..vx]
header CRC	0xA25389FF

Acknowledge code: ack_complete

Now you see the 'Time View' and 'Packet View' both display the data.
In the example window below, the Packet View is disabled and the 'Time View', 'Topology View' and 'Transaction View' are enabled.


Recorder - FireSpy810 : C:/Program Files/FireSpyTest/examples/SBP2example.fsr
File Search Hide Recorder View Windows Help
buffer size: 239 MB
20.8S
-968.852 uS
Time View
Topology View
Topology at cursor position
ID = 2
p0
p1
p2
ID = 1
p2
p0
p1
p0
ID = 0
p1
Transactions View
Packets
Flow
Data
Data Layout
0
2
ReadReq (retry_X)
ack_pending
ReadResp (retry_X, resp_complete)
ack_complete

transaction	requester	responder	data offset	data length
Read Quadlet	2	0	ROM+48	4
Read Quadlet	2	0	ROM+52	4
Read Quadlet	2	0	ROM+56	4
Read Quadlet	2	0	ROM+60	4
Read Quadlet	2	0	ROM+64	4
Read Quadlet	2	0	ROM+68	4
Read Quadlet	2	0	ROM+72	4
Read Quadlet	2	0	ROM+76	4
Read Quadlet	2	0	ROM+80	4
Read Quadlet	2	0	ROM+84	4
Read Quadlet	2	0	ROM+88	4
Read Quadlet	2	0	ROM+92	4
Read Quadlet	2	0	ROM+96	4
Read Quadlet	2	0	ROM+100	4
Read Quadlet	2	0	ROM+104	4
Read Quadlet	2	0	ROM+108	4
Read Quadlet	2	0	ROM+112	4
Read Quadlet	2	0	ROM+116	4
Read Quadlet	2	0	ROM+120	4
Read Quadlet	2	0	ROM+124	4
Read Quadlet	2	0	ROM+128	4
Read Quadlet	2	0	ROM+132	4
Write Block	2	0	FFFFF0010020	8
Read Block	0	2	00000F5E76E8	32
Read Quadlet	0	2	ROM+12	4
Read Quadlet	0	2	ROM+16	4
Write Block	0	2	00000F5E7810	16

You can use the 'Recorder' menu or toolbar buttons to record packets into internal memory of the Analyzer, and download them to the host. The recorded data can be viewed in the same way an opened file can be viewed. The recorded data can also be saved to disk as a Analyzer recorder file using the 'Save As' command in the 'File' menu. In this 'File' menu there is also an 'Export Packets' command, which can be used to save a selection of packets as a text file or as Packet file. A Packet file can be used in the (isochronous) Generator.

7.1.1.2. Record Packets

If the internal recorder memory of the Analyzer is empty, you can start recording by selecting the 'Start' command from the 'Recorder' menu, or clicking the 'Start Recording' button in the toolbar.

When the internal recorder memory is not empty, you can clear it with the 'Clear' command in the 'Recorder' menu or by clicking the 'Clear FireSpy recorder memory' button in the toolbar.

The 'Recorded bytes' box in the toolbar will display the number of recorded bytes, and the 'Record progress bar' in the toolbar will indicate the part of recorder memory that is filled with data.

Note that when the Recorder is not triggered yet, only the part before the trigger indicator in the progress bar (the little vertical line) will be filled. If this part becomes full, recording continues and old data will be removed from the memory while the new data is stored (cyclic buffer). After the Recorder is triggered, the data will be stored after the trigger position until the recorder memory is full.

If recorder memory becomes full, the recording is automatically stopped. You can also stop the recording manually by selecting the 'Stop' command in the 'Recorder' menu, or clicking the 'Stop Recording' button in the toolbar.

The hardware-Filter logic inside the Analyzer can be used to selectively store packets. See Filter/Trigger for more details.

Third Generation Analyzers:

For third generation analyzers the Recorder works differently. When the Recorder is started as described above it will start filling a buffer in on-board memory and at the same time the host will start reading data from this on-board buffer as fast as possible. Data is written to a temporary file on the host pc. This results in the possibility to make recordings of up-to 10GByte data. However, please keep in mind that if the host pc can not keep up the Recorder will simply stop when the FireSpy on-board buffer is full. Therefore, it is important to use a fast hard drive or SSD to hold the temporary Recording file. This can be configured through the settings dialog: Recording Settings.

7.1.1.3. Download recorded packets

When the recording has been stopped, you can download the recorded packets by selecting the 'Download' command in the 'Recorder' menu or clicking the 'Download recorded data from Analyzer' button in the toolbar.

After the download, the data can be viewed using the different kind of views.

Note that initially, only basic packet information is downloaded from the Analyzer. As soon as more detailed information is needed (e.g. to display the data quadlets of the packet), this information is read from the Analyzer. Thus you cannot clear the recorder memory as long as you want to view the recorded data.
You can however save the recorded data to file using the 'Save As' command in the 'File' menu and open this file using the 'Open' command in the 'File' menu. By doing so, the data in the recorder memory is not needed anymore and can be cleared.

7.1.1.4. Triggering

The Recorder can be triggered in different ways:

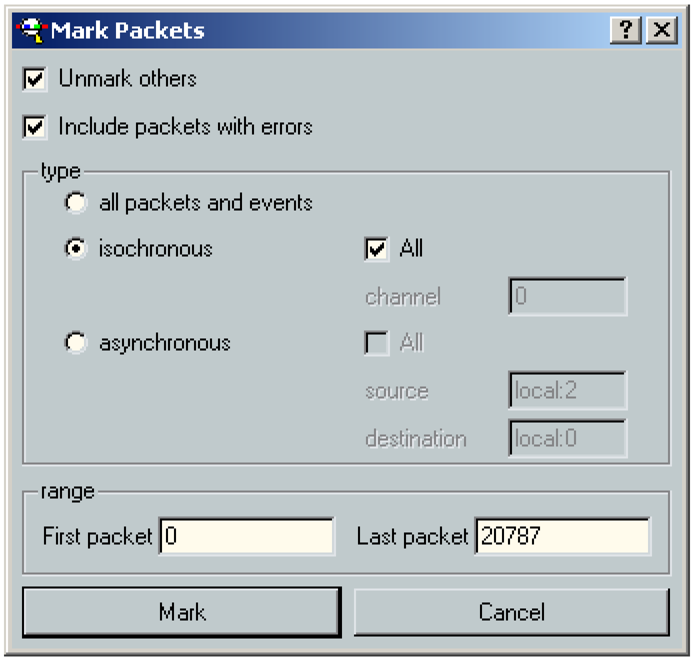
- You can press the trigger button on the Analyzer device.
- You can apply a trigger pulse on the external trigger input on the Analyzer device.
- You can click the 'Trigger' button in the toolbar or invoke the 'Trigger' command in the 'Recorder' menu.
- You can instruct the hardware-trigger logic to generate a trigger when a sequence of specified conditions are met.

The hardware-trigger logic will be described in full detail in 'Filter/Trigger'.

Each time you start the Recorder, a check is made if the user changed some settings in the Filter/Trigger Settings window. If so, the user will be asked if these changes need to be activated or ignored before starting the recording.

7.1.1.5. Marking packets

Packets and events can be marked. Marked packets are indicated by a black dot left of the packet in the Packet View. Below is an example of the Packet View with some marked packets and events.


Mark Packets

- [✓] Unmark others
- [✓] Include packets with errors

type
- ( ) all packets and events
- (●) isochronous — [✓] All; channel: 0
- ( ) asynchronous — [ ] All; source: local:2; destination: local:0

range
- First packet: 0
- Last packet: 20787

For more information about packet marking, see 'Packet Marking'.

7.1.2. Details

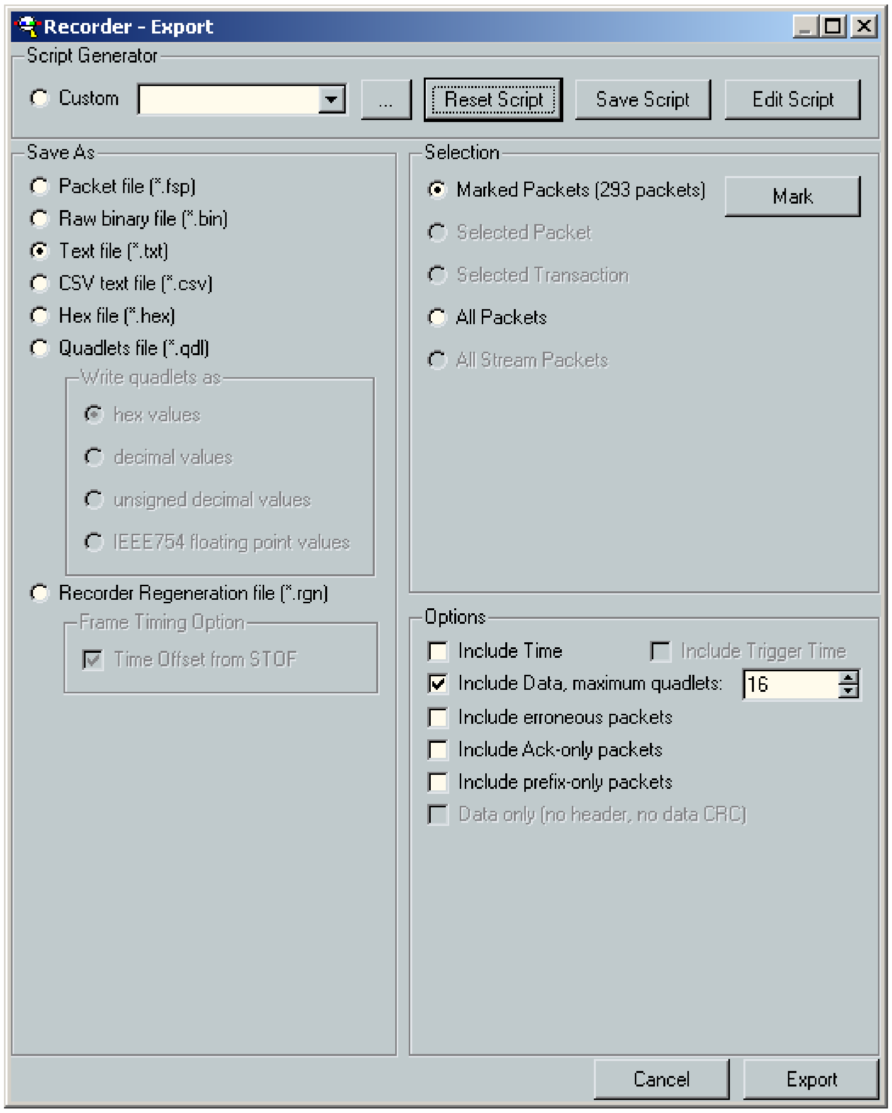
7.1.2.1. Menus

File

Open
With this command you can open an existing Analyzer recorder file. By default these files have an extension of .fsr.

Save As
With this command you can save the current Recorder data as a Analyzer recorder file. By default, these files have an extension of .fsr.
Recorder data saved this way, can later be viewed again by using the 'Open' command of the 'File' menu.

Export Packets...
With this command you can save one or more packets into a text file or Analyzer Packet file. Packet files have an extension of .fsp.
A Analyzer Packet file with isochronous packets can be used by the isochronous Generator to generate the isochronous packets stored in this file.
When selecting this command, the dialog window below is displayed.



Recorder - Export

Script Generator

- ( ) Custom
- Reset Script | Save Script | Edit Script

Save As

- ( ) Packet file (*.fsp)
- ( ) Raw binary file (*.bin)
- [✓] Text file (*.txt)
- ( ) CSV text file (*.csv)
- ( ) Hex file (*.hex)
- ( ) Quadlets file (*.qdl)
  - Write quadlets as
    - [✓] hex values
    - ( ) decimal values
    - ( ) unsigned decimal values
    - ( ) IEEE754 floating point values
- ( ) Recorder Regeneration file (*.rgn)
  - Frame Timing Option
    - [✓] Time Offset from STOF

Selection

- [✓] Marked Packets (293 packets) — Mark
- ( ) Selected Packet
- ( ) Selected Transaction
- ( ) All Packets
- ( ) All Stream Packets

Options

- [ ] Include Time
- [ ] Include Trigger Time
- [✓] Include Data, maximum quadlets: 16
- [ ] Include erroneous packets
- [ ] Include Ack-only packets
- [ ] Include prefix-only packets
- [ ] Data only (no header, no data CRC)

Cancel | Export

It has the following parts:

- *Script Generator:* All export functions, regardless of the pre-defined (the options described in the *Save As* pane) or customized, work based on scripts running behind this dialog. If you need to modify a pre-defined or customized script, you will need click on the "Edit Script" button in this pane. For detailed descriptions of the scripts, please refer to the Scriptable Export topic.
- *Selection*: In this box you specify which packets to export. You can choose between the currently marked packets, the currently selected packet or the currently selected transaction (which may consist of one or more packets). Or if you like to ignore the current selected or marked packets completely, you can then select "All Packets." Note that bus reset events may be marked too, and thus can be exported too. This however only works for text output files. "All Stream Packets" only becomes enabled when "Recorder Regeneration" format is selected.
- *Save As*: In this box you can choose one file format from three types of binary formats (*.fsp, *.bin, *.rgn) or four types of the textual file formats (*.txt, *csv, *.hex, *.qdl). Please note that the output options will be disabled unless valid license keys are present (except *.fsp and *.bin options). The text format is more versatile, it will include bus resets (if marked and the marked selection is exported). Available formats are as followings.
  - *Packet file (*.fsp)*: A file in this format can be imported by the Generator. The format is described in Analyzer packet file.
  - *Raw binary file (*.bin)*: Binary data will be written to a file in Quadlet (32-bit) boundary.
  - *Text file (*.txt)*: A packet line starts with a decimal number of the timestamp (if applicable) and the type of packets followed by attributes of the packet in "'attribute'='data'" fashion.
  - *CSV Text file (*.csv)*: Comma Separated Value File Format - a file in this format can be imported by Microsoft Excel or other applications. A file in this format can be imported by the Generator. Data payload contents can be written to a file; a column named "Data" will be

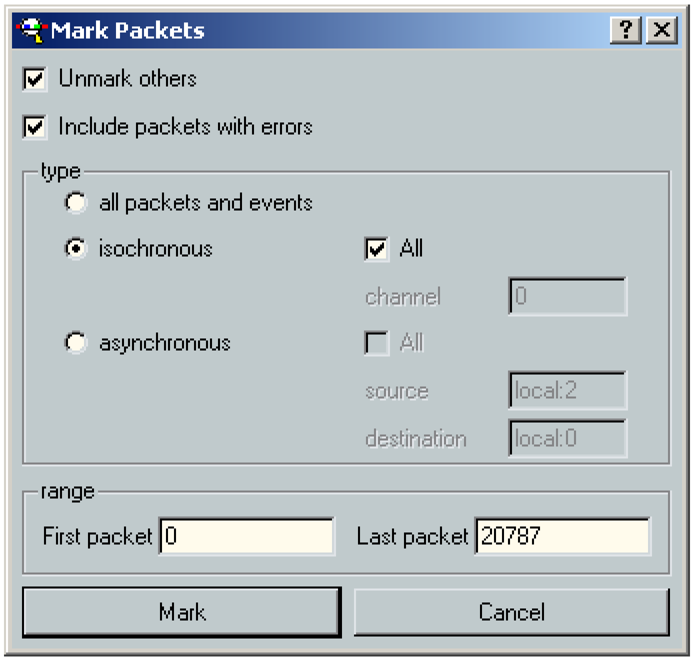
attached as the last column title and one column of hex data represents one quadlet data in the data payload.
    - *Hex file (*.hex)*: A file in this format can be imported by the Generator. The format is described in Hex Data file.
    - *Quadlets file (*.qdl)*: A file in this format can be imported by the Generator. The format is described in Quadlets Data file.
    - *Recorder Regeneration file (*.rgn)*: A file in this format can be used for the regeneration of the recorded data. Relative transmitted time interval will be preserved. The format is described in Reorder Regeneration file.
- *Options*: may be enabled/disabled depending on the selected file format in "Save As" above. Supported Save-As formats are denoted at the end of each description in parenthesis.
    - *Include Time*. This will include a decimal value in front of the packets or bus reset, indicating the start time of the packet. It expresses the value of an internal counter (at the moment of packet/bus reset start) that counts at 49.152 Mc.
    - *Include Data*. This will include the packet data (if present). You must also define the maximum number of quadlets that will be displayed for each packet. (txt, csv)
    - *Include erroneous packets*. With this option checked, the packets with errors will be included too. For erroneous packets, the packet name will be preceded by a '!' and error messages will be displayed just after the packet description (before the optional data). (txt, csv)
    - *Include Ack-only packets*. When this option is selected, the acknowledge packets that have no associated packet, will be displayed too. (txt, csv)
    - *Include prefix-only packets*. This will include the prefix-only packets, which are packets for which the Analyzer only sees the prefix signal on the bus. (txt)
    - *Data Only*: Save only data part of the packet. Header and data CRC will not be saved. (bin, hex, qdl)
- *Export*: With the export button the selected packets are saved in the selected format. A file save dialog will appear to specify the file name and location.
- *Cancel*: With the cancel button the export is canceled. Nothing will be saved to file.

Search

Advance Search…
This command will open the Advance Search dialog. You can mark and unmark selected packets with it. You can select packets on their type, speed, acknowledge type or packet header fields and data values. Using packet sets and a Boolean expression to select a combination of sets makes a very specific search possible. See 'Packet Searching' below for more information. Note that the 'Advance Search' function will be disabled if there is no license key available for version 2 software (see 'License Manager').

Redo Search
This command will search packets based on the last settings of the Advance Search dialog. You can skip opening the Advance Search dialog if you simply need to repeat the same search.

Mark
You can use this command to mark 'All' or a 'Range' of packets. When marking a 'Range' of packets, the following dialog appears:


Mark Packets

[✓] Unmark others

[✓] Include packets with errors

type

( ) all packets and events

(●) isochronous — [✓] All — channel: 0

( ) asynchronous — [ ] All — source: local:2 — destination: local:0

range

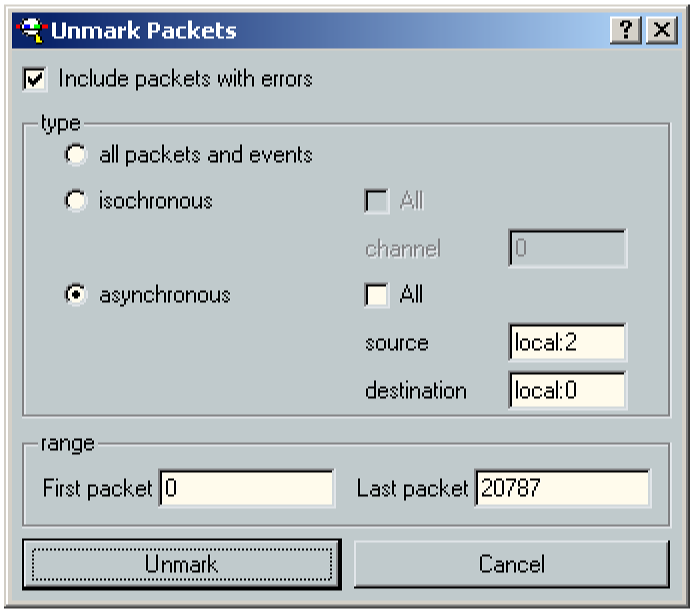
First packet: 0 — Last packet: 20787

Mark | Cancel

The dialog consists of the following parts:

- *Unmark others:* If this box is checked (default) only the packets selected by the dialog will be marked and all others will be unmarked. If this box is not checked the packets selected by the dialog will be marked additionally to the already marked packets and events.
- *Include packets with errors:* If this box is checked (default) packets with errors will be marked too, otherwise they will not be marked.
- *Type*: In this box you specify the type of packets to be saved. You can choose between isochronous and asynchronous. For isochronous packets, you can select between all isochronous packets or one specific channel number. In the second case, only packets with this channel number will be marked. For asynchronous packets you can select between all asynchronous packets or packets with specific source and destination IDs. In the second case, only asynchronous packets with this source ID and destination ID will be marked. The source and destination boxes use a special format. They specify a bus and node number, separated by a ':' character. If the bus number is 1023, the value 1023 is substituted by the word 'local'. The initial values in this box depend on the packet that is selected in the Recorder at the moment you invoke this command.
- *Range*: In this box you specify the range of packets to be stored. You specify a 'First packet' number and the 'Last packet' number. All packets in this range that fit the type above, will be stored. The number of a packet is displayed (when the packet is selected) in the toolbar of the Packet View (see below). Initially the packet range specifies all packets in the Recorder.
- *Mark*: Clicking the 'Mark' button will mark the specified packets (and optionally unmark the others). After that, the dialog quits.
- *Cancel*: Clicking the 'Cancel' button will cancel the mark operation and no packets will be marked or unmarked.

Unmark
You can use this command to unmark 'All' or a 'Range' of packets. When unmarking a 'Range' of packets, the following dialog appears:


Unmark Packets

[✓] Include packets with errors

type

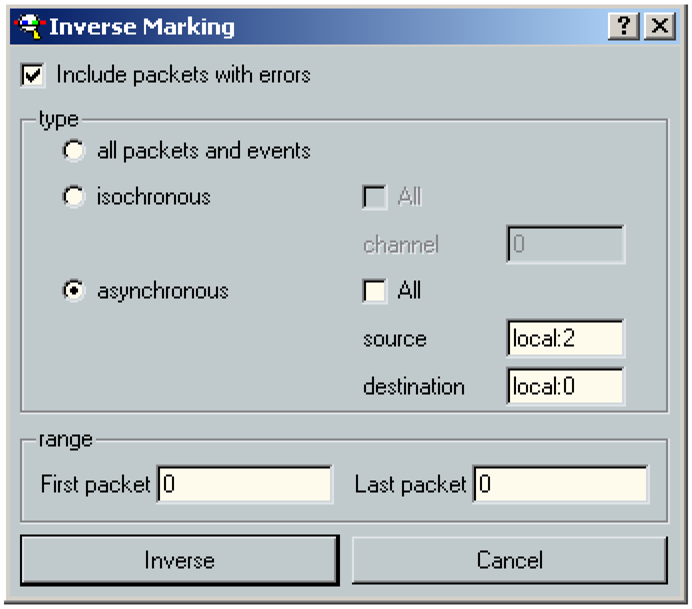
- ○ all packets and events
- ○ isochronous — ☐ All; channel: 0
- ● asynchronous — ☐ All; source: local:2; destination: local:0

range

First packet: 0 Last packet: 20787

This dialog looks almost the same as the mark dialog above. The only difference is that the 'unmark others' checkbox is missing and there is an 'Unmark" button instead of the 'Mark' button. The initial values are different in the example above because apparently an asynchronous packet with source local:0 and destination local:1 was selected when this command was invoked.
When the 'Unmark' button is clicked the specified packets will be unmarked and the dialog quits.

Inverse Mark
You can use this command to inverse 'All' or a 'Range' of packets. When inverse-marking a 'Range' of packets, the following dialog appears:


Inverse Marking

[✓] Include packets with errors

type

- ○ all packets and events
- ○ isochronous — ☐ All; channel: 0
- ● asynchronous — ☐ All; source: local:2; destination: local:0

range

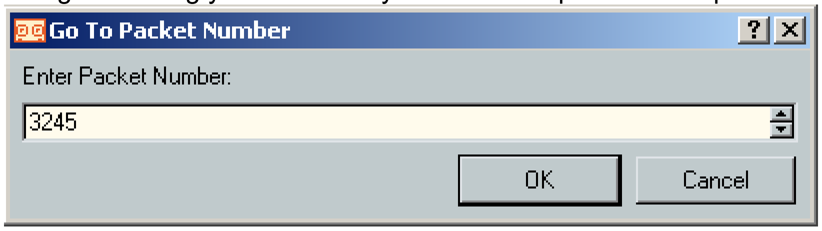
First packet: 0 Last packet: 0

This dialog looks almost the same as the unmark dialog above. The only difference is that there is an 'Inverse" button instead of the 'Unmark' button. The initial values are different in the example above because apparently an asynchronous packet with source local:0 and destination local:1 was selected when this command was invoked.
When the 'Inverse' button is clicked the specified packets will be unmarked if already marked, marked if already unmarked.

Go to first Mark
Go to previous Mark
Go to next Mark
Go to last Mark

When one or more packets are marked, you can use these commands to select the indicated packet. For the 'Go to previous Mark' and 'Go to next Mark' there are also corresponding buttons in the Packet View toolbar (see below).

Go to packet number ...
Using this dialog you can directly switch to the packet with specified number.



Next
With the 'Next' command you can select respectively the 'next Packet', 'next Request Packet', 'next Response Packet', 'next Cycle Start Packet', 'next Stream Packet' or 'next Phy Packet'.
The search for next packet will start at the currently selected packet, and if no packet is selected, at the current cursor position.

Previous
With the 'Previous' command you can select respectively the 'previous Packet', 'previous Request Packet', 'previous Response Packet', 'previous Cycle Start Packet', 'previous Stream Packet' or 'previous Phy Packet'.
The search for previous packet will start at the currently selected packet, and if no packet is selected, at the current cursor position.

Hide

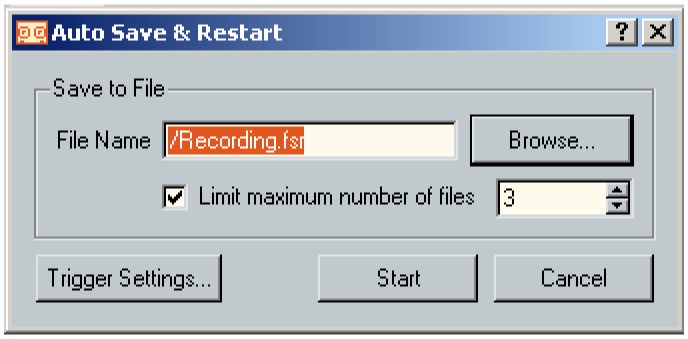
Hide marked packets
This command hides all currently marked packets and re-indexes packets that will be displayed.

Hide unmarked packets
This command hides all currently unmarked packets and re-indexes packets that will be displayed.

Unhide all
This command displays all packets regardless of marked/unmarked status.

Recorder
In this menu you will find the following Recorder control commands: 'Start', 'Stop', 'Clear', 'Download', 'Trigger' and 'AutoSave...'. They have the same functionality as the corresponding Recorder toolbar buttons described below, except 'AutoSave...' which is not present in the toolbar.

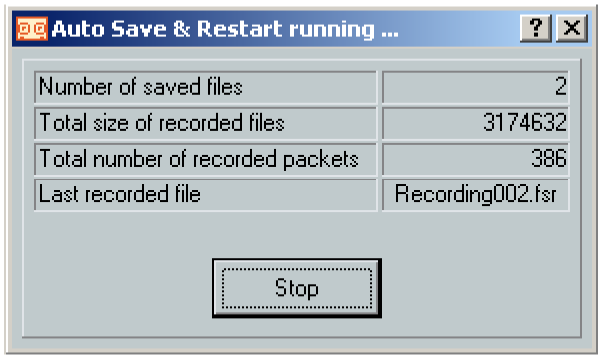
'AutoSave...' allow you to configure and make several recordings one by one. Following dialog will appear on this command.



Using this dialog you can select location and 'BaseName' for the recordings, number of the recordings and configure trigger.
Recordings will be stored on selected location in files with file name in format [BaseName][Number].fsr, where 'Number' always starts with '001'. Maximum number of recordings is defined if checked box is

checked, otherwise there is no limit and process has to be stopped manually using 'Stop' button.

First recording will start when you press 'Start' button and the rest will be done automatically. Each recording ends after a trigger, which you can configure using 'Trigger Settings...'.



Number of saved files	2
Total size of recorded files	3174632
Total number of recorded packets	386
Last recorded file	Recording002.fsr

View
With this menu you can enable and disable the different kind of Recorder views:

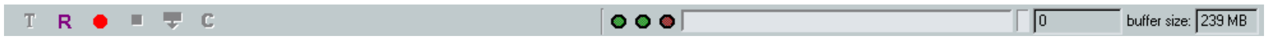
- Time View
- Topology View
- Packet View
- Transaction View
- Protocol View

All the views, except the protocol view, that are checked will be displayed. You can toggle between checked and un-checked by selecting the corresponding command. For protocol views, you select one of the supported protocols or select 'none' if you want no protocol view at all. Supported protocols will be disabled if no license is available for that protocol. See Protocol View and License Manager for more information.

Windows
From the 'Windows' menu you can open one of the other windows of the Analyzer.

7.1.2.2. Toolbar

The toolbar contains respectively the following indicators and buttons:



Buttons

Trigger
With this button the Recorder can be triggered. It is only enabled if recording is in progress and the Recorder is not triggered yet.
Start recording
With this button the Recorder can be started. After starting the Recorder, the recording will be in progress until it is stopped. It is enabled when the recorder memory is empty and the recording is not in progress yet.

Stop recording
With this button recording can be stopped. It is enabled when the recording is in progress.

Download recorded data from Analyzer
With this button the recorded data can be downloaded to the host to be displayed. It is enabled if recording is not in progress and the recorder memory is not empty.

Clear Analyzer recorder memory
With this data the recorder memory can be cleared. It will also clear the displayed data if the data of the Recorder is currently displayed. The button will be enabled if no record is in progress and the recorder

memory is not empty.

Indicators

Triggered indicator
This green indicator will light when the Recorder has been triggered since the last record start.
It has the same status as the 'trigger led' on the Analyzer front panel (see Hardware).

Ready indicator
This green indicator will light when the Recorder has been stopped and has recorded data in its memory ready for download.
It has the same status as the 'record ready led' on the FireSpy front panel (see Hardware).

Recording indicator
This red indicator will light when recording is in progress. Packets and events will be stored into the recorder memory when they appear on the IEEE1394 bus and when the Filter logic selects these events or packets to be stored.
It has the same status as the 'record led' on the Analyzer front panel (see Hardware).

Memory indicators

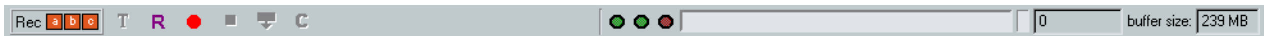
Record progress bar
This progress bar gives an indication which part of the recorder memory has been filed with packet and event data. The little vertical line indicates the trigger position. When starting recording, the memory before this trigger position (left to the trigger position) will be filled with data. If this part of the buffer gets full, the recording continues throwing away the oldest data and storing the new data (kind of cyclic buffer). After the Recorder is triggered, the data will be stored after the trigger position (right of the trigger position). If that part gets full, the Recorder is stopped.

Note that the trigger position within the recorder memory can be changed using the 'Settings' command in the 'Analyzer' menu. For more information see Settings.

Recorded bytes
This box displays the total stored number of bytes. It will increase during the recording except when the Recorder is not triggered yet and the memory part before the trigger position is full. In this case old data is removed when new data is stored and the total number of stored bytes will not change.

Recorder memory size
This box displays the total number of available memory bytes for the Recorder. It can be changed with the 'Settings' command in the 'Analyzer' menu. See Settings for more information.

Triple Analyzers
For triple Analyzers, the Toolbar looks as follows:



It contains the following additional control:

Rec
This control can be used to select the Analyzer nodes to use for recording. Clicking on the "a", "b" or "c" will activate/deactivate a node. Activated nodes are displayed with a red background color. Deactivated nodes are indicated with a white background color.

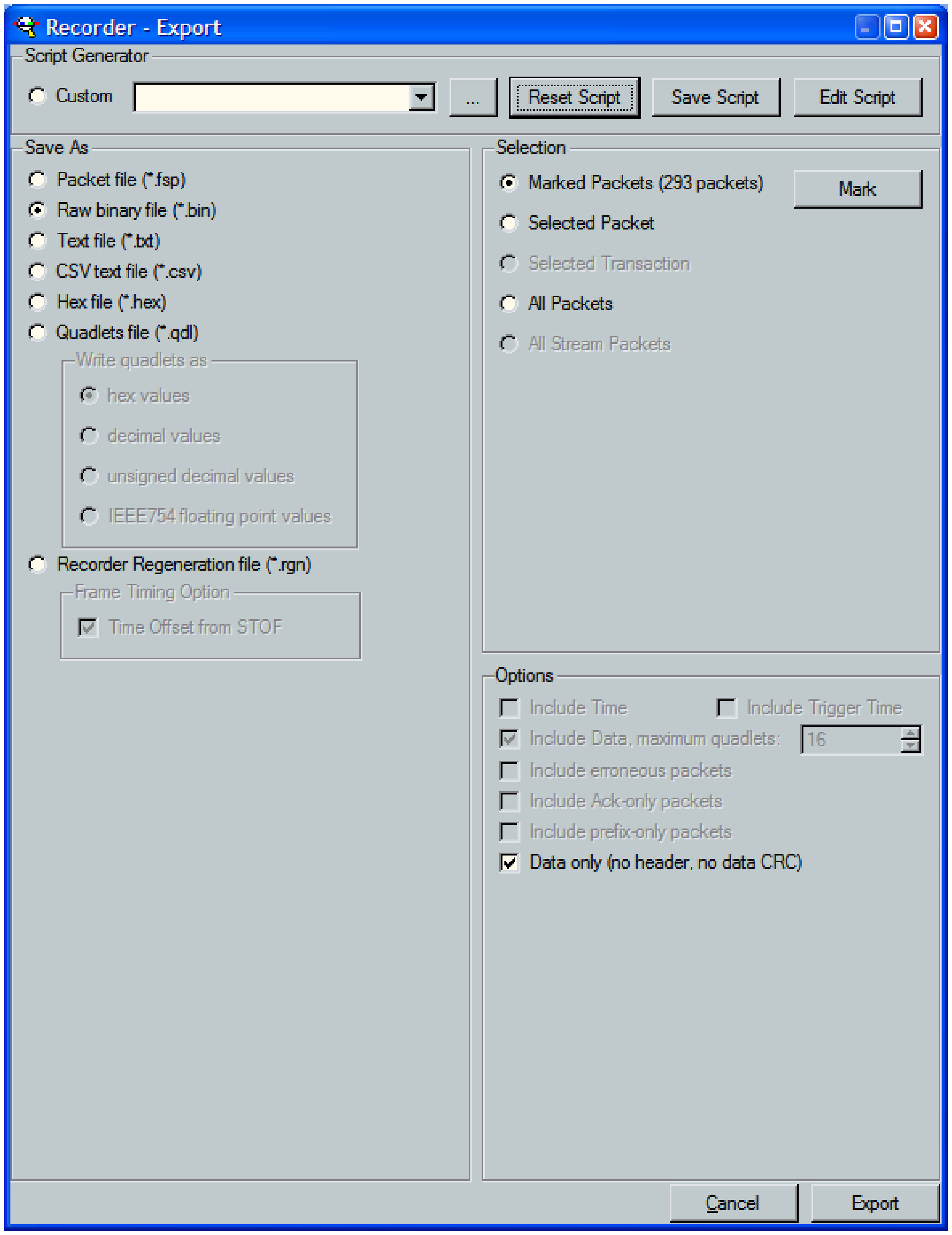
7.1.2.3. Scriptable Export

The Export functionalities have been greatly enhanced by adopting Dap Technology's proven technology: the scripting features that have been utilized in the Scriptor. The scripting engine for the Export functions runs on a host computer along with other Analyzer application components while the engine for the Scriptor runs on a RISC processor embedded on an Analyzer analyzer.

All pre-defined Export options, Packet file, Raw binary file, Text file, CSV file, Hex file, Quadlets file and Regeneration file, have their own scripts, and depending on the options selected, a different script source code will be generated, compiled and executed.

You can browse and modify actual script source code by either selecting one of the pre-defined file format or opening an external export script file (*.fes extension), and then click on the "Edit Script" button at the upper-right hand corner of the dialog. To open an export script file, please click on the "..." button to open a file open dialog.

Sample default Export dialog:


Recorder - Export

Script Generator
- Custom
- ...
- Reset Script
- Save Script
- Edit Script

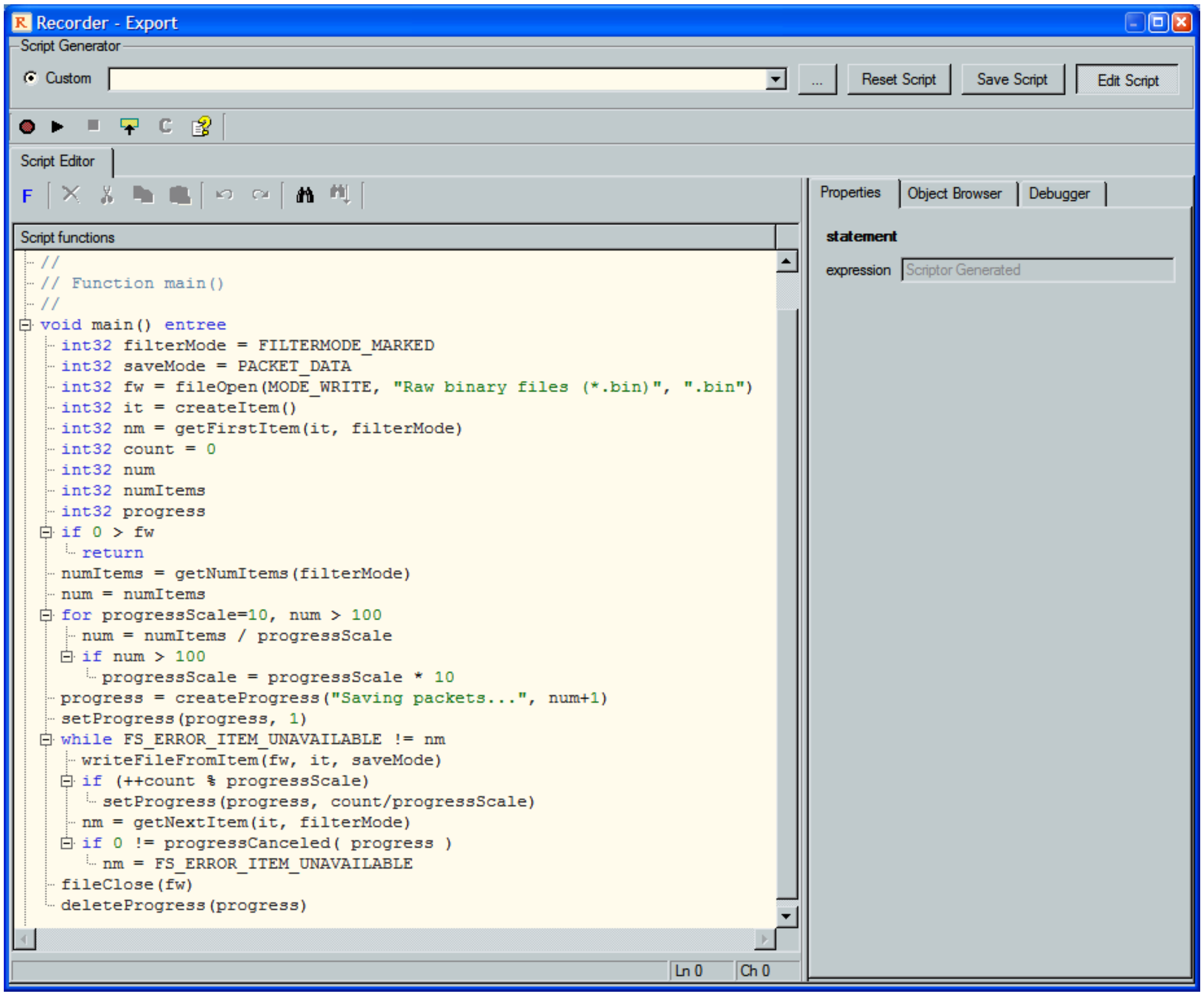
Save As
- Packet file (*.fsp)
- [✓ Raw binary file (*.bin)]
- Text file (*.txt)
- CSV text file (*.csv)
- Hex file (*.hex)
- Quadlets file (*.qdl)
  - Write quadlets as
    - [✓ hex values]
    - decimal values
    - unsigned decimal values
    - IEEE754 floating point values
- Recorder Regeneration file (*.rgn)
  - Frame Timing Option
    - [✓ Time Offset from STOF]

Selection
- [✓ Marked Packets (293 packets)]
- Mark
- Selected Packet
- Selected Transaction
- All Packets
- All Stream Packets

Options
- Include Time
- Include Trigger Time
- [✓ Include Data, maximum quadlets:] 16
- Include erroneous packets
- Include Ack-only packets
- Include prefix-only packets
- [✓ Data only (no header, no data CRC)]

Cancel
Export

Sample Export dialog in the script mode after clicking on the "Edit Script" button from the sample above:



```
//
// Function main()
//
void main() entree
  int32 filterMode = FILTERMODE_MARKED
  int32 saveMode = PACKET_DATA
  int32 fw = fileOpen(MODE_WRITE, "Raw binary files (*.bin)", ".bin")
  int32 it = createItem()
  int32 nm = getFirstItem(it, filterMode)
  int32 count = 0
  int32 num
  int32 numItems
  int32 progress
  if 0 > fw
    return
  numItems = getNumItems(filterMode)
  num = numItems
  for progressScale=10, num > 100
    num = numItems / progressScale
    if num > 100
      progressScale = progressScale * 10
  progress = createProgress("Saving packets...", num+1)
  setProgress(progress, 1)
  while FS_ERROR_ITEM_UNAVAILABLE != nm
    writeFileFromItem(fw, it, saveMode)
    if (++count % progressScale)
      setProgress(progress, count/progressScale)
    nm = getNextItem(it, filterMode)
    if 0 != progressCanceled( progress )
      nm = FS_ERROR_ITEM_UNAVAILABLE
  fileClose(fw)
  deleteProgress(progress)
```

The above picture shows the actual script source code for the Binary file option. This script is the least complex one among all pre-defined options. In this example, the script performs:
1. Sets the filter mode for traversing packets in the Recorder memory. (Marked is selected in the default dialog above.)
2. Sets the data save mode: data only or header and data. (Data only is selected because the "Data only" is checked in the default dialog.)
3. Opens a file selection dialog for output file. You can choose a file name and location.
4. Creates an item - a packet holder in this case.
5. Obtains the first item (packet) based on the filter mode. (First marked packet in this case)
6. Obtains the number of items the script needs to traverse. The number is used to calculate the ratio for displaying a progress bar.
7. Loops through the target packets (marked packets in this case), and write contents of the packet (data only in this case) to the output file.
8. Closes the output file and clean up resources upon the completion of the loop.

For complete sets of the Export script API, please expand the "Scriptor Generated" tree entry (scrolled out in the above picture) at the top of the script source code viewer.

For more information such as the script syntax, the debugger, etc., please refer to Script Editor topic.

7.1.3. Standalone Recorder

When the Standalone Recorder application is started, a similar device selection dialog is shown as when starting the main FireSpy application. However, rather than having to select a single FireSpy, this dialog allows selecting multiple FireSpys. Please note that starting the standalone Recorder with multiple FireSpys selected will only work when the selected FireSpy models support inter-FireSpy Recorder synchronization using a sync cable that needs to be purchased separately.

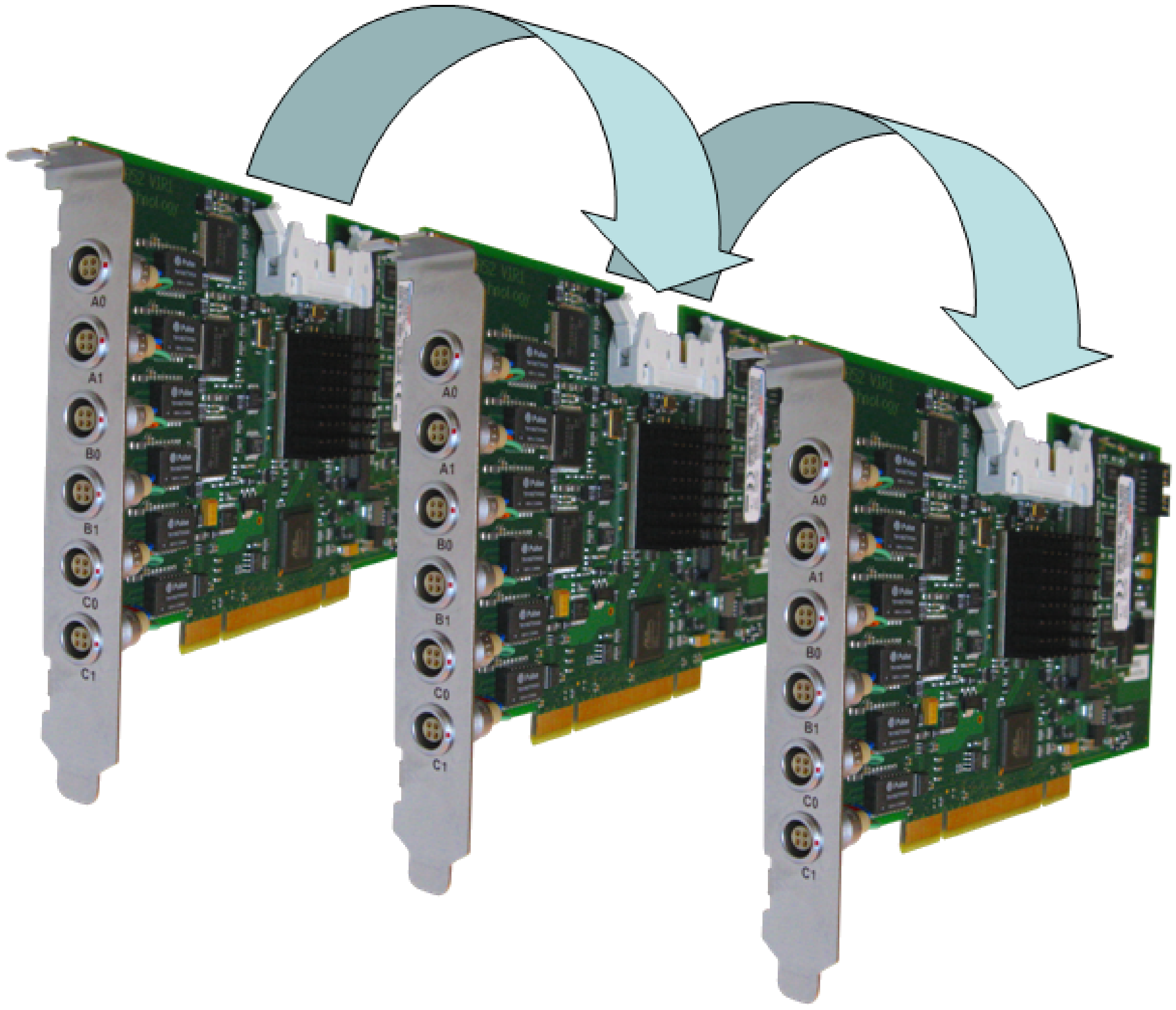
The standalone recorder can be started with the following command line options:

```
USAGE
RecorderApp [-h] or [/?]
        Shows command line usage

RecorderApp [options] [file] [commands]
        Recorder is run is a specific mode, on a specific device to start a recording or to open a recording.

OPTIONS
        -d --device serialnumber
                Opens the device which is identified by serialnumber.

        -u --unit number
                If a device contains multiple units, this option can specify which unit(s).
                Valid values are for example; 1 or 123

        --normal
                Opens the device in normal mode. When omitted it opens in the last used mode.

        --mil1394
                Opens the device in mil1394 mode. (better known as: AS5643)

        --minimized
                Opens the application minimized, so it won't be directly visible.

FILE
        filename.fsr
                Checks if the file exists, if it exists the recorder will open with that file loaded.
                Otherwise it will be created when a recording is saved.

COMMANDS
        start
                Will start the recorder immediately.

        stop [seconds]
                Will stop the recorder immediately or after a given amount of seconds.

        quit
                After the save is completed, the application will quit.

EXAMPLE
        RecorderApp.exe -d A123B --normal TenSecondRecording.fsr start stop 10 quit
```

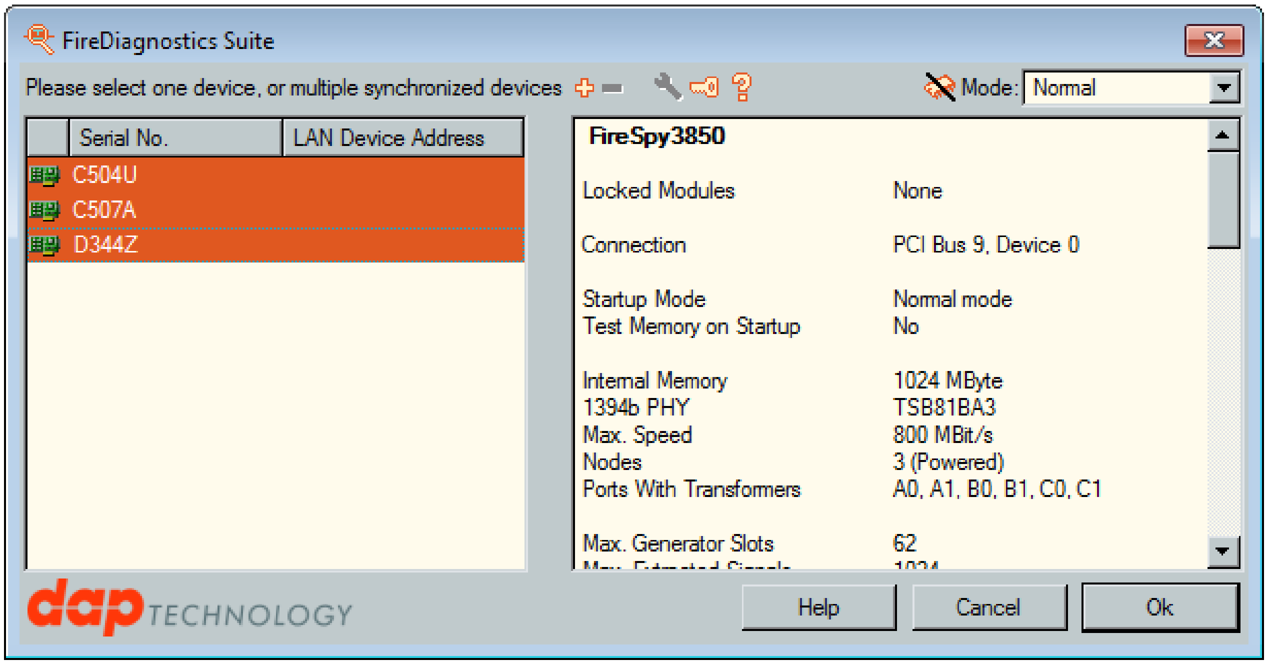
7.1.3.1. Multi-FireSpy Synchronized Recorder

When a FireSpy model supports connecting it to another FireSpy and synchronize their external timing modules, then the standalone Recorder supports controlling those synchronized FireSpy's as if it were one big analyzer with many buses. This allows making bus recordings with as many as 9-buses simultaneously in a single user interface.

The picture below shows where the synchronization cable needs to be connected for an FS3852 analyzer.



The picture below shows the option to select multiple FireSpys. Please ensure to only select multiple FireSpys that are actually time-synchronized. Otherwise, the application will exit with an error.



Please select one device, or multiple synchronized devices

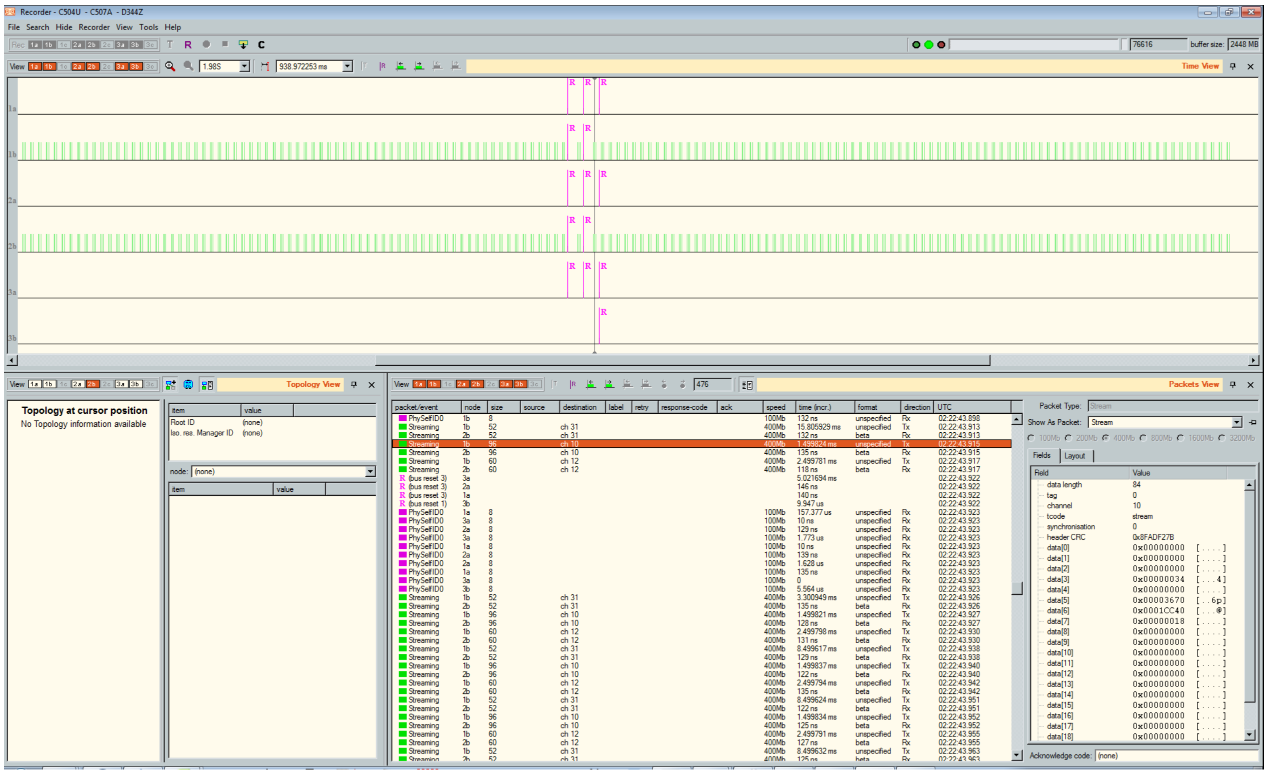
Mode: Normal

	Serial No.	LAN Device Address
	C504U	
	C507A	
	D344Z	

FireSpy3850

Locked Modules	None
Connection	PCI Bus 9, Device 0
Startup Mode	Normal mode
Test Memory on Startup	No
Internal Memory	1024 MByte
1394b PHY	TSB81BA3
Max. Speed	800 MBit/s
Nodes	3 (Powered)
Ports With Transformers	A0, A1, B0, B1, C0, C1
Max. Generator Slots	62

When multiple analyzers are properly synchronized then they can be controlled as one large FireSpy with many buses just as normally is done controlling a single FireSpy. The following picture shows the Recorder when 3 triple analyzers are open and a Recording was made with only 6 out of 9 buses selected.


Recorder - C504U - C507A - D344Z
File Search Hide Recorder View Tools Help
Rec 1a 1b 1c 2a 2b 2c 3a 3b 3c
76616
buffer size: 2448 MB
View 1a 1b 1c 2a 2b 2c 3a 3b 3c
1.98S
938.972253 ms
Time View
1a
1b
2a
2b
3a
3b
R
Topology View
Packets View
476

Topology at cursor position
No Topology information available

item	value
Root ID	(none)
Iso. res. Manager ID	(none)

node: (none)

packet/event	node	size	source	destination	label	retry	response-code	ack	speed	time (incr.)	format	direction	UTC
🟪 PhySelfID0	1b	8							100Mb	132 ns	unspecified	Rx	02:22:43.898
🟩 Streaming	1b	52		ch 31					400Mb	15.805929 ms	unspecified	Tx	02:22:43.913
🟩 Streaming	2b	52		ch 31					400Mb	132 ns	beta	Rx	02:22:43.913
🟩 Streaming	1b	96		ch 10					400Mb	1.499824 ms	unspecified	Tx	02:22:43.915
🟩 Streaming	2b	96		ch 10					400Mb	135 ns	beta	Rx	02:22:43.915
🟩 Streaming	1b	60		ch 12					400Mb	2.499781 ms	unspecified	Tx	02:22:43.917
🟩 Streaming	2b	60		ch 12					400Mb	118 ns	beta	Rx	02:22:43.917
R (bus reset 3)	3a									5.021694 ms			02:22:43.922
R (bus reset 3)	2a									146 ns			02:22:43.922
R (bus reset 3)	1a									140 ns			02:22:43.922
R (bus reset 1)	3b									9.947 us			02:22:43.922
🟪 PhySelfID0	1a	8							100Mb	157.377 us	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	3a	8							100Mb	10 ns	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	2a	8							100Mb	129 ns	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	3a	8							100Mb	1.773 us	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	1a	8							100Mb	10 ns	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	2a	8							100Mb	139 ns	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	2a	8							100Mb	1.628 us	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	1a	8							100Mb	135 ns	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	3a	8							100Mb	0	unspecified	Rx	02:22:43.923
🟪 PhySelfID0	3b	8							100Mb	5.564 us	unspecified	Rx	02:22:43.923
🟩 Streaming	1b	52		ch 31					400Mb	3.300949 ms	unspecified	Tx	02:22:43.926
🟩 Streaming	2b	52		ch 31					400Mb	135 ns	beta	Rx	02:22:43.926
🟩 Streaming	1b	96		ch 10					400Mb	1.499821 ms	unspecified	Tx	02:22:43.927
🟩 Streaming	2b	96		ch 10					400Mb	128 ns	beta	Rx	02:22:43.927
🟩 Streaming	1b	60		ch 12					400Mb	2.499798 ms	unspecified	Tx	02:22:43.930
🟩 Streaming	2b	60		ch 12					400Mb	131 ns	beta	Rx	02:22:43.930
🟩 Streaming	1b	52		ch 31					400Mb	8.499617 ms	unspecified	Tx	02:22:43.938
🟩 Streaming	2b	52		ch 31					400Mb	129 ns	beta	Rx	02:22:43.938
🟩 Streaming	1b	96		ch 10					400Mb	1.499837 ms	unspecified	Tx	02:22:43.940
🟩 Streaming	2b	96		ch 10					400Mb	122 ns	beta	Rx	02:22:43.940
🟩 Streaming	1b	60		ch 12					400Mb	2.499794 ms	unspecified	Tx	02:22:43.942
🟩 Streaming	2b	60		ch 12					400Mb	135 ns	beta	Rx	02:22:43.942
🟩 Streaming	1b	52		ch 31					400Mb	8.499624 ms	unspecified	Tx	02:22:43.951
🟩 Streaming	2b	52		ch 31					400Mb	122 ns	beta	Rx	02:22:43.951
🟩 Streaming	1b	96		ch 10					400Mb	1.499834 ms	unspecified	Tx	02:22:43.952
🟩 Streaming	2b	96		ch 10					400Mb	125 ns	beta	Rx	02:22:43.952
🟩 Streaming	1b	60		ch 12					400Mb	2.499791 ms	unspecified	Tx	02:22:43.955
🟩 Streaming	2b	60		ch 12					400Mb	127 ns	beta	Rx	02:22:43.955
🟩 Streaming	1b	52		ch 31					400Mb	8.499632 ms	unspecified	Tx	02:22:43.963
🟩 Streaming	2b	52		ch 31					400Mb	125 ns	beta	Rx	02:22:43.963

Packet Type: Stream
Show As Packet: Stream
100Mb 200Mb [✓ 400Mb] 800Mb 1600Mb 3200Mb

Fields | Layout

Field	Value	
data length	84	
tag	0	
channel	10	
tcode	stream	
synchronisation	0	
header CRC	0x8FADF27B	
data[0]	0x00000000	[....]
data[1]	0x00000000	[....]
data[2]	0x00000000	[....]
data[3]	0x00000034	[...4]
data[4]	0x00000000	[....]
data[5]	0x00003670	[..6p]
data[6]	0x0001CC40	[...@]
data[7]	0x00000018	[....]
data[8]	0x00000000	[....]
data[9]	0x00000000	[....]
data[10]	0x00000000	[....]
data[11]	0x00000000	[....]
data[12]	0x00000000	[....]
data[13]	0x00000000	[....]
data[14]	0x00000000	[....]
data[15]	0x00000000	[....]
data[16]	0x00000000	[....]
data[17]	0x00000000	[....]
data[18]	0x00000000	[....]

Acknowledge code: (none)

7.2. Time View

The Time View is one of the possible views in the Recorder which you can use to investigate the recorded packets and events. It displays all packets and events on a time line. The relative positions of the packets and events correspond to the time the packet or event was recorded and the length of packets in the view correspond to the actual duration of the packets.

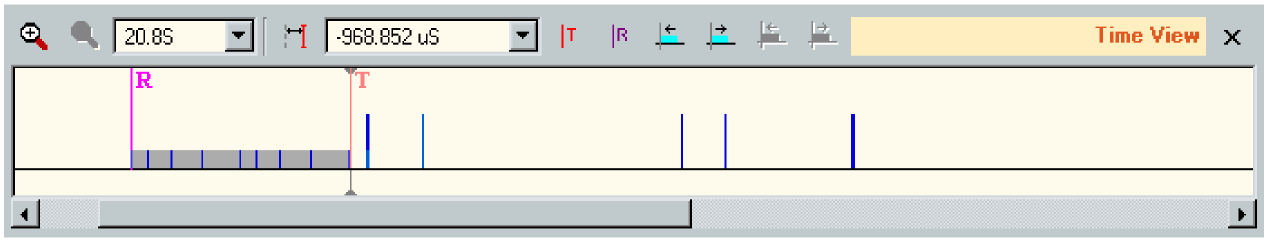
Note that before each packet you will also see the packet prefix. The duration of this prefix indicates the time the transmitting node needed to start transmitting the data from the moment that it has ownership of the bus. A shorter prefix means a faster reaction time of the node and a shorter overall packet transmit time, and thus a better bus efficiency

Besides the Reset event, some other events can be displayed in the Time View. For instance the position of subaction gaps and arbitration-reset gaps. These are only displayed if the recording of these events are enabled (see Filter/Trigger Settings) and when zoomed in far enough.

The Time View also displays the position of the trigger.

7.2.1. How to use it

You can display/remove the Time View by checking/unchecking the Time View item in the View menu of the Recorder. There is also a shortcut key to toggle the state. An example of the Time View is dispayed below:



At the top of the view you see a toolbar with some buttons and controls like zoom in and zoom out buttons, the cursor time indicator and buttons to quickly find other packets of the same type as the current selected one or packets with errors. Below the toolbar is the time line. It is a horizontal black line with the packets and events drawn on top of it.

Clicking with the mouse above the base line, will select the first packet or event that can be found to the right of the clicked position. The selected packet or event will automatically be selected in all other Recorder views.

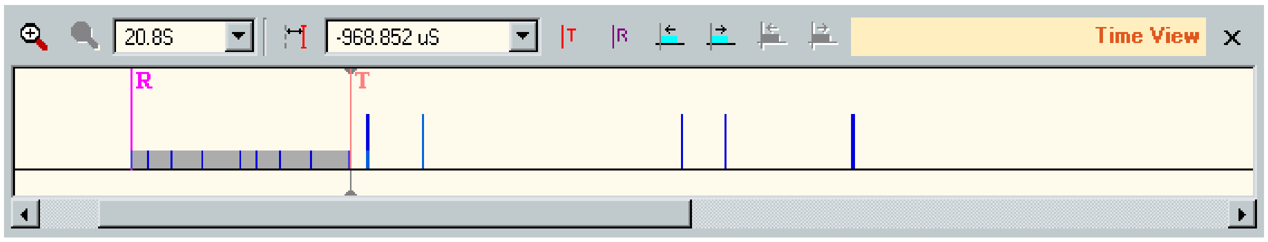
The time at the cursor position is displayed in the toolbar. It can display the absolute time or a relative time. In absolute time mode, time 0 is defined as the trigger time or if no trigger detected, the time of the very first packet or event. In relative mode the displayed cursor time is relative to a reference time. This reference time is indicated in the Time View as a vertical dotted line (see above). The reference time is set when switching from absolute to relative mode, in which case the reference time will be set to the cursor time. Additional mouse clicks will move the cursor, but not the reference. This makes time measurements very easy. To measure the time between the start of a request packet and the start of its response packet for instance, select the request packet, switch to relative mode and select the response packet. The time between them will be displayed in the toolbar. Note that the selection of the packets can also be done in one of the other views of the Recorder.

To measure packet lengths, prefix length etc., you can click with the mouse below the base line. This will move the cursor to that position, without searching the first packet to the right of it.

7.2.2. Details

The Time View displays all packets and events on a time line with a resolution of about 20 nS (1/(2*24.576 MHz)) to be exact) for a FireSpy400 and about 10 nS (1/(4*24.576 MHz)) for a FireSpy800 and newer.

It has a toolbar of its own and a part where the packets and events are displayed.



7.2.2.1. Toolbar

The Toolbar has the following indicators and buttons:

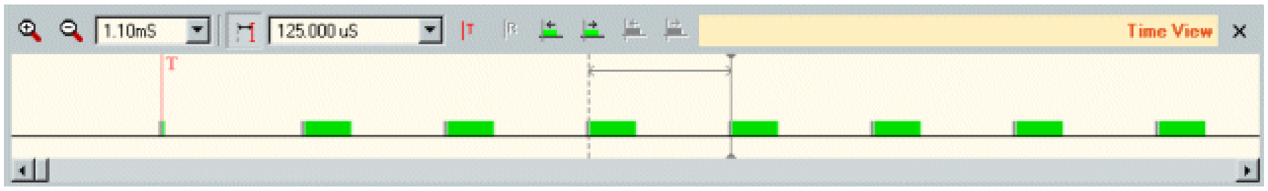
Zoom in
With this button you can zoom in. You will see more details of the packets.

Zoom out
With this button you can zoom out. You will see fewer details of the packets.

Time of display width
This box indicates the time that corresponds to the displayed time in the time view. In the example above this is 1.03 mS, or about 8 isochronous cycles.
An alternative way to zoom in and out is to select a value for this box from the popup list. The total zoom range is about 1:8000000.

Toggle between absolute and relative time
With this button you can toggle between absolute (default) and relative cursor mode. When not pressed, the cursor is in absolute mode; the time displayed in the 'Cursor time' box (next toolbar item) is the time of the cursor position expressed in seconds, after the trigger occurred. If no trigger position defined (Recorder not triggered) the time is expressed in seconds after the start of the first packet.
As soon as the button is pressed, the 'Cursor time' is relative to the cursor position at the time the button is pressed. Thus immediately after pressing this button, the 'Cursor time' will be zero (relative to itself).

When the cursor is moved now, the position of the old cursor (the reference) is indicated with a dotted line and a horizontal line with arrows is drawn between this reference and the new cursor position. See the time view above for an example. The time in the 'Cursor time' box indicates the length of the horizontal line, or the time between the old and the new cursor position. This feature can be used to easily measure time. For instance the time between the start of two isochronous packets in different cycles in the example below, which is exactly 125 uS.



When in relative mode, clicking with the right mouse button will move both the reference and the cursor to the new location. This makes time measurements easier because you do not have to switch back to normal mode to move the reference.
See also below how to move the cursor to specific locations.

Cursor time
This box indicates the absolute or relative cursor time as described above.
This box also has a popup list of the last few cursor positions. You can select one of them, and the cursor will be moved to that position.

Go to trigger position
When clicking this button, the cursor is moved to the trigger time. It is only enabled if the trigger position is defined (trigger occurred during recording).

Go to next Bus Reset
When clicking this button, the cursor is moved to the next Bus-Reset event. It is only enabled if one or more bus resets are present (recorded during the recording).

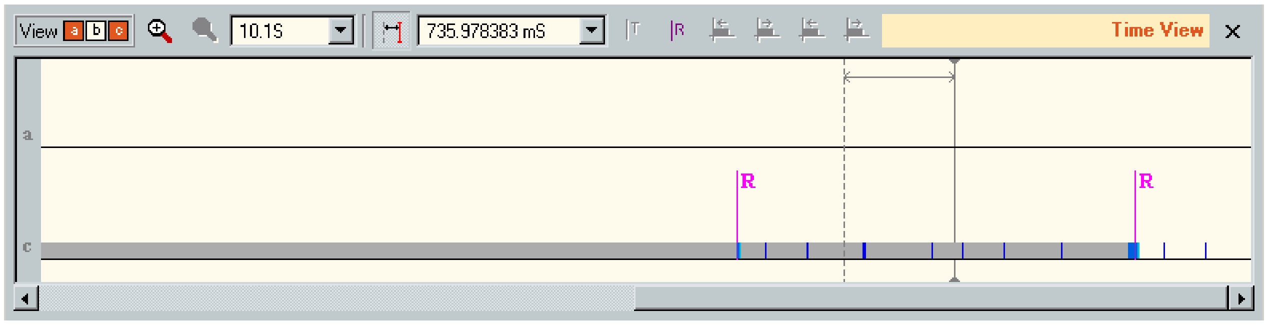
Go to previous packet of same type
When clicking this button, the cursor will be move to the previous packet of the same type. It will be enabled if such a packet is present.

Go to next packet of same type
When clicking this button, the cursor will be move to the next packet of the same type. It will be enabled if such a packet is present.

Go to previous packet with error(s)
When clicking this button, the cursor will be move to the previous packet that has one or more errors. It will be enabled if such a packet is present.

Go to next packet with error(s)
When clicking this button, the cursor will be move to the next packet that has one or more errors. It will be enabled if such a packet is present.

Triple Analyzers
For triple Analyzers, the Time View looks as follows:



It contains the following additional control:

View
This control van be used to select the Analyzer nodes to view information for. By clicking on the "a", "b" or "c", the corresponding nodes can be hidden or shown. A red background indicates a node is shown. A white background indicates a node is hidden.

7.2.2.2. Packet / event display

The packets and events are displayed below the toolbar. Packets (including acknowledge packets) are drawn as colored rectangles on a black 'base-line'. The color of the rectangle indicates the type of packet. The used packet colors are:

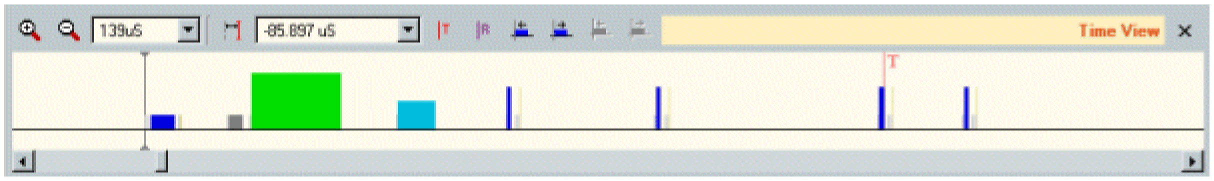
⬛ cycle start packets
🟩 stream packets
🟦 read/write/lock request packets
🟦 read/write/lock response packets
🟪 phy packets
🟥 packets with error(s)
🟨 acknowledge packets

Before each packet, a light gray rectangle is drawn, indicating the packet prefix. In case of a hidden packet, only this prefix is drawn, which has a duration of the complete packet.

The height of the packet rectangles indicates the packet speed. There are 4 different heights (from lowest to highest):

1. for the 100 Mb/s packets and packet prefix
2. for the 200 Mb/s speed packets
3. for the 400 Mb/s speed packets
4. for the 800 Mb/s speed packets (FireSpy800 only)

Below you see an example with a request and cycle-start packet at 100Mb/s speed, followed by a (big) stream packet at 800 Mb/s speed, followed by a response packet at 200Mb/s and after that a few (small) request packets at 400 Mb/s speed.

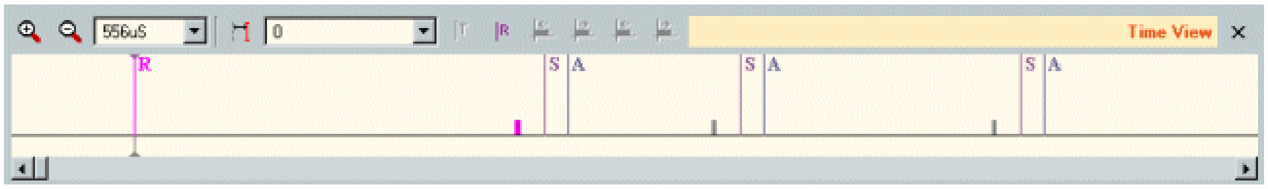


The cursor is a vertical line with little triangles on the top and bottom, as shown above (left to the first packet).
Events are drawn as vertical lines from the base line to the top, with a letter to the right of the line (somewhere at the top). The letter indicates the event type as follow:

T = Trigger
R = Bus Reset
S = Subaction Gap
A = Arbitration-Reset Gap or EventA (EventA has a darker Line)
B = EventB
C = EventC
C = Cycle Start (FireSpy800 only)
I = Phy Interrupt

The Trigger event will be displayed at the position where the trigger occurred. The position will correspond to the absolute time 0 (zero).
The Bus Reset event will be displayed at the position where the link interface signals a bus-reset status. Bus resets are always stored during the recording.
Both the Subaction Gap and Arbitration-Reset Gap events are displayed at the position where the link interface signals the corresponding status. By default these events are not stored when recording, but the

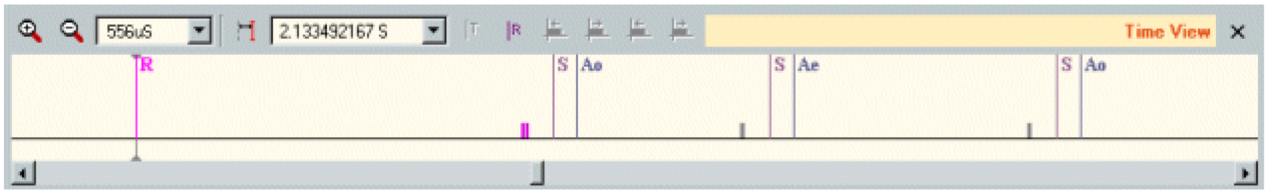
recorder filter logic can be instructed to store these events. See Filter/Trigger for more details. The example below shows the events described above. The Phy-Interrupt events are displayed at the position where the link interface signals a phy interrupt. By default this events is not stored when recording, but the recorder filter logic can be instructed to store the events. See Filter/Trigger for more details.



In this example a Bus Reset occurred, followed by two Phy-SelfID packets. After the second SelfID packet there is for some time no packet, resulting in the Subaction Gap and Arbitration-Reset Gap events. Then a cycle-start packet is transmitted and again a while nothing, resulting in the Subaction Gap and Arbitration-Reset Gap again, etc.

Note that the Subaction Gap and Arbitration-Reset Gap are only displayed when zooming in far enough, to avoid messing up the picture too much.

For the FireSpy800 the Arbitration-Reset Gap and Cycle Start can be of type 'odd' or 'even'. The example below displays the same situation recorded by a FireSpy800. The 'odd' and 'even' Arbitration-Reset Gaps are indicated with the letter 'o' respectively 'e' after the 'A'.



External events can also be recorded, external events are transitions from high to low on external IO connectors see: External Port Settings. If you want to record these events you need to check some settings in the software; In the settings menu you need to tick the /InA, /InB and/or /InC if you have connected hardware to any or all of these ports. Then in the FilterTrigger Settings go to the Filter Tab, there you can see that the default action is to skip External Events. To record them they should not be skipped and so we must uncheck the appropriate check boxes.

selecting packets
Packets can be selected by clicking with the mouse in the packet rectangle. The cursor will move to the selected packet and the selected packet will also be selected in the other views, so that the details can be seen. If a packet has an associated acknowledge rectangle, you also can click in the rectangle of this acknowledge packet or in the space between the packet and the acknowledge to select the packet. Packets and their associated acknowledges are handled as a unity in the Recorder, but not for the trigger logic as you can see in Filter/Trigger Settings. As a result, acknowledge packets cannot be selected separately if they belong to some packet. If an acknowledge has no associated packet (dangling acknowledge), which is an error condition, the acknowledge packet can be selected.

If you click with the mouse on an empty space above the base line, the first packet to the right of the clicked position will be selected and displayed.

moving cursor to specific position
If you click on the other hand below the base line, then the cursor will be moved to exactly that position. This can be used to measure the duration of, for example, the prefix of the packet itself or the time between a packet and its acknowledge.

7.3. Topology View

The Topology View is one of the possible views in the Recorder which you can use to investigate the recorded data. It displays the topology corresponding to the current cursor position. For a given cursor position, the last recorded selfIDs will be searched and the topology will be derived from these selfIDs.

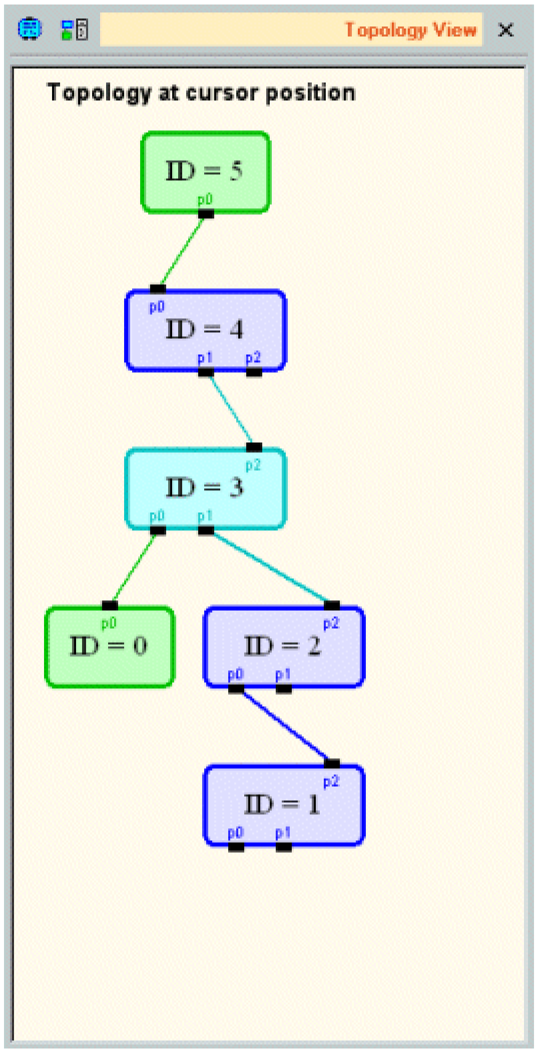
An optional pane can be switched on to display additional bus and node information. This information will be derived from recorded CSR register reads. For each node all Configuration-ROM reads will be collected and the contents of the Configuration ROM will be reconstructed if possible.

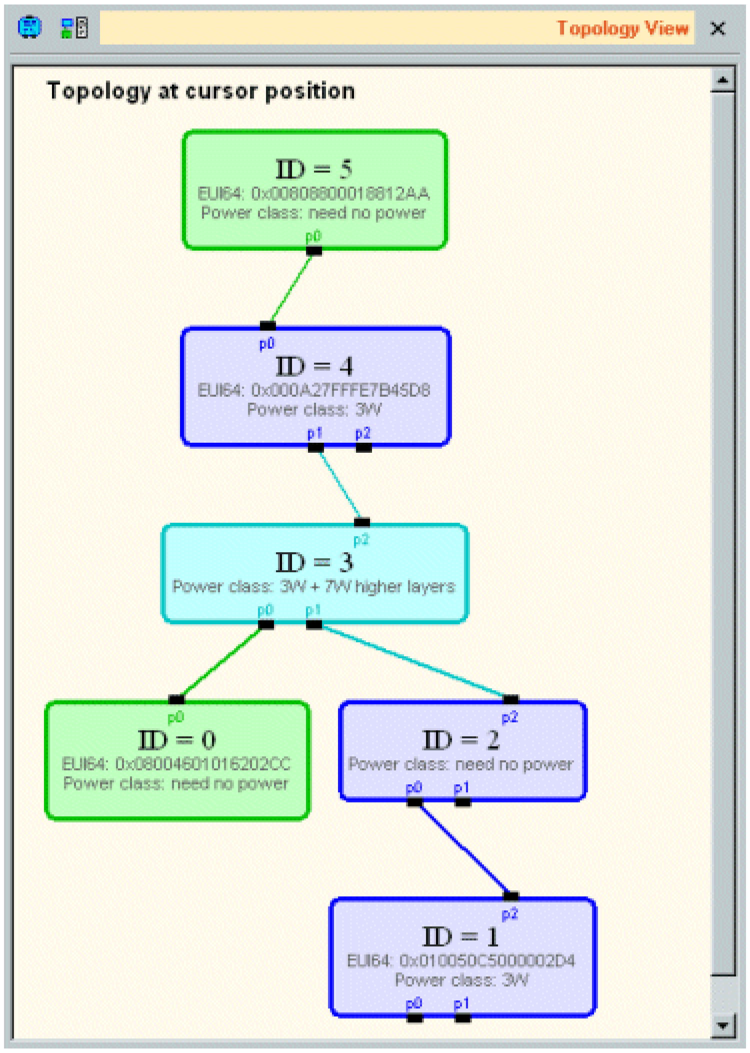
7.3.1. How to use it

You can display/remove the Topology View by checking/unchecking the Topology View item in the View menu of the Recorder. There is also a shortcut key to toggle the state.

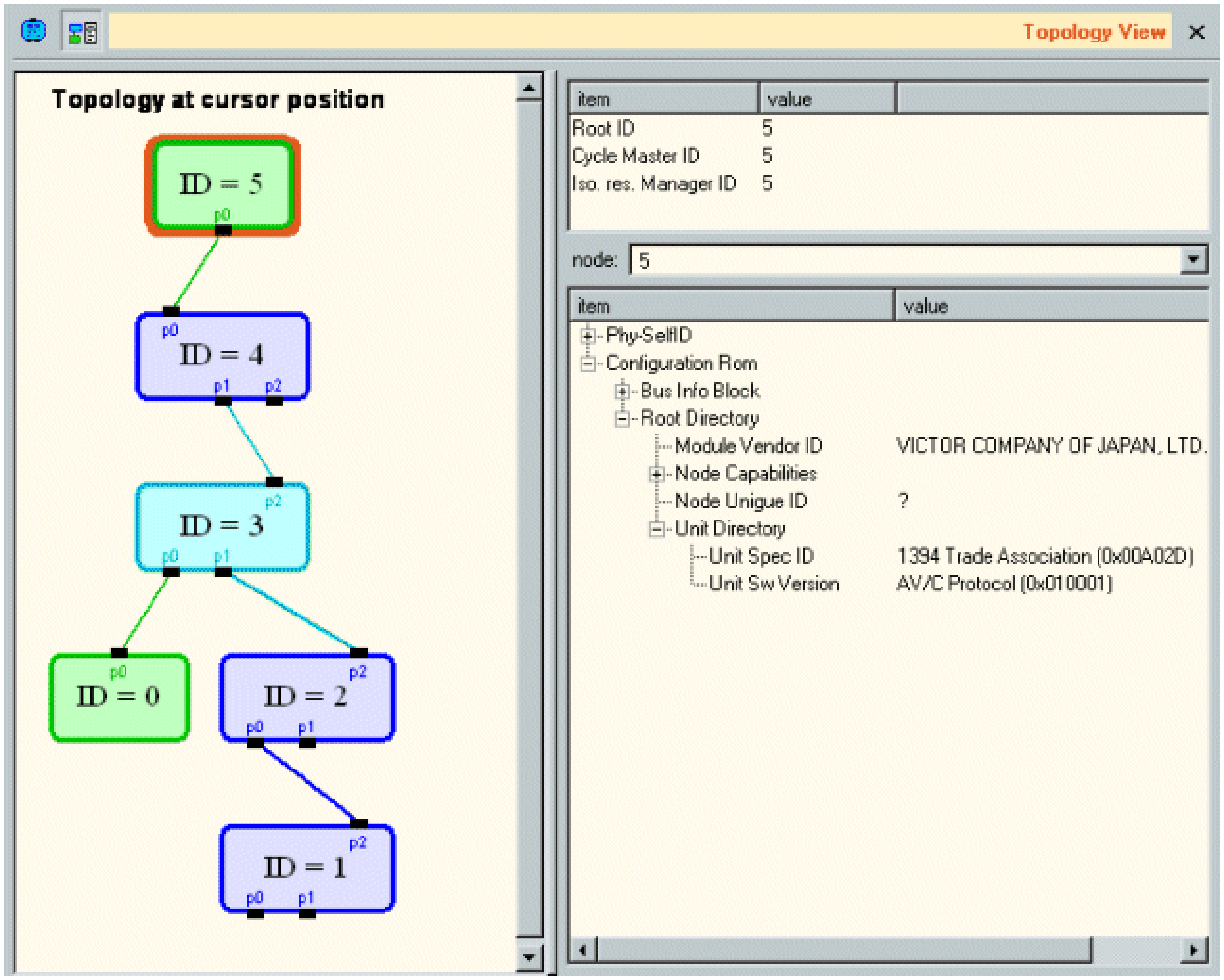
Set the cursor position by clicking in the Time View or by selecting a packet, transaction or protocol item in one of the other Recorder views. When a transaction or protocol item is selected, one of the corresponding packets will be selected too, and the cursor position is set to the selected packet. The Topology corresponding to the cursor position will be displayed in the Topology View.

Examples of the Topology View are dispayed below.
Using the 'Select node display options' button (left button on Topology Toolbar) you can display a dialog to add information to the nodes. In the right example below the 'EUI64' and 'Power class' options are added.




Topology View
Topology at cursor position
ID = 5
EUI64: 0x00808800018812AA
Power class: need no power
p0
p0
ID = 4
EUI64: 0x000A27FFFE7B45D8
Power class: 3W
p1
p2
p2
ID = 3
Power class: 3W + 7W higher layers
p0
p1
p0
ID = 0
EUI64: 0x08004601016202CC
Power class: need no power
p2
ID = 2
Power class: need no power
p0
p1
p2
ID = 1
EUI64: 0x010050C5000002D4
Power class: 3W
p0
p1

Using the 'Bus and node information' button (right button in Topology Toolbar) you can display additional bus and node information. In the example below this option is swithed on. The extra node information corresponds to node 5, the selected node.



item	value
Root ID	5
Cycle Master ID	5
Iso. res. Manager ID	5

node: 5

item	value
Phy-SelfID	
Configuration Rom	
Bus Info Block	
Root Directory	
Module Vendor ID	VICTOR COMPANY OF JAPAN, LTD.
Node Capabilities	
Node Unique ID	?
Unit Directory	
Unit Spec ID	1394 Trade Association (0x00A02D)
Unit Sw Version	AV/C Protocol (0x010001)

The extra node information consists of information from the selfIDs from this node and information from the Configuration Rom of this node. For values that could not be found (no read of the corresponding Configuration ROM address recorded) a '?' will be displayed.

7.3.2. Details

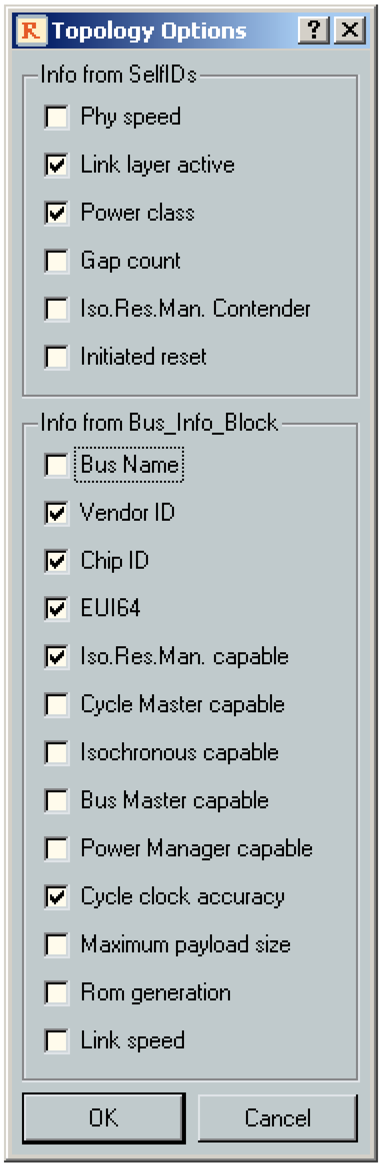
Because the topology can change during recording as a result of bus resets, the topology is time dependent. The cursor in the Time View defines the time for which the topology is drawn.

The topology view includes a little toolbar and, below that, an area where the topology picture will be drawn with optionally extra bus and node information at the right of the picture.

7.3.2.1. Toolbar

The toolbar has two buttons:

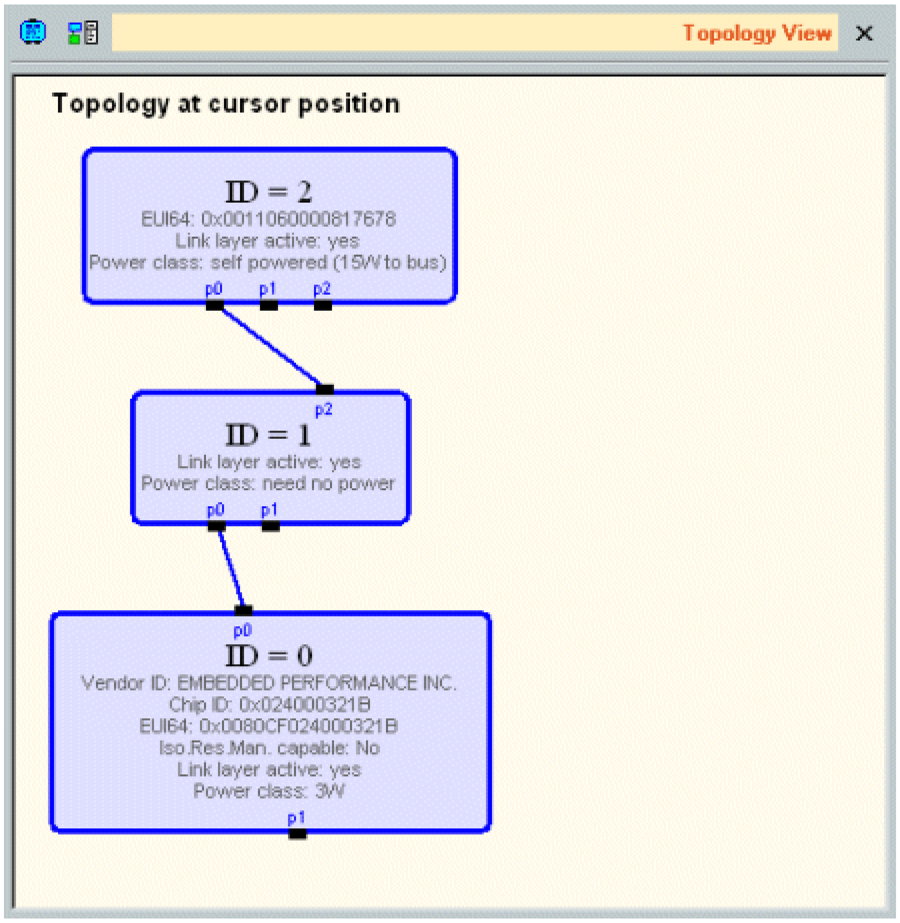
Select node display options
Clicking this button will bring up a dialog window to select the options you want to be displayed inside the nodes. The dialog with all the possible options is displayed below:



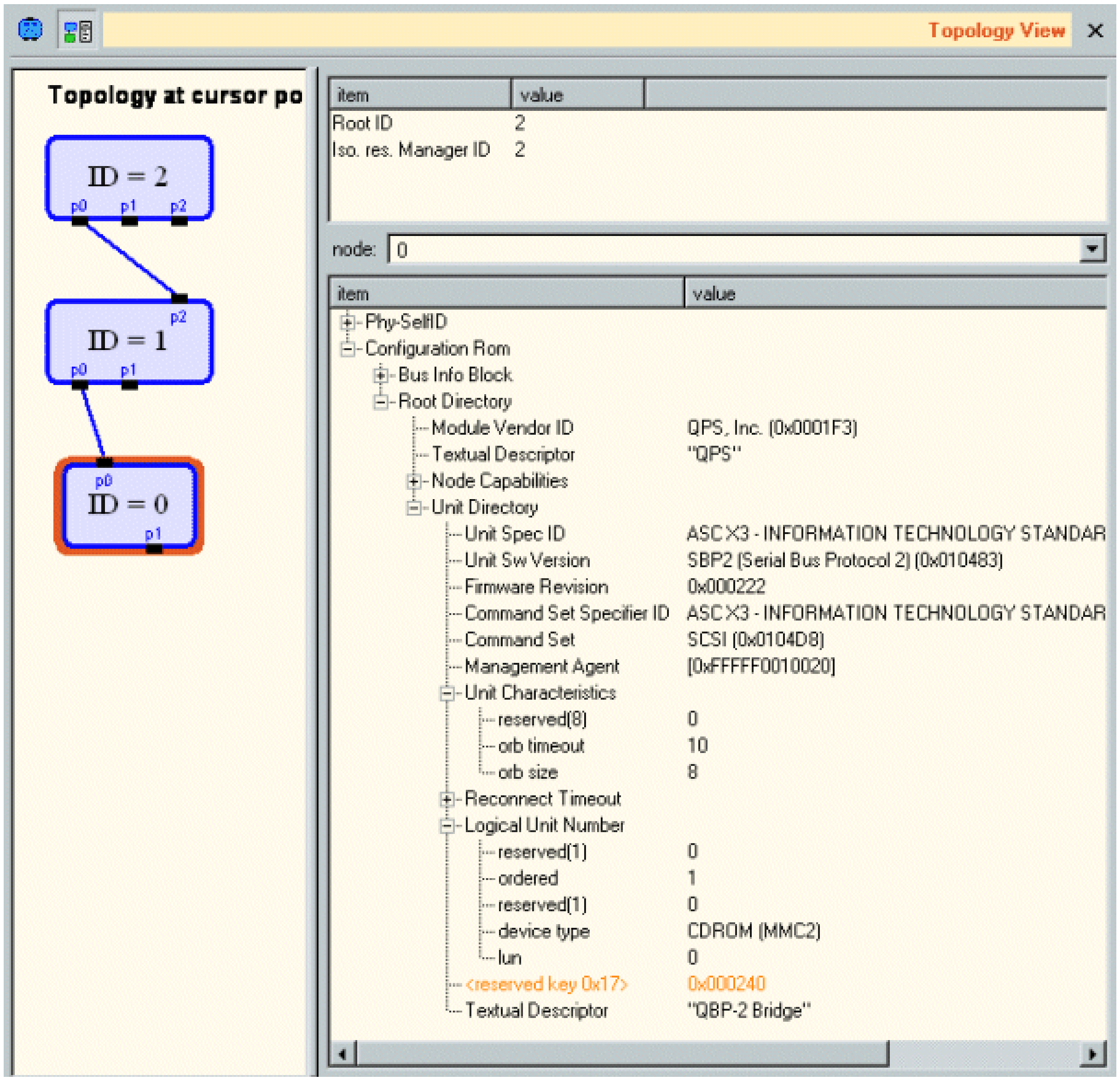
- [ ] Phy speed
- [✓] Link layer active
- [✓] Power class
- [ ] Gap count
- [ ] Iso.Res.Man. Contender
- [ ] Initiated reset

- [ ] Bus Name
- [✓] Vendor ID
- [✓] Chip ID
- [✓] EUI64
- [✓] Iso.Res.Man. capable
- [ ] Cycle Master capable
- [ ] Isochronous capable
- [ ] Bus Master capable
- [ ] Power Manager capable
- [✓] Cycle clock accuracy
- [ ] Maximum payload size
- [ ] Rom generation
- [ ] Link speed

The information for the upper options is retrieved from the selfID(s) of the corresponding node. The information for the lower options are retrieved from the Bus_Info_Block of the CSR Rom. This information will only be displayed if it can be found. See below for more information about this.
Below is an example of the topology view with the options enabled as shown in the 'Topology Options' dialog above.
Note that some information like the Vendor ID, Chip ID and EUI64 of node 1 could not be found and thus are not displayed.
Note also that the Vendor ID in the Bus_Info_Block is a 24 bit number. But vendor names are displayed inside the nodes. These names are looked up in an external file. See below for more information.


Topology View
Topology at cursor position
ID = 2
EUI64: 0x0011060000817678
Link layer active: yes
Power class: self powered (15W to bus)
p0
p1
p2
p2
ID = 1
Link layer active: yes
Power class: need no power
p0
p1
p0
ID = 0
Vendor ID: EMBEDDED PERFORMANCE INC.
Chip ID: 0x024000321B
EUI64: 0x0080CF024000321B
Iso.Res.Man. capable: No
Link layer active: yes
Power class: 3W
p1

Bus and node information
This is a toggle button. When this button is pressed, extra bus information and node information of a selected node will be displayed.
Below is an example of the topology view, with the 'Bus and node information' button pressed, and with node 0 selected. In this case the 'node display options' are all disabled.



Topology View

Topology at cursor po

ID = 2 (p0, p1, p2) — ID = 1 (p2, p0, p1) — ID = 0 (p0, p1)

item	value
Root ID	2
Iso. res. Manager ID	2

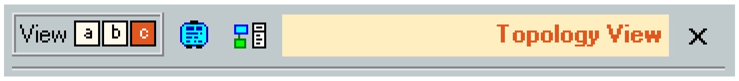
node: 0

item	value
Phy-SelfID	
Configuration Rom	
Bus Info Block	
Root Directory	
Module Vendor ID	QPS, Inc. (0x0001F3)
Textual Descriptor	"QPS"
Node Capabilities	
Unit Directory	
Unit Spec ID	ASC X3 - INFORMATION TECHNOLOGY STANDAR
Unit Sw Version	SBP2 (Serial Bus Protocol 2) (0x010483)
Firmware Revision	0x000222
Command Set Specifier ID	ASC X3 - INFORMATION TECHNOLOGY STANDAR
Command Set	SCSI (0x0104D8)
Management Agent	[0xFFFFF0010020]
Unit Characteristics	
reserved(8)	0
orb timeout	10
orb size	8
Reconnect Timeout	
Logical Unit Number	
reserved(1)	0
ordered	1
reserved(1)	0
device type	CDROM (MMC2)
lun	0
<reserved key 0x17>	0x000240
Textual Descriptor	"QBP-2 Bridge"

Note: The 'Root Directory' will not be displayed if there is no license key available for the version 2 software or higher. See License Manager.

Triple Analyzers

For triple Analyzers, the toolbar looks as in the following picture:



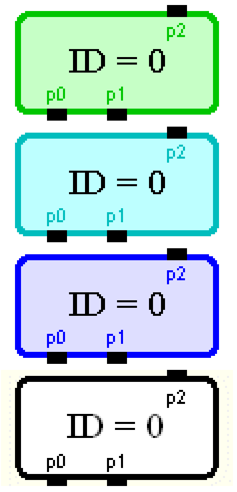
It contains the following additional control:

View
This control van be used to select the Analyzer node to view information for. By clicking on the "a", "b" or "c", the corresponding node can be hidden or shown. A red background indicates the node is shown. A white background indicates a node is hidden. In the Topology View, only one node can be shown at the same time.

7.3.2.2. *Topology picture*

The topology can only be drawn if valid topology information can be found for the cursor position. Valid information can be found if the cursor is positioned at or after a bus reset that is followed by all the PhySelfID packets that are needed to create the topology.
If the cursor is positioned before a bus-reset event, valid topology information can only be found if an earlier bus reset event plus the corresponding PhySelfIDs has been recorded.

The topology is drawn in such a way that the root (highest ID) is on the top. Each IEEE1394 node connected to the bus is displayed as a rounded rectangle. The available ports of each node are drawn as little black boxes labeled p0 and up.
The connections between these ports are drawn with colored lines.
For the optional selected node, a selection rectangle is drawn around it as can be seen in the example above, where node 1 is selected.
The Color of the node Rectangle bounding and the color of the connections indicate the maximum speed of the node and connections respectively. The used colors are:



100 Mb/s

200 Mb/s

400 Mb/s

800 Mb/s or higher

The area inside the node rectangle is colored gray if the link layer of the node is not active.
The node ID is always displayed inside the node rectangle and optionally more information will be displayed if enabled (see Select node display options above).
A node can be selected with the mouse and extra information of the selected node will be displayed right of the picture if the 'Bus and node information' button is pressed (see above).

7.3.2.3. *Extra bus and node information*

If the 'Bus and node information' button is pressed, the extra information is displayed to the right of the topology picture as can be seen in the example above.
The upper part of this extra information shows some general bus information.
See below for more information about the search for the Bus_Manager_ID register value.
The lower part of the extra information shows information on a selected node. A node can be selected by clicking on the node in the topology picture, or selecting it from the node selection box.

The extra information for the selected node consists of information extracted from the selfID(s) of the corresponding node and information extracted from the Configuration Rom of the corresponding node. The information extracted from the Configuration Rom will only be displayed if this information can be found. See below for more information about searching Configuration Rom information.
Note: The 'Root Directory' will not be displayed if there is no license key available for the version 2 software or higher. See License Manager.

Information from Configuration ROM and Bus_Manager_ID
To display the optional extra information, the information from Configuration ROM and the Bus_Manager_ID register of the CSR may be needed. The Recorder cannot read this information actively. First because the Recorder is completely passive (does not generate packets on the bus). Second because the current topology does not need to be the same as the topology valid at the cursor position of the recorder. The displayed Recorder information could be from file, in which case there is no relation at all to the current topology. Also, after the last recorded and downloaded information, there could be one or

more topology changes.
The recorder will search for recorded bus transactions that will give more information about the Configuration ROM and Bus_Manager_ID register. It searches for read transactions of the Configuration ROM and for lock transactions on the Bus_Manager_ID register. It will search, starting with the last reset event before the current cursor position until the next reset event or until the end of recorded packets. If no reset was recorded before the cursor position, it will search from the beginning of the recorded packets.

If some information cannot be found because the corresponding transactions could not be found, the corresponding information will not be displayed or a '?' will be displayed in place of the information. All information that can be found, will be displayed (if requested by the user).
Note that if a correct lock transaction on the Bus_Manager_ID register is found, but it indicates that no Bus Manager is present, the value '(none)' will be displayed. If no correct lock transaction could be found, the Bus Manager line isn't displayed at all.

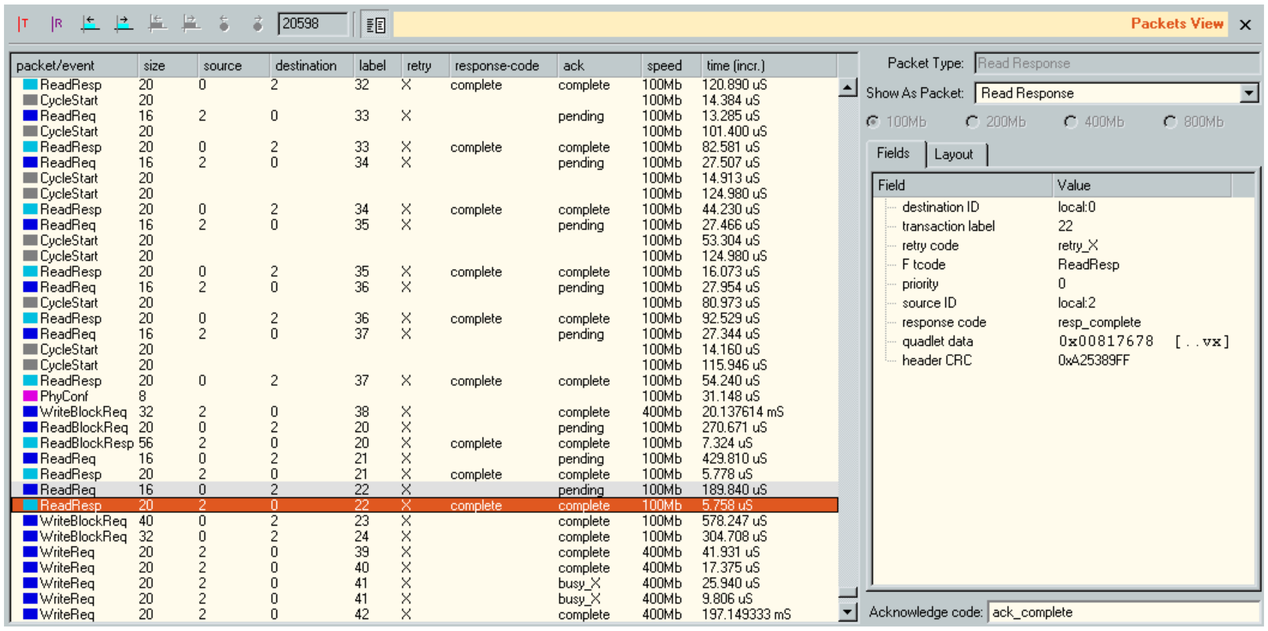
Vendor names lookup
The Vendor ID is a 24-bit value. It will be displayed, if requested, inside the node rectangle or in the extra information at the right of the topology picture.
Because a 24-bit ID is difficult to recognize, the Analyzer software will try to look up the vendor name corresponding to this 24-bit ID in an external file. If found, this name will be used inside the node rectangle (in place of the 240-bit value) and will be added to the vendor-ID value in the extra information list. The file used for this is placed in the ...\bin folder. It has the name oui.txt. Updates of this file can be found at the IEEE organization website at: http://standards.ieee.org/regauth/oui/index.shtml Here you can find free public updates of this file.
If this file is not present, only the 24-bit Vendor-ID value will be displayed.

7.4. Packet View

The Packets View is one of the possible views in the Recorder which you can use to investigate the recorded packets and events. It displays all packets and events sequentially in a table. Optionally, additional information of a selected packet can be displayed, including the packet fields and their values, the packet layout and possible warnings and errors that are detected for the packet.

7.4.1. How to use it

You can display/remove the Packets View by checking/unchecking the Packets View item in the 'View' menu of the Recorder. There is also a shortcut key to toggle the state. An example of the Packets View is dispayed below:



packet/event	size	source	destination	label	retry	response-code	ack	speed	time (incr.)
ReadResp	20	0	2	32	X	complete	complete	100Mb	120.890 uS
CycleStart	20							100Mb	14.384 uS
ReadReq	16	2	0	33	X		pending	100Mb	13.285 uS
CycleStart	20							100Mb	101.400 uS
ReadResp	20	0	2	33	X	complete	complete	100Mb	82.581 uS
ReadReq	16	2	0	34	X		pending	100Mb	27.507 uS
CycleStart	20							100Mb	14.913 uS
CycleStart	20							100Mb	124.980 uS
ReadResp	20	0	2	34	X	complete	complete	100Mb	44.230 uS
ReadReq	16	2	0	35	X		pending	100Mb	27.466 uS
CycleStart	20							100Mb	53.304 uS
CycleStart	20							100Mb	124.980 uS
ReadResp	20	0	2	35	X	complete	complete	100Mb	16.073 uS
ReadReq	16	2	0	36	X		pending	100Mb	27.954 uS
CycleStart	20							100Mb	80.973 uS
ReadResp	20	0	2	36	X	complete	complete	100Mb	92.529 uS
ReadReq	16	2	0	37	X		pending	100Mb	27.344 uS
CycleStart	20							100Mb	14.160 uS
CycleStart	20							100Mb	115.946 uS
ReadResp	20	0	2	37	X	complete	complete	100Mb	54.240 uS
PhyConf	8							100Mb	31.148 uS
WriteBlockReq	32	2	0	38	X		complete	400Mb	20.137614 mS
ReadBlockReq	20	0	2	20	X		pending	100Mb	270.671 uS
ReadBlockResp	56	2	0	20	X	complete	complete	100Mb	7.324 uS
ReadReq	16	0	2	21	X		pending	100Mb	429.810 uS
ReadResp	20	2	0	21	X	complete	complete	100Mb	5.778 uS
ReadReq	16	0	2	22	X		pending	100Mb	189.840 uS
ReadResp	20	2	0	22	X	complete	complete	100Mb	5.758 uS
WriteBlockReq	40	0	2	23	X		complete	100Mb	578.247 uS
WriteBlockReq	32	0	2	24	X		complete	100Mb	304.708 uS
WriteReq	20	2	0	39	X		complete	400Mb	41.931 uS
WriteReq	20	2	0	40	X		complete	400Mb	17.375 uS
WriteReq	20	2	0	41	X		busy_X	400Mb	25.940 uS
WriteReq	20	2	0	41	X		busy_X	400Mb	9.806 uS
WriteReq	20	2	0	42	X		complete	400Mb	197.149333 mS

Packet Type: Read Response
Show As Packet: Read Response
[✓ 100Mb] 200Mb 400Mb 800Mb

Field	Value
destination ID	local:0
transaction label	22
retry code	retry_X
F tcode	ReadResp
priority	0
source ID	local:2
response code	resp_complete
quadlet data	0x00817678 [..vx]
header CRC	0xA25389FF

Acknowledge code: ack_complete

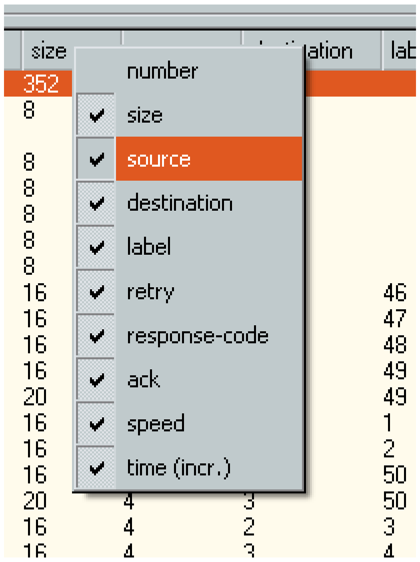
At the top of the view you see a toolbar with some buttons that enables quick selection of some event and packet types. Below the toolbar you see the packets/events table on the left and optionally the packet details on the right. Those optional details can be enabled and disabled with the right-most button on the toolbar.

Selecting a packet in the table will display the details of that packet in the (optional) details part and will also select the corresponding packet or item in all other views of the Recorder.

When clicking with the mouse left of the packet/event type in the table, the packet or event will be marked/unmarked. See Packet Marking for more information about packet and event marking

Using the Packet Search function, a large number of packets can be marked or unmarked at once, using matching criteria. See Packet Searching for more information about packet searching.

Using the 'Go to previous marked packet' and or 'Go to next marked packet' buttons, you can easily step through the marked packets.



By right-clicking on the column headers, a popup window as shown above will appear. The currently shown packets/events table columns are checked in the menu. A column is shown or hidden by clicking the corresponding menu item.

The packet/event number column shows the corresponding packet/event number. It is initially hidden and can be made visible in the packets/events table through the packet/event column menu.

7.4.2. Details

The Packets View displays all packets and events in a list and optionally the details of a selected packet. It has a toolbar of its own, a table of all the packets/events and an area to the right of the list where the details of the selected packets can be seen.

7.4.2.1. Toolbar

The Toolbar has the following indicators and buttons:

Go to trigger position
When clicking this button, the cursor is moved to the trigger time. It is only enabled if the trigger position is defined (trigger occurred during recording).

Go to next Bus Reset
When clicking this button, the next Bus Reset event will be selected. It is only enabled if one or more bus resets are present (recorded during the recording).

Go to previous packet of same type
When clicking this button, the previous packet of the same type will be selected. It will be enabled if such

a packet is present.

Go to next packet of same type
When clicking this button, the next packet of the same type will be selected. It will be enabled if such a packet is present.

Go to previous packet with error(s)
When clicking this button, the previous packet that has one or more errors will be selected. It will be enabled if such a packet is present.

Go to next packet with error(s)
When clicking this button, the next packet that has one or more errors will be selected. It will be enabled if such a packet is present.

Go to previous marked packet
When clicking this button, the previous marked packet will be selected. It will be enabled if such a packet is present.

Go to next marked packet
When clicking this button, the next marked packet will be selected. It will be enabled if such a packet is present.

Packet/Event number
In this box you can see the packet or event number. All packets and events are sequentially numbered starting from 0 (zero). The packet numbers can be used to select a range of packets to be stored in the Analyzer packet file (see the 'Save Packets' command in the 'File' menu).

Double click on this box will open '*Go to packet number ...*' dialog.

Toggle packet details on/off
With this button you can show or hide the packet details of the selected packet. In the example above, the packet-details option is enabled (shown). See below for a description of the packet details option.

7.4.2.2. Packets/Events table

In the packets/events table all recorded packets and events are sequentially listed. It has a number of columns from which one (the left most) is always visible. The number of next columns that are visible depends on the Recorder window size and the number and type of enabled views. One row (packet or event) may be selected. Details of a selected packet are optionally displayed right of the packet table (see below). If a selected packet is part of a transaction, all rows corresponding to other packets of the same transaction will have a light gray background. See the example above, where the ReadResp corresponding to the selected ReadReq has a light gray background.

If the Packet View has focus (indicated by a rectangle surrounding the highlight bar) the up-arrow and down-arrow keys can be used to step through the packet list. If the Transaction View or the Packet View has focus, the left-arrow and right-arrow keys are used to step through the packets within a transaction.

The displayed columns in the packet table are:

number
This column displays the row number. This number could be used in communication with other people looking at the same recording to point out a specific packet number.

packet/event
This column displays the packet/event type. Packets are also indicated with a colored rectangle. The same colors are used as in the Time View. Stream packets are indicated as 'stream' and have a light green color if the tag value is not 3. If the tag is 3, it is indicated as a GASP packet with a dark green color. Events are also indicated with the event character, which is also used in the Time View.
If acknowledges are recorded that do not belong to any packet, it is displayed as an 'ack only' packet without a colored rectangle.
For marked packets or events, a black dot will be displayed left in this column.

size

The size column displays the total packet size expressed in bytes. If the packet size is not a whole number of bytes, which is an error condition, the number of whole bytes is displayed plus the '+' sign. Thus a packet of 27 bits has a size of '3+'. The exact number of bits can be found in the details area, using the 'Layout' page.

source - destination - label - retry - response-code
The next 5 columns hold information of important packet fields. They are only filled with information for packets that actually include the corresponding field.
The source and destination have a special format. They are split into bus number and node number. If the bus number is 1023 (local bus), only the node number will be displayed. Otherwise the bus number and local number are displayed separated by the ':' character.
For stream packets, the destination column will display the channel number of the stream packet. Channel 31 is displayed as 'ch 31'.

ack
If a packet has a corresponding acknowledge packet, the acknowledge type is displayed in the last column.

speed
This column indicates the packet speed. Please notice that even though a physical connection at 800Mbit/s is established, it is still possible to send out packets with an effective data rate of 100Mbit/s.

destination offset
In case of Asynchronous Transaction requests this column indicates the memory address offset in the target node.

data quadlet
In case of an Asynchronous write request or read response this column shows the data quadlet embedded in the packet header.

time
This column shows the time stamp of the packet/event. By right-clicking on an item in this column it is possible to change the time reference (time equals zero point) from the first packet/event to the selected packet/event.

format
This column displays the packet format. This can be legacy, beta or unspecified.

direction
This column indicates if a packet was received or transmitted by the local node. In case of a FireStealth device, this column indicates whether the packet transferred from port A to port B or the other way round.

UTC
In case a time source like an IRIG board was used for packet time stamping, this column shows the time stamp in UTC format. Please look at the relevant Settings documentation to set this up.

7.4.2.3. Packet Details

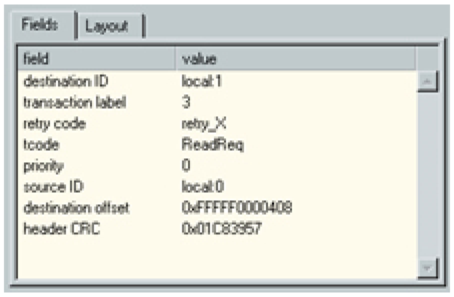
If a packet is selected, the complete line in the packet table is highlighted. If the ‘Toggle packet-details on/off’ button is pressed (on), the details of the selected packet can be seen right of the list (as in the example above). This part includes:

Packet Type
In this box the type of packet that is selected can be seen. If the packet type is unknown (e.g. a packet with erroneous header) this box will say 'unknown packet'.

Show As Packet
This box has a popup list from which you can select a packet type. The packet will be displayed in the format of the packet type selected here. Initially, it will be the same as the 'Packet Type' box. For unknown packets, it will be 'unformatted'. This feature can be used to find out for an erroneous packet what packet it most likely is ment to be, and also to display a packet in its unformatted hexadecimal quadlet form.

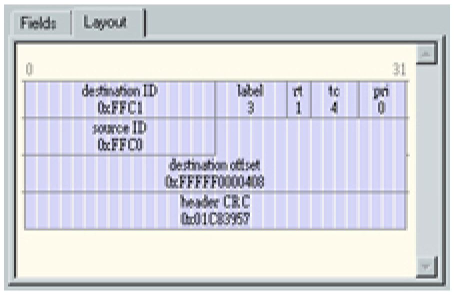
Packet speed
These selection buttons will indicate the speed of the selected packet. They are not selectable by the user, they just function as an indicator.

Fields
For primary packets the 'Fields' page shows all the fields the header of the packet is made of plus the quadlet data fields if they are present. For Phy packets it displays all the fields the phy packet is made of. And for unformatted packets, it just shows all the packet quadlets. The 'source ID' and 'destination ID' fields are displayed in a special format. The bus number and node number are displayed separately, with a ':' character in between them. If the bus number is 1023, it is substituted by the text 'local'.
Below is an example of the 'Fields' page for some read request packet.

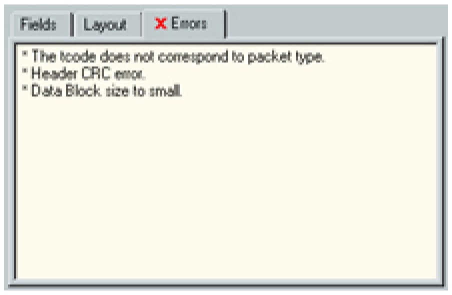


field	value
destination ID	local:1
transaction label	3
retry code	retry_X
tcode	ReadReq
priority	0
source ID	local:0
destination offset	0xFFFFF0000408
header CRC	0x01C83957

Layout
The same fields displayed in the 'Fields' page are displayed in the 'Layout' page. But now the layout of the packet is shown. Each line shows 32 bits, or one quadlet. The exact bit position of each field is shown, including the field name (mostly some abbreviation because of space limitation) and the value. Reserved fields and fields with fixed values are shown too. They have no field name, but only a value.
Below is an example of the 'Layout' page for the same packet as above.



Errors
If the selected packet has one or more errors, or if errors are detected because the selected 'Show As Packet' type does not correspond to the real packet type, an error page is added to the 'Fields' and 'Layout' pages. Selecting this page will list all detected errors.
Below, an example of the 'Errors' page is shown. In this example, the packet above is shown as a stream packet, which of course will result in errors.



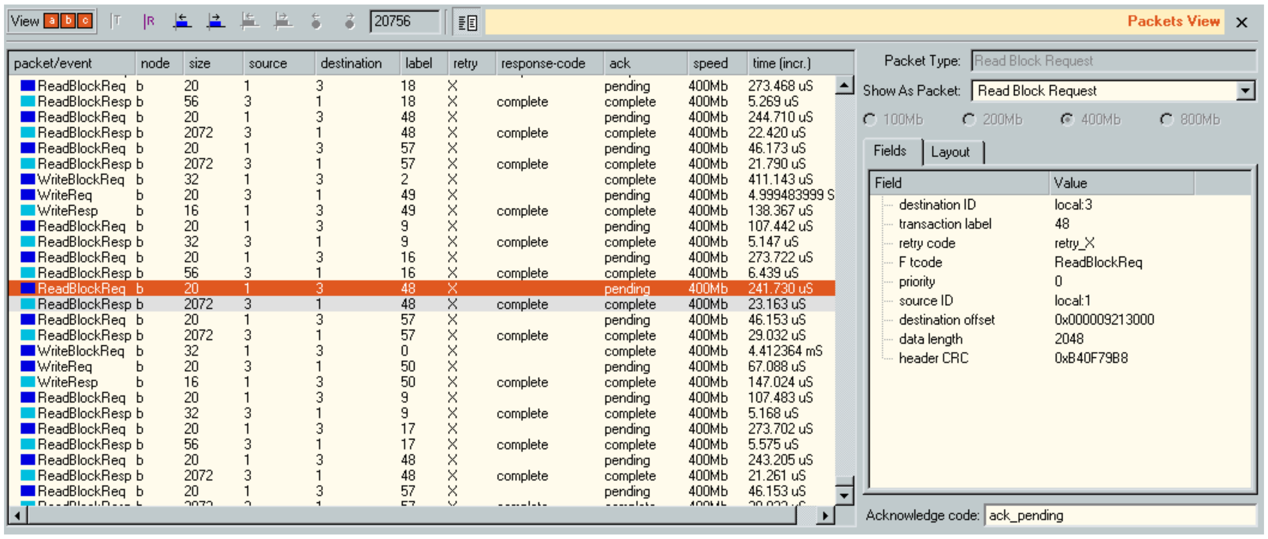
* The tcode does not correspond to packet type.
* Header CRC error.
* Data Block size to small.

If one or more fields or data is missing (like in the case of 'Data-Block size too small' error above) the 'Field' and 'Layout' pages will also reflect the error by displaying the missing parts in red. Likewise, when there is too much packet data, the parts that are too much are also displayed red in the 'Fields' and 'Layouts' pages.

Acknowledge code
If the selected packet has a corresponding acknowledge, the acknowledge code will be displayed here. If it has no acknowledge, it will say '(none)'. If the acknowledge is invalid (disallowed code) the code will be displayed too and in case of an acknowledge error (inverse-check error), the complete acknowledge byte will be displayed.

7.4.2.4. Triple Version

For triple Analyzers, the Packet View looks as follows:



packet/event	node	size	source	destination	label	retry	response-code	ack	speed	time (incr.)
ReadBlockReq	b	20	1	3	18	X		pending	400Mb	273.468 uS
ReadBlockResp	b	56	3	1	18	X	complete	complete	400Mb	5.269 uS
ReadBlockReq	b	20	1	3	48	X		pending	400Mb	244.710 uS
ReadBlockResp	b	2072	3	1	48	X	complete	complete	400Mb	22.420 uS
ReadBlockReq	b	20	1	3	57	X		pending	400Mb	46.173 uS
ReadBlockResp	b	2072	3	1	57	X	complete	complete	400Mb	21.790 uS
WriteBlockReq	b	32	1	3	2	X		complete	400Mb	411.143 uS
WriteReq	b	20	3	1	49	X		pending	400Mb	4.999483999 S
WriteResp	b	16	1	3	49	X	complete	complete	400Mb	138.367 uS
ReadBlockReq	b	20	1	3	9	X		pending	400Mb	107.442 uS
ReadBlockResp	b	32	3	1	9	X	complete	complete	400Mb	5.147 uS
ReadBlockReq	b	20	1	3	16	X		pending	400Mb	273.722 uS
ReadBlockResp	b	56	3	1	16	X	complete	complete	400Mb	6.439 uS
ReadBlockReq	b	20	1	3	48	X		pending	400Mb	241.730 uS
ReadBlockResp	b	2072	3	1	48	X	complete	complete	400Mb	23.163 uS
ReadBlockReq	b	20	1	3	57	X		pending	400Mb	46.153 uS
ReadBlockResp	b	2072	3	1	57	X	complete	complete	400Mb	29.032 uS
WriteBlockReq	b	32	1	3	0	X		complete	400Mb	4.412364 mS
WriteReq	b	20	3	1	50	X		pending	400Mb	67.088 uS
WriteResp	b	16	1	3	50	X	complete	complete	400Mb	147.024 uS
ReadBlockReq	b	20	1	3	9	X		pending	400Mb	107.483 uS
ReadBlockResp	b	32	3	1	9	X	complete	complete	400Mb	5.168 uS
ReadBlockReq	b	20	1	3	17	X		pending	400Mb	273.702 uS
ReadBlockResp	b	56	3	1	17	X	complete	complete	400Mb	5.575 uS
ReadBlockReq	b	20	1	3	48	X		pending	400Mb	243.205 uS
ReadBlockResp	b	2072	3	1	48	X	complete	complete	400Mb	21.261 uS
ReadBlockReq	b	20	1	3	57	X		pending	400Mb	46.153 uS

Packet Type: Read Block Request
Show As Packet: Read Block Request
( ) 100Mb ( ) 200Mb (•) 400Mb ( ) 800Mb

Field	Value
destination ID	local:3
transaction label	48
retry code	retry_X
F tcode	ReadBlockReq
priority	0
source ID	local:1
destination offset	0x000009213000
data length	2048
header CRC	0xB40F79B8

Acknowledge code: ack_pending

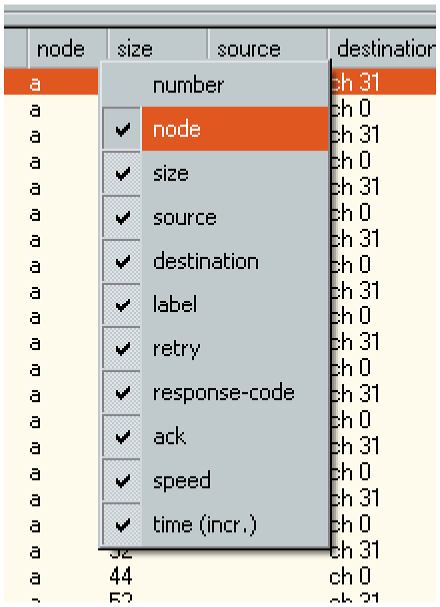
The triple version contains the following additional functionality:

Toolbar

View
This control van be used to select the Analyzer nodes to view information for. By clicking on the "a", "b" or "c", the corresponding nodes can be hidden or shown. A red background indicates a node is shown. A white background indicates a node is hidden.

Table Columns

The packets/events table columns to show are selectable through the packet/event column menu. This menu is displayed after the table header is clicked with the right mouse button. The displayed menu is shown in the figure below.



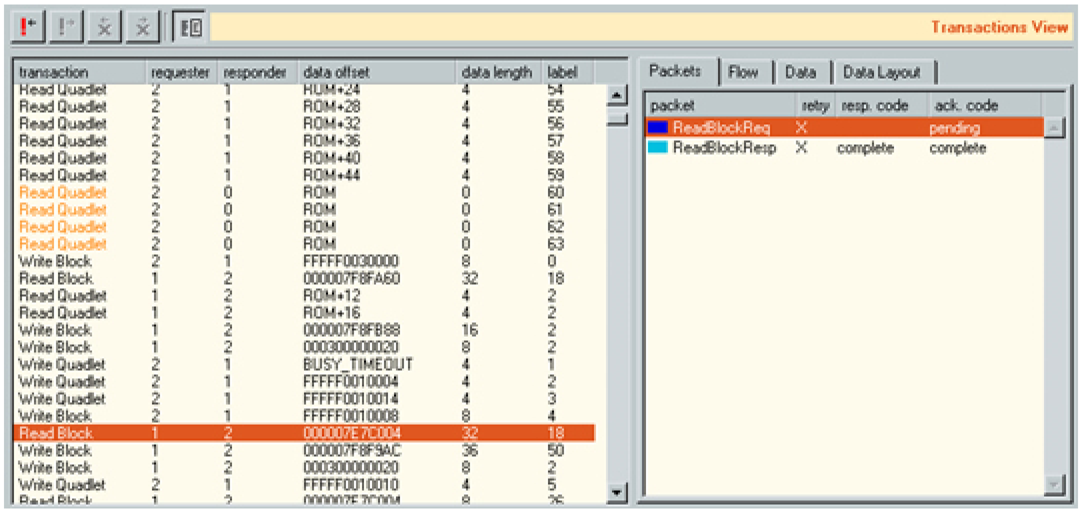
node	size	source	destination
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a			ch 31
a			ch 0
a	32		ch 31
a	44		ch 0
a	52		ch 31

7.5. Transaction View

The Transactions View is one of the possible views in the Recorder which you can use to investigate the recorded packets and events. It displays all completed and not completed transactions in a table Optionally additional information of a selected transaction can be displayed, including an overview of the involved request and response packets, a flow diagram, the transferred data and possible warnings and errors that are detected for the transaction.

7.5.1. How to use it

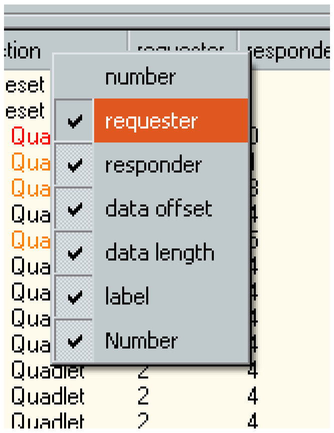
You can display/remove the Transactions View by checking/unchecking the Transactions View item in the 'View' menu of the Recorder. There is also a shortcut key to toggle the state. An example of the Transactions View is dispayed below:



transaction	requester	responder	data offset	data length	label
Read Quadlet	2	1	ROM+24	4	54
Read Quadlet	2	1	ROM+28	4	55
Read Quadlet	2	1	ROM+32	4	56
Read Quadlet	2	1	ROM+36	4	57
Read Quadlet	2	1	ROM+40	4	58
Read Quadlet	2	1	ROM+44	4	59
Read Quadlet	2	0	ROM	0	60
Read Quadlet	2	0	ROM	0	61
Read Quadlet	2	0	ROM	0	62
Read Quadlet	2	0	ROM	0	63
Write Block	2	1	FFFFF0030000	8	0
Read Block	1	2	000007F8FA60	32	18
Read Quadlet	1	2	ROM+12	4	2
Read Quadlet	1	2	ROM+16	4	2
Write Block	1	2	000007F8FB88	16	2
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	BUSY_TIMEOUT	4	1
Write Quadlet	2	1	FFFFF0010004	4	2
Write Quadlet	2	1	FFFFF0010014	4	3
Write Block	2	1	FFFFF0010008	8	4
Read Block	1	2	000007E7C004	32	18
Write Block	1	2	000007F8F9AC	36	50
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	FFFFF0010010	4	5
Read Block	1	2	000007E7C004	8	26

packet	retry	resp. code	ack. code
🟦 ReadBlockReq	X		pending
🟦 ReadBlockResp	X	complete	complete

At the top of the view you see a toolbar with some buttons that enables quick selection of transactions with warnings or errors. Below the toolbar you see the transactions table at the left and optionally the transactions details on the right. Those optional details can be enabled and disabled with the right most button on the toolbar.



The currently shown transactions table columns are checked in the menu. A column is shown or hidden by clicking the corresponding menu item.

The transaction number column shows the corresponding transaction number. It is initially hidden and can be made visible in the transaction table through the transaction column menu.

Selecting a transaction in the table will display the details of that transaction in the (optional) details part and will also select the corresponding item in all other views of the Recorder.

7.5.2. Details

The Transaction View displays all completed and not completed transactions.
When all the packet data is loaded, transactions are formed by grouping all requests and responses that make up a transaction together. All these transactions are displayed in a list. The details of a transaction can be seen to the right of the list by selecting one of the transactions.
It has a toolbar of its own, a table with all the transactions and optionally an area to the right of the table where the details of the selected transaction can be seen.

7.5.2.1. Toolbar

The Toolbar has the following buttons:

Go to previous packet with warning(s)
When clicking this button, the previous transaction that has one or more warnings will be selected. It will be enabled if such a transaction is present.

Go to next packet with warning(s)
When clicking this button, the next transaction that has one or more warnings will be selected. It will be enabled if such a transaction is present.

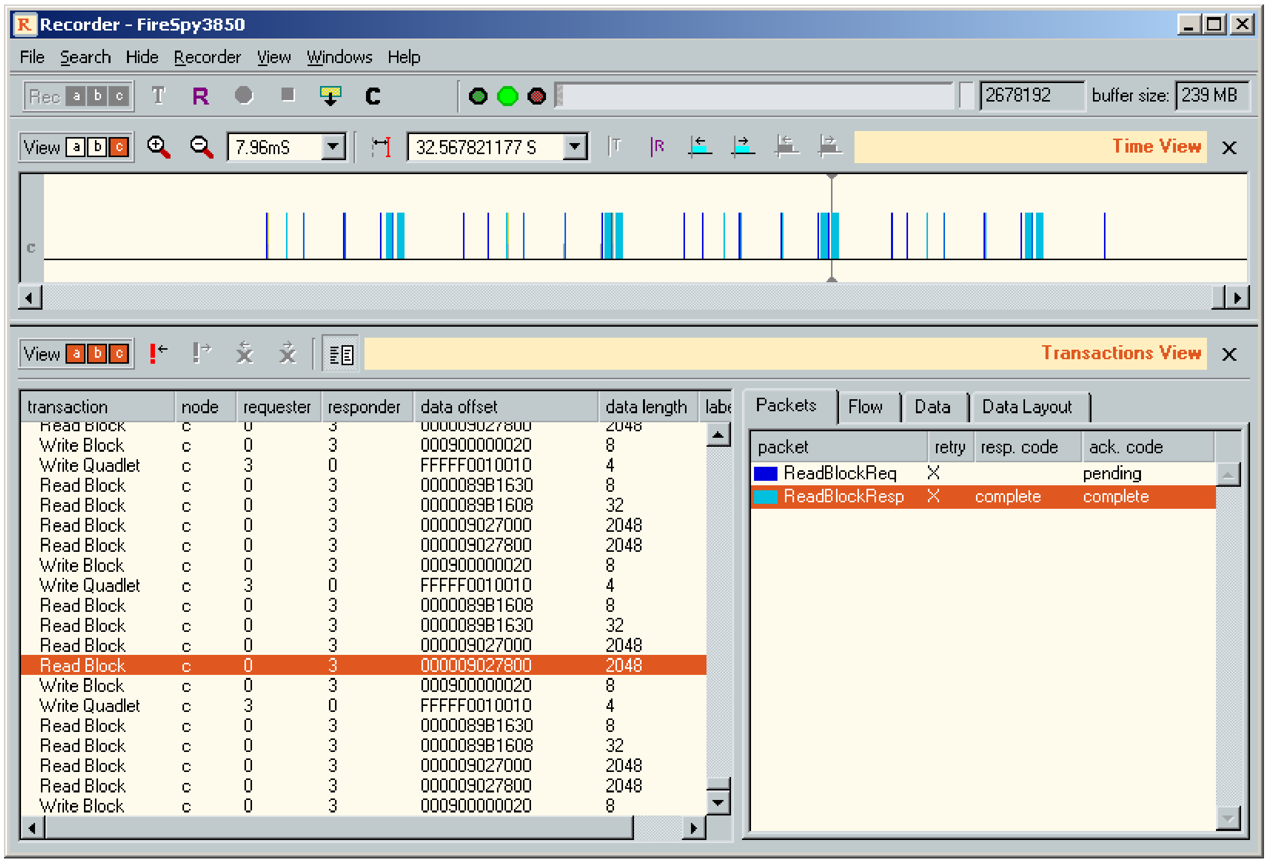
Go to previous packet with error(s)
When clicking this button, the previous transaction that has one or more errors will be selected. It will be enabled if such a transaction is present.

Go to next packet with error(s)
When clicking this button, the next transaction that has one or more errors will be selected. It will be enabled if such a transaction is present.

Toggle transaction details on/off
With this button you can show or hide the transaction details of the selected transaction. In the example above, the transaction details option is on (shown). See below for a description of the transaction details option.

Triple Analyzer

For triple Analyzers, the Transaction View looks as follows:



transaction	node	requester	responder	data offset	data length	labe
Read Block	c	0	3	000009027800	2048	
Write Block	c	0	3	000900000020	8	
Write Quadlet	c	3	0	FFFFF0010010	4	
Read Block	c	0	3	0000089B1630	8	
Read Block	c	0	3	0000089B1608	32	
Read Block	c	0	3	000009027000	2048	
Read Block	c	0	3	000009027800	2048	
Write Block	c	0	3	000900000020	8	
Write Quadlet	c	3	0	FFFFF0010010	4	
Read Block	c	0	3	0000089B1608	8	
Read Block	c	0	3	0000089B1630	32	
Read Block	c	0	3	000009027000	2048	
Read Block	c	0	3	000009027800	2048	
Write Block	c	0	3	000900000020	8	
Write Quadlet	c	3	0	FFFFF0010010	4	
Read Block	c	0	3	0000089B1630	8	
Read Block	c	0	3	0000089B1608	32	
Read Block	c	0	3	000009027000	2048	
Read Block	c	0	3	000009027800	2048	
Write Block	c	0	3	000900000020	8	

packet	retry	resp. code	ack. code
ReadBlockReq	X		pending
ReadBlockResp	X	complete	complete

It contains the following additional Toolbar control:

View
This control van be used to select the Analyzer nodes to view information for. By clicking on the "a", "b" or "c", the corresponding nodes can be hidden or shown. A red background indicates a node is shown. A white background indicates a node is hidden.

7.5.2.2. Transactions Table

The transactions table lists all complete and incomplete transactions. They are sorted by transmit time of the first packet of the transaction. The table has a number of columns from which one (the left most) is always visible. The number of next columns that are visible depend on the Recorder window size and the number and type of enabled views.

One row (transaction) can be selected and the details of that transaction can optionally be seen to the right of the transaction table (see below).
If the Transaction View has focus (indicated by a rectangle surrounding the highlight bar) the up- and down-arrow keys can be used to step through the transaction list. If the Transaction View or the Packet View has focus, the left- and right- arrow keys are used to step through the packets within a transaction.

The transaction table has the following columns:

transaction
The first column displays the transaction type. The possible transaction types are:

- Read Quadlet
- Read Block
- Write Quadlet
- Write Block
- Lock

Bus resets are also included in this list for the convenience of the user.Transactions are displayed in orange if one or more warnings are found within the transactions and red when one or more errors are found within the transaction. Details of the warnings and/or errors can be found on the optional right part of the view (see below).

requester - responder
These columns display the source ID and destination ID respectively of the request packet for this transaction. Response transaction will have a source ID equal to the responder column and destination ID equal to the requester ID.
The format of these values follows the same rule as the source and destination columns in the list of the Packet View.

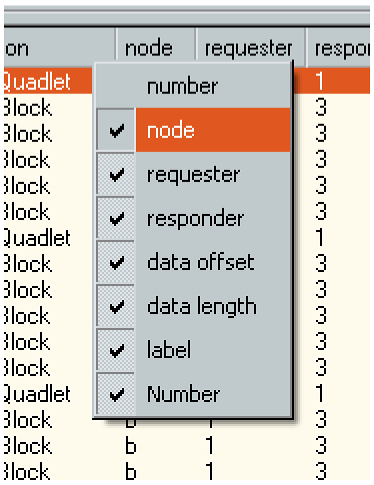
data offset
This column will hold the data offset of the associated data transfer. It equals the data-offset field of the request packet.

data length
This column will hold the data length of the associated data transfer.

label
The last column has the value of the label field for both the request and response packets.

Triple Analyzer

The transactions table columns to show are selectable through the transaction column menu. This menu is displayed after the table header is clicked with the right mouse button. The displayed menu is shown in the figure below.

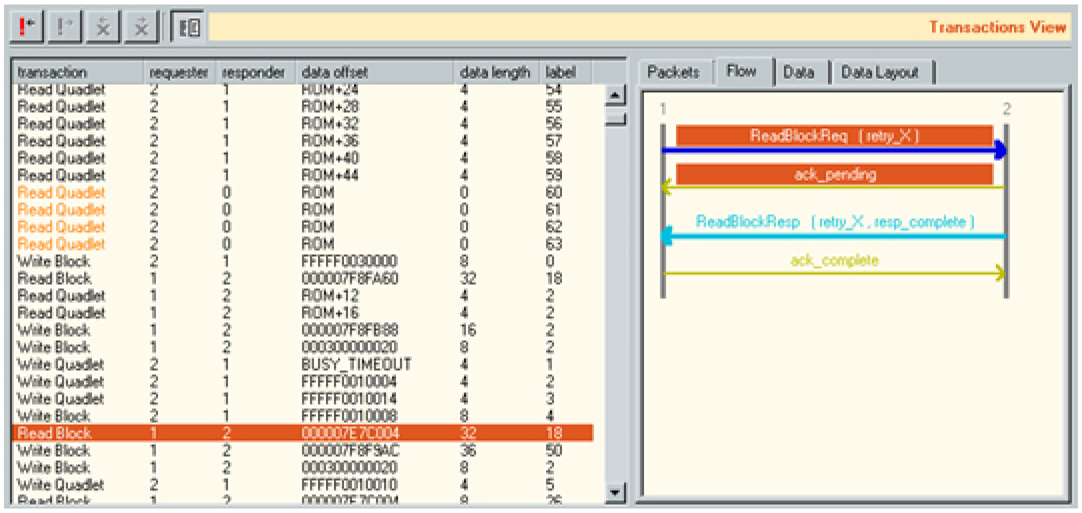


7.5.2.3. Transactions Details

Right to the transaction table, the details of the selected transaction can be seen if the ‘Toggle transaction details on/off button’ is pressed (on).
The transaction details box has 4 to 6 pages (depending of the presence of errors and or warnings):

Packets
Selecting this page will show the page with a list of all the packets that make up the transaction. In the example above, the selected transaction consists of a read request packet, followed by a read response packet.
In this packets page you also can see the retry-code and response-code (if present) of the packets and (if the packet was followed by an acknowledge packet) the acknowledge code sent.
If you select a packet in this list, it will also be selected in the other views. So you can select it and investigate the packet details in the Packet View or check its position on the time line in the Time View.
If you want to know the time between request transmission start and response transmission start for instance, simply select the request, switch the Time View to relative cursor mode, and click the response packet. The Time View will display the difference in transmission start time in its 'Cursor Time' box.
You can use the left- and right- arrow keys to step through the list of transaction packets.

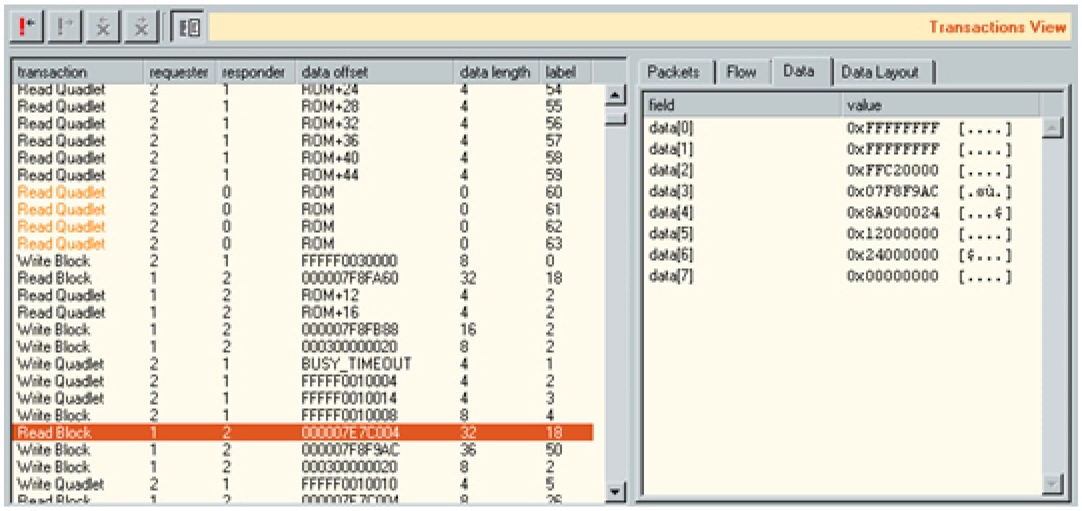
Flow
Selecting this page will show the same packets but now as a flow diagram. The requester is drawn left (vertical dark gray line) and the responder right (also vertical dark gray line). The packets transmitted between these two nodes, for the selected transaction, are drawn as arrows. The arrows point into the direction of transmission (from source to destination). Acknowledge packets are drawn as separate arrows. The color of the lines is the same as used in the Time View to easily identify the packet types. You can select a packet in the flow by clicking with your mouse on the arrow or on the labels above the arrow. For selection, both packet and associated acknowledge (if present) are a unity. If you select a packet (with acknowledge) this way, it will also be selected in the other views, as in the case of packet selection on the Packets page.
Below, the same transaction is selected as the example above, but now with the Flow page selected.



transaction	requester	responder	data offset	data length	label
Read Quadlet	2	1	ROM+24	4	54
Read Quadlet	2	1	ROM+28	4	55
Read Quadlet	2	1	ROM+32	4	56
Read Quadlet	2	1	ROM+36	4	57
Read Quadlet	2	1	ROM+40	4	58
Read Quadlet	2	1	ROM+44	4	59
Read Quadlet	2	0	ROM	0	60
Read Quadlet	2	0	ROM	0	61
Read Quadlet	2	0	ROM	0	62
Read Quadlet	2	0	ROM	0	63
Write Block	2	1	FFFFF0030000	8	0
Read Block	1	2	000007F8FA60	32	18
Read Quadlet	1	2	ROM+12	4	2
Read Quadlet	1	2	ROM+16	4	2
Write Block	1	2	000007F8FB88	16	2
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	BUSY_TIMEOUT	4	1
Write Quadlet	2	1	FFFFF0010004	4	2
Write Quadlet	2	1	FFFFF0010014	4	3
Write Block	2	1	FFFFF0010008	8	4
Read Block	1	2	000007E7C004	32	18
Write Block	1	2	000007F8F9AC	36	50
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	FFFFF0010010	4	5
Read Block	1	2	000007E7C004	8	26

As you can see above, the retry-code, response-code and acknowledge codes are also displayed in the flow graph, so that you easily can see how the transaction is build up.

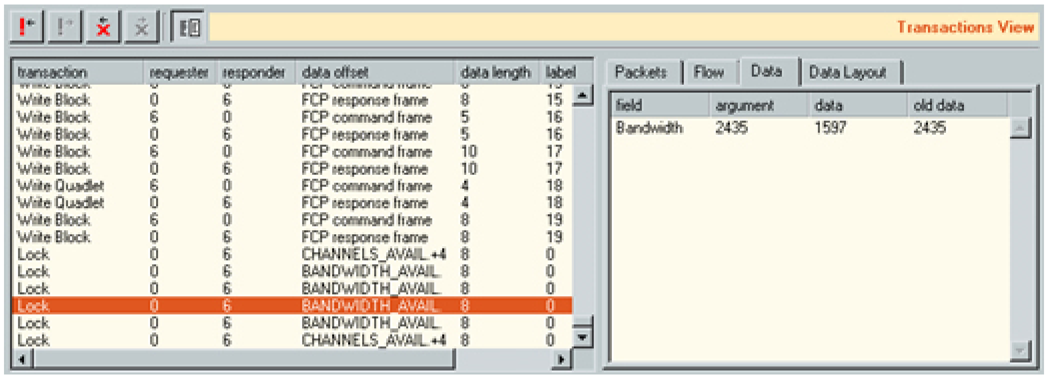
Data
In the Data page, the data transferred by the transaction is displayed. Below the data of the same transaction as selected above is displayed.



transaction	requester	responder	data offset	data length	label
Read Quadlet	2	1	ROM+24	4	54
Read Quadlet	2	1	ROM+28	4	55
Read Quadlet	2	1	ROM+32	4	56
Read Quadlet	2	1	ROM+36	4	57
Read Quadlet	2	1	ROM+40	4	58
Read Quadlet	2	1	ROM+44	4	59
Read Quadlet	2	0	ROM	0	60
Read Quadlet	2	0	ROM	0	61
Read Quadlet	2	0	ROM	0	62
Read Quadlet	2	0	ROM	0	63
Write Block	2	1	FFFFF0030000	8	0
Read Block	1	2	000007F8FA60	32	18
Read Quadlet	1	2	ROM+12	4	2
Read Quadlet	1	2	ROM+16	4	2
Write Block	1	2	000007F8FB88	16	2
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	BUSY_TIMEOUT	4	1
Write Quadlet	2	1	FFFFF0010004	4	2
Write Quadlet	2	1	FFFFF0010014	4	3
Write Block	2	1	FFFFF0010008	8	4
Read Block	1	2	000007E7C004	32	18
Write Block	1	2	000007F8F9AC	36	50
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	FFFFF0010010	4	5
Read Block	1	2	000007E7C004	8	26

field	value
data[0]	0xFFFFFFFF [....]
data[1]	0xFFFFFFFF [....]
data[2]	0xFFC20000 [....]
data[3]	0x07F8F9AC [.øù.]
data[4]	0x8A900024 [...¢]
data[5]	0x12000000 [....]
data[6]	0x24000000 [¢...]
data[7]	0x00000000 [....]

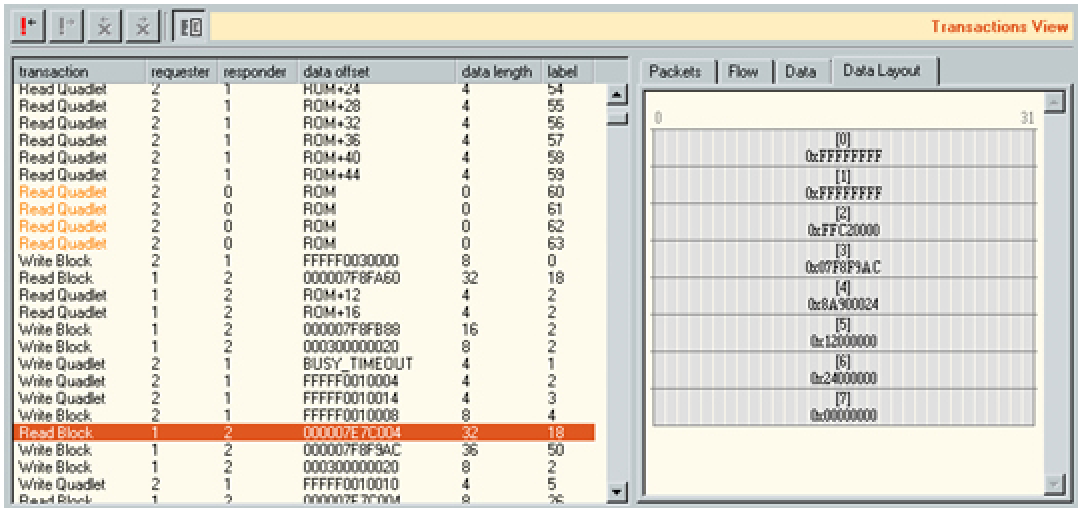
Note that for Lock transactions, the ‘value’ column will be replaced by tree columns:
argument, data and old data. The argument and data values are from the Lock Request packet and the old value is extracted from the response packet. In the example below the ‘Bandwidth Available’ register is accessed with a Lock transaction.



transaction	requester	responder	data offset	data length	label
Write Block	0	6	FCP response frame	8	15
Write Block	6	0	FCP command frame	5	16
Write Block	0	6	FCP response frame	5	16
Write Block	6	0	FCP command frame	10	17
Write Block	0	6	FCP response frame	10	17
Write Quadlet	6	0	FCP command frame	4	18
Write Quadlet	0	6	FCP response frame	4	18
Write Block	6	0	FCP command frame	8	19
Write Block	0	6	FCP response frame	8	19
Lock	0	6	CHANNELS_AVAIL+4	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	CHANNELS_AVAIL+4	8	0

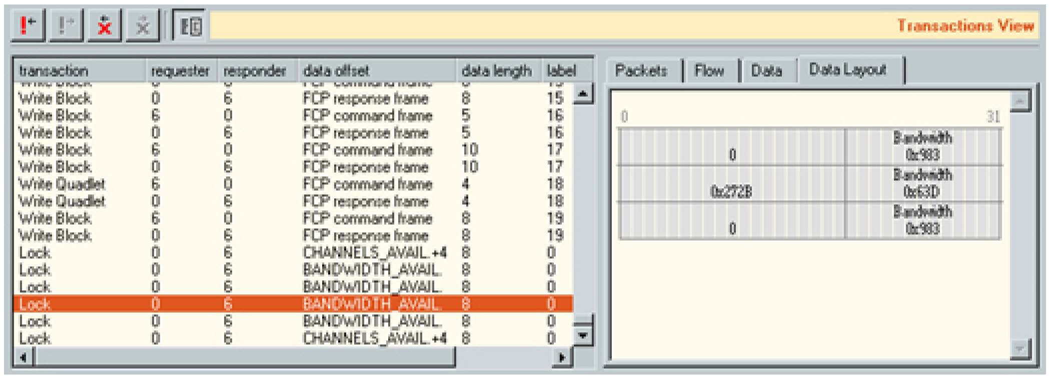
field	argument	data	old data
Bandwidth	2435	1597	2435

Data Layout
The same data as displayed in the 'Data' page, is also displayed in the 'Data Layout' page, but now shown as a packet layout. An example of the ReadBlock transaction of above is shown below.



transaction	requester	responder	data offset	data length	label
Read Quadlet	2	1	ROM+24	4	54
Read Quadlet	2	1	ROM+28	4	55
Read Quadlet	2	1	ROM+32	4	56
Read Quadlet	2	1	ROM+36	4	57
Read Quadlet	2	1	ROM+40	4	58
Read Quadlet	2	1	ROM+44	4	59
Read Quadlet	2	0	ROM	0	60
Read Quadlet	2	0	ROM	0	61
Read Quadlet	2	0	ROM	0	62
Read Quadlet	2	0	ROM	0	63
Write Block	2	1	FFFFF0030000	8	0
Read Block	1	2	000007F8FA60	32	18
Read Quadlet	1	2	ROM+12	4	2
Read Quadlet	1	2	ROM+16	4	2
Write Block	1	2	000007F8FB88	16	2
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	BUSY_TIMEOUT	4	1
Write Quadlet	2	1	FFFFF0010004	4	2
Write Quadlet	2	1	FFFFF0010014	4	3
Write Block	2	1	FFFFF0010008	8	4
Read Block	1	2	000007E7C004	32	18
Write Block	1	2	000007F8F9AC	36	50
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	FFFFF0010010	4	5
Read Block	1	2	000007E7C004	8	26

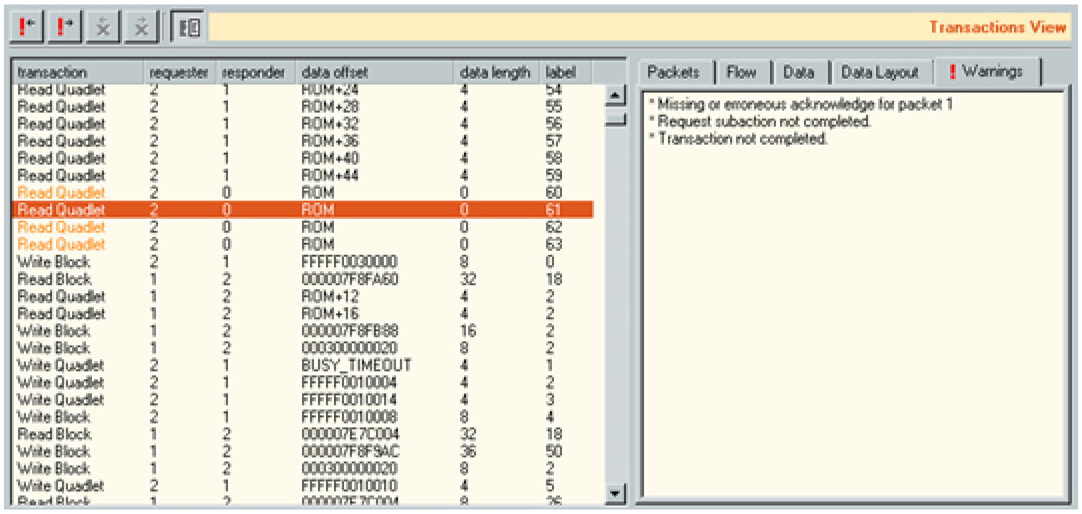
For Lock transactions the tree data values (argument, data and old data) are now displayed underneath each other. The example below shows the same Lock transaction as the lock transaction above the previous example.



transaction	requester	responder	data offset	data length	label
Write Block	0	6	FCP response frame	8	15
Write Block	6	0	FCP command frame	5	16
Write Block	0	6	FCP response frame	5	16
Write Block	6	0	FCP command frame	10	17
Write Block	0	6	FCP response frame	10	17
Write Quadlet	6	0	FCP command frame	4	18
Write Quadlet	0	6	FCP response frame	4	18
Write Block	6	0	FCP command frame	8	19
Write Block	0	6	FCP response frame	8	19
Lock	0	6	CHANNELS_AVAIL+4	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	BANDWIDTH_AVAIL	8	0
Lock	0	6	CHANNELS_AVAIL+4	8	0

Warnings - Errors
Errors and or warnings may be detected for a transaction. In that case, an 'Error' page and or 'Warning' page will be added to the Packets and Flow pages. Errors are situation where the IEEE1394 rules are violated. For instance a request that is acknowledged with an ack_pending, but where the response is never sent. Warnings are situations where the transaction could not be completed, but that are not errors. For instance a request packet that is never acknowledged (because the node doesn't exist for instance). A transaction that is not completed will always have minimal one warning ('Transaction not completed') and transactions with one or more errors, will always have minimal one warning.
You easily can search the transactions with errors and or warnings, by using the toolbar buttons.
Below is an example of a transaction with warnings. In this case the requesting packet was never been

answered with an acknowledge. So the request is not completed. Because there was no response, there is also a warning that the transaction is not completed.


Transactions View

Packets | Flow | Data | Data Layout | Warnings

transaction	requester	responder	data offset	data length	label
Read Quadlet	2	1	ROM+24	4	54
Read Quadlet	2	1	ROM+28	4	55
Read Quadlet	2	1	ROM+32	4	56
Read Quadlet	2	1	ROM+36	4	57
Read Quadlet	2	1	ROM+40	4	58
Read Quadlet	2	1	ROM+44	4	59
Read Quadlet	2	0	ROM	0	60
Read Quadlet	2	0	ROM	0	61
Read Quadlet	2	0	ROM	0	62
Read Quadlet	2	0	ROM	0	63
Write Block	2	1	FFFFF0030000	8	0
Read Block	1	2	000007F8FA60	32	18
Read Quadlet	1	2	ROM+12	4	2
Read Quadlet	1	2	ROM+16	4	2
Write Block	1	2	000007F8FB88	16	2
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	BUSY_TIMEOUT	4	1
Write Quadlet	2	1	FFFFF0010004	4	2
Write Quadlet	2	1	FFFFF0010014	4	3
Write Block	2	1	FFFFF0010008	8	4
Read Block	1	2	000007E7C004	32	18
Write Block	1	2	000007F8F9AC	36	50
Write Block	1	2	000300000020	8	2
Write Quadlet	2	1	FFFFF0010010	4	5

* Missing or erroneous acknowledge for packet 1
* Request subaction not completed.
* Transaction not completed.

7.6. Protocol View

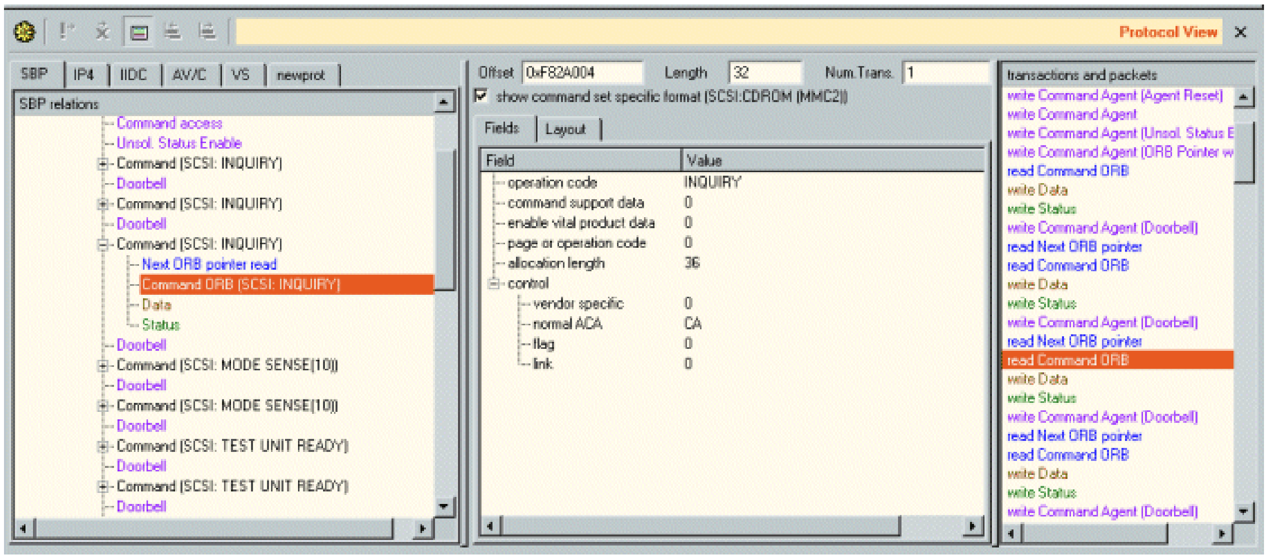
The Protocol View is part of the Recorder and will display the result of the protocol analysis.

Currently the protocol analyzer supports the following protocols:

- Serial Bus Protocol (SBP)
- Internet Protocol version 4 (IP4)
- Industrial / Instrumentation Digital Camera protocol (IIDC)
- Audio Video / Control protocol (AV/C)
- Mil1394 Protocol
- AMI-C Protocol
- Custom

Each protocol analyzer needs information to do the analysis correctly. Mostly this information can be found automatically. Using the Protocol Settings dialog, you can view the analyzer settings and change or add information manually.

The Protocol View is positioned in the lower part of the Recorder. The Protocol View comprises 3 panes and a toolbar, see figure below.


Protocol View

SBP | IP4 | IIDC | AV/C | VS | newprot

```
SBP relations
    Command access
    Unsol. Status Enable
  + Command (SCSI: INQUIRY)
    Doorbell
  + Command (SCSI: INQUIRY)
    Doorbell
  - Command (SCSI: INQUIRY)
      Next ORB pointer read
      Command ORB (SCSI: INQUIRY)
      Data
      Status
    Doorbell
  + Command (SCSI: MODE SENSE(10))
    Doorbell
  + Command (SCSI: MODE SENSE(10))
    Doorbell
  + Command (SCSI: TEST UNIT READY)
    Doorbell
  + Command (SCSI: TEST UNIT READY)
    Doorbell
```

Offset 0xF82A004 Length 32 Num.Trans. 1

☑ show command set specific format (SCSI:CDROM (MMC2))

Fields | Layout

Field	Value
operation code	INQUIRY
command support data	0
enable vital product data	0
page or operation code	0
allocation length	36
control	
vendor specific	0
normal ACA	CA
flag	0
link	0

```
transactions and packets
write Command Agent (Agent Reset)
write Command Agent
write Command Agent (Unsol. Status E
write Command Agent (ORB Pointer w
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
```

relations pane

The left pane of the Protocol View is the 'relations' pane. It displays the results of the protocol analyzer in the terminology of the supported protocols. Each supported protocol has a tab page in the 'relations' pane. The tab page shows the results found for the protocol it supports. A tab page comprises a tree, which displays the hierarchy of the found results.

transactions and packets pane
The right pane of the Protocol View is the 'transactions and packets' pane. It displays all involved transactions and packets which are part of the results found by the protocol analyzer, as well as the bus resets. The transactions, packets and bus resets are displayed in a chronological order.

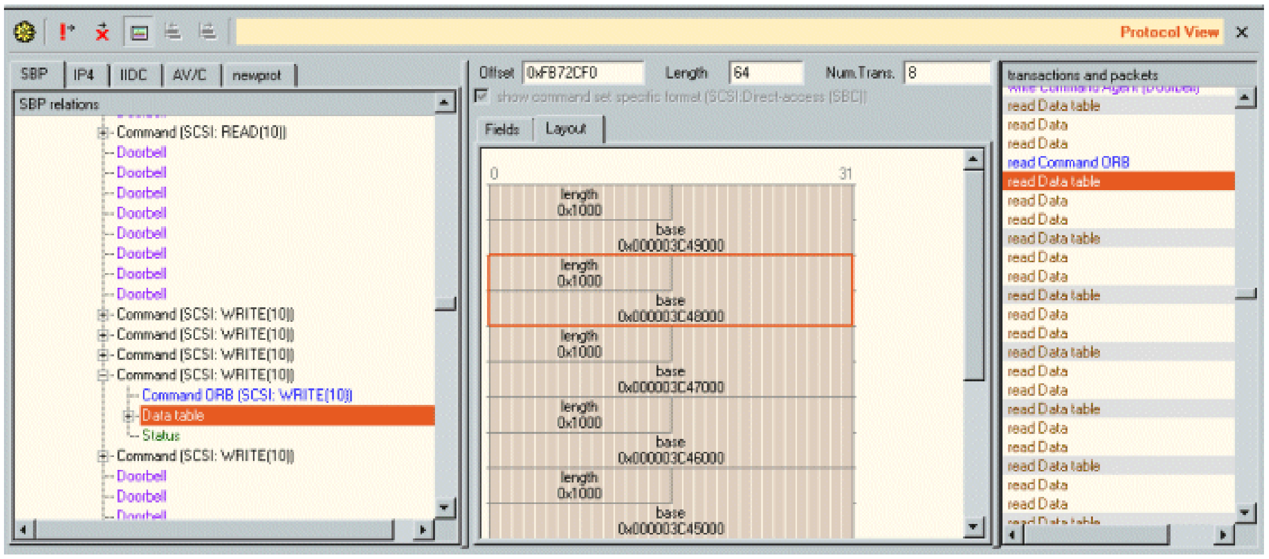
If you click an item in a tree of the 'relations' pane, the item will highlight. If one or more transactions or packets in the 'transactions and packets' pane correspond to the item, then the first of these transactions or packets will highlight. The remaining of these transactions or packets will be indicated with a gray background color, see also the figure above and below. The up-arrow and down-arrow keys can be used to step through the relation list. The left-arrow and right-arrow keys can be used to expand and collapse items in the relation list.

Conversely, if you click a transaction or packet in the 'transactions and packets' pane, the transaction or packet will highlight. The item in a tree of the 'relations' pane to which the highlighted transaction or packet corresponds will also highlight. The remaining transactions or packets, which also correspond to the highlighted item, will be indicated with a gray background color again. The up-arrow and down-arrow keys can be used to step through the transactions and packets list.

details pane
The middle pane of the Protocol View is the 'detail' pane. It displays the details of a highlighted item of a tree in the 'relations' pane. The details of a highlighted item are its fields and their values. The fields are defined by the protocol definition of the highlighted item. The values of the fields are retrieved from the corresponding transactions or packets of the highlighted item.

The 'detail' pane displays the fields and their values in 2 different forms. Each form has a separate tab page. The Fields tab page shows the fields and their values in a tree and the Layout tab page shows the fields and their values in the layout of the highlighted item.

If multiple transactions or packets correspond to the highlighted item, the protocol of the highlighted item determines how the fields and their values of the highlighted item are displayed in the Layout tab page. For example protocol SBP shows all fields and their values in one large block, see figure below.



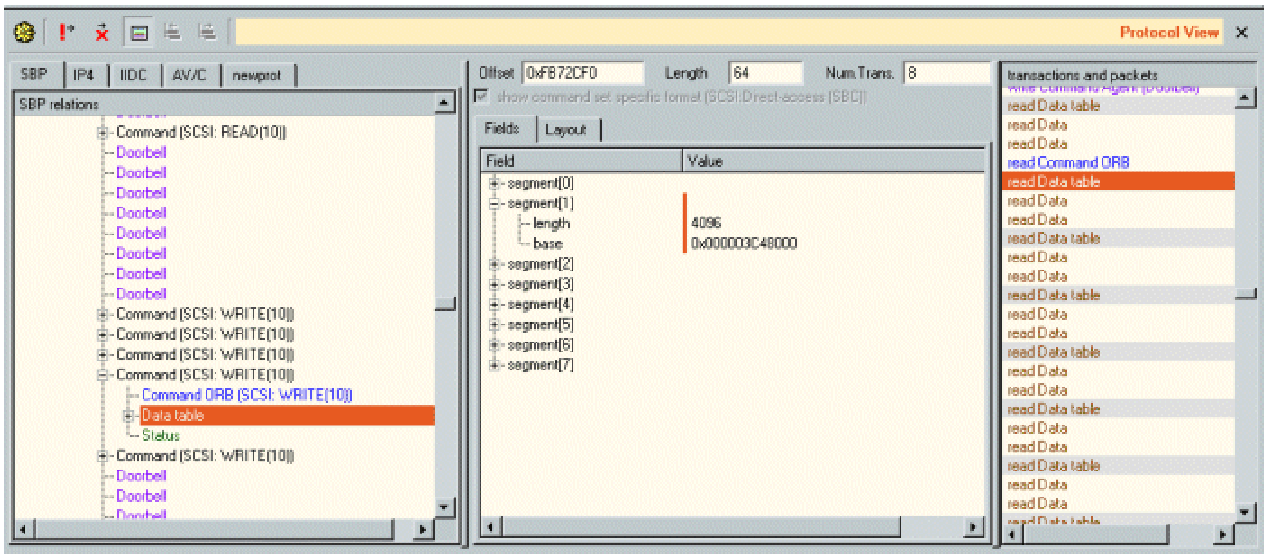
Offset 0xFB72CF0 Length 64 Num.Trans. 8

```
Command (SCSI: READ(10))
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
Command (SCSI: WRITE(10))
Command (SCSI: WRITE(10))
Command (SCSI: WRITE(10))
Command (SCSI: WRITE(10))
  Command ORB (SCSI: WRITE(10))
  Data table
  Status
Command (SCSI: WRITE(10))
  Doorbell
  Doorbell
```

```
0                                31
length 0x1000
              base 0x000003C49000
length 0x1000
              base 0x000003C48000
length 0x1000
              base 0x000003C47000
length 0x1000
              base 0x000003C46000
length 0x1000
              base 0x000003C45000
```

```
read Data table
read Data
read Data
read Command ORB
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
```

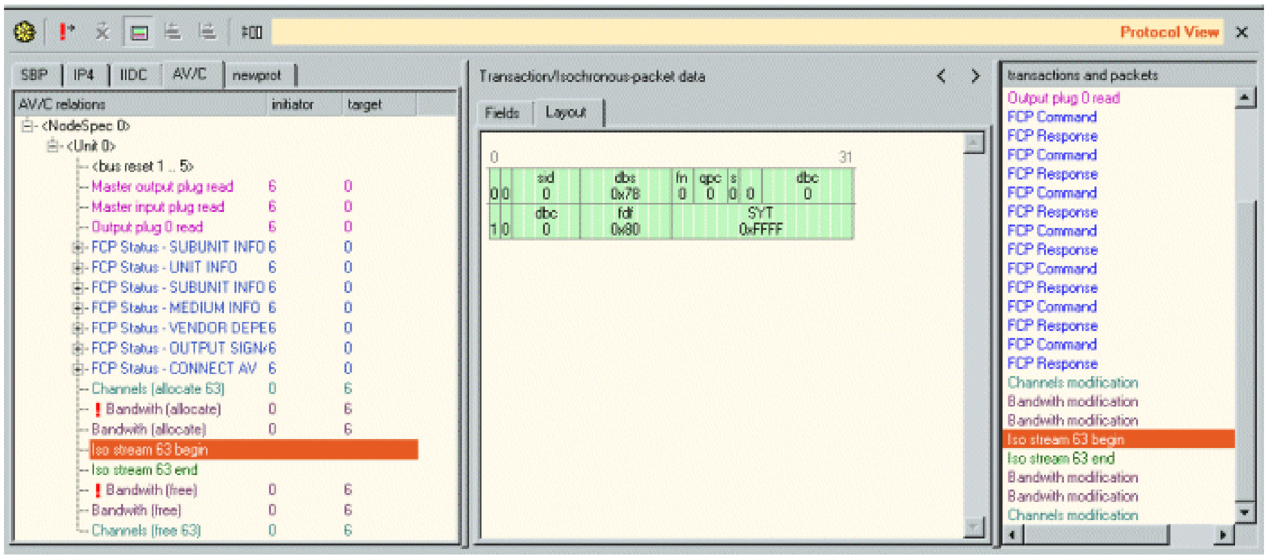
In the figure above, the red frame indicates the highlighted transaction or packet in the 'transactions and packets' pane. The Fields tab page of the 'detail' pane is shown in the figure below.



```
Protocol View

SBP | IP4 | IIDC | AV/C | newprot

SBP relations
  Command (SCSI: READ(10))
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Doorbell
  Command (SCSI: WRITE(10))
  Command (SCSI: WRITE(10))
  Command (SCSI: WRITE(10))
  Command (SCSI: WRITE(10))
    Command ORB (SCSI: WRITE(10))
    Data table
    Status
  Command (SCSI: WRITE(10))
  Doorbell
  Doorbell

Offset 0xFB72CF0    Length 64    Num.Trans. 8
show command set specific format (SCSI:Direct-access (SBC))

Fields | Layout
Field            Value
segment[0]
segment[1]
  length         4096
  base           0x000003C48000
segment[2]
segment[3]
segment[4]
segment[5]
segment[6]
segment[7]

transactions and packets
read Data table
read Data
read Data
read Command ORB
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
```

The red bar indicates the highlighted transaction in the 'transactions and packets' pane, which is in this case the second corresponding transaction of the highlighted item in the 'relations' pane.

Another example is protocol AV/C, see figure below.



```
Protocol View

SBP | IP4 | IIDC | AV/C | newprot

AV/C relations                 initiator   target
<NodeSpec 0>
  <Unit 0>
    <bus reset 1 .. 5>
    Master output plug read        6          0
    Master input plug read         6          0
    Output plug 0 read             6          0
    FCP Status - SUBUNIT INFO      6          0
    FCP Status - UNIT INFO         6          0
    FCP Status - SUBUNIT INFO      6          0
    FCP Status - MEDIUM INFO       6          0
    FCP Status - VENDOR DEPE       6          0
    FCP Status - OUTPUT SIGN       6          0
    FCP Status - CONNECT AV        6          0
    Channels (allocate 63)         0          6
    ! Bandwith (allocate)          0          6
    Bandwith (allocate)            0          6
    Iso stream 63 begin
    Iso stream 63 end
    ! Bandwith (free)              0          6
    Bandwith (free)                0          6
    Channels (free 63)             0          6

Transaction/Isochronous-packet data    <  >
Fields | Layout
0                                              31
0 0 | sid 0 | dbs 0x78 | fn 0 | qpc 0 | s 0 | 0 | dbc 0
1 0 | dbc 0 | fdf 0x80 | SYT 0xFFFF

transactions and packets
Output plug 0 read
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
Channels modification
Bandwith modification
Bandwith modification
Iso stream 63 begin
Iso stream 63 end
Bandwith modification
Bandwith modification
Channels modification
```

Protocol AV/C shows only the fields and their values of the highlighted transaction in the 'transactions and packets' pane and offers navigation buttons to walk through the corresponding transactions of the highlighted item.

IMPORTANT:
The Protocol View can only be used if a license key for software version 1 or higher is installed. For each protocol an additional license key needs to be installed. The 'relations' pane of the Protocol View doesn't show a supported protocol if no license key is installed for it. An exception is made for recorder files produced by a Analyzer with a valid license key for that protocol. For details, see License Manager.

7.6.1. How to use it

To show the Protocol View of the recorder click menu item: 'View | Protocol View'. The Protocol View will be positioned in the lower part of the Recorder.

In order to analyze the protocol for some Unit inside a node, the protocol analyzer needs information of the Unit. The information needed is protocol dependent. The protocol analyzer will scan the recorded data in the Recorder to retrieve this information automatically. This information is normally present in the

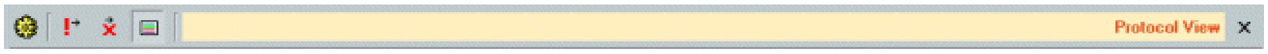
Configuration ROM of a node. The Configuration ROM of a node is often read after a bus reset. If this information can't be retrieved from the recorded data it is to be provided manually. The Protocol Settings dialog shows all information needed by each protocol. It is filled initially with the information that could be found automatically. This information can be changed or new information can be added manually.

7.6.2. Details

This section describes the following parts of the Protocol View: the toolbar, the 'relations' pane, the 'detail' pane, the 'transactions and packets' pane and the 'Protocol-Analyzer Settings'.

7.6.2.1. Toolbar

The toolbar is positioned at the top of the Protocol View.



The left side of the toolbar is divided into three parts by vertical lines: the first two parts are general and apply to all protocols; the third section contains the buttons that apply only to the selected tab page in the 'relations' pane. The general buttons will be explained below in the order they are displayed on the toolbar. The right side with the name 'Protocol View' will display a star '*' if the protocol-analyzer settings have been modified.

Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See 'Protocol Settings' for more information.

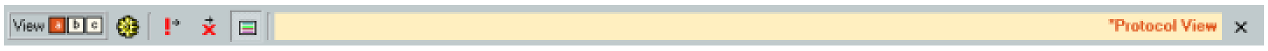
Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane at the right side of the Protocol View.

Triple Analyzer

For triple Analyzers the toolbar looks as follows:



It contains the following additional control:

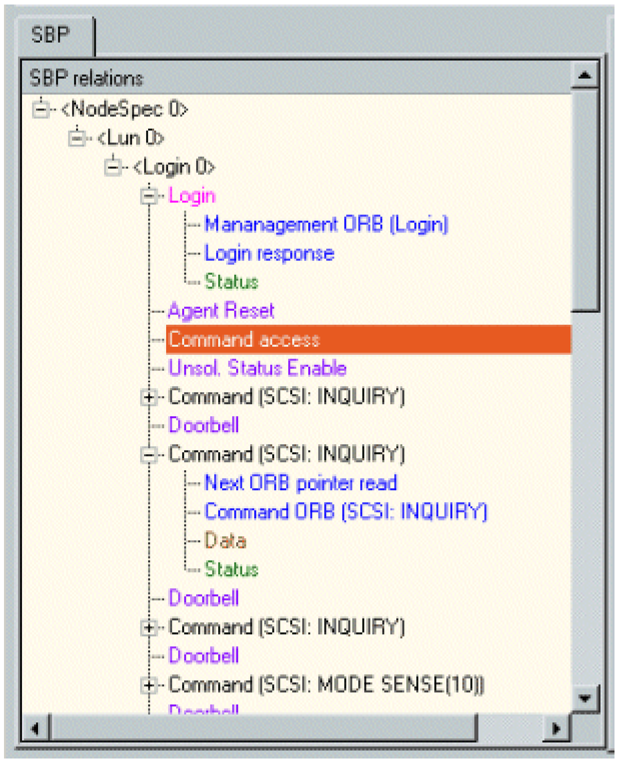
View a b c

Analyzer node select control
With this control you are able to select the protocol analysis result of each supported Analyzer node. The protocol view will only show the protocol information for the selected node.

7.6.2.2. Relations pane

The 'relations' pane shows the results of the protocol analyzer in the terminology of the supported protocols. Each supported protocol has a tab page. However, if no valid license key is installed for a supported protocol, the tab page of the supported protocol isn't displayed. The found items of a protocol are displayed in the corresponding tab page in a tree structure. The items are not necessarily recorded in the same order as displayed in this tree. However, all items inside the same item of the tree (at the same level) are listed in the same order as the order of recording the first corresponding packet of each item. To find out the order of corresponding transactions and packets, see the 'transactions and packets' pane or take a look at one of the other views of the Recorder (e.g. the Transactions View).

The example below shows the relations pane of an SBP analysis.



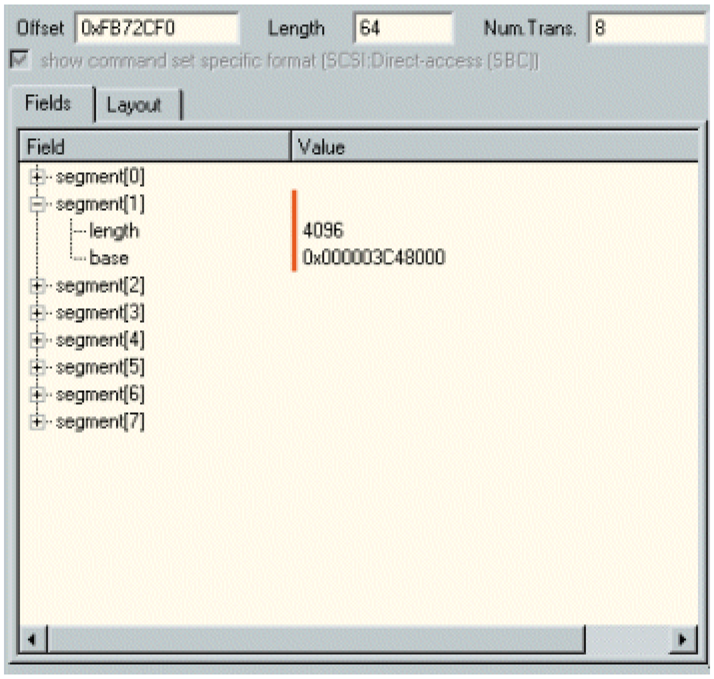
```
SBP
SBP relations
<NodeSpec 0>
    <Lun 0>
        <Login 0>
            Login
                Mananagement ORB (Login)
                Login response
                Status
            Agent Reset
            Command access
            Unsol. Status Enable
            Command (SCSI: INQUIRY)
            Doorbell
            Command (SCSI: INQUIRY)
                Next ORB pointer read
                Command ORB (SCSI: INQUIRY)
                Data
                Status
            Doorbell
            Command (SCSI: INQUIRY)
            Doorbell
            Command (SCSI: MODE SENSE(10))
            Doorbell
```

The hierarchy of the tree shows the nodes as root items, the units as child items of the nodes and the found results of the protocol analyzer as child items of the units. The terminology of the protocol determines how units and found results of the protocol analyzer are displayed in a tree, see also figure above. In a tree, the nodes are represented by their node specification ID. For more information about node specification ID, see Protocol Settings.

Click in the tree to highlight an item. If one or more transactions or packets in the 'transactions and packets' pane correspond to the highlighted item, the first of these transactions or packets will highlight and the remaining of these transactions or packets will be indicated with a gray background color.

7.6.2.3. Details pane

The 'details' pane shows the details of a highlighted item in the 'relations' pane. This part is protocol dependent, but mostly the 'details' pane will show the fields or layout of the selected item. This item may correspond to one or multiple transactions. In the example below for instance, the 'details' pane of a 'Data Table' item from a SBP analysis is shown. We can see that this 'Data Table' has a size of 64 bytes and is transferred using 8 transactions. The second transaction was selected and the data corresponding to the selected transaction is indicated with a red bar.



Offset 0xFB72CF0 Length 64 Num.Trans. 8

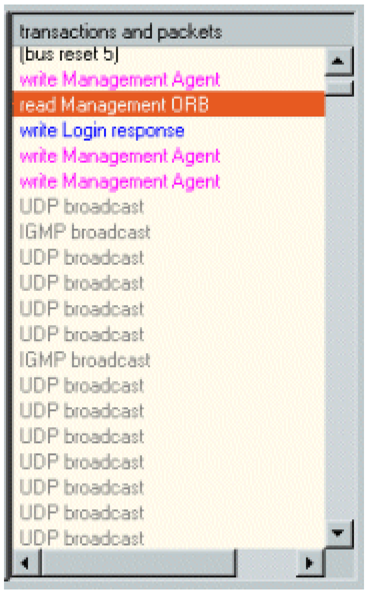
show command set specific format (SCSI:Direct-access (SBC))

Field	Value
segment[0]	
segment[1]	
length	4096
base	0x000003C48000
segment[2]	
segment[3]	
segment[4]	
segment[5]	
segment[6]	
segment[7]	

For more informattion about the 'details' pane, see the corresponding protocol chapters.

7.6.2.4. Transactions and packets pane

The 'transactions and packets' pane shows the involved transactions and packets of the results found by the protocol analyzer. It also shows all bus resets, see figure below. The 'transactions and packets' pane lists the transactions, packets and bus resets in a chronological order.



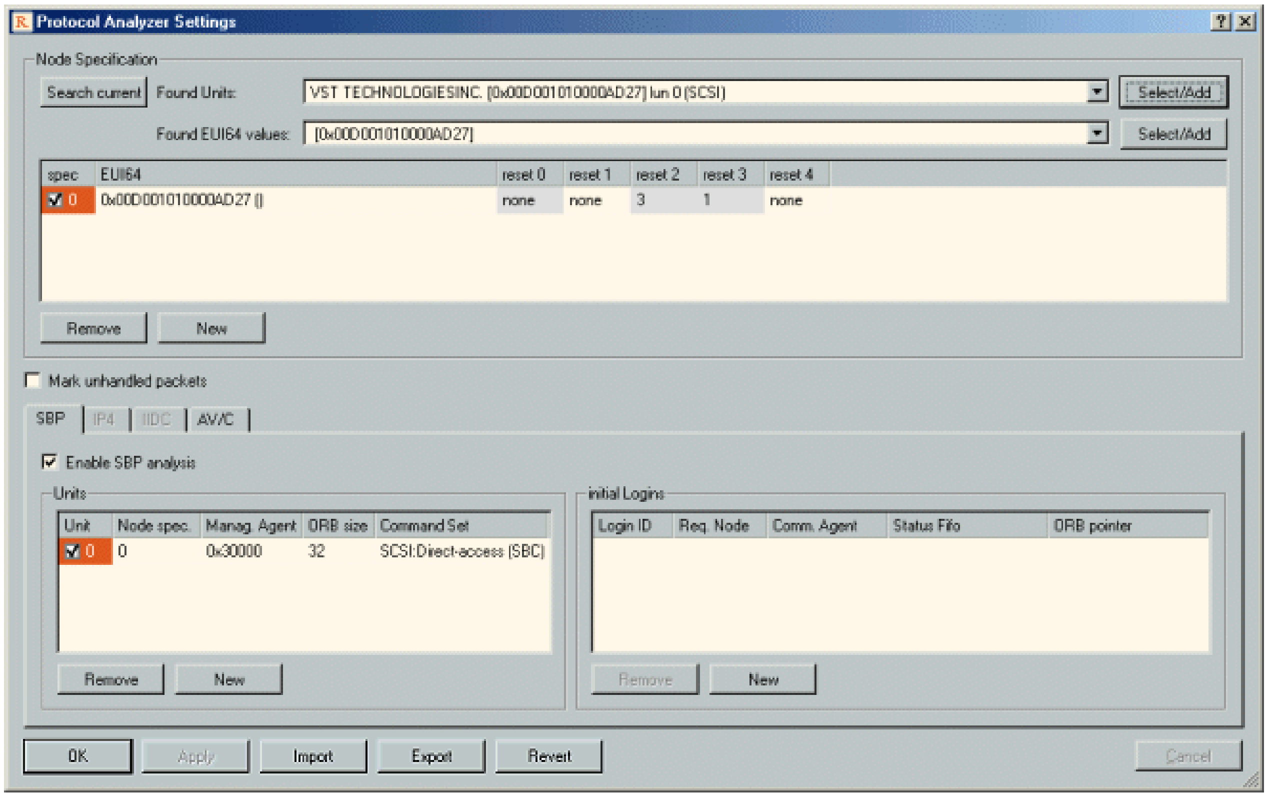
transactions and packets
[bus reset 5]
write Management Agent
read Management ORB
write Login response
write Management Agent
write Management Agent
UDP broadcast
IGMP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
IGMP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast

The 'transactions and packets' pane may list transactions and packets of multiple protocols. If this is the case all transactions and packets of the protocol of the highlighted item in the 'relations' pane are displayed in their normal text color and all transactions and packets of the other protocols are displayed with a gray text color, see also figure above.

7.6.3. Protocol Settings

The Protocol View displays the result of the protocol analysis. To do the analysis correctly, the protocol analyzer may need some information about the nodes connected to the Analyzer during the Recording. Mostly, this information can be found automatically by the analyzer software. The Protocol-Settings dialog will display this information and you may change or add information manually if needed. Manual changes in this dialog you can store using 'Export' button, or import some saved protocol settings using 'Import' button. To get original protocol settings use 'Revert' button.

Below is an example of the Protocol-Settings dialog. In this example you can see the settings for the SBP protocol.



Found Units: VST TECHNOLOGIESINC. [0x00D001010000AD27] lun 0 (SCSI)

Found EUI64 values: [0x00D001010000AD27]

spec	EUI64	reset 0	reset 1	reset 2	reset 3	reset 4
✓ 0	0x00D001010000AD27 ()	none	none	3	1	none

Mark unhandled packets

✓ Enable SBP analysis

Unit	Node spec.	Manag. Agent	ORB size	Command Set
✓ 0	0	0x30000	32	SCSI:Direct-access (SBC)

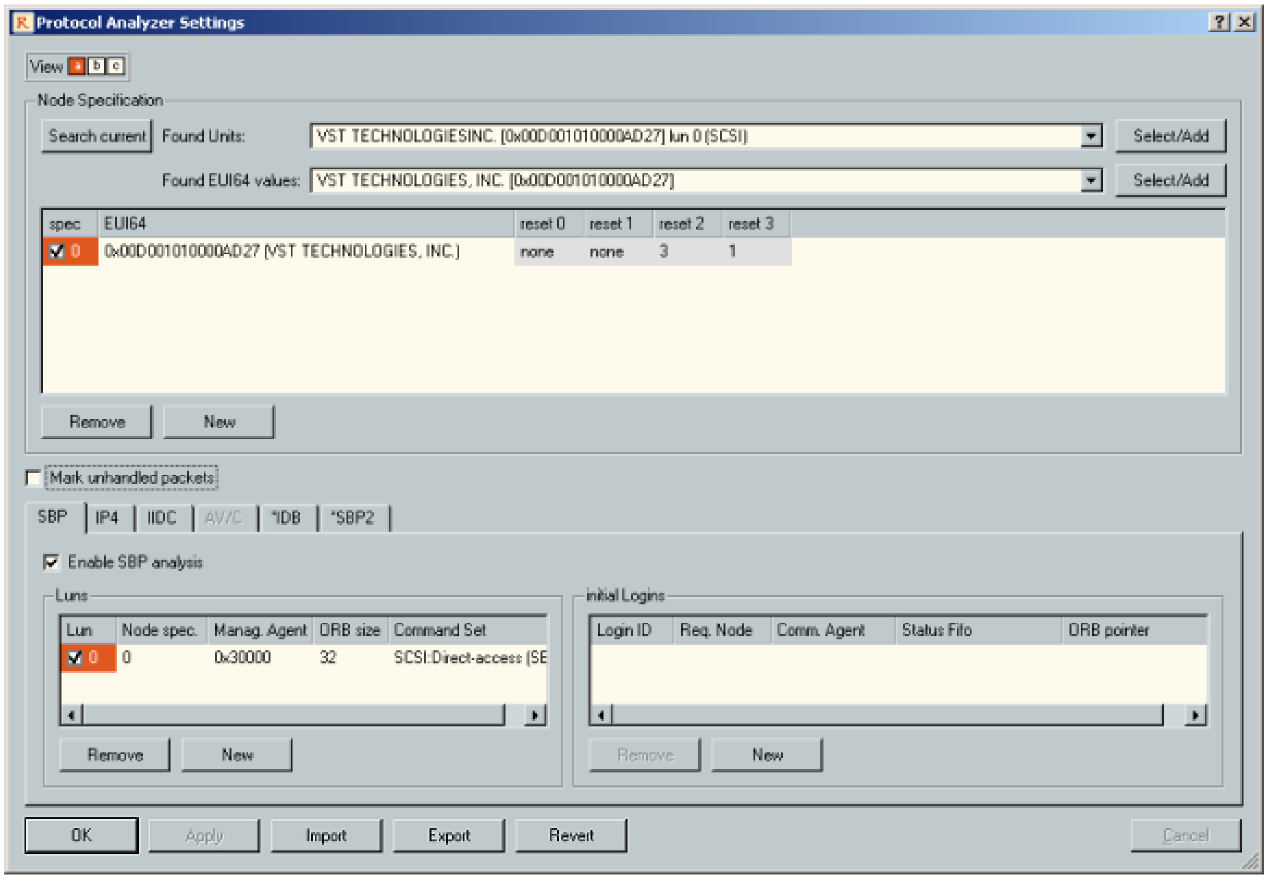
Login ID	Req. Node	Comm. Agent	Status Fifo	ORB pointer

Mostly the protocol analyzer will need information about the Units inside the nodes that comply to the protocol. For example the SBP protocol will need to know which nodes contain SBP Units and for these Units it needs to know the 'Management Agent, 'ORB size' and supported 'Command Set. All this information can be found in the Configuration ROM of the node. The analyzer will search for Configuration-ROM reads to find out this information automatically.

Node Specifications
To specify a device on the bus within the recorded data, we use a Node Specification. We can not specify a device with only one node ID, because the node ID of a device may change after a bus reset. So to fully specify a device, we will need a node ID for each reset segment.

Reset Segments
With a reset segment we refer that part of a recording where the node IDs do not change. Thus resets are the boundaries of reset segments. All recorded data, up to the first reset event (reset 1) is called reset-segment-0. All data up to the next reset (between reset 1 and reset 2) is called reset-segment-1, etc.
Below we will explain how you can make Node Specifications using reset segments.

Triple Analyzer
Below is an example of the Protocol-Settings dialog for a triple Analyzer. It shows the settings per Analyzer node. In this example you can see the settings for the SBP protocol of Analyzer node a. Settings of the other Analyzer nodes are selected with the control above the Node Specification.



Node Specification

Found Units: VST TECHNOLOGIESINC. [0x00D001010000AD27] lun 0 (SCSI)

Found EUI64 values: VST TECHNOLOGIES, INC. [0x00D001010000AD27]

spec	EUI64	reset 0	reset 1	reset 2	reset 3
☑ 0	0x00D001010000AD27 (VST TECHNOLOGIES, INC.)	none	none	3	1

☐ Mark unhandled packets

☑ Enable SBP analysis

Luns

Lun	Node spec.	Manag. Agent	ORB size	Command Set
☑ 0	0	0x30000	32	SCSI:Direct-access (SE

initial Logins

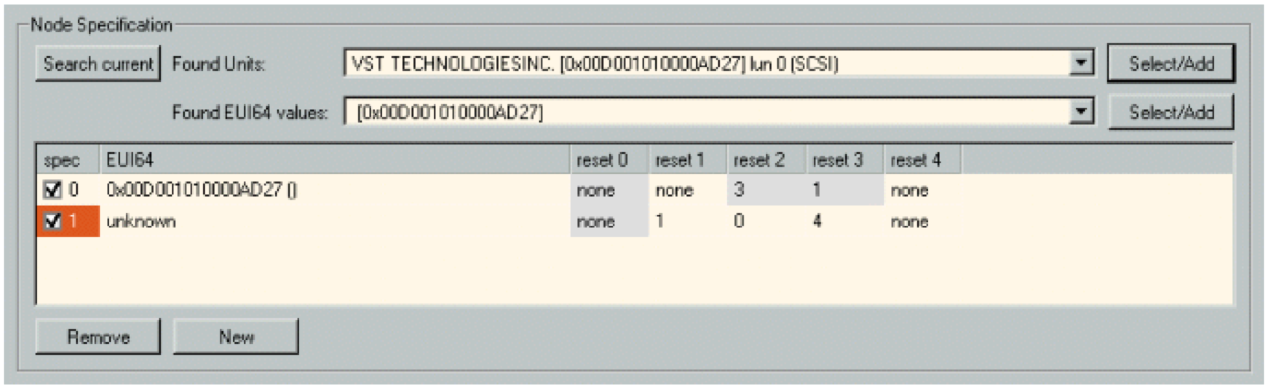
Login ID	Req. Node	Comm. Agent	Status Fifo	ORB pointer

7.6.3.1. How to use it

If you need to specify Units for some protocol manually, then you first have to make a Node Specification that specifies the node in which the Unit resides. Then you can specify the Unit and the information that is needed for the Unit. This Unit information is protocol dependent and will be described in the chapers describing the corresponding protocol.

Creating Node Specifications manually

To specify a node, you can specify the node ID for each reset segment that is present in the recorder data. You can create a new Node Specification by clicking the 'New' button below the Node-Specifications table. In the example below Node Specification 1 specifies a (manually added) device that was not present (or unknown) during reset-segment 0, and 4, and had a node ID of 1, 0 and 4 for reset segments 1, 2 and 3 respectively. Note that a gray table cell indicates that the node ID for that segment is not editable. In the case of Node Specification 1, above, the node ID for reset-segment 0 is not editable because no transactions were recorded during that segment at all, thus no node ID is needed.



Node Specification

Found Units: VST TECHNOLOGIESINC. [0x00D001010000AD27] lun 0 (SCSI)

Found EUI64 values: [0x00D001010000AD27]

spec	EUI64	reset 0	reset 1	reset 2	reset 3	reset 4
☑ 0	0x00D001010000AD27 ()	none	none	3	1	none
☑ 1	unknown	none	1	0	4	none

Adding Node Specifications semi automatically by EUI64

Another way to specify a node across reset boundaries is to use the EUI64 of that node. This value will (normally) be present in the Configuration ROM of the device. All EUI64 values for which one or more reads have been recorded are listed in the 'Found EUI64 values' box somewhere at the top of the dialog. When selecting such a value and clicking the 'Select/Add' button to the right of it, the Node Specification for that device will be added if it is not already present. The node IDs for each reset-segment will be filled in automatically as follows: If during a particular reset segment, the read of the selected EUI64 is detected, the node ID of that segment will be filled in with the corresponding node ID, and the cell is set to not editable (gray cell). For all other segments, the value 'none' will be filled in and the cell will not be editable (gray) if no transactions are recorded at all for that segment (and thus no node ID is needed). For the segments for which no corresponding EUI64 read was found you can specify a node ID manually (unless the cell is gray, indicating that no node needs to be specified).

Adding Node specifications semi automatically by Unit
The protocol-analysis software does not only search for EUI64 reads, but also for Configuration-ROM reads that read information about Units inside the nodes. Each supported protocol can recognize Units that comply to the corresponding protocol. These Units are listed in the 'Found Units' box. Selecting such a Unit in this box and clicking the 'Select/Add' button to the right of it, will add the corresponding Node Specification just as described in 'Adding Node Specifications semi automatically by EUI64'. In addition, the Unit specification is also added to the 'Units' table below the 'Node Specifications' table.

Using currently connected devices to add Units
If the 'Found Units' box does not contain the Lun you want to add, but the device containing this Unit is still connected to the Analyzer, you can click the 'Search current' button. This will add all Units of the currently connected devices to the 'Found Units' box. Clicking the 'Select/Add' button for such a Unit, will add the correct Unit information, but the correct node IDs for each reset segment can only be found automatically if the read of the corresponding EUI64 has been recorded for that segment.

Save Settings
The protocol-analysis settings are stored in the recording file. So, if after making a recording you save the recording to a file, all current protocol settings are stored in that file as well. If you open a recording file and change any of the protocol-analysis settings, a dialog box will be displayed before the currently loaded recording will be removed from memory. This allows you to cancel the action and save the modified recording.

7.6.3.2. Details

The Protocol-Settings dialog consists of two main parts. The Node-Specifications part at the top to specify nodes across resets (as explained above), and at the bottom, a protocol-specific part for protocol-specific information (including Unit information). Between these two parts, general options can be selected.

Node Specifications
The 'Node Specifications' part consists of a table with the Node Specifications and some combo boxes and buttons to help with the creation of new Node Specifications.

Combo box Found Units
The combo box shows the found units of the nodes on the bus.

Combo box Found EUI64 values
The combo box shows the found EUI64 values of the nodes on the bus.

Table Node specification
Each row in the table 'Node specification' relates to a node on the bus. The table shows in column 'spec' the defined logical ID of a node; it shows in column 'EUI64' the EUI64 value of a node and it shows in the columns 'reset 0 .. reset n' the node IDs of a node retrieved from the recorded data in Recorder. The node ID of a node may change after a bus reset. So the node ID is to be known for each reset segment. With the reset segment that part of a recording is referred to, where the node IDs do not change. Thus, bus resets are the boundaries of reset segments. All recorded data up to the first bus-reset event (reset 1) is called reset segment 0. All recorded data up to the next reset (between reset 1 and reset 2) is called reset segment 1, etc.
Using the checkbox before each Node Specification in the table you can enable or disable each node individually for analysis. If the box is not checked, all units inside this node will be excluded from analysis by any protocol.

General Options
At this moment, one general option exists:

Mark unhandled packets
This checkbox can be used to have all packets marked that aren't part of any protocol. It works like this: before the protocol analysis starts, all packets are marked that might be part of a protocol (request, response and stream packets). During analysis, any packet that is part of a protocol is unmarked. After the analysis, the Packet View or Transaction View can be used to check which transactions or packets are not included in the Protocol View.

Protocol-specific information
The dialog shows in the lower part the supported protocols. Each supported protocol has a tab page. If a protocol has no installed license key the tab page is disabled.

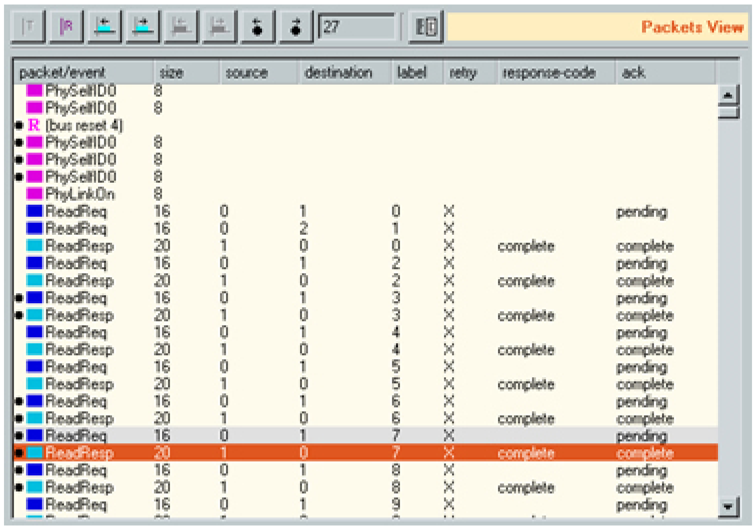
Protocol tab pages
Click on the tab to show the associated protocol tab page on top. The information on the tab pages are protocol dependent, but most will include Unit information as described above. See the seperate protocol chapters for more information.

Table Units
For most protocols, a Units table will be present. The rows in the table 'Units' relate to units, which support the corresponding protocol. The table shows in column 'Unit' the unit ID of a unit; it shows in column 'Node spec.' the Node-Specification number for this unit and in the remaining columns it shows the protocol-dependent settings. Notice that the Node Specification for this Unit is indicated with a number. This number corresponds to the 'spec' number in the Node-Specifiaction table.

See the corresponding protocol chapters for more information.

7.7. Packet Marking

Packets and events can be marked. You can mark or unmark them individually or use a packet search. A packet search can be peformed on all packets or on the selected packets so that you can narrow the search step by step. You can easily step through marked packets to investigate them and marked packets can be exported to a Analyzer packet file (*.fsp) or text file.
Marked packets will have a black dot left in the packet list of the Packet View (left to the colored packet rectangle). In the Packet-View example below a reset event and some packets are marked.



	packet/event	size	source	destination	label	retry	response-code	ack
	PhySelfID0	8						
	PhySelfID0	8						
●	R [bus reset 4]							
●	PhySelfID0	8						
●	PhySelfID0	8						
●	PhySelfID0	8						
	PhyLinkOn	8						
	ReadReq	16	0	1	0	X		pending
	ReadReq	16	0	2	1	X		
	ReadResp	20	1	0	0	X	complete	complete
	ReadReq	16	0	1	2	X		pending
	ReadResp	20	1	0	2	X	complete	complete
●	ReadReq	16	0	1	3	X		pending
●	ReadResp	20	1	0	3	X	complete	complete
	ReadReq	16	0	1	4	X		pending
	ReadResp	20	1	0	4	X	complete	complete
	ReadReq	16	0	1	5	X		pending
	ReadResp	20	1	0	5	X	complete	complete
●	ReadReq	16	0	1	6	X		pending
●	ReadResp	20	1	0	6	X	complete	complete
●	ReadReq	16	0	1	7	X		pending
●	ReadResp	20	1	0	7	X	complete	complete
●	ReadReq	16	0	1	8	X		pending
●	ReadResp	20	1	0	8	X	complete	complete
	ReadReq	16	0	1	9	X		pending

You can easily step through the marked packets using the ‘Go to previous marked packet’ and ‘Go to next marked packet’ buttons in the toolbar of the Packet View.

Packets can be marked in several ways:

- Clicking left in the packet list of the Packet View (the area of the black dots).
- Using the Mark and Unmark menu functions.

- Using the Advance Search function.

Marked packets can be exported to a Analyzer Packet file (*.fsp) or to a text file using the 'Export' function in the 'File' menu of the Recorder.

Mark / Unmark with mouse clicks
To mark/unmark a packet with a mouse click, click left to the colored packet rectangles in the Packet View (completely left in the table). A black dot (left of the colored packet rectangle) indicates marked packets. If you only need to mark or unmark a few packets, this is a quick and easy way to do this.

Mark / Unmark with menu functions
To mark/unmark a lot of packets, you can use the Mark and Unmark menu functions described above in the ‘menu’ paragraph. Using these functions you can mark or unmark all packets or a range of packets. Using the range mark/unmark function you can also choose between isochronous and asynchronous packets and some additional conditions (channel, source and destination ID).

Mark / Unmark with Packet Search function
Using the Advanced Packet Search function you will be able to mark/unmark a more specific selection of packets. See the Packet Search function for more information about packet searching.

7.8. Packet Searching

Packets and events can be marked. You can mark or unmark them individually or use a Packet Search dialog. You can mark or unmark packets by specifying search criteria, such as packet type, its acknowledge type, packet speed and values of specific header or data fields. A packet search can be peformed on all packets or on the marked packets so that you can narrow the search step by step. You can easily step through marked packets to investigate them and marked packets can be exported to a Analyzer packet file (*.fsp) or text file.

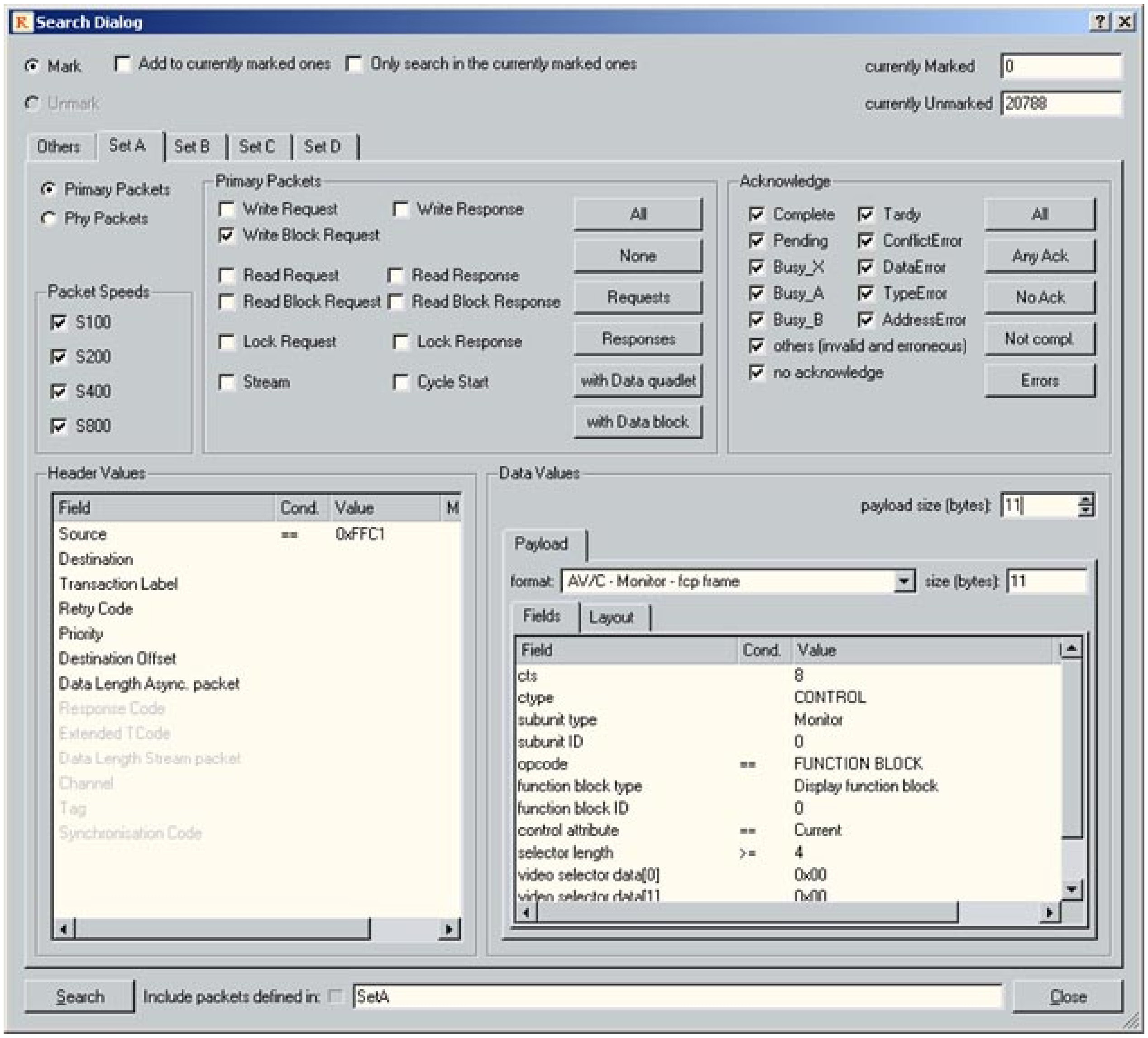
Marked packets will have a black dot left in the packet list of the Packet View (left to the colored packet rectangle). See Packet Marking for more information about packet marking.

7.8.1. How to use it

When selecting the Advanced Search in the Search menu of the Recorder window, the Search dialog will be shown. An example is shown below. Here you can specify a selection of packets that have to be marked or unmarked. The selection consists of a combination of sets. A set consists of all packets that satisfy a number of selection conditions. The conditions on which packets in a set are selected are: Packet Type, Acknowledge type, Packet speed, specific field values of the Packet Header and specific field values of the Packet Data, or a combination of these.

You can choose if the selected packets need to be marked or unmarked. When marking, you can choose if the current marked packets need to be unmarked, and if you only want to search in the currently marked ones (see below for more details).

Apart from selecting packets on specific conditions, some packet types (like phy packets) and events (like bus resets) may be included in the selection directly.
Below we will describe the different parts of the Search Dialog in more detail.



Search Dialog

[✓ Mark] Unmark   [ ] Add to currently marked ones   [ ] Only search in the currently marked ones

currently Marked: 0
currently Unmarked: 20788

Others | Set A | Set B | Set C | Set D

[✓ Primary Packets] Phy Packets

Packet Speeds
- [✓] S100
- [✓] S200
- [✓] S400
- [✓] S800

Primary Packets
- [ ] Write Request
- [ ] Write Response
- [✓] Write Block Request
- [ ] Read Request
- [ ] Read Response
- [ ] Read Block Request
- [ ] Read Block Response
- [ ] Lock Request
- [ ] Lock Response
- [ ] Stream
- [ ] Cycle Start

Acknowledge
- [✓] Complete
- [✓] Tardy
- [✓] Pending
- [✓] ConflictError
- [✓] Busy_X
- [✓] DataError
- [✓] Busy_A
- [✓] TypeError
- [✓] Busy_B
- [✓] AddressError
- [✓] others (invalid and erroneous)
- [✓] no acknowledge

Header Values

Field	Cond.	Value	M
Source	==	0xFFC1	
Destination			
Transaction Label			
Retry Code			
Priority			
Destination Offset			
Data Length Async. packet			
Response Code			
Extended TCode			
Data Length Stream packet			
Channel			
Tag			
Synchronisation Code			

Data Values

payload size (bytes): 11

Payload

format: AV/C - Monitor - fcp frame   size (bytes): 11

Fields | Layout

Field	Cond.	Value
cts		8
ctype		CONTROL
subunit type		Monitor
subunit ID		0
opcode	==	FUNCTION BLOCK
function block type		Display function block
function block ID		0
control attribute	==	Current
selector length	>=	4
video selector data[0]		0x00
video selector data[1]		0x00

Include packets defined in: [ ] SetA

7.8.2. Details

Below, the different parts of the Search dialog are described in more detail.

7.8.2.1. Mark / Unmark options

At the top of the dialog you can choose if you want to mark or unmark packets and set some other options.

Mark/Unmark radio buttons
With these radio buttons you choose between marking and unmarking packets and events.

Add to currently marked ones
When marking packets, and this checkbox is checked, the selected packets will be marked and the currently marked packets will stay marked too. Otherwise, the currently marked packets will be unmarked before the selected packets are marked.

Only search in the currently marked ones
When marking packets, and this checkbox is checked, only the currently marked packets will be checked. Unmarked packets are left unmarked.
When unmarking packets, the search is always done only on packets that are currently marked.

7.8.2.2. Search result

At the top-right of the dialog you will find an indication of the number of currently marked packets and the number of currently unmarked packets.
When the search is performed (Search button clicked), new packets may be marked or marked packets may be unmarked (depending on the chosen options). These changes will be reflected in the current number of marked and unmarked packets.

Note that you can easily examine the marked packets using the buttons to get to the previous and next marked packet in the toolbar of the Packet View and using the menu functions in the 'Search' menu of the Recorder window to go to the first or last marked packet.

7.8.2.3. Set selection, Search button and Close button

At the bottom of the dialog you will find the Search button, Close button and the Include packets defined in: box.

Search
When clicking the Search button, the search is performed. Packets and events may be marked or unmarked and the number of currently marked and currently unmarked packets will reflect the result (top-right of dialog).

Close
When clicking the Close button, the dialog will quit without doing the search. No packets will be marked or unmarked and the number of currently marked and currently unmarked packets will not change.

Include packets defined in:
Here you can type in a set name or any boolean combination of the four sets. All packets that are included in the search and that fit the Set or combination of Sets, will be marked or unmarked (depending on the chosen option). In the example above, all packets that fit the description of SetA will be marked, others will be unmarked.
The set names that can be used are 'seta', 'setb', 'setc' and 'setc'. The names are not case sensitive, thus 'SetA' is valid too. You can also use just 'a', 'b', 'c' or 'd' (or 'A', 'B' etc.) for the set names. The Boolean operators are 'not', 'and' and 'or'. But they may be substituted by the characters '!', '&' and '|' respectively. Parenthesis may be used to group the Boolean operations.
Some examples of valid expressions are:

```
setc
NOT SETB
(seta or setb) and not setd
!SetA & (SetC | SetB)
a
C or D
```

Note that a red cross will be displayed before the boolean expression when there is an error in the expression.

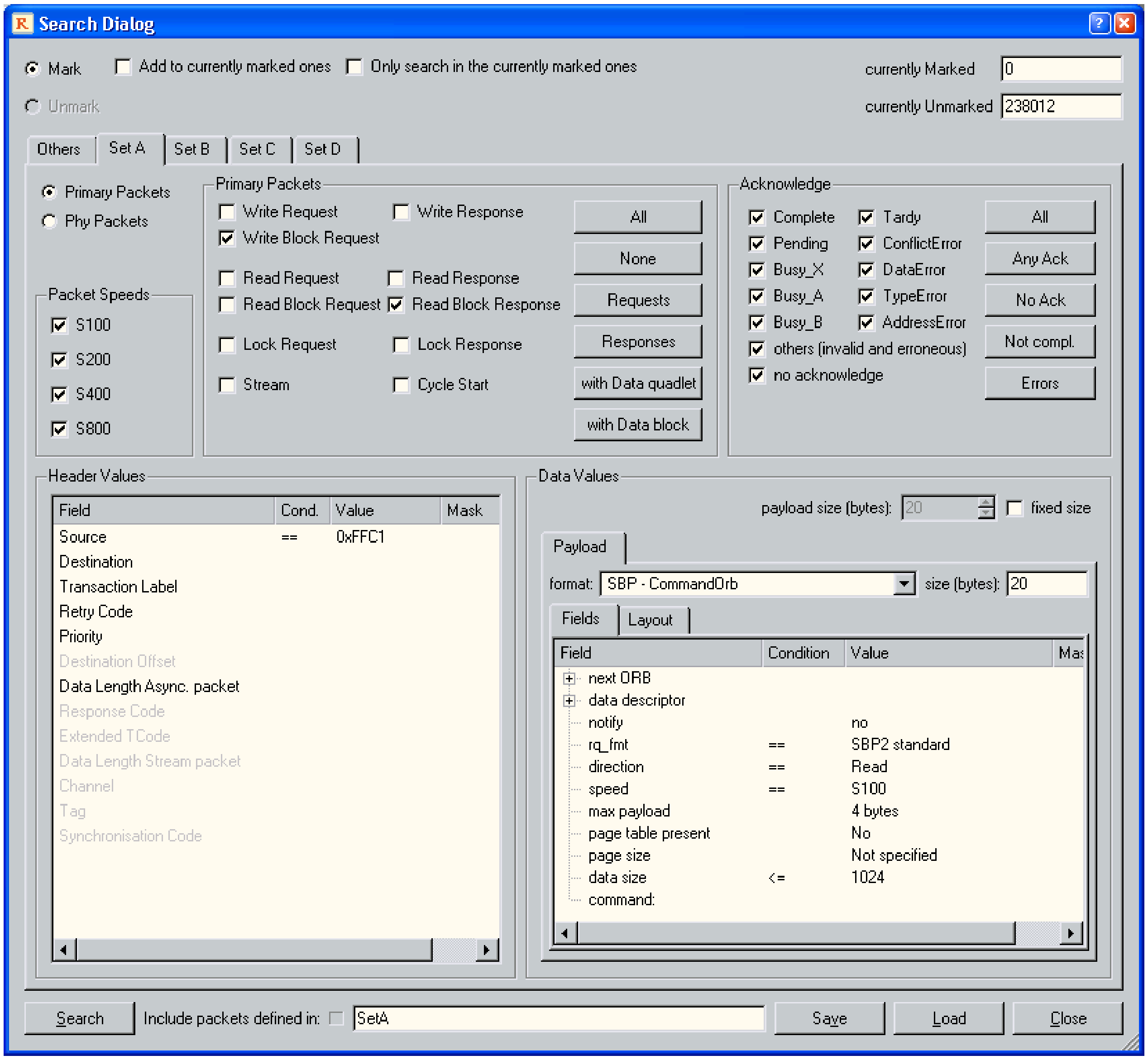
7.8.2.4. Packet Sets

Up to four Sets (SetA, SetB, SetC and SetD) can be used for the packet search. A Set defines a subset of all asynchronous and isochronous packets by specifying packet types, packet speeds, packet acknowledges, and optionally value conditions for one or more header and or data fields.
A packet is said to 'fit' the Set if all specified conditions are met. Thus, when the packet is of the specified type, and the packet is of the specified speed, and the packet is acknowledged by the specified acknowledge, and the optional header-field and data-field conditions are met, then the packet 'fits' the Set.

A Set page has the following parts:

- Primary Packets or Phy Packets selection
- Primary Packets box to select primary types
- Packet Speeds box to select which packet speeds are included
- Acknowledge box to select which acknowledge should follow the packet
- Header Values box to specify optional header field conditions

- Data Values box to specify optional data field conditions



Mark [✓] · Unmark
Add to currently marked ones [ ] · Only search in the currently marked ones [ ]
currently Marked: 0
currently Unmarked: 238012

Primary Packets [✓] · Phy Packets

Packet Speeds: S100 [✓] · S200 [✓] · S400 [✓] · S800 [✓]

Primary Packets:
Write Request [ ] · Write Response [ ]
Write Block Request [✓]
Read Request [ ] · Read Response [ ]
Read Block Request [ ] · Read Block Response [✓]
Lock Request [ ] · Lock Response [ ]
Stream [ ] · Cycle Start [ ]

Acknowledge:
Complete [✓] · Tardy [✓]
Pending [✓] · ConflictError [✓]
Busy_X [✓] · DataError [✓]
Busy_A [✓] · TypeError [✓]
Busy_B [✓] · AddressError [✓]
others (invalid and erroneous) [✓]
no acknowledge [✓]

Header Values

Field	Cond.	Value	Mask
Source	==	0xFFC1	
Destination			
Transaction Label			
Retry Code			
Priority			
Destination Offset			
Data Length Async. packet			
Response Code			
Extended TCode			
Data Length Stream packet			
Channel			
Tag			
Synchronisation Code			

Data Values
payload size (bytes): 20 · fixed size [ ]
format: SBP - CommandOrb · size (bytes): 20

Field	Condition	Value	Mas
next ORB			
data descriptor			
notify		no	
rq_fmt	==	SBP2 standard	
direction	==	Read	
speed	==	S100	
max payload		4 bytes	
page table present		No	
page size		Not specified	
data size	<=	1024	
command:			

Include packets defined in: [ ] SetA

Primary Packets or Phy Packets
Here you can select whether this set specifies primary packets or phy packets. Primary packets are Request packets, Response packets, Cycle Start packets or Stream (isochronous or asynchronous) packets.
If you select Phy Packets, then you can specify optional phy-packet field-value conditions using the Data-Values box. The Header-Values box will be disabled for the Phy Packets.
If you select Primary Packets, then you can select packet types in the Primary-Packets box. You can specify optional header-value conditions in the Header-Values box and optional data-value conditions in the Data-Values box.

Packet Speeds
Here you can select which speed a packet should have to fit the Set. If no speed is selected at all, no packet will fit the Set at all. If you select all speeds, the speed of a packet doesn't matter. In the example above, packets of speed 100Mb/s or 200Mb/s do not fit SetA.

Primary Packets
Here you can specify which primary packet types may fit the Set. All types for which the corresponding checkbox is not checked, do not fit the Set. So in the example above, only ReadBlockResponse packets may fit SetA.
If one or more types are selected, then only those lines in the Header-Values table will be enabled for which the corresponding field is present in all the selected types. If you select, for instance, only request

types, then the Destination-Offset field will be enabled. you can put a condition on this field. But if you also select a Response, this field will be disabled because it is not present in a Response packet.

There are also some buttons to quickly select multiple types at once:

All
Clicking this button will select all types.

None
Clicking this button will deselect all types.

Requests
Clicking this button will select all request types.

Responses
Clicking this button will select all response types.

With Data quadlet
Clicking this button will select all packet types, for which a data quadlet is present in the header.

With Data block
Clicking this button will select all packet types, for which a data block may follow the header.

Acknowledge
Here you can select all different acknowledge types, including invalid and erroneous acknowledge or packets without acknowledge. A packet will only fit the set if it has one of the specified acknowledge types. By default they are all checked, which in fact means that there is no condition on acknowledge type.
When the no acknowledge checkbox is checked, the packet does fit the set when it is not acknowledged by any acknowledge packet (any 8-bit packet).

When the others (invalid and erroneous) checkbox is checked, the packet fits the set when an invalid or erroneous acknowledge is detected for the packet. An erroneous acknowledge is an acknowledge packet (8-bit packet) for which the inverse check fails. An invalid acknowledge is an acknowledge for which the 4-bit invert check is ok, but with an invalid ack type.

There are also some buttons to quickly select multiple types at once:

All
Clicking this button will check all checkboxes, which in fact means that there is no condition on the acknowledge.

Any Ack
Clicking this button will check all checkboxes except the no acknowledge checkbox. This means that the packet will fit the set if there is an acknowledge. It doesn't matter if it is an invalid acknowledge or even erroneous one, as long as there is one.

No Ack
Clicking this button will check only the no acknowledge checkbox. This means that a packet does fit the set when it does not have an acknowledge.

Not Compl.
Clicking this button will select all acknowledge types indicating that a request is not completed yet. The selected types are the Pending, Tardy and the three Busy types.

Errors
Clicking this button will select all packet types indicating an error and will select the others (invalid and erroneous> checkbox.

Header Values
Here you can specify additional header-field conditions. As mentioned above, only those fields will be enabled that are present in all packet types selected in the Primary-Packets box. For each enabled field, a

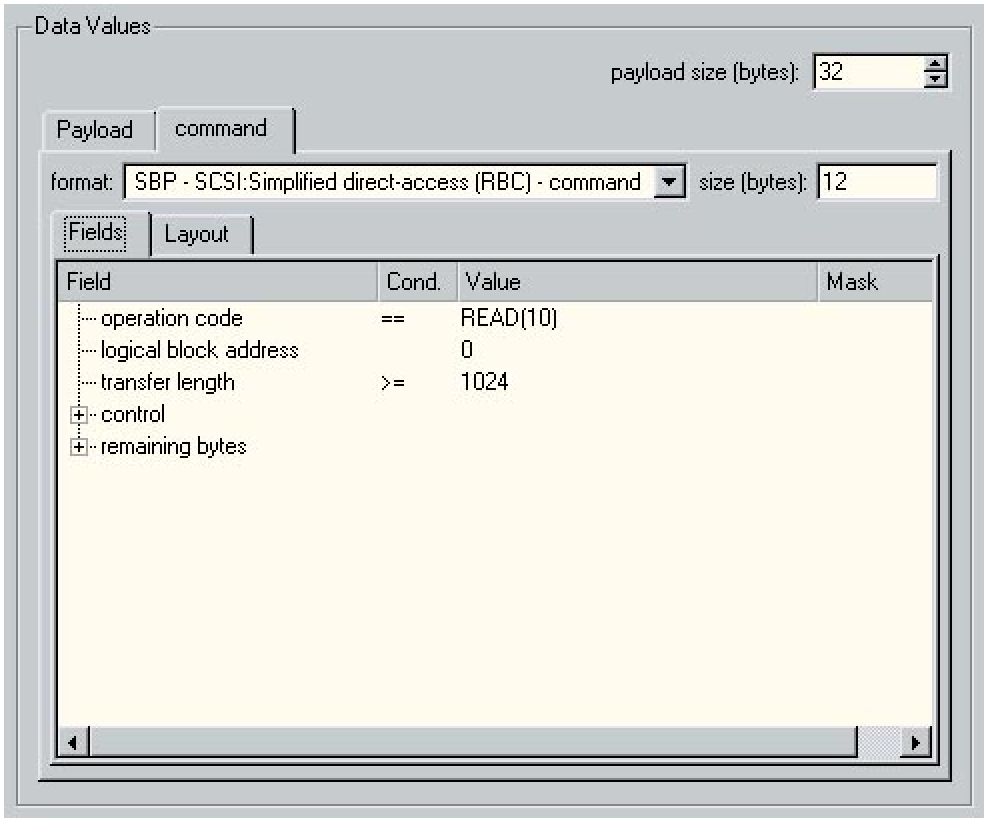
condition may be entered.
A condition can be set by specifying a condition and a value. In the example above for instance, a condition is set on the Source field.
A packet only fits the Set if all specified conditions (and that are not disabled) are true for the packet.
In the example above for instance, a packet does not fit the Set if the Source field is not equal to 0xFFC1.

The Header Values table has the following columns:

- *Field.* The Field column displays the header-field name. Note that only those fields are enabled that are present in all the selected types in the Primary-Packets box. Fields not enabled are displayed with a gray color.
- *Cond.* Here a compare type can be selected. You define a compare type by clicking in this field and select the type from the combo box. You can select the following types:
  - == condition met if the packet-field value equals the specified Value.
  - != condition met if the packet-field value does not equal the specified Value.
  - <= condition met if the packet-field value is less than or equal to the specified Value.
  - >= condition met if the packet-field value is greater than or equal to the specified Value.

  You can also select an 'empty' operation, meaning that the field condition has to be removed. Selecting this will also remove the field Value.
  Note that for each byte in the packet, only one condition may exist, unless the condition is '=='. You can for instance not put a '<=' condition on the Transaction Label field if there is also a condition other than '==' on the Retry Code field. If both conditions are '==', there is no problem. The software will inform you about possible conflicts as soon as you try to Apply the settings.
- *Value.*Here you can specify the value for the condition. You can enter values in decimal or hexadecimal notation. If you clear this field, the compare type in the Cond. columns will be removed too. Together with the compare type from the Cond. columnm it specifies the condition the packet field should have to fit the Set.
- *Mask.* Before the packet field is compared with the Value, a logical and is performed with this optional Mask value. It can be used to mask one or more bits before the compare is done. For each bit in the mask with value 0, the corresponding bit in the packet field and the corresponding bit in the specified Value will be cleared before they are compared. If, in the example above, a mask value of 0x003F was specified for the Source field, then effectively only the lower 6 bits are compared because all other bits will be zero-ed by the mask operation.

Data Values
Here you can specify additional data-field conditions. Basically, it works the same as the header-field conditions, but because the format and size of data fields are not fixed, a powerful packet-data editor has been added. Here you can select the format for all kinds of data-payload types.
In the example above for instance, the CommandOrb format (part of the SBP2 support) has been selected as the packet payload format. This gives you some fields on which conditions can be set. In the example above, the conditions are such that a packet will fit the set only if the rq_fmt field equals "SBP2 standard", the direction field equals "Read", the speed field equals "S100" and the 'data size' field is greater than or equal to 1024. Note that some values can be selected by selecting a symbolic value from a combo box, so you do not have to look up these values. If you want to know the values of these symbolic names, you can look inside the 'Layout' tab, or you can use a right mouse click and select the 'Decimal'-values option for instance.
For some formats, there may be a data field for which you can define a separate sub-format in an additional tab. Here you can also add additional conditions for the fields in that sub-format. In the CommandOrb format example above for instance, the 'command' field can be sub-formatted in an additional tab. In the example below this additional 'command' tab has been selected. The 12-bytes command is shown here. Now you can select a separate format for this 12-bytes command. In the example, we chose the 'Simplified direct access' command set, which is part of the SCSI specifications. This format can show all SCSI commands that are part of that set. When you select another operation code, the format will adapt to the format for that particular command. This way it is very easy to add additional conditions on fields that are specific for a command set and specific to a command from that set. In the example below for instance, additional conditions are added so that only a READ(10) command that has also a 'transfer length' field value of 1024 or more fits the set.



Data Values

payload size (bytes): 32

Payload | command

format: SBP - SCSI:Simplified direct-access (RBC) - command   size (bytes): 12

Fields | Layout

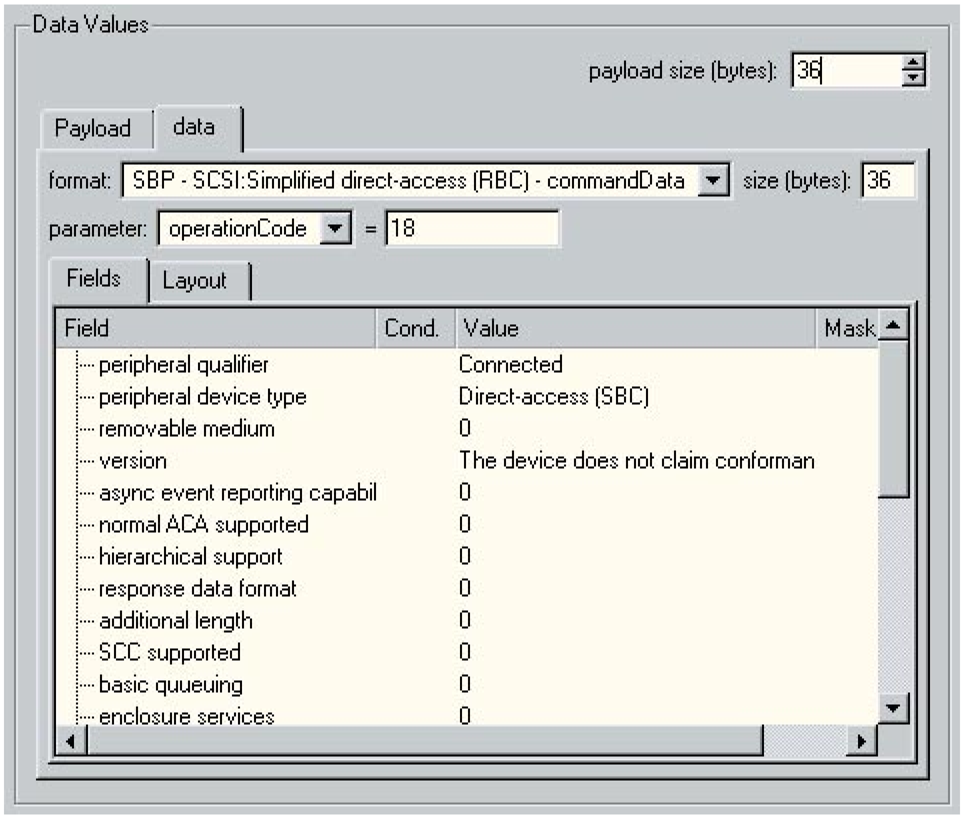
Field	Cond.	Value	Mask
operation code	==	READ(10)	
logical block address		0	
transfer length	>=	1024	
control			
remaining bytes			

Depending on the license keys that are installed, you can select all kinds of formats and optionally sub formats where needed. There are formats for the SBP protocol, IIDC protocol, IP4 protocol and AV/C protocol. You can also select the format 'unformatted' to be able to set any kind of condition on any bits. Now we will describe the different parts of the Data Values box:

payload size (bytes):
Here you can enter the payload size of the packet expressed in number of bytes. It will specify the size of the data shown in the 'Payload' tab. Changing the value will also change the size (bytes) value in the 'Payload' tab. This value will automatically be adapted if you specify a new value for the 'Data-Length Async. packet' field in the Header Values box.

Format tabs
The 'Payload' tab will always be present. It describes the whole payload. If a field can be described with a sub-format, an additional tab will be present for the format of that field. See 'command' tab in the example above. All tabs have the same parts in it:

- *format.* With this combo box you can select the format of the packet or sub-format of the field. Note that the formats you can select from depend on the license keys that are installed for the currently connected Analyzer. If no Analyzer is connected, there will only be the unformatted option.
- *size (bytes).* Here the number of bytes will be displayed that correspond to the data in the selected tab. For the 'Payload' tab this will be the size of the complete payload as indicated in the payload size (bytes): above it.
- *parameter.* Sometimes a format may need extra parameters to be able to format the data correctly. In the example below for instance, the selected format is one of the command sets for the command data for the SBP2 protocol. Command data is returned as a reaction on a command. In this case, the command has an operation code that determines the format of the command data. Thus to be able to format a command data packet, you will need to enter an operation code. In such situations, a parameter combo box will appear. Here you can select a parameter and enter the value for it in the box

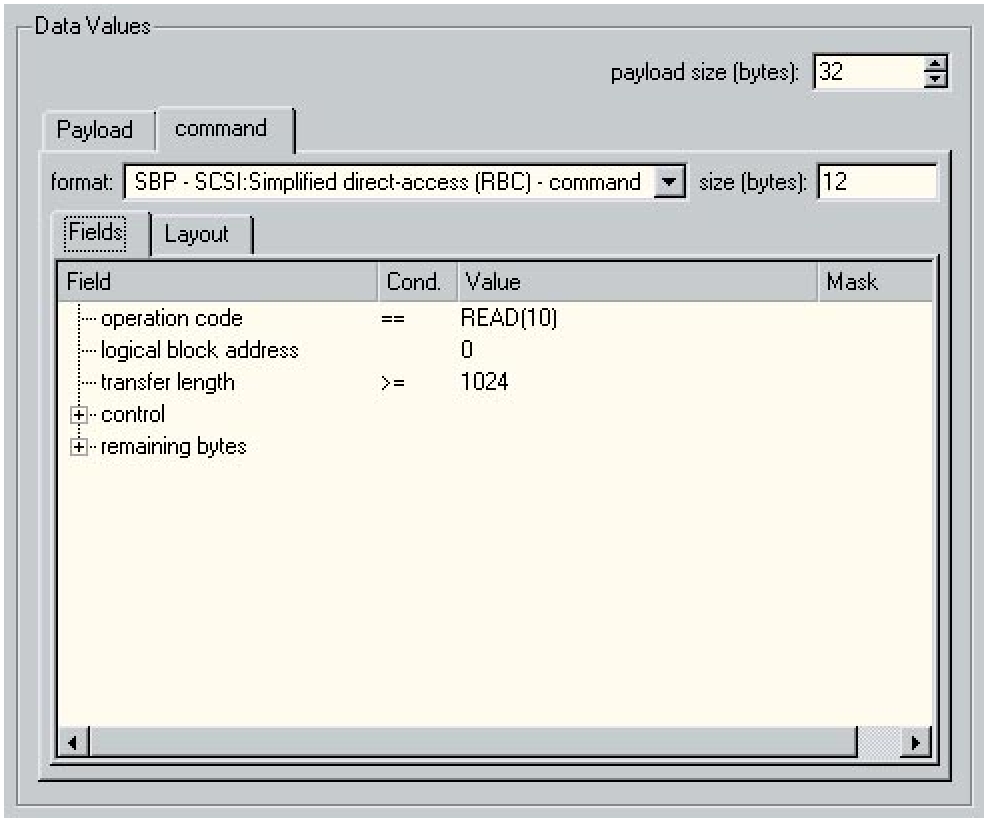

Data Values
payload size (bytes): 36
Payload
data
format: SBP - SCSI:Simplified direct-access (RBC) - commandData
size (bytes): 36
parameter: operationCode = 18
Fields
Layout

Field	Cond.	Value	Mask
peripheral qualifier		Connected	
peripheral device type		Direct-access (SBC)	
removable medium		0	
version		The device does not claim conforman	
async event reporting capabil		0	
normal ACA supported		0	
hierarchical support		0	
response data format		0	
additional length		0	
SCC supported		0	
basic queueing		0	
enclosure services		0	

right of it.
For some parameters, the value can be entered by selecting a symbolic value from a list of symbolic values in a combo box.

- *Fields.* The formatted payload or field can be shown as a table or a layout picture. The Fields tab shows the table. Here you can enter the data-field conditions just as you could do with the header fields. The table has the following columns:
  - *Field.* The Field column displays the data field name.
  - *Cond.* Here a compare type can be selected. You define a compare type by clicking in this field and select the type from the combo box. You can select the following types:
    - == condition met if the packet-field value equals the specified Value.
    - != condition met if the packet-field value does not equal the specified Value.
    - <= condition met if the packet-field value is less than or equal to the specified Value.
    - >= condition met if the packet-field value is greater than or equal to the specified Value.

    You can also select an 'empty' operation, meaning that the field condition has to be removed. Selecting this will also remove the field Value.
    Note that for each byte in the packet, only one condition may exist, unless the condition is '=='. The software will inform you about possible conflicts as soon as you try to Apply the settings.
  - *Value.* Here you can specify the value for the condition. Note that the values are dispayed in a form depending on the field type. For instance the 'transfer length' field in the example below is displayed as a decimal number. The 'logical block address' field is displayed as hexadecimal number and the 'operation code' field is shown using symbolic values.



Field	Cond.	Value	Mask
operation code	==	READ(10)	
logical block address		0	
transfer length	>=	1024	
control			
remaining bytes			

Decimal and hexadecimal numbers can be entered decimal or hexadecimal. For fields with symbolic values, a combo box will be displayed when clicking on the value so that a value can be chosen. You can change this behaviour by selecting another display form (overruling the automatic form) using the right mouse button as described below.
If you clear this field, the compare type in the Cond. columns will be removed too.
Together with the compare type from the Cond. column it specifies the condition the packet field should have to fit the set.

- *Mask.* Before the packet field is compared with the Value, a logical-and is performed with this Mask value. It can be used to mask one or more bits before the compare is done. For each bit in the mask with value 0, the corresponding bit in the packet field and the corresponding bit in the specified Value will be cleared before they are compared.

Using a right mouse click in the Field table you can popup a menu to select one of the following options:

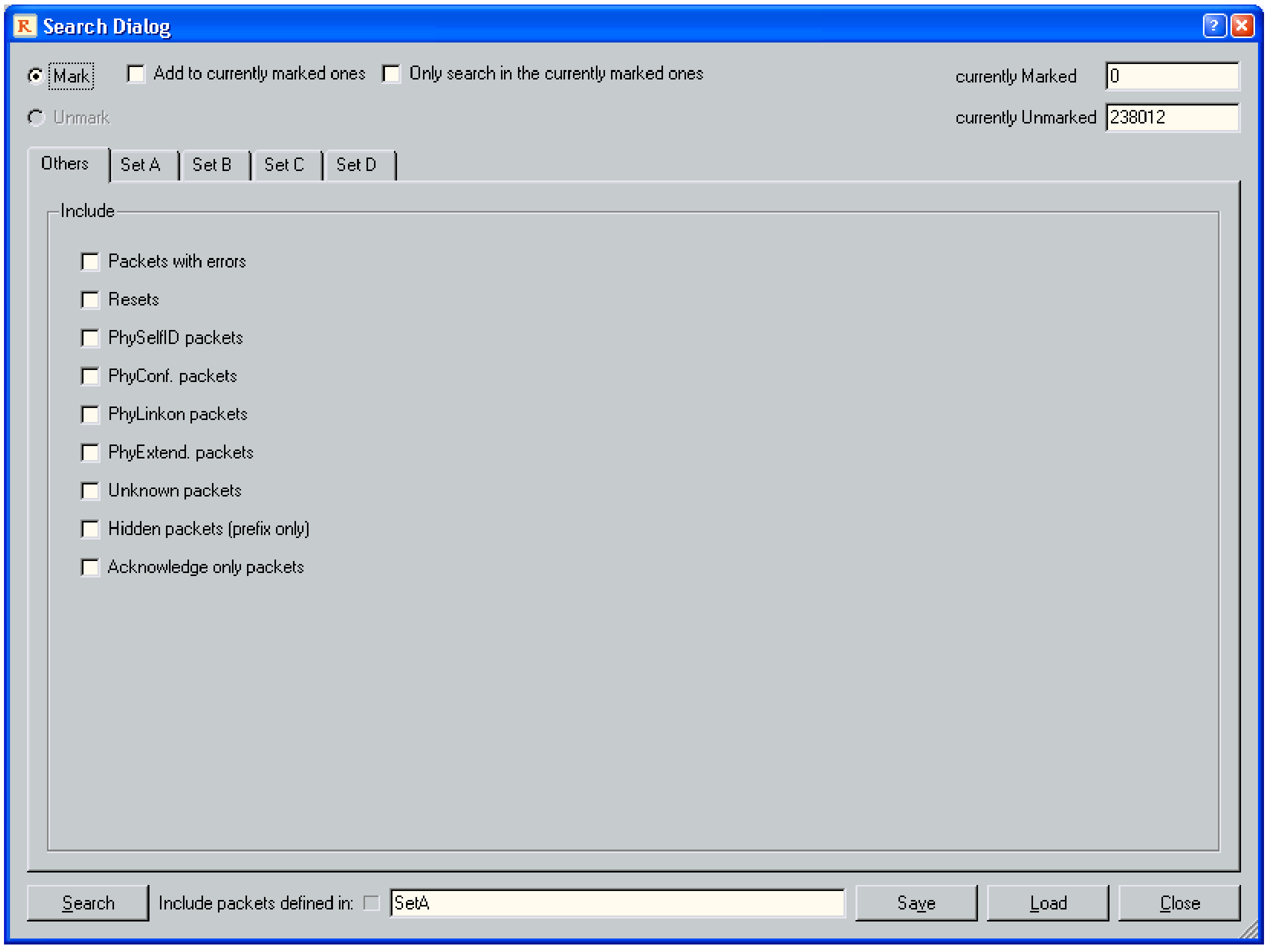
- *display form.* You can select if the value of the fields should be displayed Hexadecimal, Decimal or Automatic. Automatic (default) means that the format will determine the display form for each field separately. One of the automatic forms can be a symbolic value.
- *Select All.* With this option you can quickly select all fields in the table.
- *Import and Export.* With this option you can import or export the selected field(s) or all fields. If you choose to import, a file dialog is presented. You can select to import from a hex data file (*.hex) file or quadlet data file *.qdl file. See File Formats for more information on these formats. If you choose to export, you can select the type of export in a little dialog and then with a file dialog you can select a file to export to.
- *Hex/Ascii edit.* With this option you can edit the selected field(s) or all fields using a hex/ascii editor.

- *Layout.* The formatted payload or field can be shown as a table or a layout picture. The Layout tab shows the layout picture. The same fields are presented as in the 'Fields' tab, but now you can see the sizes of the fields and how they are positioned in the data block. Note that reserved fields are shown here too. Reserved fields are not shown in the 'Fields' table.

Save/Load Search dialog settings file
Your customized search criteria can be saved from and loaded to the dialog. An files has the extension of "*.frs". Save and Load buttons are located at the bottom of the dialog.

7.8.2.5. Others page

As well as specifying packets using the packet sets, we can also quickly include packets and events using the 'Others' tab page. On this page you will find a number of packet types and events that can be included by checking those check boxes. All the checked packets and events will be marked or unmarked. Below is an example of the 'Others' page. In this example all reset events and Phy-SelfID packets will be marked when the 'Search' button is pressed.


Search Dialog

[✓ Mark] Unmark

☐ Add to currently marked ones ☐ Only search in the currently marked ones

currently Marked: 0

currently Unmarked: 238012

Others | Set A | Set B | Set C | Set D

Include

☐ Packets with errors
☐ Resets
☐ PhySelfID packets
☐ PhyConf. packets
☐ PhyLinkon packets
☐ PhyExtend. packets
☐ Unknown packets
☐ Hidden packets (prefix only)
☐ Acknowledge only packets

Include packets defined in: ☐ SetA

Chapter 8. Mil1394 Player

The Mil1394 Player is a FireSpy application which is able to transmit AS5643 stream packets as recorded by the Recorder. This allows the user to recreate a previously recorded situation on demand, including dropped packets, timing faults and missing data. The Mil1394 Player can help reproduce specific use cases, which can be used in developing the system around it.

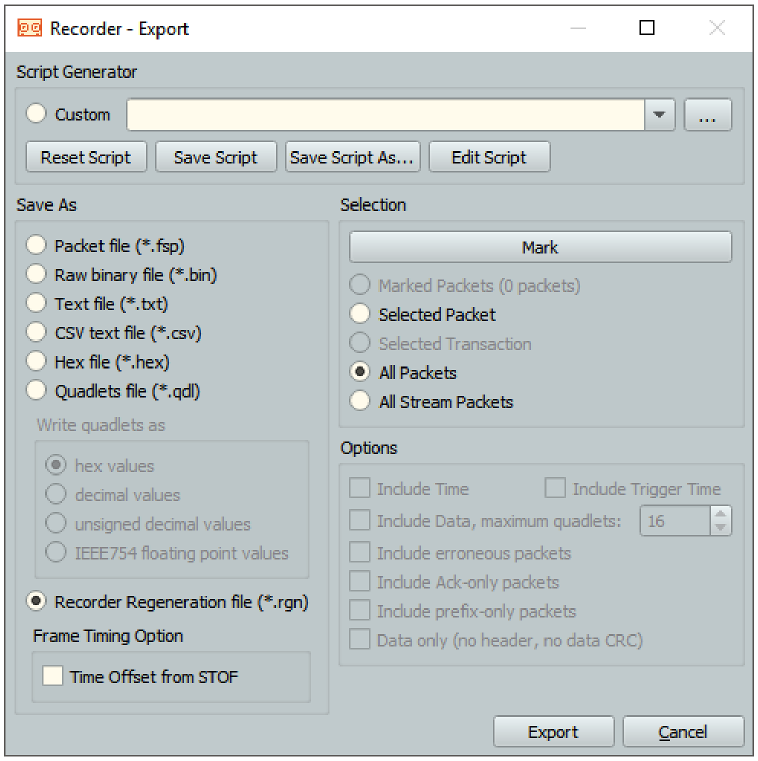
8.1. How to use it

The Mil1394 Player imports the AS5643 stream packet data from regeneration files. (*.rgn) These files can be created using the Recorder. Recorded stream packets can be exported using the Export Packets dialog. Once the regeneration file is loaded into the application, settings such as channel selection and STOF packet generation can be selected. After that the stream packet data will be uploaded to the FireSpy, and it will play the regeneration file.

8.1.1. Export Recorder Data

The Recorder will be used to collect the stream packet data. This can be done by making a recording, or by opening an .fsr Recorder file. It is possible to export a specific set of stream packets, by using the packet marking functionality.

To export the stream packet data, select File > Export Packets... from the Recorder's toolbar. This will bring up the following dialog.



Script Generator: Custom

Save As:
- Packet file (*.fsp)
- Raw binary file (*.bin)
- Text file (*.txt)
- CSV text file (*.csv)
- Hex file (*.hex)
- Quadlets file (*.qdl)

Write quadlets as:
- [✓ hex values]
- decimal values
- unsigned decimal values
- IEEE754 floating point values

[✓ Recorder Regeneration file (*.rgn)]

Frame Timing Option:
- Time Offset from STOF

Selection:
- Marked Packets (0 packets)
- Selected Packet
- Selected Transaction
- [✓ All Packets]
- All Stream Packets

Options:
- Include Time
- Include Trigger Time
- Include Data, maximum quadlets: 16
- Include erroneous packets
- Include Ack-only packets
- Include prefix-only packets
- Data only (no header, no data CRC)

Save As
In the Save As field, select the Recorder Regeneration file (*.rgn) option.

Frame Timing Option
To generate stream packets with a timestamp relative to the previous STOF packet, check the Time Offset from STOF option. Uncheck this field to create stream packets with absolute timestamps.

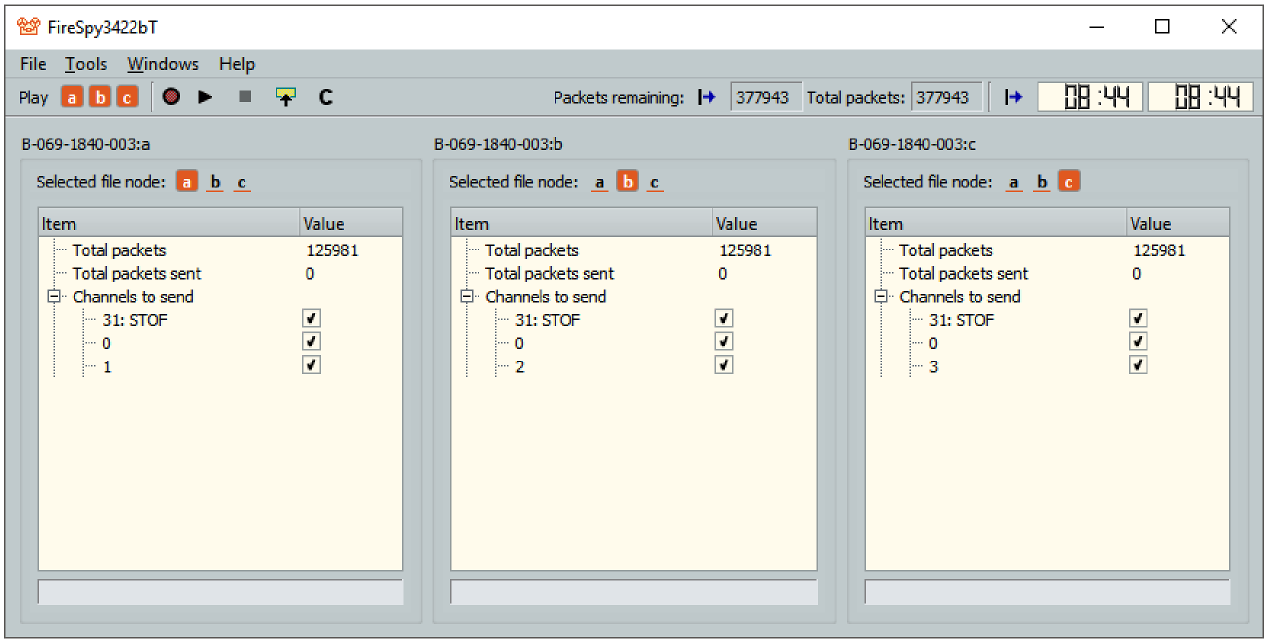
For the most accurate timing data, do not select the Time Offset from STOF option.

Selection
Use this field to select whether the Recorder should export all packets, the marked packets or a selected packet(s).

Export packets
Click on Export to create the Regeneration file (*.rgn). This is the file that will be used by the Mil1394 Player.

8.1.2. Play Regeneration File

You can open the Mil1394 Player by clicking on the Player button in the main window. When opening the Mil1394 Player, an empty user interface will appear. To use the Player, load a Regeneration file (*.rgn) into the Mil1394 Player using the 'Open' command of the 'File' menu.



Packets remaining: 377943 Total packets: 377943 08:44 08:44

B-069-1840-003:a

Selected file node: a b c

Item	Value
Total packets	125981
Total packets sent	0
Channels to send	
31: STOF	✓
0	✓
1	✓

B-069-1840-003:b

Selected file node: a b c

Item	Value
Total packets	125981
Total packets sent	0
Channels to send	
31: STOF	✓
0	✓
2	✓

B-069-1840-003:c

Selected file node: a b c

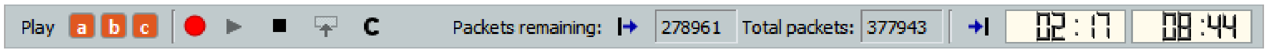
Item	Value
Total packets	125981
Total packets sent	0
Channels to send	
31: STOF	✓
0	✓
3	✓

To play the selected Regeneration file, simply click the Play button in the toolbar at the top. This will upload the Regeneration data to the FireSpy, and play the stream packets on the FireSpy nodes. Use the Upload button will only upload the data to the FireSpy, without starting the Player.

8.2. Details

8.2.1. Toolbar

The toolbar is placed at the top of the Mil1394 Player window.



Packets remaining: 278961 Total packets: 377943 02:17 08:44

From left to right, the toolbar has the following elements.

Play node selector
For FireSpy analyzers with multiple buses, the Play node selector can be used to enable and disable nodes on the FireSpy.

Mil1394 Player active indicator
This status LED will turn on when the Mil1394 Player is active.

Play, Stop, Upload data, Clear FireSpy memory
These buttons are used to start and stop the Mil1394 Player, as well as uploading the data to the FireSpy, or clearing its memory.

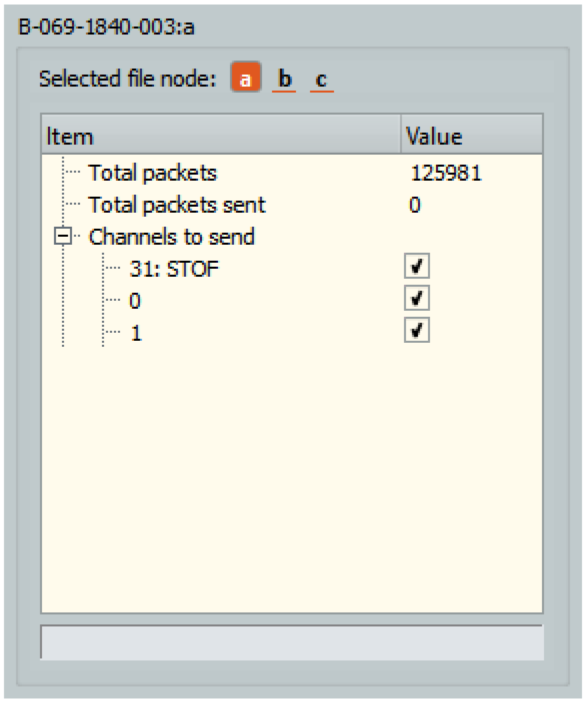
Packet counters
The first 'packet counter' shows either the remaining or sent packets, which can be switched with the

toggle button. The second counter shows the total number of packets to be sent.

Time indicators
The first ‘time indicator’ shows either the remaining or elapsed time, which can be switched with the toggle button. The second time indicator shows the total time needed for the Regeneration file to be played completely.

8.2.2. Bus settings

The user interface consists of a number Mil1394 Player bus setting dialogs, one for each bus on the used FireSpy. These setting dialogs contain information about the Regeneration file, and can be used to setup how the Player will play the loaded data.



Item	Value
Total packets	125981
Total packets sent	0
Channels to send	
31: STOF	✓
0	✓
1	✓

The Player bus settings contain 3 elements: A node selection, a table with packet information, and a memory bar.

Selected file node
The 'Selected file node' dialog shows how many nodes are available in the Regeneration file. These nodes correspond to the nodes that are used in the Recorder to create the Regeneration file. Use this dialog to select which recorded node must be played on which FireSpy bus.

Packet Information table
The 'Packet Information table' shows an overview of the data from the selected regeneration file node. It displays the total number of packets that are recorded on this node, and how many are already sent when playing the file.

This dialog can also be used to select which channels will be used when playing the Regeneration file.

Memory bar
The bar at the bottom of the box displays the memory status of the FireSpy. As the data from the Regeneration file will be uploaded to the FireSpy, the memory bar will be filled. More memory space will become available while playing the file, so playing large files is also possible.

Chapter 9. Symbol Recorder

NOTE: The Symbol Recorder module is only available on FireStealth analyzers with a Symbol Recorder Module license.

The 'Time View' enables the user to view the timing of symbols, packets and toning as they occur on the connection. A time cursor is used to make time measurements and for selecting individual symbols.

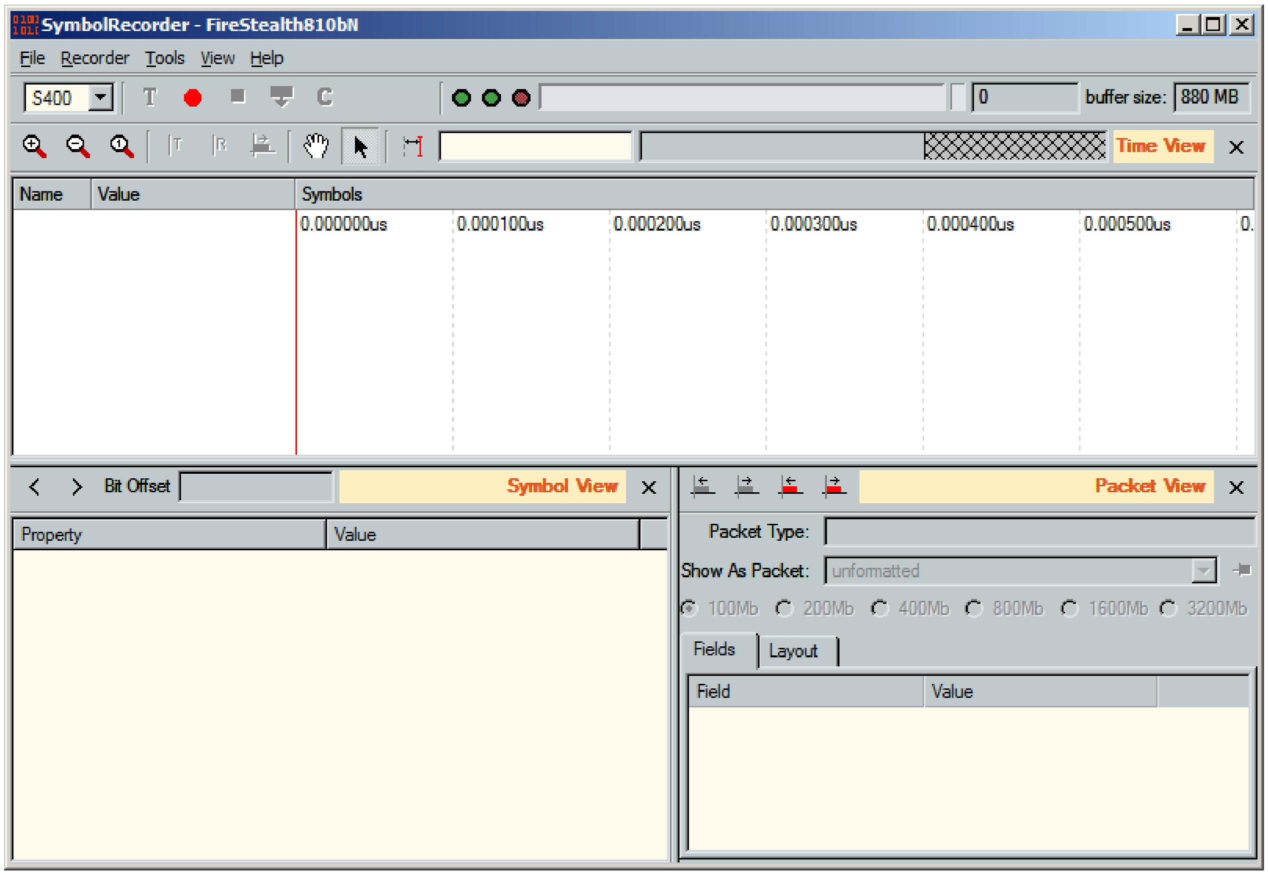
The 'Packet View' displays all packets in a list. Each packet can be viewed in detail, by showing its packet fields, packet layout or packet errors. Packets can be displayed as any possible packet type, so the user can find out what kind of packet an erroneous packet may be.

The 'Symbol View' displays the details of a selected item.

9.1. Main Window

You can open the Symbol Recorder by clicking on the Symbol Recorder button in the main window, or select the Symbol Recorder from the 'FireSpy' menu at the top of the main window or one of the other open windows. It is also possible to start the Symbol Recorder stand alone application from the Windows Start menu. Only the standalone version is capable of controlling multiple analyzers at the same time.

When you open the Recorder, the window below will appear.

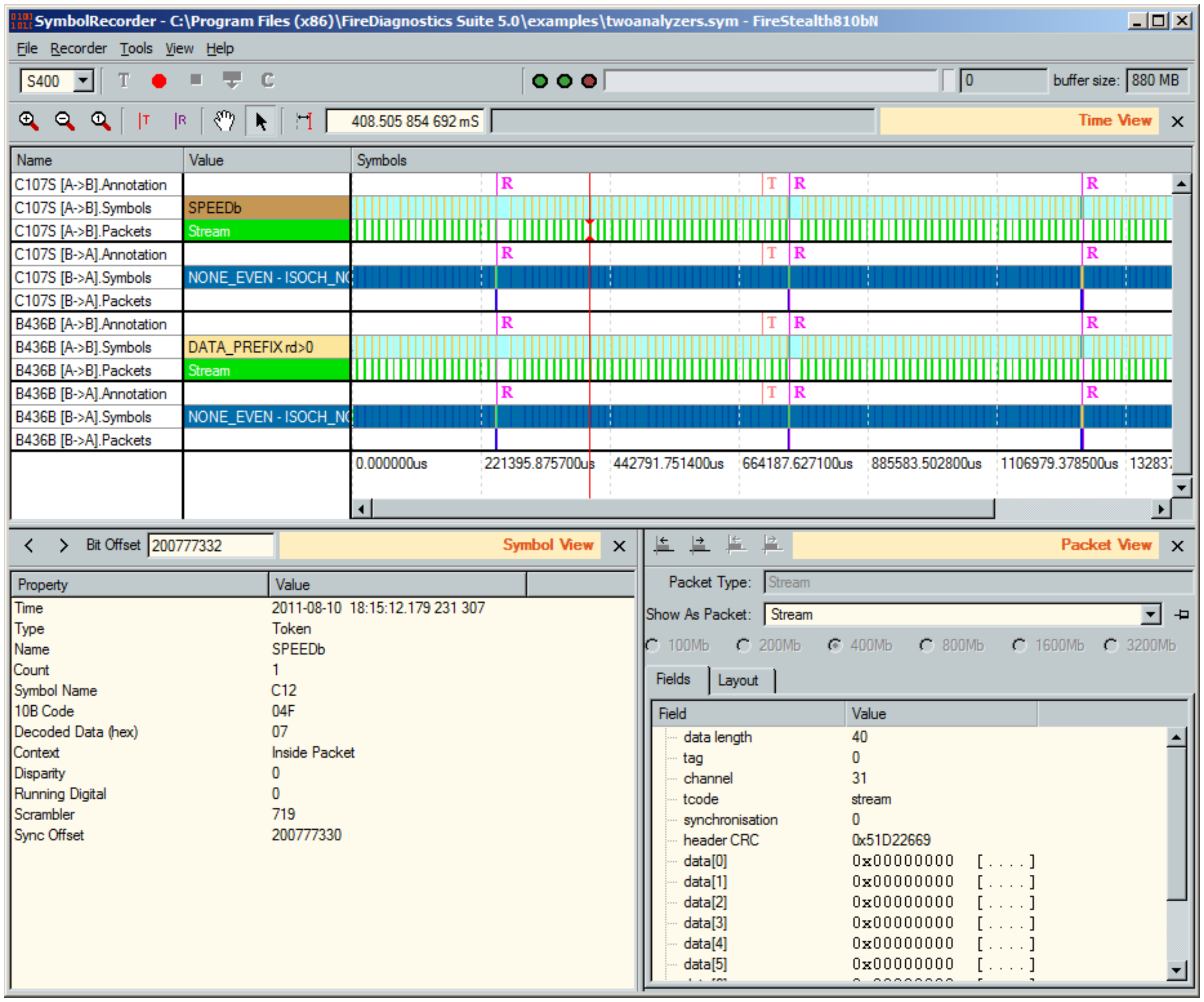

SymbolRecorder - FireStealth810bN
File Recorder Tools View Help
S400
0
buffer size: 880 MB
Time View
Name Value Symbols
0.000000us 0.000100us 0.000200us 0.000300us 0.000400us 0.000500us
Bit Offset
Symbol View
Property Value
Packet View
Packet Type:
Show As Packet: unformatted
100Mb 200Mb 400Mb 800Mb 1600Mb 3200Mb
Fields Layout
Field Value

The window is filled with different kinds of views for the recorded data. You can switch them on and off. The Time View is always displayed (if enabled) at the top. The remaining views share the area between them . You can enable and disable the different kinds of views using the 'View' menu. In the picture above, the Time View, Symbol View and the Packet View are visible.

9.1.1. How to use it

9.1.1.1. Display Symbol Recorder Files

Initially, there is no recorded data and the views are empty. You can load a symbol recorder file into the Symbol Recorder using the 'Open' command of the 'File' menu. You can select a file (extension .sym) and it will be loaded. In the example below, the recorder file 'twoanalyzers.sym' is opened.


SymbolRecorder - C:\Program Files (x86)\FireDiagnostics Suite 5.0\examples\twoanalyzers.sym - FireStealth810bN
File Recorder Tools View Help
S400
buffer size: 880 MB
408.505 854 692 mS
Time View
Symbol View
Packet View

Name	Value
C107S [A->B].Annotation	
C107S [A->B].Symbols	SPEEDb
C107S [A->B].Packets	Stream
C107S [B->A].Annotation	
C107S [B->A].Symbols	NONE_EVEN - ISOCH_NC
C107S [B->A].Packets	
B436B [A->B].Annotation	
B436B [A->B].Symbols	DATA_PREFIX rd>0
B436B [A->B].Packets	Stream
B436B [B->A].Annotation	
B436B [B->A].Symbols	NONE_EVEN - ISOCH_NC
B436B [B->A].Packets	

0.000000us 221395.875700us 442791.751400us 664187.627100us 885583.502800us 1106979.378500us 132837

Bit Offset 200777332

Property	Value
Time	2011-08-10 18:15:12.179 231 307
Type	Token
Name	SPEEDb
Count	1
Symbol Name	C12
10B Code	04F
Decoded Data (hex)	07
Context	Inside Packet
Disparity	0
Running Digital	0
Scrambler	719
Sync Offset	200777330

Packet Type: Stream
Show As Packet: Stream
100Mb 200Mb [✓ 400Mb] 800Mb 1600Mb 3200Mb
Fields Layout

Field	Value
data length	40
tag	0
channel	31
tcode	stream
synchronisation	0
header CRC	0x51D22669
data[0]	0x00000000 [....]
data[1]	0x00000000 [....]
data[2]	0x00000000 [....]
data[3]	0x00000000 [....]
data[4]	0x00000000 [....]
data[5]	0x00000000 [....]

You can use the 'Recorder' menu or toolbar buttons to start recording raw data into internal memory of the Analyzer, and download them to the host. The recorded data can be viewed in the same way an opened file can be viewed. The recorded data can also be saved to disk as a symbol recorder file using the 'Save As' command in the 'File' menu.

9.1.1.2. Record Symbols

Before making a recording it is important to set the correct speed and polarity. The speed should be set to the actual connection speed of the connection the FireStealth is placed in line with. The polarity should be set according to the actual polarity of the two cables plugged in. Please refer to the section about FireStealth connection settings.

If the internal symbol recorder memory of the Analyzer is empty, you can start recording by selecting the 'Start' command from the 'Recorder' menu, or clicking the 'Start Recording' button in the toolbar.

When the internal recorder memory is not empty, you can clear it with the 'Clear' command in the 'Recorder' menu or by clicking the 'Clear FireSpy recorder memory' button in the toolbar.

The 'Recorded bytes' box in the toolbar will display the number of recorded bytes, and the 'Record progress bar' in the toolbar will indicate the part of recorder memory that is filled with data.

Note that when the Recorder is not triggered yet, only the part before the trigger indicator in the progress bar (the little vertical line) will be filled. If this part becomes full, recording continues and old data will be removed from the memory while the new data is stored (cyclic buffer). After the Recorder is triggered, the data will be stored after the trigger position until the recorder memory is full.

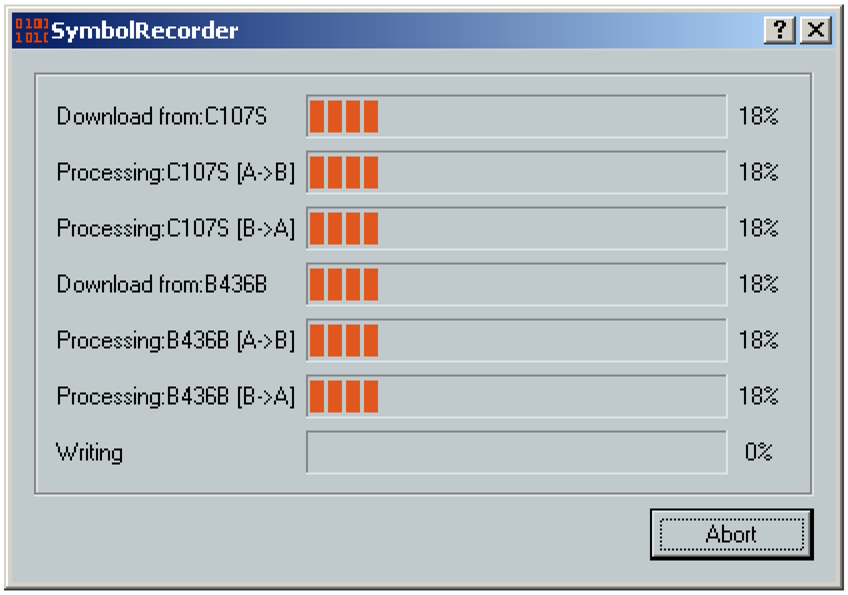
If recorder memory becomes full, the recording is automatically stopped. You can also stop the recording manually by selecting the 'Stop' command in the 'Recorder' menu, or clicking the 'Stop Recording' button in the toolbar.

9.1.1.3. Download and process symbols

When the recording has been stopped, you can download the recorded data by selecting the 'Download' command in the 'Recorder' menu or clicking the 'Download recorded data from Analyzer' button in the toolbar.

After the download, the data can be viewed using the different kind of views.

The recorder records every bit that is send over the bus, the processing of the recorded data is done in software on the host. The processing starts simultaneously with the downloading of the data. If more than one analyzer is used to examine a bus, the downloading can also be done in parallel. The progress dialog will show the progress of each process. As the amount of data can easily get too large to fit in memory, the decoded data is saved to file during the process.



Download from:C107S 18%
Processing:C107S [A->B] 18%
Processing:C107S [B->A] 18%
Download from:B436B 18%
Processing:B436B [A->B] 18%
Processing:B436B [B->A] 18%
Writing 0%

9.1.1.4. Triggering

The Recorder can be triggered in different ways:

- You can click the 'Trigger' button in the toolbar or invoke the 'Trigger' command in the 'Recorder' menu.
- You can apply a trigger pulse on the external trigger input on the Analyzer device.

The hardware-trigger logic will be described in full detail in 'Filter/Trigger'. Please be aware that for FireStealth only the General Trigger functionality is available.

Each time you start the Recorder, a check is made if the user changed some settings in the Filter/Trigger Settings window. If so, the user will be asked if these changes need to be activated or ignored before starting the recording.

9.1.1.5. *Using multiple analyzers*

When multiple FireStealth devices are connected to the same topology it is important that they are synchronized to an IRIG-B122 time source. It is also possible to control multiple devices without time synchronization but then then Recorder simply assumes all analyzer recordings started at exactly the same time.

To control multiple analyzers please open the standalone "Symbol Recorder" from the Start menu. (Start->Programs->FireDiagnostics Suite x.x->FireSuite Tools -> Symbol Recorder). The application will first show a device selection dialog which allows selecting multiple devices before pressing ok.

When each FireStealth is connected to an Irig-B122 Decoder using a small Sub-D cable and the Irig source is properly connected to the Irig Decoder boxes then time synchronization can be setup from the settings dialog:

- Please make sure all devices used are setup in Master as described here.
- Please make sure external timing is used for recordings as described here.

9.1.2. Details

9.1.2.1. *Menus*

File

Open
With this command you can open an existing Symbol recorder file. By default these files have an extension of .sym. You can also choose to open a raw .bin file. These files are saved during the download phase and only contain the raw bits and a direction. There is no timing information available in these .bin files which means the two streams will most likely not be correctly time aligned.

Save As
With this command you can save the current Recorder data as a Symbol recorder file. By default, these files have an extension of .sym.
Recorder data saved this way, can later be viewed again by using the 'Open' command of the 'File' menu.

Recorder
In this menu you will find the following Recorder control commands: 'Start', 'Stop', 'Clear', 'Download' and 'Trigger'. They have the same functionality as the corresponding Recorder toolbar buttons described below.

View
With this menu you can enable and disable the different kind of Recorder views:

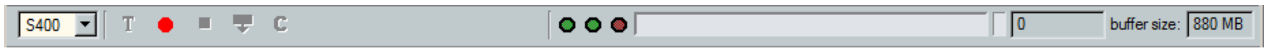
- Time View
- Symbol View
- Packet View

All the views that are checked will be displayed. You can toggle between checked and un-checked by selecting the corresponding command.

Windows
From the 'Windows' menu you can open one of the other windows of the Analyzer.

9.1.2.2. *Toolbar*

The toolbar contains respectively the following indicators and buttons:



Indicators

Triggered indicator
This green indicator will light when the Recorder has been triggered since the last record start.

It has the same status as the 'trigger led' on the Analyzer front panel (see Hardware).

Ready indicator
This green indicator will light when the Recorder has been stopped and has recorded data in its memory ready for download.
It has the same status as the 'record ready led' on the FireSpy front panel (see Hardware).

Recording indicator
This red indicator will light when recording is in progress. Symbols will be stored into the recorder memory. Note that on an idle bus, a continuous stream of symbols is transmitted and will fill up the memory.
It has the same status as the 'record led' on the Analyzer front panel (see Hardware).

Buttons

Trigger
With this button the Recorder can be triggered. It is only enabled if recording is in progress and the Recorder is not triggered yet.

Start recording
With this button the Recorder can be started. After starting the Recorder, the recording will be in progress until it is stopped. It is enabled when the recorder memory is empty and the recording is not in progress yet.

Stop recording
With this button recording can be stopped. It is enabled when the recording is in progress.

Download recorded data from Analyzer
With this button the recorded data can be downloaded to the host to be displayed. It is enabled if recording is not in progress and the recorder memory is not empty.

Clear Analyzer recorder memory
With this data the recorder memory can be cleared. It will also clear the displayed data if the data of the Recorder is currently displayed. The button will be enabled if no record is in progress and the recorder memory is not empty.

Memory indicators

Record progress bar
This progress bar gives an indication which part of the recorder memory has been filed with data. The little vertical line indicates the trigger position. When starting recording, the memory before this trigger position (left to the trigger position) will be filled with data. If this part of the buffer gets full, the recording continues throwing away the oldest data and storing the new data (kind of cyclic buffer). After the Recorder is triggered, the data will be stored after the trigger position (right of the trigger position). If that part gets full, the Recorder is stopped.

Note that the trigger position within the recorder memory can be changed using the 'Settings' command in the 'Analyzer' menu. For more information see Settings.

Recorded bytes
This box displays the total stored number of bytes. It will increase during the recording except when the Recorder is not triggered yet and the memory part before the trigger position is full. In this case old data is removed when new data is stored and the total number of stored bytes will not change.

Recorder memory size
This box displays the total number of available memory bytes for the Recorder. It can be changed with the 'Settings' command in the 'Analyzer' menu. See Settings for more information.

Rec
This control can be used to select the Analyzer nodes to use for recording. Clicking on the "a", "b" or "c" will activate/deactivate a node. Activated nodes are displayed with a red background color. Deactivated nodes are indicated with a white background color.

9.2. Time View

The Time View is one of the possible views in the Recorder which you can use to investigate the recorded data. It displays all packets and symbols on a time line. The relative positions of the packets and symbols correspond to the time the packet or symbol was recorded and the length of symbols and packets in the view correspond to the actual duration of the items.

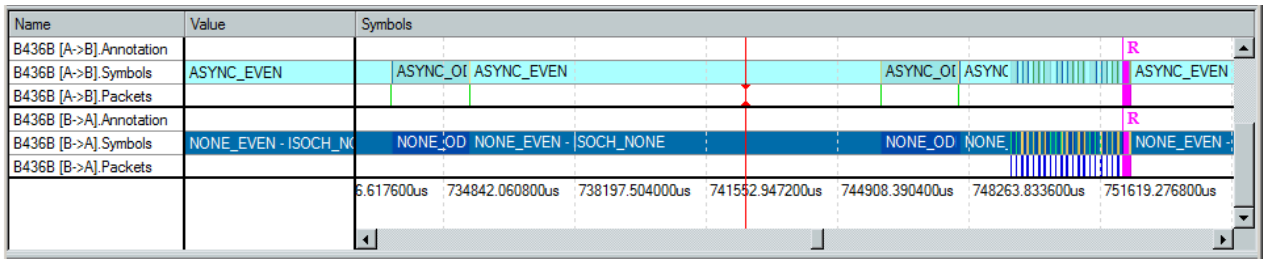
Per direction, the Time View shows

- Annotation, Busreset events and Trigger position
- Symbols, Low level representation of symbols and toning seen on the connection
- Packets, Higher level grouping of symbols that form packets

The cursor can be moved inside the Time View allowing for inspection of packets and symbols. When a packet is selected, the PacketView will show the content of the packet.

9.2.1. How to use it

You can display/remove the Time View by checking/unchecking the Time View item in the View menu of the Recorder. An example of the Time View is displayed below:



At the top of the view you see a toolbar with the buttons zoom in, zoom out, zoom fit and a relative time button and the cursor time.

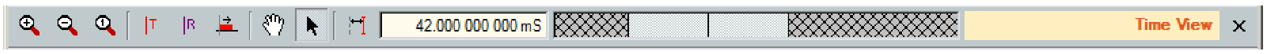
The time at the cursor position is displayed in the toolbar. It can display the absolute time or a relative time. In absolute time mode, time 0 is defined as the left-most position in the time view. In relative mode the displayed cursor time is relative to a reference time. This reference time is indicated in the Time View as a vertical dotted line (see above). The reference time is set when switching from absolute to relative mode, in which case the reference time will be set to the cursor time. Additional mouse clicks will move the cursor, but not the reference. This makes time measurements very easy. To measure the time between the start of a request packet and the start of its response packet for instance, select the request packet, switch to relative mode and select the response packet. The time between them will be displayed in the toolbar. Note that the selection of the packets can also be done in one of the other views of the Recorder.

The cursor will snap to a symbol transition when the mouse is close enough.

9.2.2. Details

9.2.2.1. Toolbar

The Toolbar has the following indicators and buttons:



Zoom in
With this button you can zoom in. You will see more details of the symbols.

Zoom out
With this button you can zoom out. You will see fewer details of the symbols.

Zoom reset
With this button you can reset the zoom amount to its original value.

Goto Next Trigger
With this button you can move the cursor to the next trigger location.

Goto Next Bus Reset
With this button you can move the cursor to the next Bus Reset symbol.

Goto Next Error
With this button you can move the cursor to the Error or Raw symbol.

Grabbing Cursor
With this button you can change the mouse cursor for the Time View into a grabbing mouse cursor allowing to scroll the view with a click and drag motion.
To inspect an item below the mouse cursor you can double click it to move the time view cursor to this position.

Normal Cursor
With this button you can change the cursor for the Time View into a pointing cursor allowing you to select an item and inspect its details in the other view(s).
Switching between a Grabbing mouse cursor and the normal mouse cursor is also possible by right clicking in the time view.

Toggle between absolute and relative time
With this button you can toggle between absolute (default) and relative cursor mode. When not pressed, the cursor is in absolute mode; the time displayed in the 'Cursor time' box (next toolbar item) is the time of the cursor position.

As soon as the button is pressed, the 'Cursor time' is relative to the cursor position at the time the button is pressed. Thus immediately after pressing this button, the 'Cursor time' will be zero (relative to itself). When the cursor is moved now, the position of the old cursor (the reference) is indicated with a dotted line and a horizontal line is drawn between this reference and the new cursor position. The time in the 'Cursor time' box indicates the duration of the horizontal line, or the time between the old and the new cursor position. This feature can be used to easily measure time.

Cursor time
This box indicates the absolute or relative cursor time as described above.

9.2.2.2. Data display

The Time View uses colored bars to display the decoded symbols on the bus. The used colors are:

Color Codes

Control Symbols

CYCLE_START_EVEN
CYCLE_START_ODD
ATTACH_REQUEST
SPEEDa
DATA_END
DATA_NULL
SPEEDb
GRANT
DATA_PREFIX>0
DATA_PREFIX<0
GRANT_ISOCH
SPEEDc
ASYNC_EVEN
ASYNC_ODD
BUS_RESET

Configuration Request Symbols

TRAINING
DISABLE_NOTIFY
CHILD_NOTIFY
OPERATION
STANDBY
SUSPEND
PARENT_NOTIFY

Packet Data

Data

Comma Symbol

Comma

Erroneous

Error

Toning

TONING

Raw (undecodable)

Raw Data

Packets have their own color codes. The used packet colors are:

⬛ cycle start packets
🟩 stream packets
🟦 read/write/lock request packets
🟦 read/write/lock response packets
🟪 phy packets
🟥 packets with error(s)
🟨 acknowledge packets

The cursor is a vertical line with little triangles on the top and bottom.
Events are drawn as vertical lines from the base line to the top, with a letter to the right of the line (somewhere at the top). The letter indicates the event type as follow:

T = Trigger
R = Bus Reset

The Trigger event will be displayed at the position where the trigger occurred.
The Bus Reset event will be displayed at the position where the link interface signals a bus-reset status.

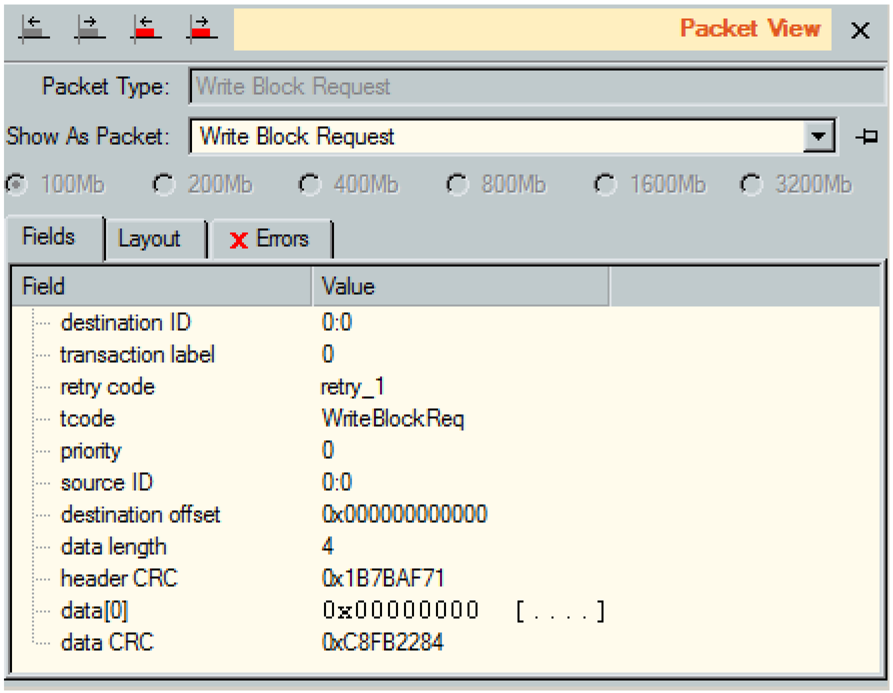
selecting packets
Packets can be selected by clicking with the mouse in the packet rectangle. The cursor will move to the selected packet and the selected packet will also be selected in the other views, so that the details can be seen.

9.3. Packet View

The Packets View is one of the possible views in the Recorder which you can use to investigate the recorded packets.Information of a selected packet can be displayed, including the packet fields and their values, the packet layout and possible warnings and errors that are detected for the packet.

9.3.1. How to use it

You can display/remove the Packets View by checking/unchecking the Packets View item in the 'View' menu of the Recorder. An example of the Packets View is displayed below:


Packet View
Packet Type: Write Block Request
Show As Packet: Write Block Request
100Mb
200Mb
400Mb
800Mb
1600Mb
3200Mb
Fields
Layout
Errors

Field	Value
destination ID	0:0
transaction label	0
retry code	retry_1
tcode	WriteBlockReq
priority	0
source ID	0:0
destination offset	0x000000000000
data length	4
header CRC	0x1B7BAF71
data[0]	0x00000000 [....]
data CRC	0xC8FB2284

Selecting a packet in the time view will display the details of that packet.

Four buttons in the toolbar of the packet view assist in selecting previous and next (erroneous) packets.

9.3.2. Details

The Packets View displays all packets and events in a list and optionally the details of a selected packet. It has a toolbar of its own, a table of all the packets/events and an area to the right of the list where the details of the selected packets can be seen.

9.3.2.1. Toolbar

The Toolbar has the following indicators and buttons:

Go to previous packet

When clicking this button, the previous packet will be selected. It will be enabled if a previous packet is present.

Go to next packet
When clicking this button, the next packet will be selected. It will be enabled if another packet is present.

9.3.2.2. Packet Detail View

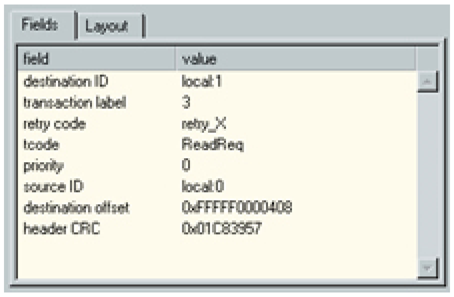
If a packet is selected, the packet view will show its contents. The view has the following parts:

Packet Type
In this box the type of packet that is selected can be seen. If the packet type is unknown (e.g. a packet with erroneous header) this box will say 'unknown packet'.

Show As Packet
This box has a popup list from which you can select a packet type. The packet will be displayed in the format of the packet type selected here. Initially, it will be the same as the 'Packet Type' box. For unknown packets, it will be 'unformatted'. This feature can be used to find out for an erroneous packet what packet it most likely is meant to be, and also to display a packet in its unformatted hexadecimal quadlet form.

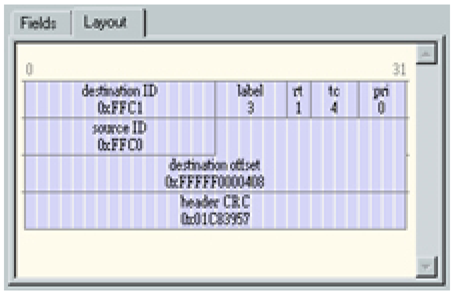
Packet speed
These selection buttons will indicate the speed of the selected packet. They are not selectable by the user, they just function as an indicator.

Fields
For primary packets the 'Fields' page shows all the fields the header of the packet is made of plus the quadlet data fields if they are present. For Phy packets it displays all the fields the phy packet is made of. And for unformatted packets, it just shows all the packet quadlets. The 'source ID' and 'destination ID' fields are displayed in a special format. The bus number and node number are displayed separately, with a ':' character in between them. If the bus number is 1023, it is substituted by the text 'local'.
Below is an example of the 'Fields' page for some read request packet.



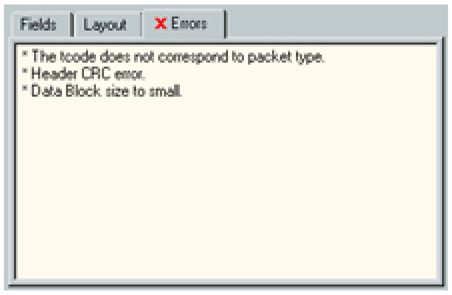
field	value
destination ID	local:1
transaction label	3
retry code	retry_X
tcode	ReadReq
priority	0
source ID	local:0
destination offset	0xFFFFF0000408
header CRC	0x01C83957

Layout
The same fields displayed in the 'Fields' page are displayed in the 'Layout' page. But now the layout of the packet is shown. Each line shows 32 bits, or one quadlet. The exact bit position of each field is shown, including the field name (mostly some abbreviation because of space limitation) and the value. Reserved fields and fields with fixed values are shown too. They have no field name, but only a value.
Below is an example of the 'Layout' page for the same packet as above.



Errors
If the selected packet has one or more errors, or if errors are detected because the selected 'Show As

Packet' type does not correspond to the real packet type, an error page is added to the 'Fields' and 'Layout' pages. Selecting this page will list all detected errors.
Below, an example of the 'Errors' page is shown. In this example, the packet above is shown as a stream packet, which of course will result in errors.



```
* The tcode does not correspond to packet type.
* Header CRC error.
* Data Block size to small.
```

If one or more fields or data is missing (like in the case of 'Data-Block size too small' error above) the 'Field' and 'Layout' pages will also reflect the error by displaying the missing parts in red. Likewise, when there is too much packet data, the parts that are too much are also displayed red in the 'Fields' and 'Layouts' pages.

9.4. Symbol View

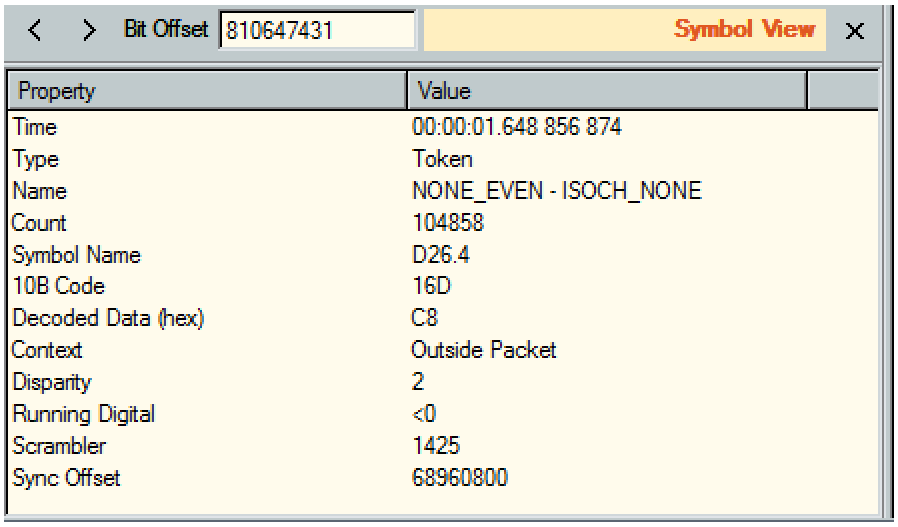
The Symbol View is one of the possible views in the Recorder which you can use to investigate the item selected by the cursor in the Time View. The following items can be found when investigating a recording.

- SignalDetect
- Raw
- Toning
- Token
- Error

These items are described below.

9.4.1. How to use it

The Symbol View allows for navigating to previous or next items. The Bit Offset field shows the current bit position in the recording. It can be used to jump to a different position by editing the field. The list below the toolbar shows the properties of a selected symbol.



Bit Offset 810647431

Property	Value
Time	00:00:01.648 856 874
Type	Token
Name	NONE_EVEN - ISOCH_NONE
Count	104858
Symbol Name	D26.4
10B Code	16D
Decoded Data (hex)	C8
Context	Outside Packet
Disparity	2
Running Digital	<0
Scrambler	1425
Sync Offset	68960800

9.4.2. Details

9.4.2.1. Toolbar

The Toolbar has the following buttons:

Go to previous item
When clicking this button, the previous item is displayed.

Go to next item
When clicking this button, the next item is displayed.

Bit Offset
Displays the current bit offset. It can be edited to jump to a different location in the recording. After editing, press enter to jump to the required location.

9.4.2.2. Detail View

The information shown in the Symbol View depends on the selected item. There are five different types of items visible which are described below. All items have a few common properties.

Common Properties

Bit Offset
This value is the real bit offset of the recorded data in the selected stream.

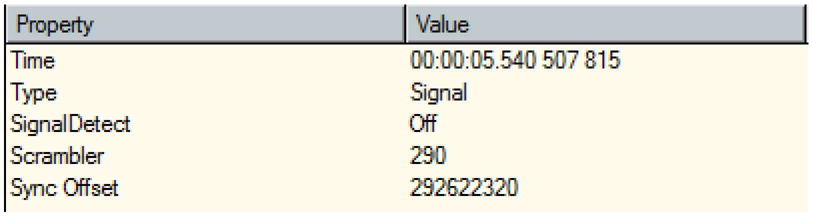
Time
The calculated time at the start of the item.

Type
The type of the item as summarized below.

Scrambler
The value of the scrambler of the sending 1394 port at the current location.

Sync Offset
The offset from the position where the software could find bit synchronization and determine the scrambler value. If this field is negative then there was no bit synchronization at this point in the recording and must be found further ahead.

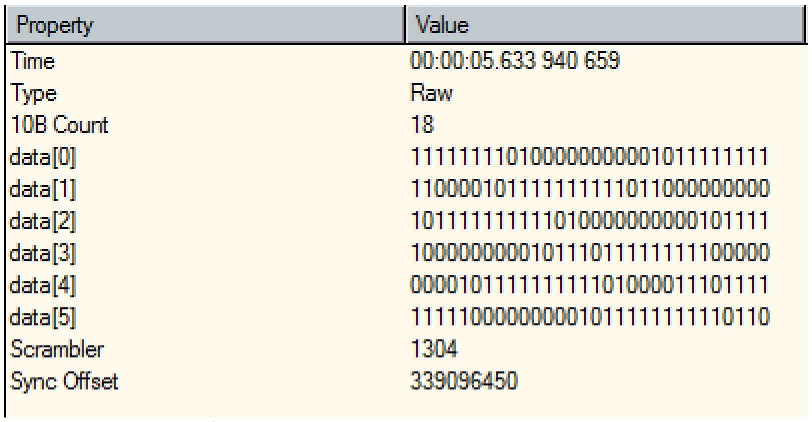
Signal Detect



Property	Value
Time	00:00:05.540 507 815
Type	Signal
SignalDetect	Off
Scrambler	290
Sync Offset	292622320

This item can have two states, 'On' and 'Off'. Signal detect off means the pickup logic of the analyzer do not detect and data. This can be seen when no cable is attached or in between toning pulses and right before a connection is made.
Signal Detect On can be seen at the beginning of a recording and right before the decoding algorithm can decode data properly.

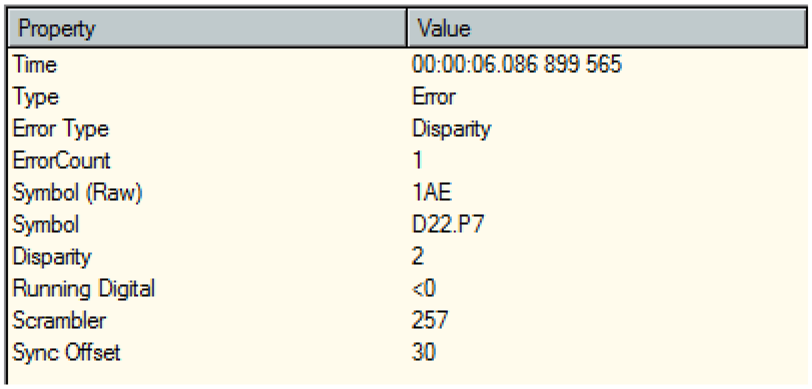
Raw



Property	Value
Time	00:00:05.633 940 659
Type	Raw
10B Count	18
data[0]	111111110100000000001011111111
data[1]	110000101111111111011000000000
data[2]	101111111111101000000000101111
data[3]	100000000010111011111111100000
data[4]	000010111111111101000011101111
data[5]	111110000000000101111111110110
Scrambler	1304
Sync Offset	339096450

This item can be found when the decoding algorithms can not decode the data in any way. This means the pickup logic can detect data on the line but it cannot decode it into toning or tokens.
10B Count is the number of supposed symbols
data[x] shows the raw data of 30 bits a row. the last row may be padded with zeros.

Error



Property	Value
Time	00:00:06.086 899 565
Type	Error
Error Type	Disparity
ErrorCount	1
Symbol (Raw)	1AE
Symbol	D22.P7
Disparity	2
Running Digital	<0
Scrambler	257
Sync Offset	30

This item shows a single symbol that has been recorded on the bus. There are a few types of errors:
Error Type

- InvalidSymbol : The Symbol is not a valid 10B symbol
- Disparity : The Symbol is a valid symbol but belongs to the wrong column in the 10B table
- Reserved : The Symbol is decoded as a reserved token. Control or Request.

ErrorCount
Subsequent errors are counted, after 10 errors the decoding algorithm will check if the bit and scrambler synchronization are still correct.

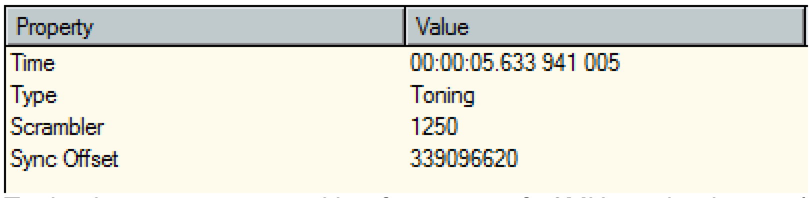
Symbol (Raw)
Shows the hex representation of the 10 bits of the symbol.

Symbol
If the symbols is valid, it shows the name of the symbol. ie: D6.4 or C0.

Disparity
Data and Request symbols can have a disparity (more or less ones as zeros).

Running Digital
All symbols with a disparity have two 10B versions. If this value is not 0 it shows which column of the symbol table is used.

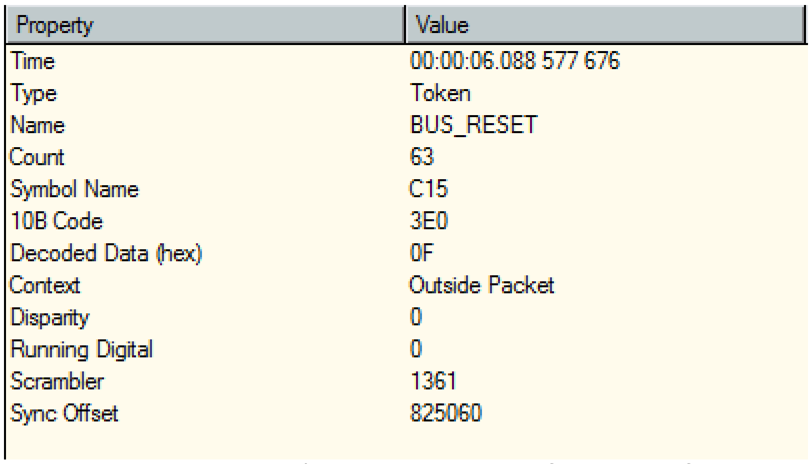
Toning



Property	Value
Time	00:00:05.633 941 005
Type	Toning
Scrambler	1250
Sync Offset	339096620

Toning is a square wave with a frequency of 50MHz and a duty cycle of 50%. If the software detects this, it is marked as a toning item.

Token



Property	Value
Time	00:00:06.088 577 676
Type	Token
Name	BUS_RESET
Count	63
Symbol Name	C15
10B Code	3E0
Decoded Data (hex)	0F
Context	Outside Packet
Disparity	0
Running Digital	0
Scrambler	1361
Sync Offset	825060

Tokens are descrambled/ decoded symbols. If a token is found repeatedly, only the first occurrence is saved and a counter is increased.

Name
For request tokens outside a packet boundary and control tokens, a friendly name is shown here. For data tokens (inside a packet boundary) it shows the hex representation of the decoded value.

Count
The number of repeating tokens.

Symbol Name
The name of the symbol. ie D13.4 or C13

10B Code
Shows the hex representation of the 10 bits of the symbol.

Decoded Data (hex)
Shows a hex representation of the descrambled value.

Context
Context shows if the symbol is found to be inside or outside a packet boundary.

Disparity
Data and Request symbols can have a disparity (more or less ones as zeros).

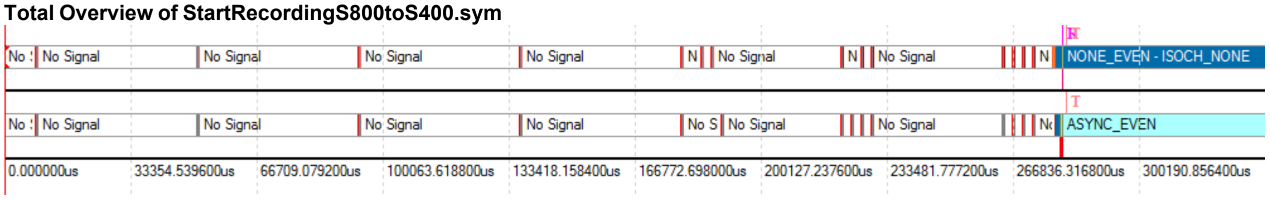
Running Digital
All symbols with a disparity have two 10B versions. If this value is not 0 it shows which column of the symbol table is used.

9.5. Examples

9.5.1. Start Connection

The example file StartConnectionS800toS400.sym is a recording of a plug-in event. It what happens when a connection is made on a IEEE1394 bus.
In this case the FireStealth was set to S400 and placed between a S400 and a S800 capable node.

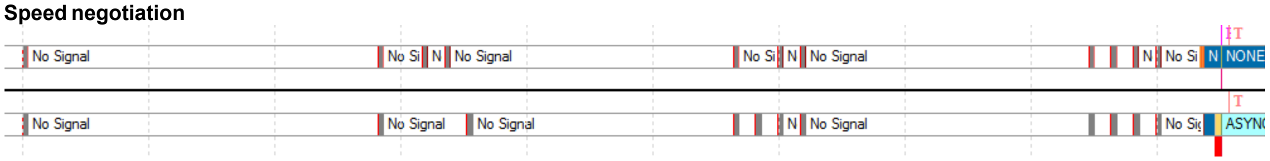
Total Overview of StartRecordingS800toS400.sym



First Part: Toning

In the IEEE1394-2008 specification this state is named the "Disconnected" State (P0). Toning and speed negotiation takes place here.

Speed negotiation



In the picture above we see the S400 capable node sending toning on the upper part of the screenshot and the S800 capable node on the lower part.
Toning is used for speed negotiation. For detailed workings please refer to chapter 14.8.3 Beta-mode speed negotiation of the IEEE1394-2008 specification.

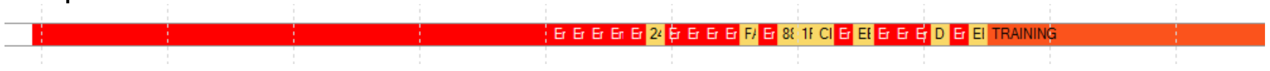
The image above shows a total of 8 speed codes, 4 for each node. If we represent the presence of a tone as a 1 and the absence as a 0, the top speed codes can be written as:
10000 - 10110 - 10110 - 11110
The bottom stream:
10000 - 10001 - 11110 - 11110

Decoded it means:
10000 : (Only Start of Toning)
10110 : Capable of S400
10001 : Capable of S800
11110 : Acknowledge, lets do S400

Second Part: Training

In the IEEE1394-2008 specification this state is named the "Untested" State (P11).

Startup effects

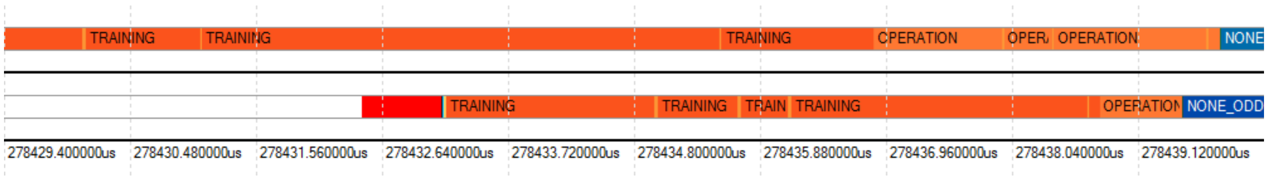


In the picture above we see the first few symbols of a new connection.

The errors are expected and caused by startup effects like clock synchronization between the two nodes making the connection. FireStealth itself also needs to synchronize to the bit stream on the bus. In most situations FireStealth will be able to lock on to the signal before the two nodes complete their training period. This can also be seen in the picture.

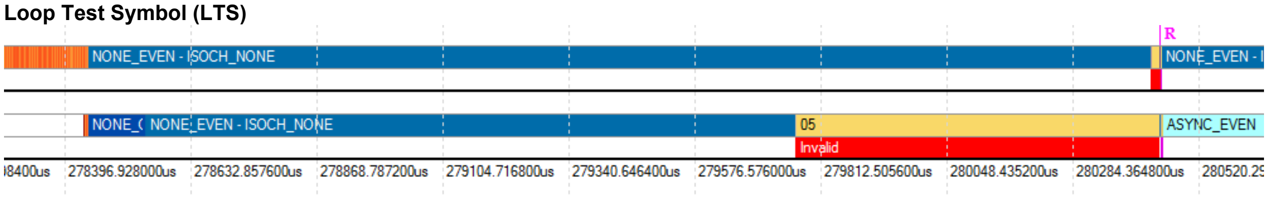
In this example the software receives correct data after approximately 30 symbols. The bottom stream was properly received after 17 symbols (in the example).

Training to Operation to Idle



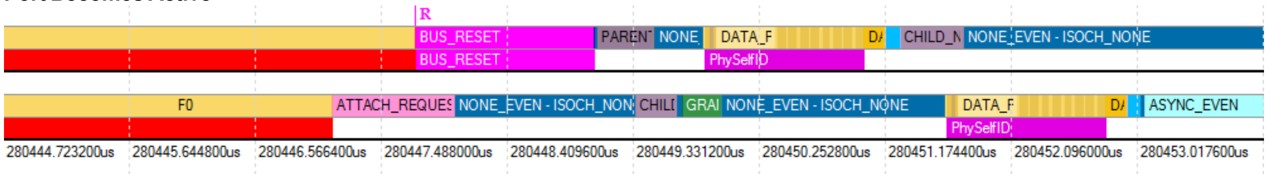
(For details about training please refer to 13.3.2.1 of the IEEE1394-2008 specification)
TRAINING/OPERATION symbols are used to synchronize bits/character and the scrambler. When a node is ready (in sync) it starts sending OPERATION symbols.
After both ports receive OPERATION symbols, they are synchronized and can start testing for a possible loop. described in chapter 14.7 of the IEEE1394 Specification.

Loop Test Symbol (LTS)



The picture above show something that looks like an invalid packet, but are in fact the transmission of Loop Test Symbols. It starts with a SPEEDb control followed by a number of DATA_PREFIX symbols. Please refer to 14.7.2 of the IEEE1394-2008 Specification for more info on the format of the symbol.

Port Becomes Active



After the port detects making it active will not result in a loop, it sends an ATTACH_REQUEST which is responded with a BUS_RESET control. (See Chapter 14.7.9 and 14.7.10 of the specification).

After the Bus reset, each node enters their "Tree identification state machine" (see chapter 16.4.6). Directly followed by the Self-identification state machine (see chapter 16.4.7).

Chapter 10. Generator

The Generator can generate large amounts of stream packets. It includes a powerful graphical editor to specify the sequences of stream packets that should be sent; up to 64 channels simultaneously.

You can do this by defining so-called slots. Each slot represents a number of cycles; a packet will be sent once per cycle. A sequence of packets can be loaded from file or generated from a template. For each sequence you can select various options such as speed, packet size and header fields, including erroneous values.

The Generator has two modes of operation: Isochronous and Mil1394 Streams. For Mil1394 Protocol users, the Generator can be set to Mil1394 mode from the device's Configuration Settings in the Settings Dialog, or in the Device Selection Dialog when you start the application.

Please continue reading the section that applies to your usage:

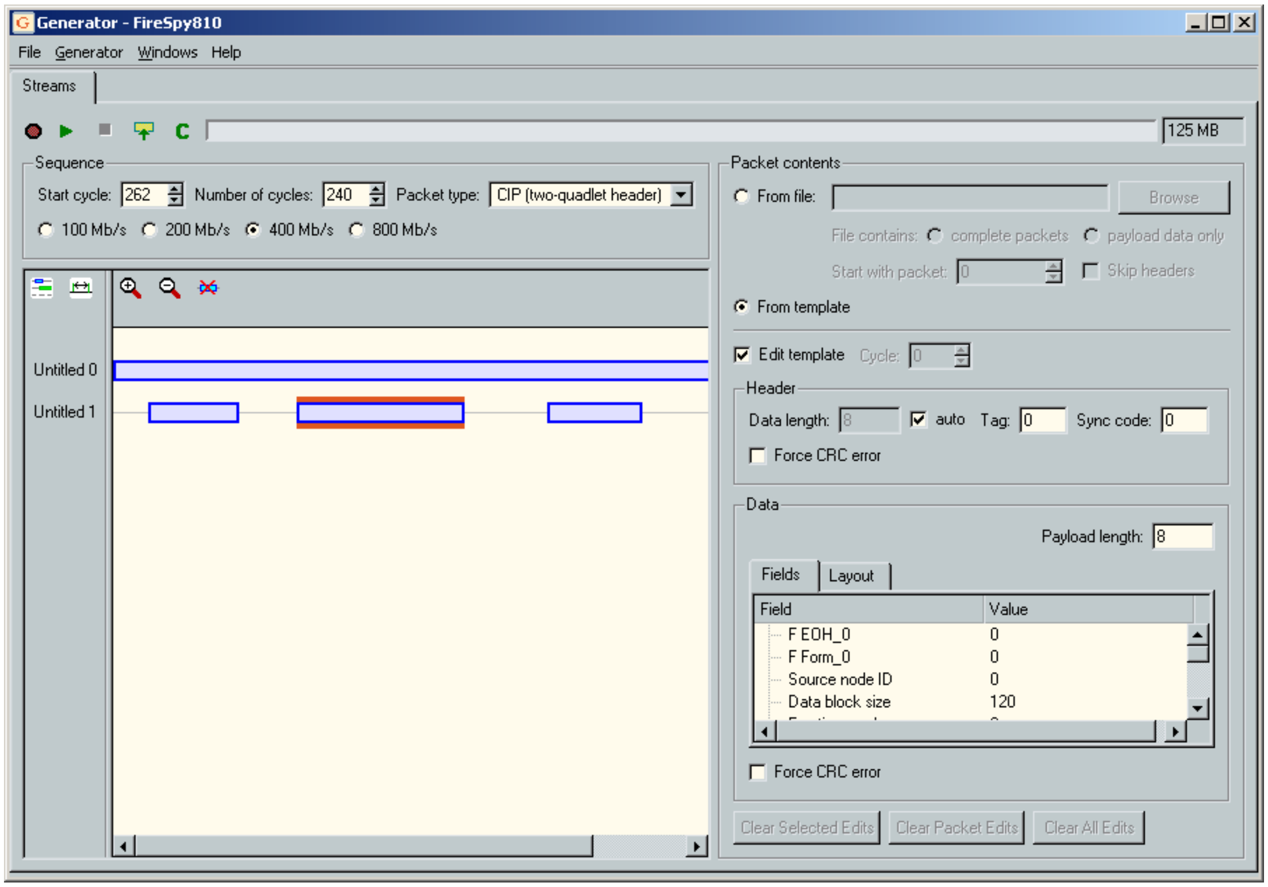
- Iso Stream Generator
- Mil1394 Stream Generator

10.1. Iso Stream Generator

10.1.1. How to use it

You can open the Generator by clicking on the Generator button in the main window, or select the Generator from the 'Analyzer' menu at the top of the main window or one of the other open windows.

When you open the Generator, a window like the following will appear. (although empty)



Start cycle: 262 Number of cycles: 240 Packet type: CIP (two-quadlet header)

100 Mb/s [✓ 400 Mb/s] 200 Mb/s 800 Mb/s

From file:

File contains: complete packets payload data only

Start with packet: 0 Skip headers

[✓ From template]

[✓ Edit template] Cycle: 0

Header

Data length: 8 [✓ auto] Tag: 0 Sync code: 0

Force CRC error

Data

Payload length: 8

Field	Value
F EOH_0	0
F Form_0	0
Source node ID	0
Data block size	120

Force CRC error

At the top of this window you can find the toolbar with buttons for starting, uploading and stopping the Generator. Before starting the Generator, you first have to upload the generator data to the Analyzer. This data can be specified in the "Streams" generator page, where you can define isochronous sequences. Sequences are indicated by blue rectangles on the time line of a slot. This can be seen in the picture above.

An isochronous sequence is specified by a a start cycle number, the length in number of cycles, the speed and contents of the isochronous packets to be sent during these cycles. During generation each sequence will result in the generation of an isochronous packet for each isochronous cycle at which the sequence is defined. The channel, speed and contents will correspond to the values specified for the slot the sequence is placed in. If more than one slot is defined for some isochronous cycle, then the order of isochronous cycles will depend on the specified priorities.

An isochronous sequence is placed in a 'slot'. In the example window above, a couple of slots are defined. The names of the slots are listed on the left side of the window and their 'time' lines are drawn to the right of their names. Before creating any sequences, at least one slot must be defined. To add a slot, open the slot definitions by selecting it from the Generator menu or pressing the corresponding toolbutton.

To create an isochronous sequence, click on the 'time' line of the slot you would like to create a sequence for. Initially, the sequence will occupy the complete slot. However, the left and right edges of the sequence can be moved by dragging them with the mouse. By doing this, the sequence start and end cycles can be changed.

Now you have defined the start cycle and length. Both start cycle and length can be changed at any time as long as they do not overlap another sequence in the same slot.

To change the packet contents for the packets in a specific sequence, click on the sequence rectangle to select it. The selection is indicated by a red surrounding. If a sequence is selected, the right side of the generator window will represent the packet contents of the selected sequence. It is now possible to fill the sequence with data from a template, file or manually.

Note that isochronous data only can be generated if cycle start packets are present on the bus.

10.1.2. Details

10.1.2.1. Menu

File
Open
With this command you can open an existing Analyzer generator file. By default these files have an extension of .fsg.

Save
With this command you can save the current generator data as a Analyzer generator file. By default these files have an extension of .fsg.
Generator data saved this way, can later be used again by invoking the 'Open' command of the 'File' menu. If no file name was given before, a file dialog is displayed to enter the file name and path.

Save As
This commands does the same as the 'Save' command, except that it will always display the file dialog, so that you can specify a file name and path.

Generator
Start
This command can be used to start the generator. If the Generator data has not yet been uploaded, a dialog box will popup to confirm Generator data upload. If successful, the generator will start generating the packets.

Stop
This command can be used to stop the generator. The generator data will remain in the Analyzer for an optional next run.

Upload
Upload the current Generator definition to the Analyzer.

Clear
Clear the uploaded Generator definition from the Analyzer.

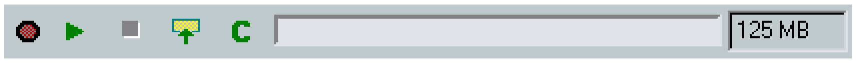
Slot Definitions
With this command, the slot definition editor can be opened. For a description of how it works, please refer to the Slot Definitions section.

Frame Settings
With this command, the frame settings dialog can be opened. For a description of how it works, please refer to the Frame Settings section.

Windows
From the 'Windows' menu you can open one of the other windows of the Analyzer.

10.1.2.2. Toolbar

The toolbar contains respectively the following items:



Generator active indicator
This red indicator lights when the isochronous Generator is active. The Generator is active when it is started and has not stopped yet. An active isochronous Generator does not always mean that it is actually generating packets, because it can be waiting for the next isochronous cycle start to generate the next packet or it can be in the middle of cycles for which no isochronous packets are defined.

Start isochronous generator
Clicking this button will activate the isochronous Generator. The uploaded isochronous packets will be generated for the defined cycle, where cycle 0 corresponds to the current isochronous cycle at the moment the Generator is started. If the isochronous generator data has been changed since the last upload, the user will first be asked if the changed data must be uploaded again.

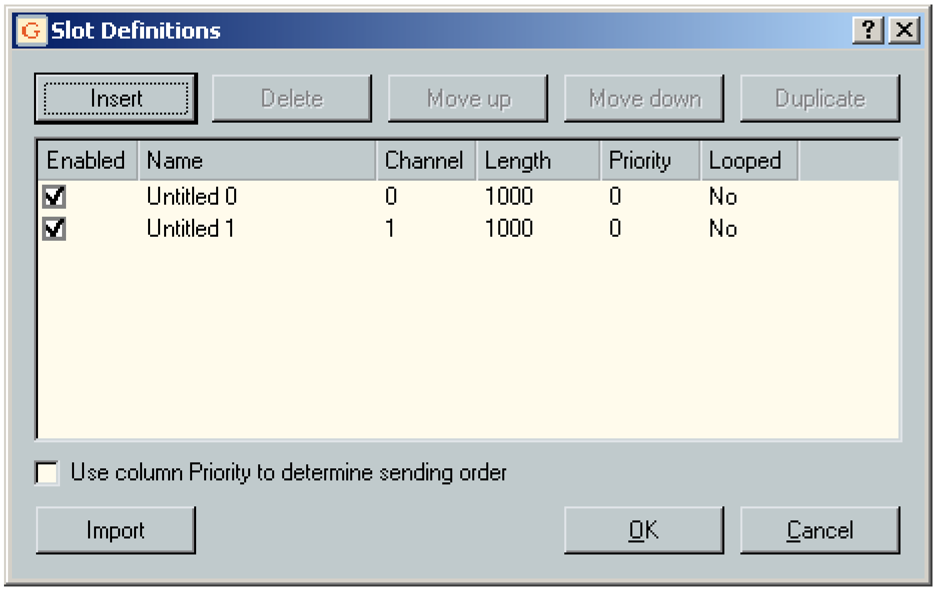
Stop isochronous generator
Clicking this button will stop (deactivate) the isochronous Generator.

Upload isochronous generator data to Analyzer
Clicking this button will result in the uploading of the isochronous generator data to the Analyzer. During the upload, the packets are actually created. This means that some errors may be detected. If packet data is to be read from a file for instance, the file may not be found. Or the generated packet length may be larger than specified in the 'Max Data Length' box. In both cases the user will be asked if to abort the upload or to ignore the error.

Clear isochronous generator memory
Clicking this button will clear the isochronous data, which is uploaded to the Analyzer generator memory. Before new data can be uploaded, the old must be cleared. When uploading new data the user will be asked if the old data should be cleared (if there is already some uploaded isochronous generator data), but you could also clear the memory manually before uploading new data. The generator memory also needs to be cleared when changing memory sizes, see Settings for more information about changing the generator memory size.

10.1.2.3. Slot Definitions

You can add and remove slots and change the properties of slots (like associated channel number and offset time), by clicking the 'Select slots' button. This button can be found in the upper left corner. When clicked, a dialog shows up. In this dialog you can change slot properties and add or remove slots. Using the import button in this dialog, you can read the slot information from a file (comma separated values). The Analyzer supports a maximum of 61 slots except the FireSpy 400b and 800 which support a maximum of 31 slots.



Enabled	Name	Channel	Length	Priority	Looped
✓	Untitled 0	0	1000	0	No
✓	Untitled 1	1	1000	0	No

☐ Use column Priority to determine sending order

Buttons
The slot definitions window contains the following buttons

Insert
This button can be used to insert a new slot definition.

Delete
This button can be used to delete the currently selected slot.

Move up
This button can be used to move the currently selected slot one line up. This means that the priority will be increased by one.

Move down
This button can be used to move the currently selected slot one line down. This means that the priority will be decreased by one.

Duplicate
This button can be used to insert a copy of the currently selected slot definition.

Import
This button can be used to import slot definitions from a file. See Mil1394SlotExample.csv in the example directory for an example of such a file.

Ok
This button can be used to commit all changes made to the slot definitions.

Cancel
This button can be used to close the dialog, clearing all changes made to the slot definitions.

Slot Definition table
The center of the dialog contains a table with all the slot definitions that are defined. Each slot is defined by the following fields.

Enabled
If this field is checked, this slot definition is actually used when generating the packets. If this field is not checked, this slot definition is discarded on packet generation.

Name
This can be given a user-defined name for the slot definition. This name will also show up in the sequence editor on the main window of the generator.

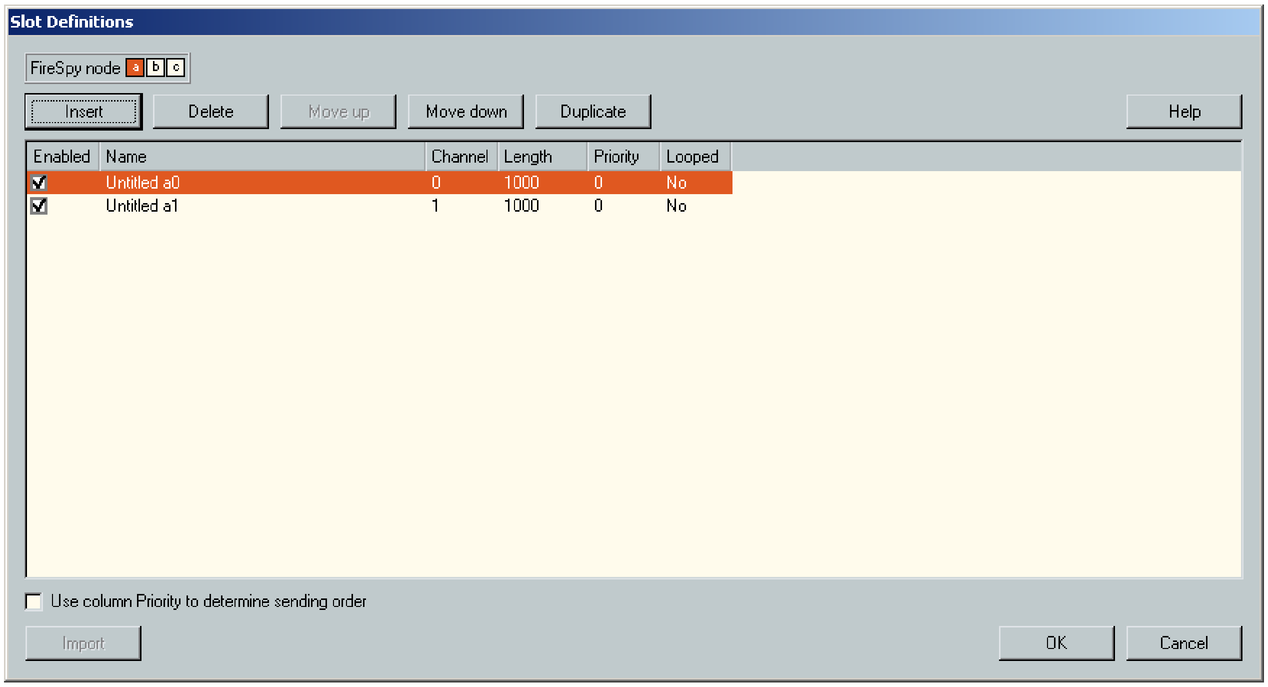
Channel
The channel to use for all packets for this slot.

Length
The number of cycles represented in the slot.

Priority
If the check-box "Use column priority for sending order" is checked, this determines sending order. Otherwise, the channel numbers.

Looped
If set to true, the generator will continuously repeat sending the packets for this slot when the end of the slot is reached.

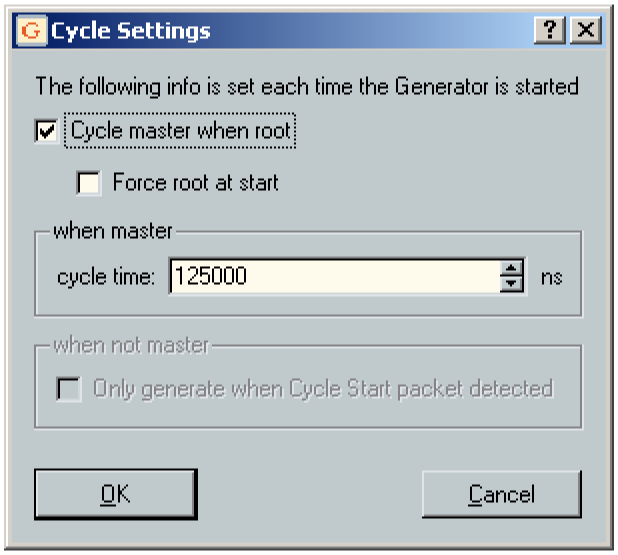
Triple Analyzers
For triple Analyzers the slot definitions dialog contains an additional node selector as can be seen in the following picture. Only the slots from the currently selected node are shown. There is a similar node selector on the Generator window; only the streams from the currently selected node are shown, but all nodes are used when starting the generator.



Enabled	Name	Channel	Length	Priority	Looped
✓	Untitled a0	0	1000	0	No
✓	Untitled a1	1	1000	0	No

☐ Use column Priority to determine sending order

10.1.2.4. Settings


Cycle Settings

The following info is set each time the Generator is started

[✓] Cycle master when root

[ ] Force root at start

when master

cycle time: 125000 ns

when not master

[ ] Only generate when Cycle Start packet detected

The Cycle Settings dialog can be used to change the following settings:

Cycle master when root
If the Analyzer is the root node and this setting is enabled, it will also act as Cycle Master

Force Root at Start
If this setting is enabled, the Analyzer will try to become the root node when the Generator is started

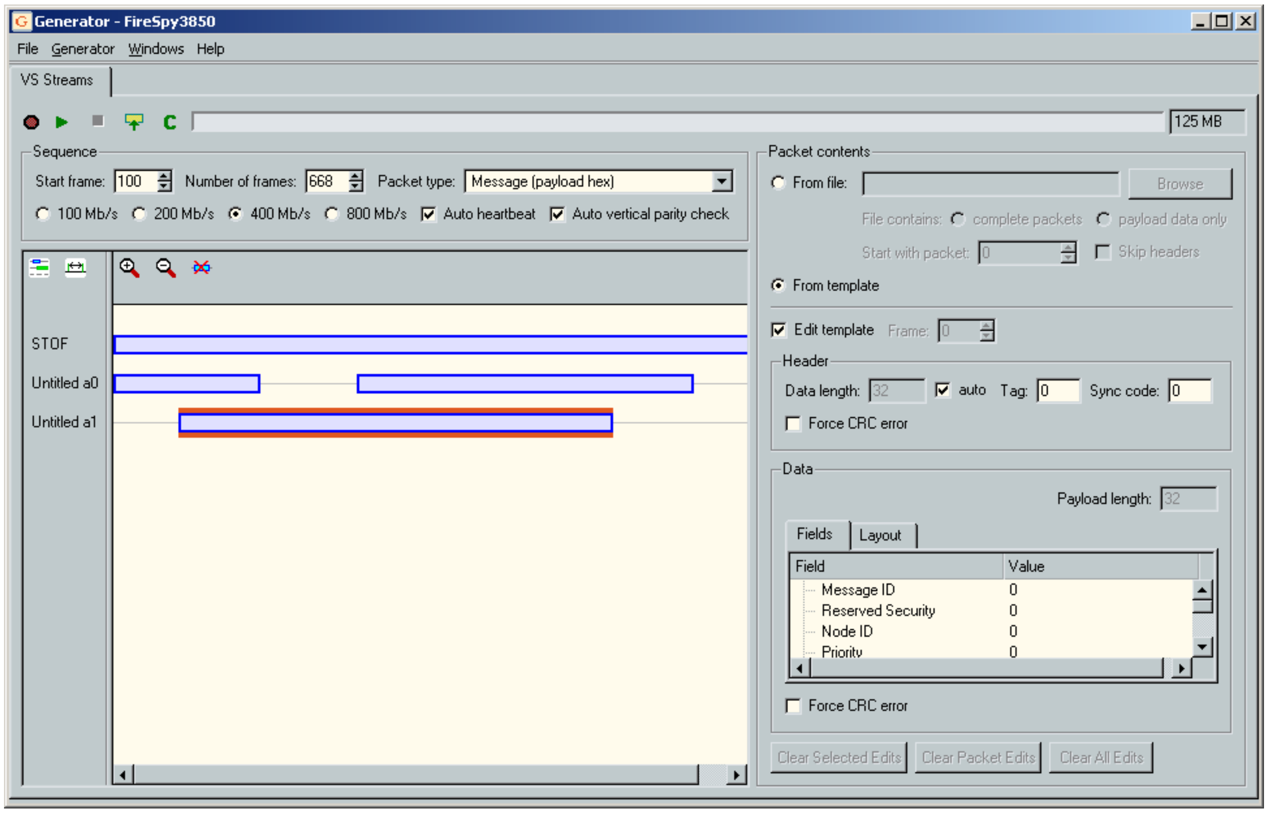
Cycle Time
If the Analyzer is cycle master, this setting determines the cycle time in nano seconds.

10.2. Mil1394 Stream Generator

10.2.1. How to use it

You can open the Generator by clicking on the Generator button in the main window, or select the Generator from the 'Analyzer' menu at the top of the main window or one of the other open windows.

When you open the Generator, a window like the following will appear. (although empty)



Sequence

Start frame: 100 Number of frames: 668 Packet type: Message (payload hex)

100 Mb/s [✓ 400 Mb/s] 200 Mb/s 800 Mb/s [✓ Auto heartbeat] [✓ Auto vertical parity check]

Packet contents

From file:

File contains: complete packets payload data only

Start with packet: 0 Skip headers

[✓ From template]

[✓ Edit template] Frame: 0

Header

Data length: 32 [✓ auto] Tag: 0 Sync code: 0

Force CRC error

Data

Payload length: 32

Field	Value
Message ID	0
Reserved Security	0
Node ID	0
Priority	0

Force CRC error

At the top of this window you can find the toolbar with buttons for starting, uploading and stopping the Generator. Before starting the Generator, you first have to upload the generator data to the Analyzer. This data can be specified in the "Mil1394 Streams" generator page, where you can define isochronous sequences. Sequences are indicated by blue rectangles on the time line of a slot. This can be seen in the picture above.

An isochronous sequence is specified by a a start frame number, the length in number of frames, the speed, a setting for auto heartbeat, a setting for auto VPC calculation and contents of the isochronous packets to be sent during these frames. During generation each sequence will result in the generation of an isochronous packet for each isochronous frame at which the sequence is defined. The channel, speed and contents will correspond to the values specified for the slot the sequence is placed in. If more than one slot is defined for some isochronous frame, then the order of isochronous frames will depend on the specified priorities.

An isochronous sequence is placed in a 'slot'. In the example window above, a couple of slots are defined. The names of the slots are listed on the left side of the window and their 'time' lines are drawn to the right of their names. Before creating any sequences, at least one slot must be defined. To add a slot, open the slot definitions by selecting it from the Generator menu or pressing the corresponding toolbutton.

To create an isochronous sequence, click on the 'time' line of the slot you would like to create a sequence for. Initially, the sequence will occupy the complete slot. However, the left and right edges of the sequence can be moved by dragging them with the mouse. By doing this, the sequence start and end frames can be changed.

Now you have defined the start frame and length. Both start frame and length can be changed at any time as long as they do not overlap another sequence in the same slot.

To change the packet contents for the packets in a specific sequence, click on the sequence rectangle to select it. The selection is indicated by a red surrounding. If a sequence is selected, the right side of the generator window will represent the packet contents of the selected sequence. It is now possible to fill the sequence with data from a template, file or manually.

10.2.2. Details

10.2.2.1. Menu

File

Open
With this command you can open an existing Analyzer generator file. By default these files have an extension of .fsg.

Save
With this command you can save the current generator data as a Analyzer generator file. By default these files have an extension of .fsg.
Generator data saved this way, can later be used again by invoking the 'Open' command of the 'File' menu. If no file name was given before, a file dialog is displayed to enter the file name and path.

Save As
This commands does the same as the 'Save' command, except that it will always display the file dialog, so that you can specify a file name and path.

Generator
Start
This command can be used to start the generator. If the Generator data has not yet been uploaded, a dialog box will popup to confirm Generator data upload. If successful, the generator will start generating the packets.

Stop
This command can be used to stop the generator. The generator data will remain in the Analyzer for an optional next run.

Upload
Upload the current Generator definition to the Analyzer.

Clear
Clear the uploaded Generator definition from the Analyzer.

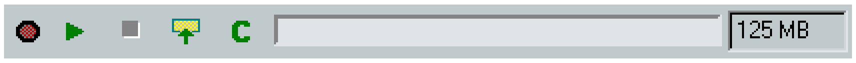
Slot Definitions
With this command, the slot definition editor can be opened. For a description of how it works, please refer to the Slot Definitions section.

Frame Settings
With this command, the frame settings dialog can be opened. For a description of how it works, please refer to the Frame Settings section.

Windows
From the 'Windows' menu you can open one of the other windows of the Analyzer.

10.2.2.2. Toolbar

The toolbar contains respectively the following items:



Generator active indicator
This red indicator lights when the isochronous Generator is active. The Generator is active when it is started and has not stopped yet. An active isochronous Generator does not always mean that it is actually generating packets, because it can be waiting for the next isochronous cycle start to generate the next packet or it can be in the middle of cycles for which no isochronous packets are defined.

Start isochronous generator
Clicking this button will activate the isochronous Generator. The uploaded isochronous packets will be

generated for the defined cycle, where cycle 0 corresponds to the current isochronous cycle at the moment the Generator is started. If the isochronous generator data has been changed since the last upload, the user will first be asked if the changed data must be uploaded again.

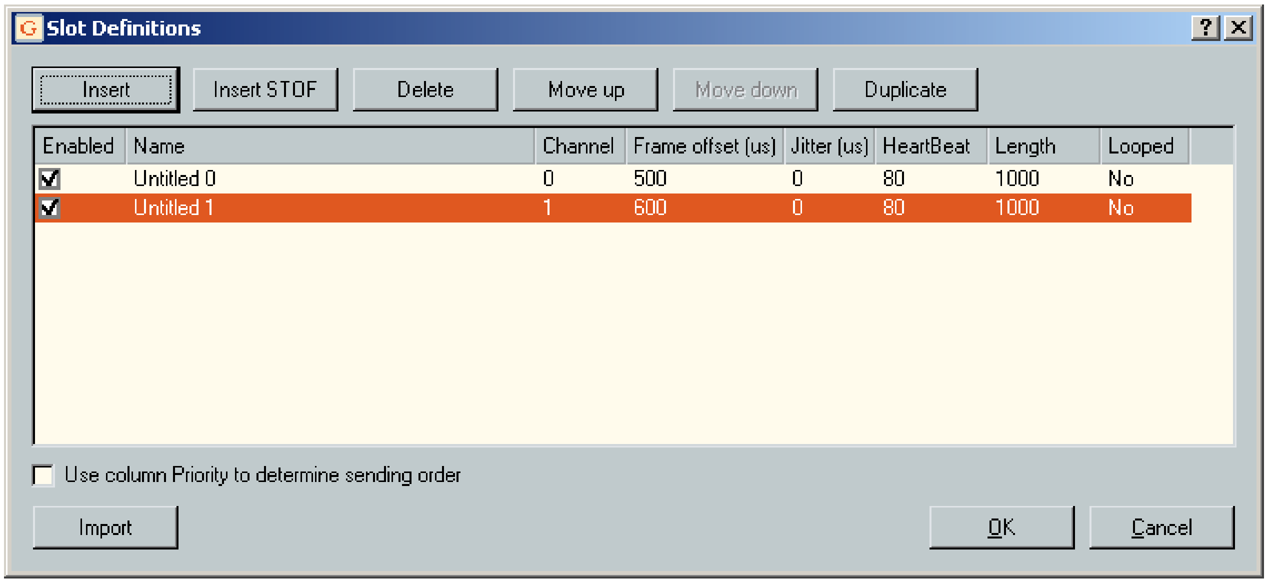
Stop isochronous generator
Clicking this button will stop (deactivate) the isochronous Generator.

Upload isochronous generator data to Analyzer
Clicking this button will result in the uploading of the isochronous generator data to the Analyzer. During the upload, the packets are actually created. This means that some errors may be detected. If packet data is to be read from a file for instance, the file may not be found. Or the generated packet length may be larger than specified in the 'Max Data Length' box. In both cases the user will be asked if to abort the upload or to ignore the error.

Clear isochronous generator memory
Clicking this button will clear the isochronous data, which is uploaded to the Analyzer generator memory. Before new data can be uploaded, the old must be cleared. When uploading new data the user will be asked if the old data should be cleared (if there is already some uploaded isochronous generator data), but you could also clear the memory manually before uploading new data. The generator memory also needs to be cleared when changing memory sizes, see Settings for more information about changing the generator memory size.

10.2.2.3. Slot Definitions

You can add and remove slots and change the properties of slots (like associated channel number and offset time), by clicking the 'Select slots' button. This button can be found in the upper left corner. When clicked, a dialog shows up. In this dialog you can change slot properties and add or remove slots. Using the import button in this dialog, you can read the slot information from a file (comma separated values). The Analyzer supports a maximum of 61 slots except the FireSpy 400b and 800 which support a maximum of 31 slots.



Enabled	Name	Channel	Frame offset (us)	Jitter (us)	HeartBeat	Length	Looped
✓	Untitled 0	0	500	0	80	1000	No
✓	Untitled 1	1	600	0	80	1000	No

☐ Use column Priority to determine sending order

Buttons
The slot definitions window contains the following buttons

Insert
This button can be used to insert a new slot definition.

Insert STOF
This button can be used to insert a new slot definition with all settings set to the defaults for STOF packets.

Delete
This button can be used to delete the currently selected slot.

Move up
This button can be used to move the currently selected slot one line up. This means that the priority will be increased by one.

Move down
This button can be used to move the currently selected slot one line down. This means that the priority will be decreased by one.

Duplicate
This button can be used to insert a copy of the currently selected slot definition.

Import
This button can be used to import slot definitions from a file. See Mil1394SlotExample.csv in the example directory for an example of such a file.

Ok
This button can be used to commit all changes made to the slot definitions.

Cancel
This button can be used to close the dialog, clearing all changes made to the slot definitions.

Slot Definition table
The center of the dialog contains a table with all the slot definitions that are defined. Each slot is defined by the following fields.

Enabled
If this field is checked, this slot definition is actually used when generating the packets. If this field is not checked, this slot definition is discarded on packet generation.

Name
This can be given a user-defined name for the slot definition. This name will also show up in the sequence editor on the main window of the generator.

Channel
The channel to use for all packets for this slot.

Frame Offset
The offset time relative to the start of the frame (STOF) in microseconds at which al packets in this slot should be sent. (one per frame)

Jitter
Not implemented yet

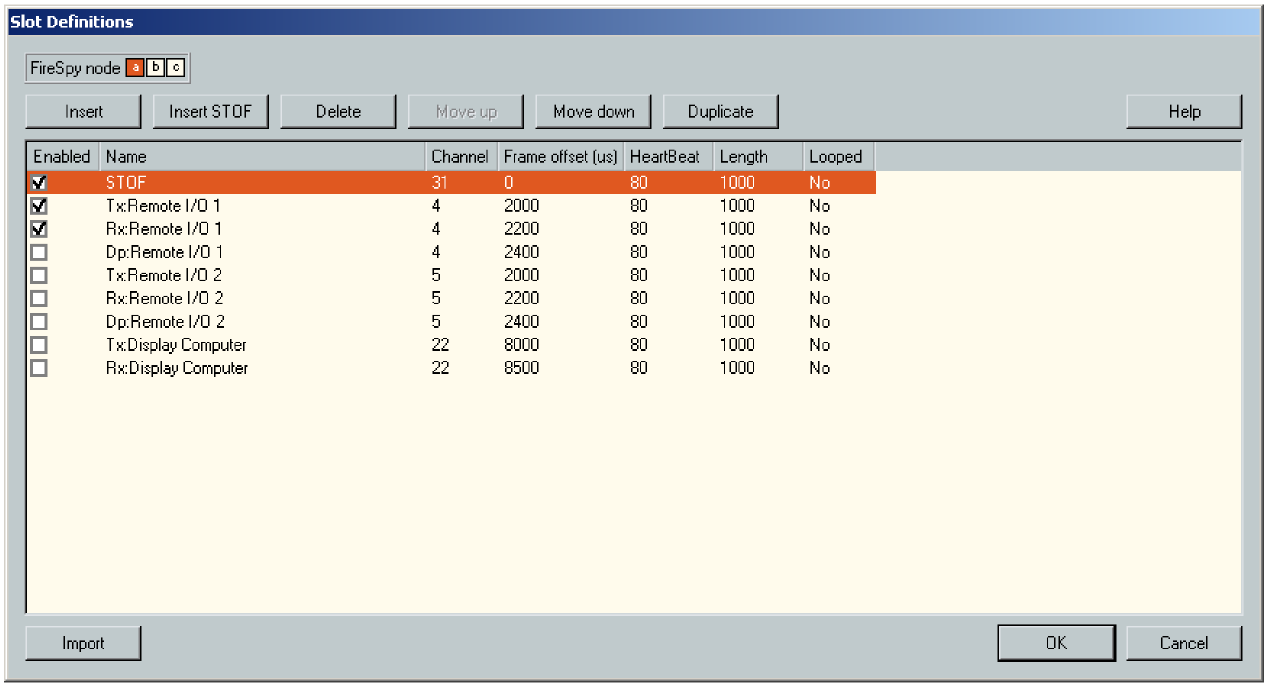
Heartbeat
For a selected sequence you can select the 'Auto heartbeat' option in the main window. When selected, the heartbeat value will increment for each frame. Note that these values are not shown in the packets.

Length
The number of cycles represented in the slot.

Looped
If set to true, the generator will continuously repeat sending the packets for this slot when the end of the slot is reached.

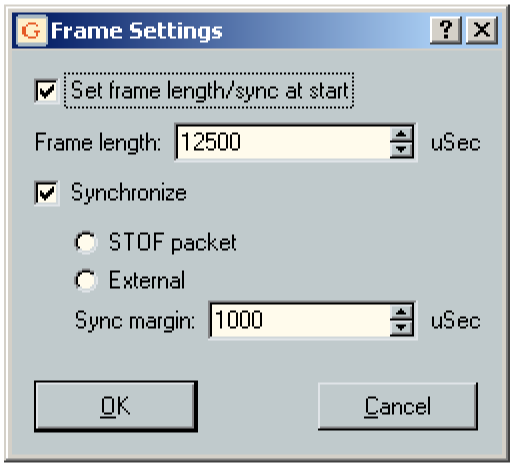
Triple Analyzers
For triple Analyzers the slot definitions dialog contains an additional node selector as can be seen in the following picture. Only the slots from the currently selected node are shown. There is a similar node selector on the Generator window; only the streams from the currently selected node are shown, but all

nodes are used when starting the generator.



Enabled	Name	Channel	Frame offset (us)	HeartBeat	Length	Looped
✓	STOF	31	0	80	1000	No
✓	Tx:Remote I/O 1	4	2000	80	1000	No
✓	Rx:Remote I/O 1	4	2200	80	1000	No
☐	Dp:Remote I/O 1	4	2400	80	1000	No
☐	Tx:Remote I/O 2	5	2000	80	1000	No
☐	Rx:Remote I/O 2	5	2200	80	1000	No
☐	Dp:Remote I/O 2	5	2400	80	1000	No
☐	Tx:Display Computer	22	8000	80	1000	No
☐	Rx:Display Computer	22	8500	80	1000	No

10.2.2.4. Settings



[✓] Set frame length/sync at start
Frame length: 12500 uSec
[✓] Synchronize
STOF packet
External
Sync margin: 1000 uSec

The Frame Settings dialog can be used to change the following settings:

Set frame length/sync start
If this option is enabled, the generator will synchronize its frame timing when started. The following options determine how synchronization will take place.

Frame Length
If the synchronize option is not enabled, internal frame timing is used. This setting determines the length of a frame in micro seconds. The maximum frame length is 128000 micro seconds expect for the FireSpy 400b and 800 which have a maximum frame length of 32000 micro seconds. The minimum frame length which the Analyzer except is 7500 micro seconds.

Synchronize
If this option is enabled, the frame timing is synchronized to either STOF packets received on the bus or an external source.

10.2.3. Mil1394 Example

We will explain the Generator and Recorder function with regard to the Mil1394-protocol using an example file.
To run the examples, connect the FireSpy800 and make sure the Mil1394-protocol key is installed.

Load example file
Open the Generator window and select File->Open. Browse to the Mil1394GenExample.fgs file in the examples folder and open it.
In the Generator window you see at the left a graphical impression of what the Generator will generate. Each colored block specifies a sequence of stream packets for the corresponding time slot.
When clicking on such a sequence, it will be selected and the upper part of the window shows the properties of this sequence. You can see for instance the start frame and number of frames to be generated.
In the example, 1000 STOF packets will be generated. On channel 4 and 5 1000 packets will be generated with an offset of 2000 and 3000 usec form the STOF packet recpectively. On channel 22 a sequence of 370 packets, followed by a pause and then 460 packets will be generated with an offset of 8000 uSec from the STOF packet.

Edit packets
When a sequence is selected you will be able to choose if the packet data comes from a file or from a template packet. In this example, it comes from a template packet.
When the 'Edit template' checkbox is checked, the packet shown is the template packet. This packet can be changed. When the checkbox is not checed, individual packets of the selected sequence can be editted. Using the 'Frame' box you can select the packet to be editted. The changed values will be colored red and they overrule the values from the template packet.
When packet data comes from a file, there is no template packet, but individual packets can still be editted, overruling the values from the file.

Heartbeat
For a selected sequence you can select the 'Auto heartbeat' option. When selected, the heartbeat value will increment for each frame. Note that these values are not shown in the packets.

Vertical parity check
For a selected sequence you can select the 'Auto vertical parity check' (VPC) option. When selected, the VPC value will be calculated for each packet in that sequence. Note that these values are not shown in the packets.

Start/stop generate
Before the Generator can be started, you first need to upload the information. Then the Generator can be started and stopped. The buttons to do this can be found in the upper left corner of the window. When the example is started, the 1000 STOF packets will be generated on channel 31 and at the same time the 1000 packets on channels 4 and 5 and the 370 plus 460 packets on channel 22 will be generated inbetween the STOF packets with the specified offsets. In total 1000 frames will be generated and the Generator will stop after that (it takes 12.5 sec). If one or more channels are in loop-mode, the Generator continues generating until stopped by the user.

Recorder example
An recording of the result of this generated example has been saved into Mil1394RecExample1.fsr. For more information about the recorder, please see the manual.

Chapter 11. Scriptor

Using the Analyzer Scriptor feature, it is possible to send/receive packets, extract fields from a packet and display the results on a Control Panel by using different kinds of indicators. The Scriptor window is divided in four parts, each represented by a tab page.

- Script Editor
- Control Panel
- Data Editor
- Script Properties

For a quick impression of the features of the Scriptor, see the following examples:

- **simpleSend** : A script showing how to define a packet using the built-in macros and how to send the packet multiple times.
- **simConfRom** : A script showing how to handle the configuration ROM read requests using a data file such that the presence of a configuration ROM is simulated.
- **timedSend** : A script showing how to define a packet using the built-in macros and how to send the packet at user-controlled frame offset times. This example makes use of the Mil1394 Protocol and will only work if Mil1394 mode is enabled.
- **erroneousSend** : A script that allows the user to send packets that contain errors. The control panel is used to select what kind of error the packet should contain and to set some other options.
- **Mil1394MessageMonitor** : A script for the Mil1394 Protocol that displays signal values for a specific message ID and channel number. Up to 20 signals can be configured and displayed simultaneously. This example will only work in Mil1394 mode.

The Script Editor can be used to enter scripts for the Analyzer device. To make the Scriptor functionality as intuitive as possible, assistance is provided by showing relevant property editors while entering the script. These property editors are shown to the right of the Script Editor and can be used to fill in function parameters, loop conditions and much more.

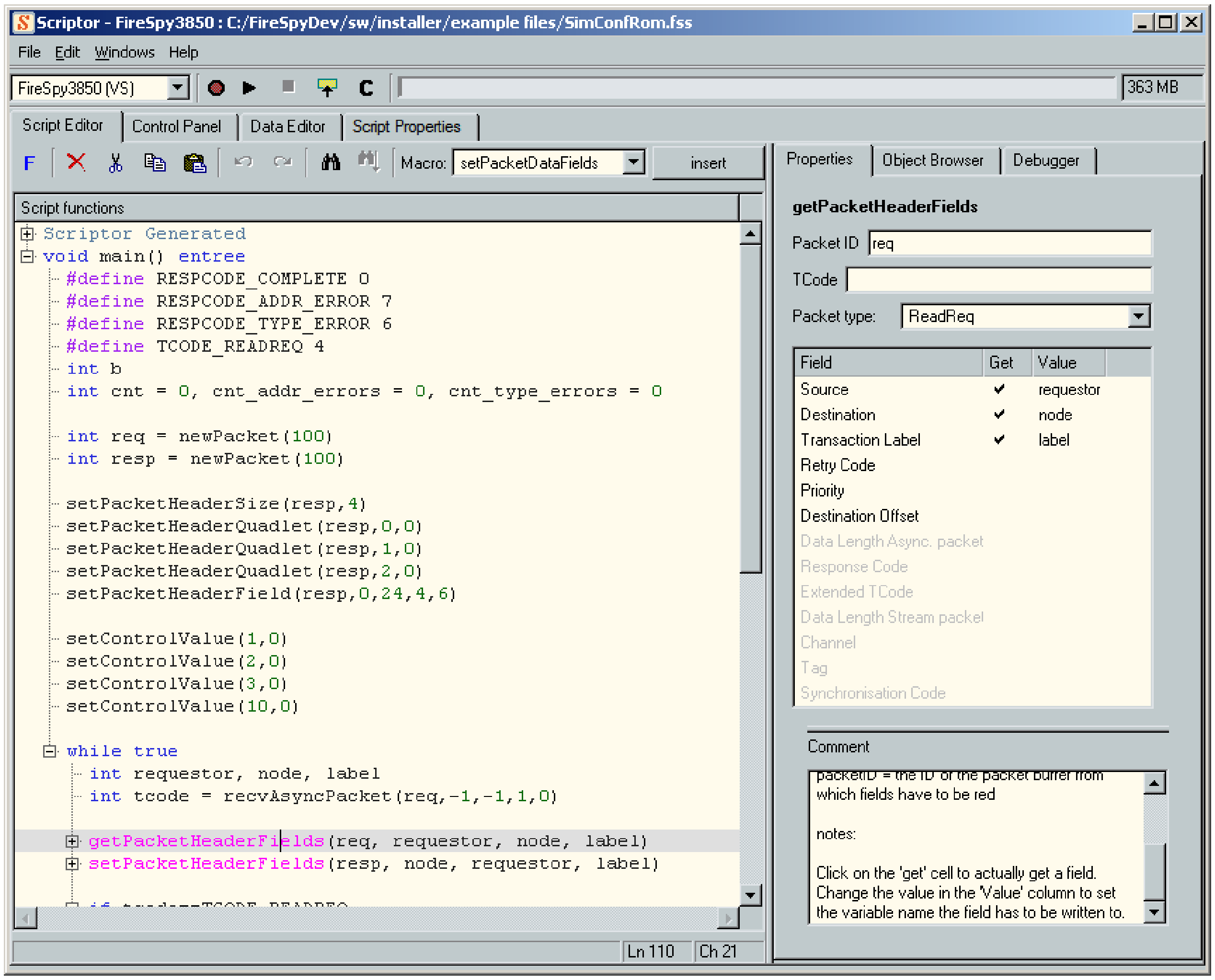
The Control Panel contains a wide range of indicators for showing the values of user-defined parameters in a graphical way. Alarms can be set by specifying parameter conditions. It is also possible to send values from the control panel to a running script, thereby controlling it.

The Data Editor contains different kinds of editors for entering constant data which can be used by the script to write to a packet.

The Scriptor can be used to control almost anything of the Analyzer, including packet sending and reading/writing of Phy registers. For a reference of the script language and API, see the Language reference and Function List section of the manual.

11.1. Main Window

After opening the Scriptor, a window will be shown that contains a toolbar at the top and three tabs below it: Script Editor, Control Panel and Data Editor. These tabs will be explained in the following sections. The picture below shows the Scriptor main window.



```
Scriptor Generated
void main() entree
    #define RESPCODE_COMPLETE 0
    #define RESPCODE_ADDR_ERROR 7
    #define RESPCODE_TYPE_ERROR 6
    #define TCODE_READREQ 4
    int b
    int cnt = 0, cnt_addr_errors = 0, cnt_type_errors = 0

    int req = newPacket(100)
    int resp = newPacket(100)

    setPacketHeaderSize(resp,4)
    setPacketHeaderQuadlet(resp,0,0)
    setPacketHeaderQuadlet(resp,1,0)
    setPacketHeaderQuadlet(resp,2,0)
    setPacketHeaderField(resp,0,24,4,6)

    setControlValue(1,0)
    setControlValue(2,0)
    setControlValue(3,0)
    setControlValue(10,0)

    while true
        int requestor, node, label
        int tcode = recvAsyncPacket(req,-1,-1,1,0)

        getPacketHeaderFields(req, requestor, node, label)
        setPacketHeaderFields(resp, node, requestor, label)

        if tcode==TCODE_READREQ
```

getPacketHeaderFields

Packet ID: req
TCode:
Packet type: ReadReq

Field	Get	Value
Source	✓	requestor
Destination	✓	node
Transaction Label	✓	label
Retry Code		
Priority		
Destination Offset		
Data Length Async. packet		
Response Code		
Extended TCode		
Data Length Stream packet		
Channel		
Tag		
Synchronisation Code		

Comment

packetID = the ID of the packet buffer from which fields have to be red

notes:

Click on the 'get' cell to actually get a field.
Change the value in the 'Value' column to set the variable name the field has to be written to.

Getting help

When editing a script, one may press the F1 key to get help about the word at the current cursor position. Additionally, when moving the mouse to an identifier and leaving it there for approximately one second, a tooltip pops up that contains the declaration information for that identifier if known.

File menu

Open

With this command you can open an existing script file. By default these files have an extension of ".fss".

Save

With this command you can save the current script to file. By default these files have an extension of ".fss". Scripts saved this way, can later be used again by invoking the 'Open' command from the 'File' menu. Both the script and the control panel are saved to file for later use.

Save As

This commands does the same as the 'Save' command, except that it will always display the file dialog, so that you can specify a (different) file name and path.

Import

With this command it is possible to import the textual part of a script from a text file. The file should follow the same hierarchical lay out as the tree view of the Scriptor. Indention should be done by using 'tab' characters, NOT spaces. The best way to learn the format is to use the export function on an existing script and see what it looks like.
Note that if the script contains macro's (fillPacket, setPacketHeaderFields, ...) then the macro itself is removed and the generated script is inserted at the position of the macro call.

Export

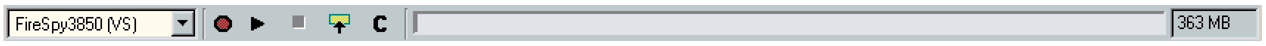
This command enables exporting the text part of a script to an external file. Two formats are currently support: (*.txt) and (*.html). Scripts exported to a text file can be imported again with the import command. However, when doing so, the Data and Control Panel are lost. The other format exports the text part of a script to a web page. The web page contains the text of the script in highlighted form.

Edit menu
The contents of this menu depend on the tab that is selected. All tabs have their own edit menu. These menus contain the windows-standard edit functions like copy, paste, undo and redo. In addition, this menu also contains some editor dependend items like function insertion for the Script Editor and control insertion for the Control Panel.

Windows menu
This menu contains commands to move to other Analyzer windows like the Commander or the Recorder. The menu items are self-explanatory.

Help menu
This menu contains commands for accessing different sections of the manual. The menu items are self-explanatory.

Toobar



Looking from the left to the right, the main toolbar contains the following items:

Selected Target Device
This combo box contains a list of all devices a script can be created for. The name of the device indicates whether the device is currently configured in Mil1394 mode or not and whether the device is online or offline. If developing for an offline device, the script can be compiled but not uploaded.

Scriptor active indicator
This led control is turned on (red light) whenever a script is being executed on the Analyzer.

Start script
This button starts the current script. If the current script is not yet compiled and uploaded to the Analyzer, this is done automatically before execution.

Stop script
If a script is currently running on the connected Analyzer, this button stops the script.

Upload script data to Analyzer
This button uploads the current script to a connected Analyzer. If the script is not yet compiled, this is done on the fly.

Clear scriptor memory
This button clears the compiled script from the memory of a connected Analyzer if present. Note that the script will remain loaded in the script editor.

Scriptor Memory bar
This gray thermometer indicates how much of the Scriptor memory is in used and how much is still available to the running script. The number on the right side displays the amount of Analyzer memory allocated to the Scriptor. Use the global settings window to allocate a different amount of memory to the scriptor.

11.2. Script Editor

The Script Editor tab is used for editing and debugging a user-defined script. The tab page is divided in two parts. The left part is a textual script editor, which shows the current script. The right part contains a

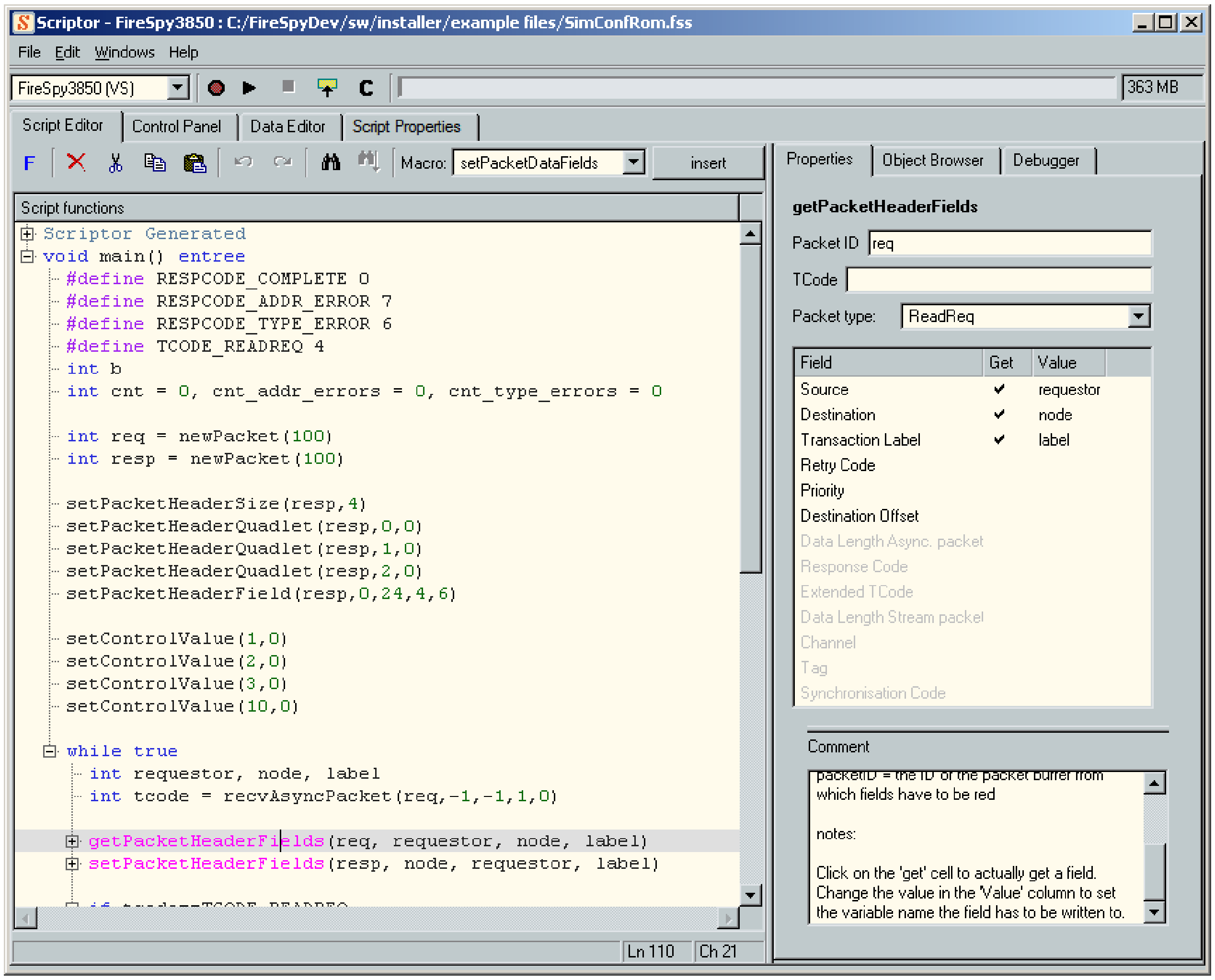
Property Editor, an Object Browser and a Debugger.

The Property Editor is used to edit the properties of the script source-code line that is currently selected.

The Object Browser contains an hierarchical overview of the processes, functions and variables that are defined in the script source code. It is mainly used to provide an overview of the script and easier navigation in the source code.

The Debugger is used to analyse the behavior of a running script.

The Script Editor is shown in the picture below and displays a hierarchical representation of the current script, which can be edited by changing individual lines of script code. This view behaves much similar to a text editor, however, there are some differences. These differences will be explained below.



```
Scriptor Generated
void main() entree
  #define RESPCODE_COMPLETE 0
  #define RESPCODE_ADDR_ERROR 7
  #define RESPCODE_TYPE_ERROR 6
  #define TCODE_READREQ 4
  int b
  int cnt = 0, cnt_addr_errors = 0, cnt_type_errors = 0

  int req = newPacket(100)
  int resp = newPacket(100)

  setPacketHeaderSize(resp,4)
  setPacketHeaderQuadlet(resp,0,0)
  setPacketHeaderQuadlet(resp,1,0)
  setPacketHeaderQuadlet(resp,2,0)
  setPacketHeaderField(resp,0,24,4,6)

  setControlValue(1,0)
  setControlValue(2,0)
  setControlValue(3,0)
  setControlValue(10,0)

  while true
    int requestor, node, label
    int tcode = recvAsyncPacket(req,-1,-1,1,0)

    getPacketHeaderFields(req, requestor, node, label)
    setPacketHeaderFields(resp, node, requestor, label)
```

getPacketHeaderFields

Packet ID: req

TCode:

Packet type: ReadReq

Field	Get	Value
Source	✓	requestor
Destination	✓	node
Transaction Label	✓	label
Retry Code		
Priority		
Destination Offset		
Data Length Async. packet		
Response Code		
Extended TCode		
Data Length Stream packet		
Channel		
Tag		
Synchronisation Code		

Comment

packetID = the ID of the packet buffer from which fields have to be red

notes:

Click on the 'get' cell to actually get a field. Change the value in the 'Value' column to set the variable name the field has to be written to.

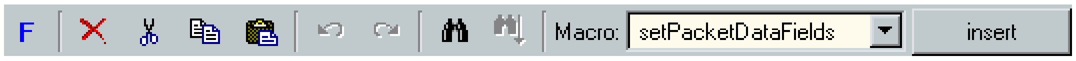
Cursor moveability
In a normal text editor the user can freely move the cursor around on every line. As in this hierarchical view line indention is indicated by a dotted tree on the left, the cursor is limited to the area right of this tree.

Selection/deselection
Whenever the script-functions view is active, there is at least one line active as indicated by a gray highlight bar. Altough the gray area only marks one line for editing, all lines that are leaves of the current line are implicitly selected as well. So, when deleting or copying, all child lines are also involved in the action. It is also possible to select multiple lines that are not childs of eachother. However, only lines on the same hierarchical level can be selected simultaneously.

By using the mouse or shift-left or shift-right it is also possible to select just part of a line. A partly selected line shows its selection by using a blue highlight bar for the selected part and a gray highlight bar for the remainder of the line.

Collapsing and expanding
Whenever a line contains child lines, as indicated by the branches in the plotted tree, the childs may be hidden by collapsing the tree node corresponding to this line. Expansion and collapsing can be controlled by using the '{' (expand) and '}' (collapse) keys on the keyboard or by using the mouse to click on the '+' or '-' symbols in front of the line respectively to expand or collapse.

11.2.1. Toolbar



The toolbar is shown in the picture above. From the left to the right, the toolbar contains the following buttons:

- Insert new Function
- Delete selected (gray) line (and its childs)
- Cut selected (gray) line (and its childs)
- Copy selected (gray) line (and its childs)
- Paste copied lines
- Undo
- Redo
- Find
- Find Next
- Select Macro Combo box
- Insert selected macro

The Macros that can be selected will be explained in the Macros section.

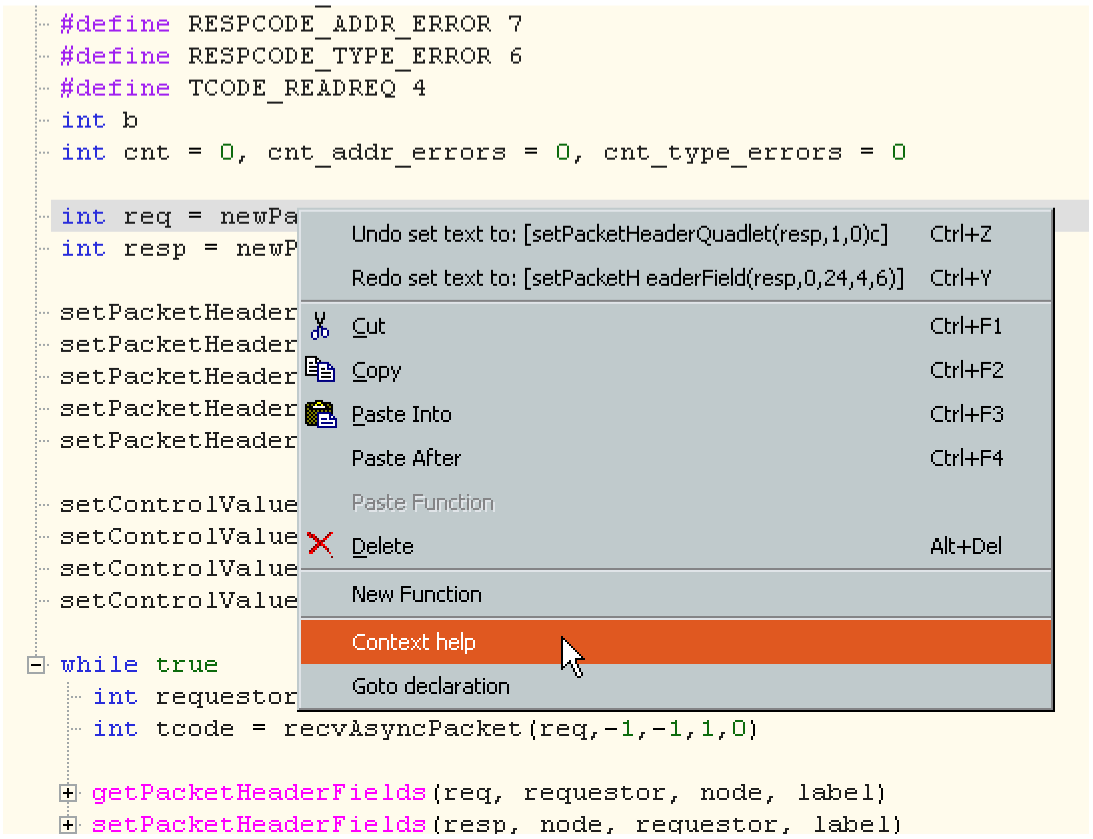
11.2.2. Context menu

When pressing the right mouse button while the mouse cursor is above the Script treeview, a context menu as in the picture below will pop up. This menu is the same as the "Edit" menu in the main menu at the top of the window. The contents of this menu are always the same, but individual items may be enabled/disabled depending on the cursor position in the script. The menu contains windows-standard items like copy, paste, undo and redo. In addition it contains the following items:

New Function
This will insert a new function in the script. In previous versions this was the only way to insert a function in the script, but now it is also possible to enter new functions by using the text editor.

Context help
This will open the manual in a web browser and locate a section that corresponds to the keyword at the current cursor position. This is especially handy to get help about API functions and such.

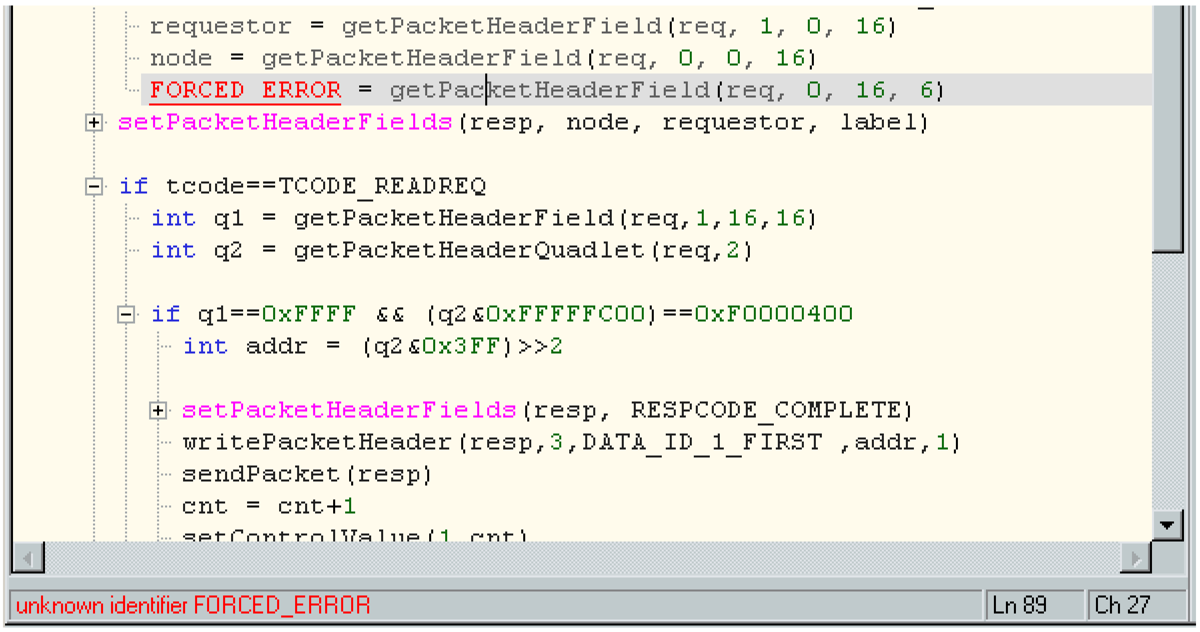
Goto declaration
When scripts become larger and larger, it might be difficult to remember where a function or variable was defined. With this menu option it is possible to navigate to the declaration of the identifier at the current cursor position. This also works for standard functions the Scriptor provides.



```
#define RESPCODE_ADDR_ERROR 7
#define RESPCODE_TYPE_ERROR 6
#define TCODE_READREQ 4
int b
int cnt = 0, cnt_addr_errors = 0, cnt_type_errors = 0

int req = newPa
int resp = newP

setPacketHeader
setPacketHeader
setPacketHeader
setPacketHeader
setPacketHeader

setControlValue
setControlValue
setControlValue
setControlValue

while true
  int requestor
  int tcode = recvAsyncPacket(req,-1,-1,1,0)

  getPacketHeaderFields(req, requestor, node, label)
  setPacketHeaderFields(resp, node, requestor, label)
```

11.2.3. Status Bar

The status bar at the bottom possibly contains an error message. While editing a script, error messages will be shown in red. During the edit phase, error messages should be considered as hints about the current statement. If, during compilation, errors still exist, the line corresponding to the error will be highlighted right after the compiler finishes and the script part that contains the error will be underlined. The script in the picture below deliberately contains an error to show what the status bar looks like in case of errors.



```
    requestor = getPacketHeaderField(req, 1, 0, 16)
    node = getPacketHeaderField(req, 0, 0, 16)
    FORCED_ERROR = getPacketHeaderField(req, 0, 16, 6)
  setPacketHeaderFields(resp, node, requestor, label)

  if tcode==TCODE_READREQ
    int q1 = getPacketHeaderField(req,1,16,16)
    int q2 = getPacketHeaderQuadlet(req,2)

    if q1==0xFFFF && (q2&0xFFFFFC00)==0xF0000400
      int addr = (q2&0x3FF)>>2

      setPacketHeaderFields(resp, RESPCODE_COMPLETE)
      writePacketHeader(resp,3,DATA_ID_1_FIRST ,addr,1)
      sendPacket(resp)
      cnt = cnt+1
      setControlValue(1,cnt)
```

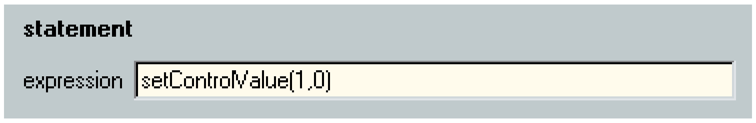
```
unknown identifier FORCED_ERROR          Ln 89   Ch 27
```

11.2.4. Property Editor

The Property Editor tab contains a properties editor for the currently selected script line in the script source code. The lay out of this editor differs for the different kinds of statements that may be inserted in the script. An overview of currently supported statements will be presented in the next sections.

11.2.4.1. Statement

When entering a statement, a simple edit box is shown on the properties page for entering an expression. The property editor is shown in the picture below.

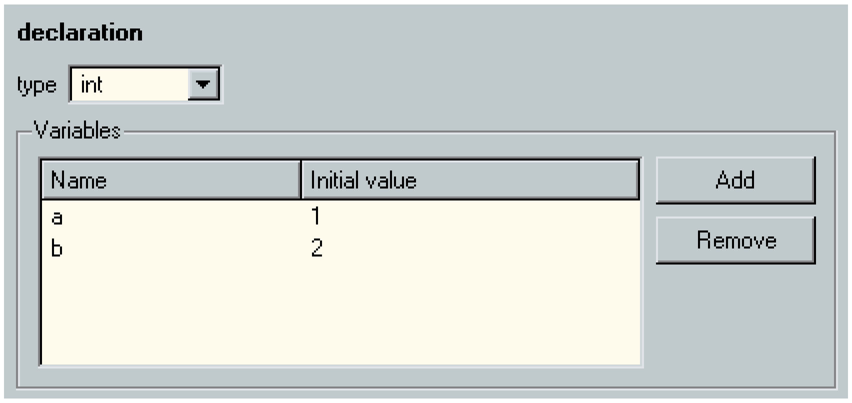


statement

expression: setControlValue(1,0)

11.2.4.2. Declaration

When entering a declaration, the properties page shows a list box to select the appropriate type and a table to fill in one or more variable name and initial value pairs. The property editor is shown in the picture below.



declaration

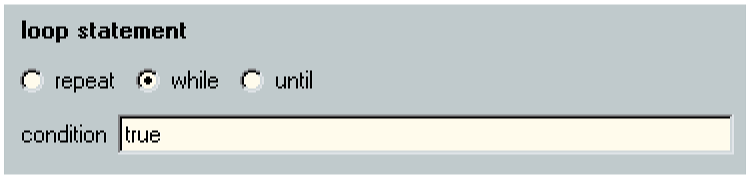
type: int

Variables

Name	Initial value
a	1
b	2

11.2.4.3. Loop Statement

When entering a repeat/while/until statement, the properties editor shows a group of radio buttons to select what kind of loop should be defined and an edit box to enter the loop condition. The property editor is shown in the picture below.



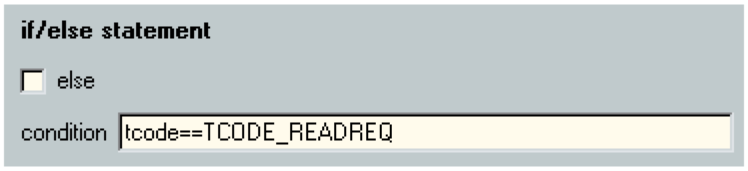
loop statement

repeat [✓ while] until

condition: true

11.2.4.4. Conditional Statement

When entering an if/else statement, the properties editor shows a check box to select whether an 'if' or an 'else' statement should be defined and an edit box to enter the condition. The property editor is shown in the picture below.



if/else statement

☐ else

condition: tcode==TCODE_READREQ

11.2.4.5. Function Declaration

Functions can be inserted by pressing the "New Function" button or by typing a function definition at the highest level in the script tree. When starting to type a function definition in the script source, the property editor will first show a property editor for a declaration. When enough of the function definition is entered for the parser to decide that it is indeed a function definition and not a declaration, the property editor will change into a function property editor.

This property editor contains the following items:

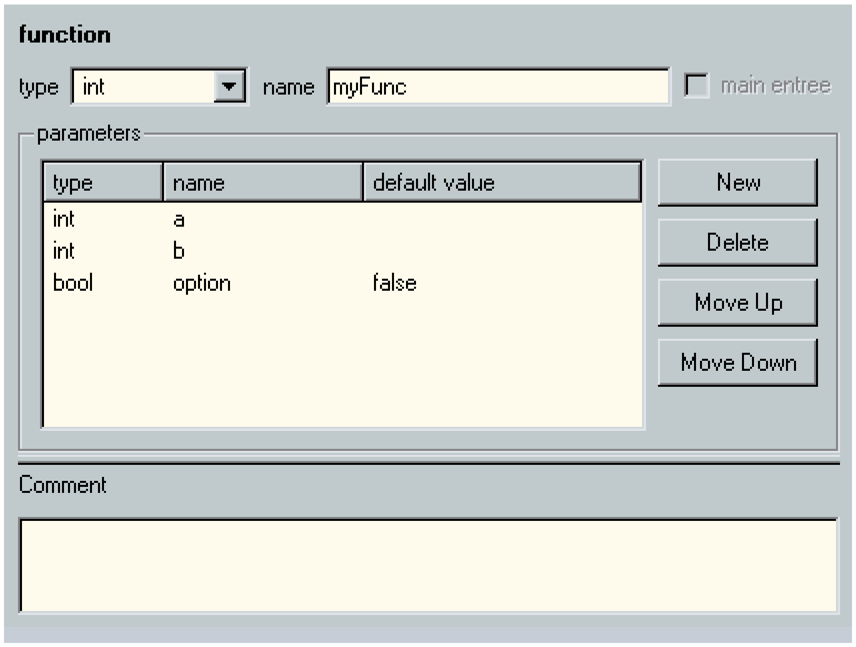
Type
The return type of the function.

Name
A user-defined name.

Main entry
When the script is being executed on the Analyzer, all functions that have this option checked will be started. Function priorities are determined by the order of function definitions, the first one having the highest priority. Functions that donot have this option checked will only be executed if they are being called by main-tree functions. Furthermore, main-tree functions cannot have a result type.

Parameters
This is a list of function parameters. Parameters are combinations of a data type, a variable name and optionally a default value. Variables listed here should be provided whenever the function is called, with exception of parameters with a default value. The order in which they have to be provided may be changed by using the up and down buttons.

Comment
This text editor does not influence the behavior of the function. It is a way for the developer to describe the purpose of the function being defined.



type int name myFunc ☐ main entree

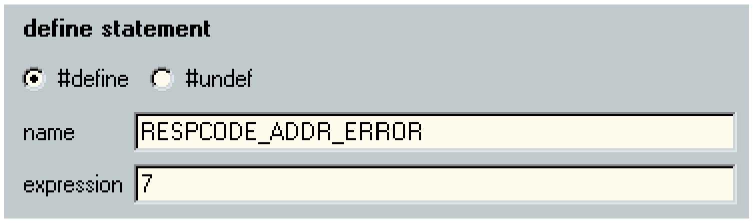
parameters

type	name	default value
int	a	
int	b	
bool	option	false

Comment

11.2.4.6. Define Statement

When entering a `#define` / `#undef` statement, the properties editor shows a check box to select whether a '#define' or an '#undef' statement should be defined and two edit boxes to enter the name and expression. The property editor is shown in the picture below.



[✓ #define] #undef

name RESPCODE_ADDR_ERROR

expression 7

11.2.4.7. Macros

In addition to the statements described in the previous section, the Script Editor also supports the use of macros. Macros can be used by providing their name on an empty line in the script editor. After such a name is entered, a property editor corresponding to that type of macro will show up. Available macro editors will be covered in the next sections.

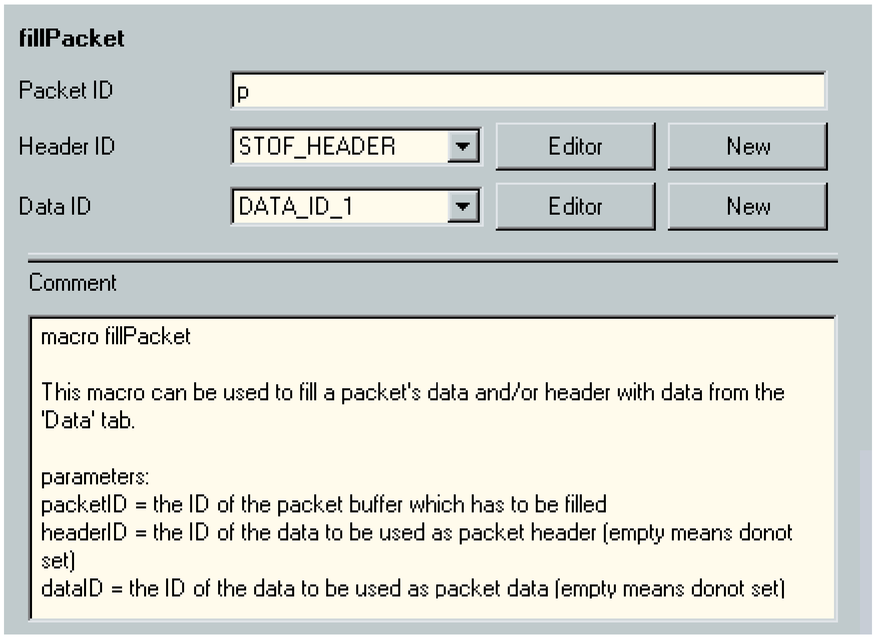
11.2.4.7.1 fillPacket

The fillPacket macro can be used to completely fill a packet buffer with constant data. Such constant data is created with the Data Editor. The property editor should at least be provided a PacketID to fill and a HeaderID or a DataID or both to read the data from.

If data has already been defined, the Header ID combo box and the Data ID combo box will contain the names of all available data objects. Otherwise, the user can click on the "New" button to create a new data object. When a data object has been selected in one of the combo boxes, the user can edit the data by clicking on the "Editor" button. When filling a packet with data, the corresponding data size is updated in the packet header if the header is already defined. The header can be defined by providing this macro a data ID for the header or by setting the header in the script before this macro is invoked.

In the image below, an example is given that fills a packet with data from the "MyData" data object. As can also be seen in the image below, the compiler 'knows' about the data object's name because it has been defined by a "#define" statement in the script. So the user has the choice to provide this macro with a defined name for the data, a numeric data ID (0 in this case) or even an integer variable set to the correct value. Note that when providing a numeric Data ID or integer variable, the user should make sure it really exists.

After filling in values in the property editor, the macro generates subnodes. Those subnodes are generated by the macro and read-only to the user (as indicated by a gray color). When the script is compiled, the line containing the macro call is not compiled itself, but only its subnodes. In this example the generated code is a call to an API function to fill the packet.



Packet ID: p
Header ID: STOF_HEADER
Data ID: DATA_ID_1

macro fillPacket

This macro can be used to fill a packet's data and/or header with data from the 'Data' tab.

parameters:
packetID = the ID of the packet buffer which has to be filled
headerID = the ID of the data to be used as packet header (empty means donot set)
dataID = the ID of the data to be used as packet data (empty means donot set)

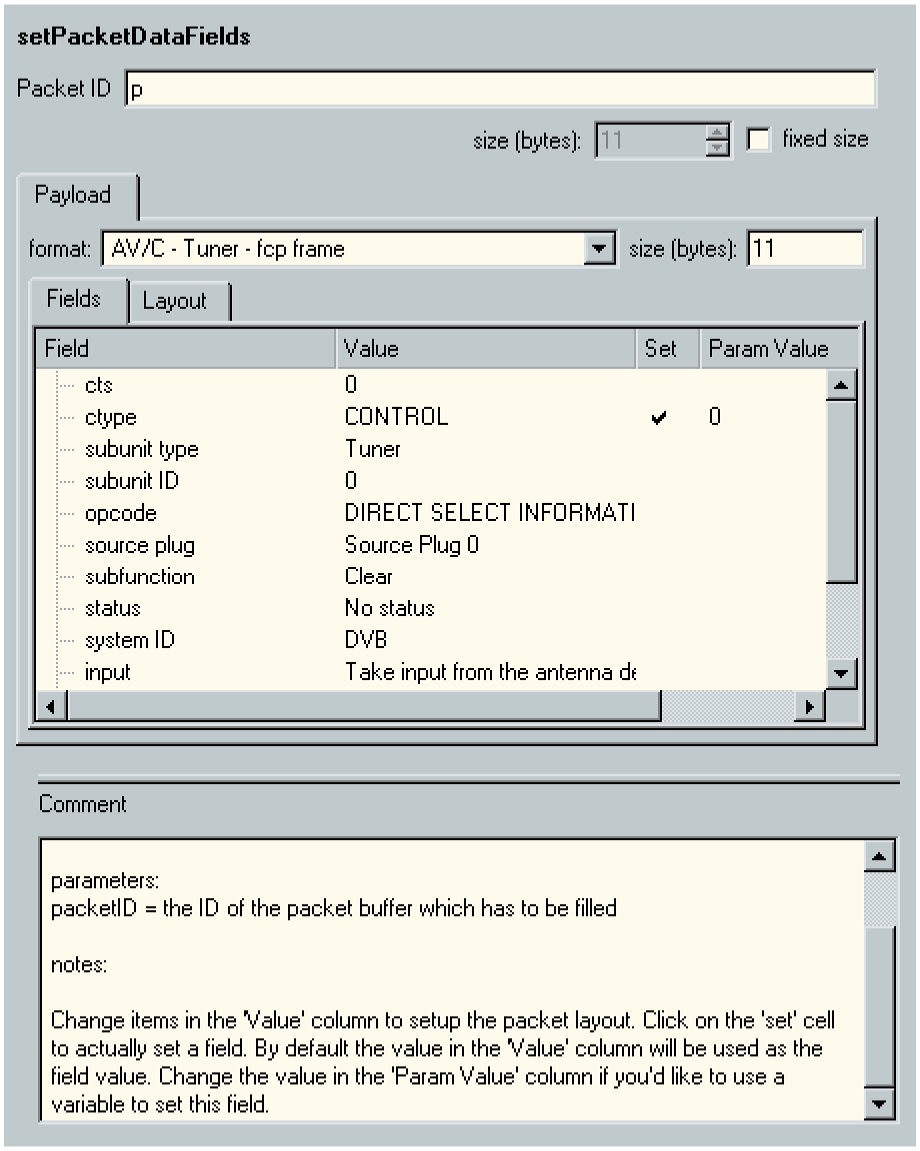
11.2.4.7.2 setPacketDataFields and getPacketDataFields

This macro can be used to set individual field values in a packet. It is invoked by entering "setPacketDataFields", and an editor shows up as can be seen on the right side of the picture below.

The packetID is the ID of the packet buffer which has to be filled. Change items in the 'Value' column to setup the packet layout. Setting values in the value column will not write these values in the packet, these values are only used to setup the correct packet layout in the editor. To actually write a field to the packet, click on the 'set' cell. By default the value in the 'Value' column will be used as the field value. Change the value in the 'Param Value' column if you'd like to use a variable or a different numeric value to set this field.

To create a packet that is larger than the size that is automatically determined by the packet layout, enable the "Fixed size" checkbox set the size to an arbitrary value.

While filling in the property editor, the script on the left side is updated and child nodes are generated. In de example below, the offset field has been given a value in the property editor and the script carries this out by calling two API functions.



setPacketDataFields

Packet ID: p

size (bytes): 11 ☐ fixed size

Payload

format: AV/C - Tuner - fcp frame size (bytes): 11

Fields | Layout

Field	Value	Set	Param Value
cts	0		
ctype	CONTROL	✓	0
subunit type	Tuner		
subunit ID	0		
opcode	DIRECT SELECT INFORMATI		
source plug	Source Plug 0		
subfunction	Clear		
status	No status		
system ID	DVB		
input	Take input from the antenna de		

Comment

parameters:
packetID = the ID of the packet buffer which has to be filled

notes:

Change items in the 'Value' column to setup the packet layout. Click on the 'set' cell to actually set a field. By default the value in the 'Value' column will be used as the field value. Change the value in the 'Param Value' column if you'd like to use a variable to set this field.

The getPacketDataFields macro is much the same as the setPacketDataFields macro. This macro is used to extract individual field values from a packet and set them to variables which have already been declared. The only difference in the way the macro is used is that the property editor now contains a "get" column instead of a "set" column and the "Param Value" should be provided a variable name to which the field value should be set.

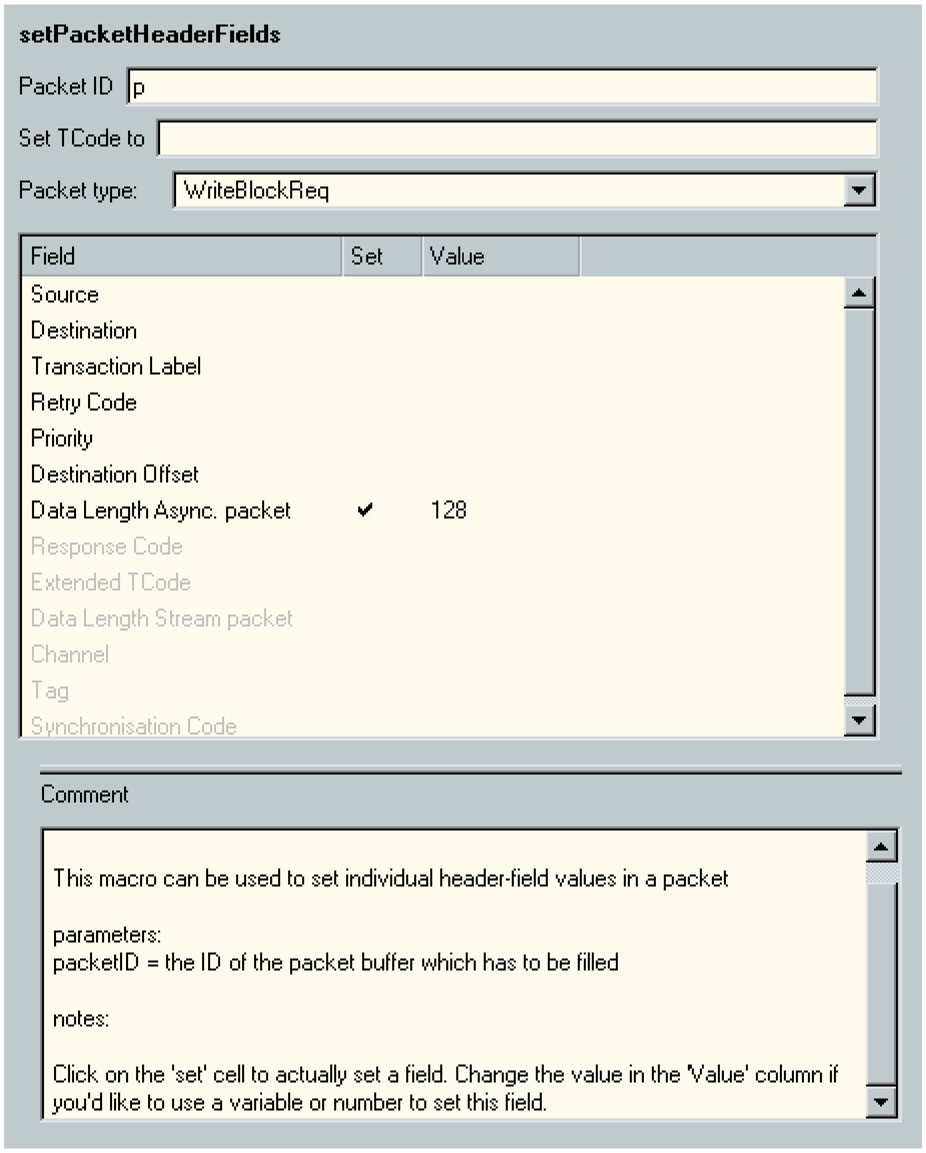
11.2.4.7.3 setPacketHeaderFields and getPacketHeaderFields

These macros are used to set and get individual header fields of a packet. The macros are invoked by entering "setPacketHeaderFields" or "getPacketHeaderFields" in the script. The property editor for this type of macro can be seen on the right side of the picture below.

The property editor again contains a line editor to provide a packet ID. In addition a combo box is provided to select the type of packet to set the header fields for and a table containing the available header fields for this type of packet to set or get the values of. Just like setting packet data, click on the "set" column to set a header field and provide the value to set in the "Param Value" column.

Use the tCode edit box to set the tCode of the packet. When no value is provided, nothing is done.

Just like the other macros, when filling in the property editor, subnodes are generated containing the required API calls to set or get the header fields.


setPacketHeaderFields

Packet ID: p

Set TCode to:

Packet type: WriteBlockReq

Field	Set	Value
Source		
Destination		
Transaction Label		
Retry Code		
Priority		
Destination Offset		
Data Length Async. packet	✔	128
Response Code		
Extended TCode		
Data Length Stream packet		
Channel		
Tag		
Synchronisation Code		

Comment

This macro can be used to set individual header-field values in a packet

parameters:
packetID = the ID of the packet buffer which has to be filled

notes:

Click on the 'set' cell to actually set a field. Change the value in the 'Value' column if you'd like to use a variable or number to set this field.

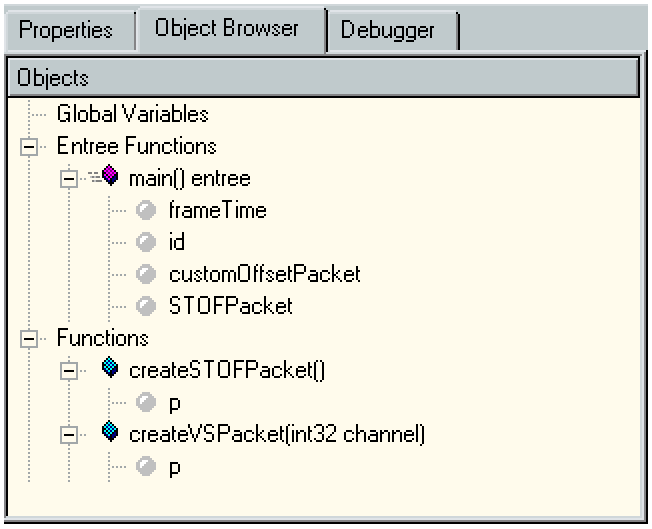
The getPacketHeaderFields macro is much the same as the setPacketHeaderFields macro. This macro is used to extract individual field values from a packet and set them to variables which have already been declared. The only difference in the way the macro is used is that the property editor now contains a "get" column instead of a "set" column and the "Value" should be provided a variable name to which the field value should be set.

11.2.5. Object Browser

The Object Browser contains an overview of the items that are defined in the current script. The tree contains all functions that are defined together with the variables that are defined in their bodies. Note that variables defined in the bodies of other constructions like loops are not shown here. To move the cursor to the definition of an item in the script source, simply click on its name.

Entry Functions
The list of entry functions reflects the processes that will run on the Analyzer when the script is started. The order of the entry functions in this list corresponds to the priority of the processes, the top one having the highest priority.

Functions
The list of functions contains all the functions that are defined in the source code of the script. This does not necessarily mean that a function in this list is actually executed. Regular functions themselves are not started as a process, but can only be called by other functions or processes. The order of this list only reflects the order the functions are defined in the script. This has nothing to do with priorities. The priority is determined by the process that calls a function.



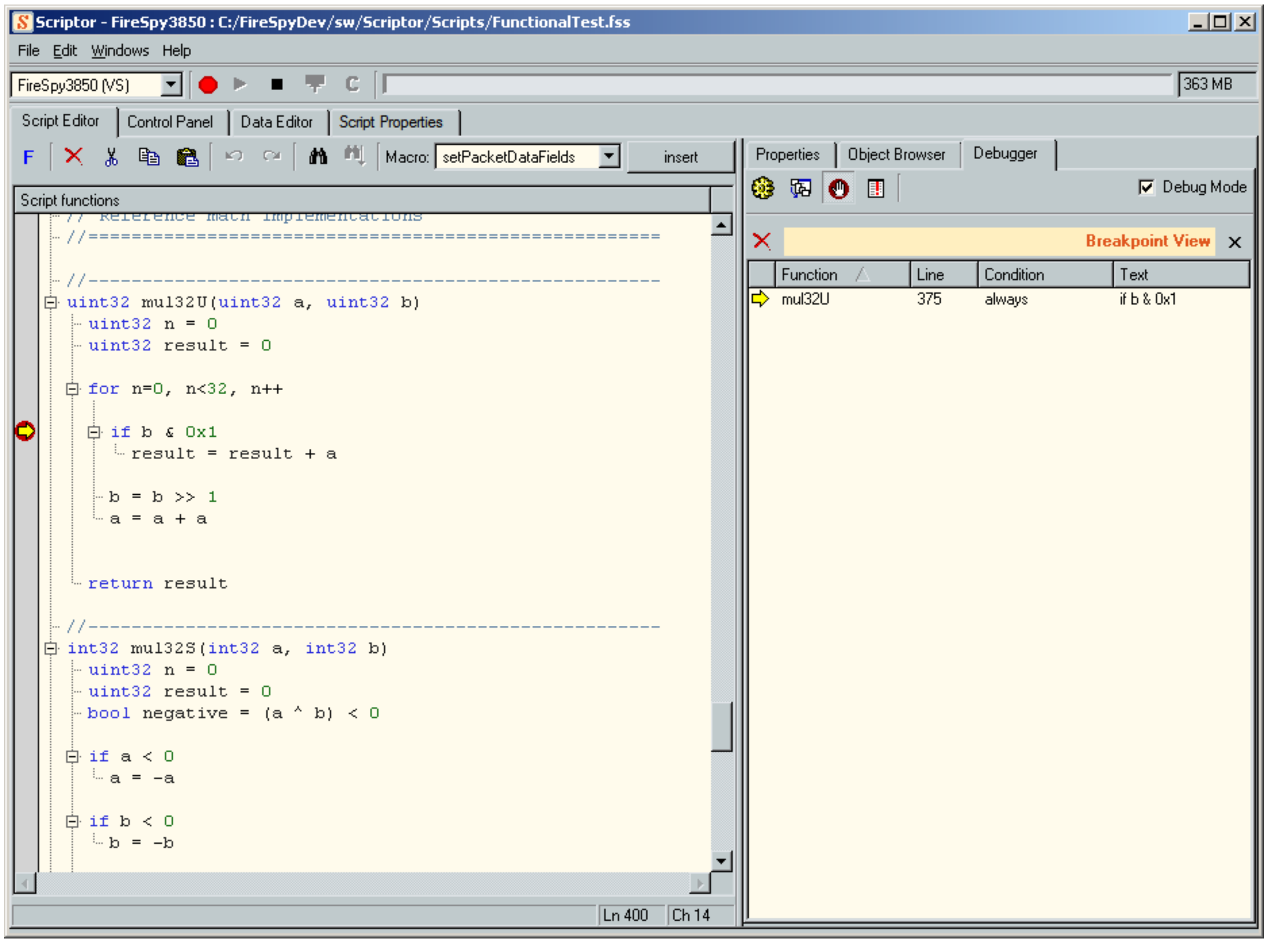
11.2.6. Debugger

To use the debugger, a main software license version 5.4 or higher is required. For more information about licenses see License Manager.

The Debugger can be used to analyze and control a running script. If the debugger is enabled, the main Scriptor window will look as in the picture below. On the left side of the script-editor pane a gray area is visible. This area is used to indicate that something is happening at specific lines. For example, in the image below a red circle with a yellow arrow in it is used to indicate that a process is broken because of a break point at that specific line in the script. Available debug icons will be explained in the next section.

On the right side, in the Debugger tab, several debug views are available and the user is free to chooses which one(s) to display. Each of the views will be explained in the next sections.

To set a breakpoint on a line of code, click on the gray border to the right of the line you want to set it for. A red circle will show up, indicating that a breakpoint has been set.



```
//===========================================================
    //-----------------------------------------------------------
uint32 mul32U(uint32 a, uint32 b)
    uint32 n = 0
    uint32 result = 0

    for n=0, n<32, n++

        if b & 0x1
            result = result + a

        b = b >> 1
        a = a + a


    return result

//-----------------------------------------------------------
int32 mul32S(int32 a, int32 b)
    uint32 n = 0
    uint32 result = 0
    bool negative = (a ^ b) < 0

    if a < 0
        a = -a

    if b < 0
        b = -b
```

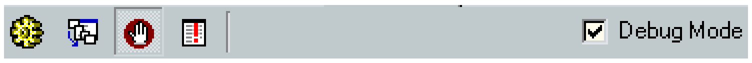
Function	Line	Condition	Text
mul32U	375	always	if b & 0x1

11.2.6.1. Debug Icons

Currently the following icons exist for display in the debugger:

A process is currently executing this line
The selected process is currently executing this line
This line is currently being executed by multiple processes
This line is currently being executed by multiple processes, including the selected one
If for the currently selected process, in the call stack view, a different call depth is selected, this icon is shown at the line where the selected function call took place.
The same as the icon above with as only difference that other processes are also currently executing this line.
This line contains a breakpoint. This icon can be the only icon visible at a script line, but it can also be combined with one of the other icons. If combined with a yellow arrow, it means that a script process stopped at this break point.

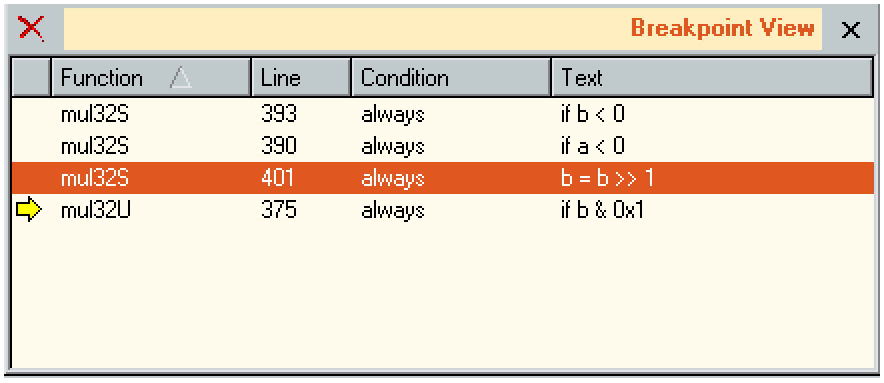
11.2.6.2. Toolbar



The toolbar is shown in the picture above. From the left to the right, the toolbar contains the following buttons:

- Enable/disable the Process View
- Enable/disable the Call Stack View
- Enable/disable the Breakpoint View
- Enable/disable the Error View
- Enable/disable the debugger

11.2.6.3. Breakpoint View



	Function	Line	Condition	Text
	mul32S	393	always	if b < 0
	mul32S	390	always	if a < 0
	mul32S	401	always	b = b >> 1
⇨	mul32U	375	always	if b & 0x1

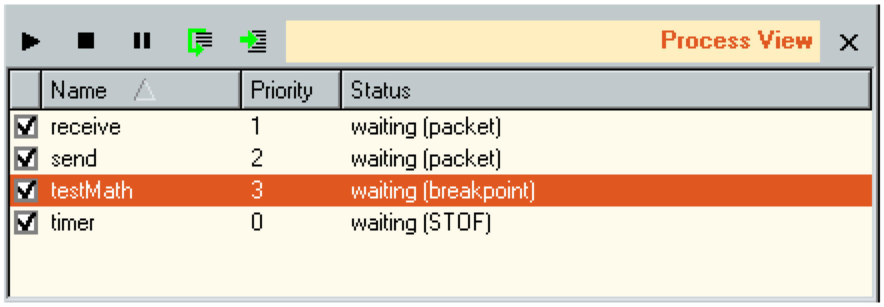
The Breakpoint view displays a list of all the breakpoints that are currently set in the script. For each breakpoint listed, the following information is shown:

- If a process is broken at this breakpoint, an arrow is shown. See row 4 in the picture above
- The name of the function the breakpoint is positioned in
- The line number the breakpoint is set at
- The condition that must be met for a process to break at this breakpoint (conditions cannot be edited yet, so a process always breaks at a breakpoint)
- The text of the script line the breakpoint is positioned at

By clicking on a breakpoint, it will be highlighted in red and the script-tree editor will move to the corresponding line in the script. The selection in the breakpoint view is not related to the selection in any other Debug View.

By clicking on the red cross in the toolbar of this view, the currently selected breakpoint will be deleted.

11.2.6.4. Process View



	Name	Priority	Status
✓	receive	1	waiting (packet)
✓	send	2	waiting (packet)
✓	testMath	3	waiting (breakpoint)
✓	timer	0	waiting (STOF)

The Process View displays a list of processes. For each process listed, the following information is displayed:

- Enable checkbox: If checked, the process is currently in debug mode, otherwise not
- The name of the process
- The priority of the process (smaller number means higher priority)
- The current status of the process.

Possible process state values are:

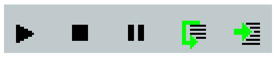
- unknown: The process is not being debugged, or the state has not been updated yet
- not started, The process is not yet started
- running, The process is currently running
- stopped, The process stopped running
- ready, The process is ready to execute the next code, however another process is already running
- paused, The process is paused by the user
- waiting (packet), The process is waiting for a packet to be sent or received
- waiting (STOF), The process is waiting for the start of the next frame (Mil1394 mode only)
- waiting (STOF packet), The process is waiting to receive a STOF packet (Mil1394 mode only)
- waiting (bus reset), The process is waiting for a bus reset

- waiting (sleep), The process is waiting for a sleep statement (not yet implemented)
- waiting (breakpoint), The process is broken at a breakpoint or the user is stepping with the debuggger
- waiting (stream write), The process is writing to a stream buffer, or waiting until space is available to write
- waiting (stream read), The process is reading from a stream buffer or waiting for data to be available in the buffer
- waiting (host function), The process is executing a function on the host and waiting for it to return

If the debug mode is enabled and the script is running, the Scriptor will periodically request information for the processes that are enabled in the process view. Enabling a process is done by clicking in the check box to the left of it.

If a process is not only enabled (checked), but also selected as the current process (red highlight bar), the script tree will follow the yellow arrow that marks the line that is executed by the process. When no process is selected, no process is followed and only gray arrows are drawn in the tree view. If the Call Stack view is shown, the information displayed in it corresponds to the selected process.

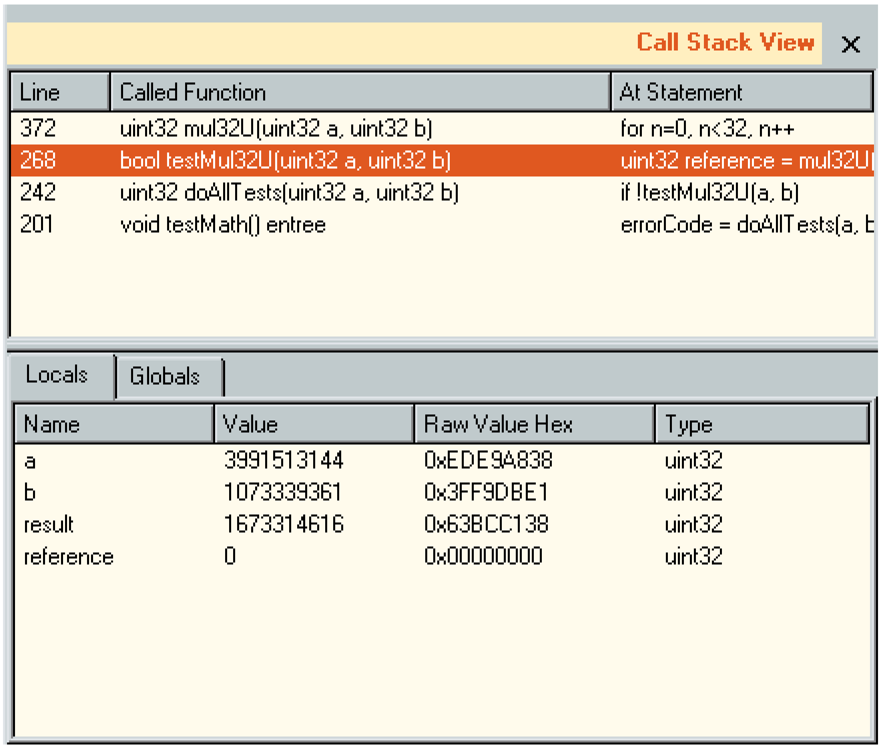
As the info per process is requested in a periodical way, it might seem that a process is constantly at the same code line. In reality, however, the process could be executing other code lines in between two process info updates. Check the process state to verify if a process is indeed waiting for some event or if it is still running/ready.



The Process View also contains a toolbar with some controls on it to control script execution. The toolbar is shown in the picture above and from the left to the right contains the following controls:

- Start/resume process: This control will start or resume the currently selected process. Ofcoarse this function is only available if the current process state allows resuming or starting.
- Stop process: This control will try to stop the current process. If the current process is waiting for something, it will not stop until it is done waiting. Pressing the stop control several times will not speedup the stop request. If the process is not waiting and not stopped already, it will stop immediately.
- Pause process: This control will try to pause the current process. If the current process is waiting for something, it will not pause until it is done waiting. Pressing the pause control several times will not speedup the pause request. If the process is not waiting and not stopped already, it will stop immediately.
- Step Over: If the script is currently at a breakpoint or paused, this control will make it execute the current script line and break at the next one.
- Step Into: If the script is currently at a breakpoint or paused, this control will make it execute the current script line and break at the next one. However, if the current script line contains any function calls, it will break at the first line in the function that is called.

11.2.6.5. Call Stack view



Line	Called Function	At Statement
372	uint32 mul32U(uint32 a, uint32 b)	for n=0, n<32, n++
268	bool testMul32U(uint32 a, uint32 b)	uint32 reference = mul32U(
242	uint32 doAllTests(uint32 a, uint32 b)	if !testMul32U(a, b)
201	void testMath() entree	errorCode = doAllTests(a, b

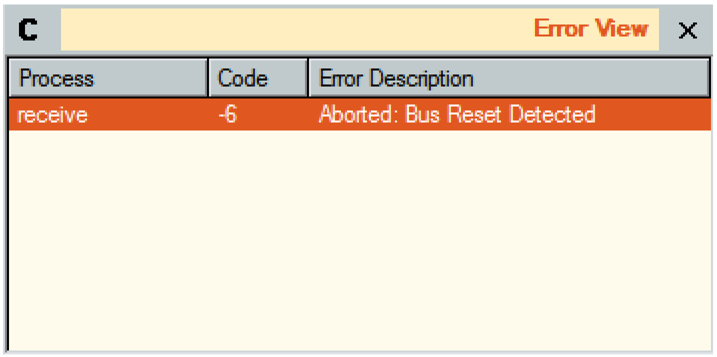
Name	Value	Raw Value Hex	Type
a	3991513144	0xEDE9A838	uint32
b	1073339361	0x3FF9DBE1	uint32
result	1673314616	0x63BCC138	uint32
reference	0	0x00000000	uint32

The Call Stack View displays symbol and function-call information for the process that is currently selected in the Process View. If the Process View has no selection, this view will be empty.

The upper part of the Call Stack View displays a list of function calls made by the selected process. The top one is the line that the process is currently executing. All other items point to the script line at which a function call was made to the function the item above it is currenlty executing. In the example above, currently the function "mul32U" is being executed. This function was called by "testMul32U", which on its turn was called by "doAllTests", which on its turn was called by "testMath". The last one is an entry function and therefore the root of the call stack.

The lower part of the Call Stak View displays a list of local and global variables that were available to the script at the script line corresponding to the selected function call. The values of these variables are displayed in both their native format and in raw HEX format. The last column displays the data type of the variable. In the example above, the function call is selected that was made at script line 268. At this line, the variabled "a, b, result and reference" were available and their values can be seen in the picture above. If the Process View is used to "step into" or "step" through the script, the symbol list will indicate changes by displaying them in red.

11.2.6.6. Error View



Process	Code	Error Description
receive	-6	Aborted: Bus Reset Detected

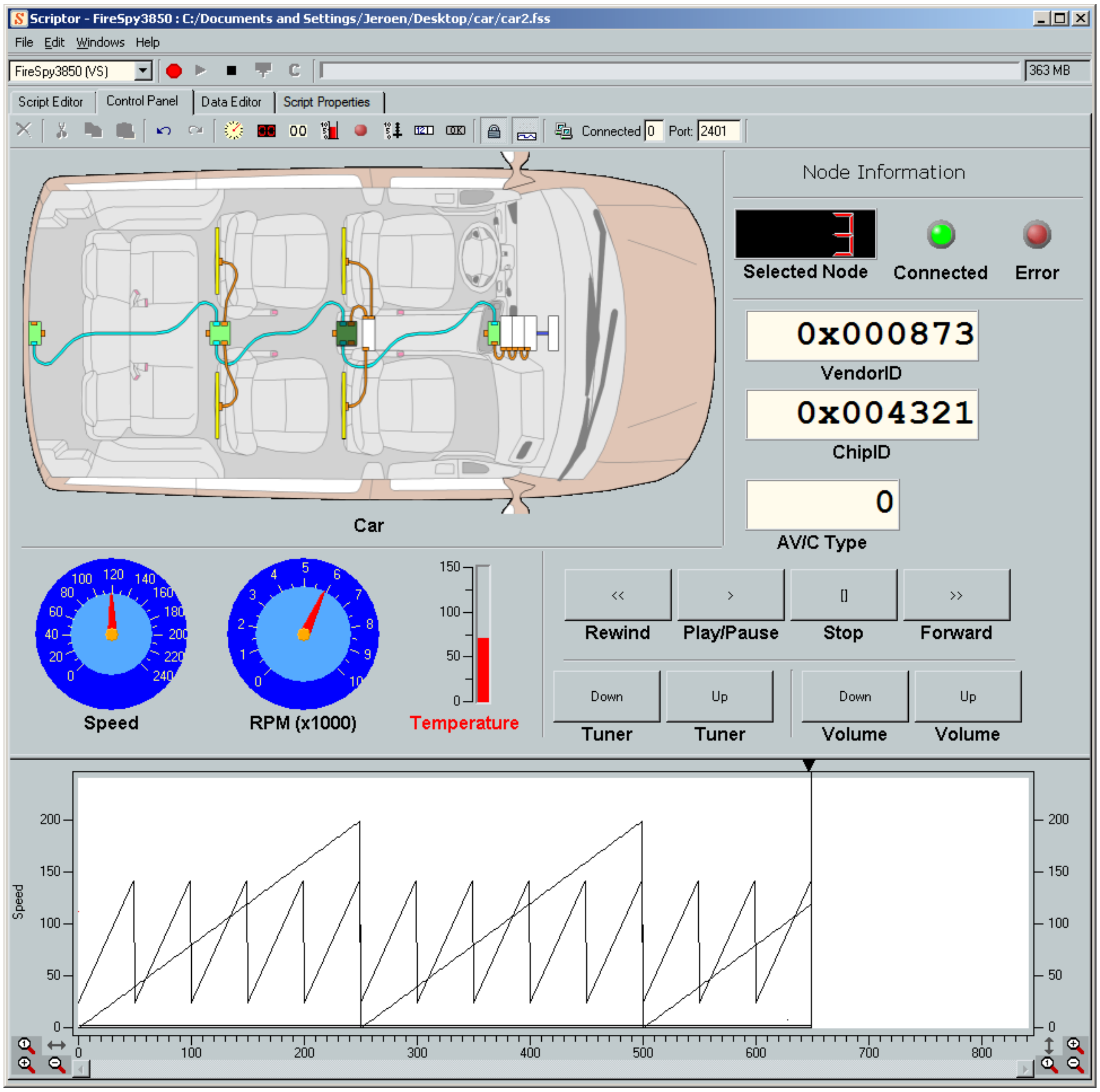
Many of the Scriptor library functions will return a negative value in case of error. When running in debug mode, these negative return codes will always be sent to the debugger, even if the script does not contain any return value checking.

Whenever the debugger receives a negative return value it is appended to the error view. The picture above shows an example of a negative value (-6) returned by the recvStreamPacket function. In this case a bus reset was detected before a packet was received.

The error view will always get focus when it received the first error during script execution. Now you can click on the error and the script editor on the left will highlight the exact line that resulted in a negative return code.

11.3. Control Panel

The control panel is used to display the output of a script and/or send values to a script. You can, for instance, send the value of a specific field of all stream packets on some channel number to a control on the control panel such that you can easily monitor the value of this field. These values can also be added to a history graph. An example Control Panel is shown in the picture below.



Adding controls
To add a control, use your secondary (right) mouse button on the control panel and choose one of the above controls. To view the controls' properties you can click with your secondary mouse button on the

control. The Gauge is the most complex control with many properties to customize it. The Led has just two specific properties (color and threshold). All Controls share a few settings, these are also the most important settings.
An added control is positioned below the cursor, the position and size might not be what you want. To position it just drag and drop the control to its place. The size can be changed by using the size grips in the corners of the control.

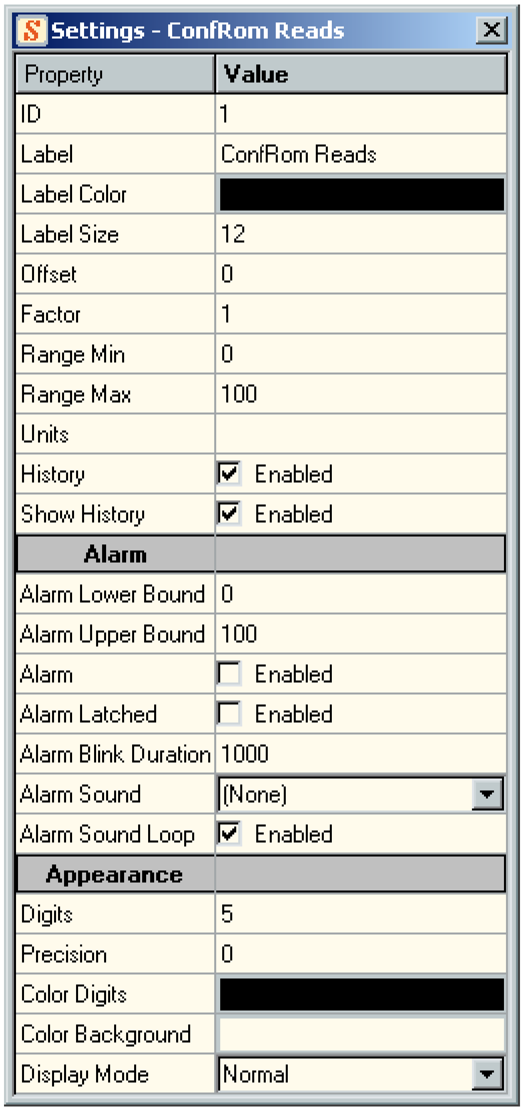
Locking controls
When all controls are in place and sized according to your preferences you can lock the controls and hide the grid by right clicking on the control panel and selecting "Lock Controls". Now the controls cannot accidentally be resized or moved. Note that this setting is also available on the tool bar.

Hiding the history graph
If you don't want to see the history graph in the bottom of the panel you can hide it by right clicking on the control panel and toggle the option "Show History Graph". This setting will be saved for the current script. Note that this setting is also available on the tool bar.

Script interaction
This section and its subsections mainly describe the Control Panel components themselves. To learn how to make the Control Panel actually interact with a running script, see the Script function list for the Control Panel.

11.3.1. Control Properties

All controls on the control panel have properties associated with them. These properties can be changed to influence the behavior of the control. A property editor is available that can be opened by right-clicking the control to edit. The picture below shows an example property editor. Note that all changes made are applied immediately. By using undo/redo, changes can be reverted.



Settings - ConfRom Reads

Property	Value
ID	1
Label	ConfRom Reads
Label Color	■
Label Size	12
Offset	0
Factor	1
Range Min	0
Range Max	100
Units	
History	☑ Enabled
Show History	☑ Enabled
Alarm	
Alarm Lower Bound	0
Alarm Upper Bound	100
Alarm	☐ Enabled
Alarm Latched	☐ Enabled
Alarm Blink Duration	1000
Alarm Sound	(None)
Alarm Sound Loop	☑ Enabled
Appearance	
Digits	5
Precision	0
Color Digits	■
Color Background	□
Display Mode	Normal

This section explains the most common properties.

ID
Every Control should get its own unique ID. When the script is being executed, the function setControlValue(ID,Value) sets all the controls with that ID to "Value". It is possible to assign the same ID to multiple controls. A setControlValue(ID,Value) then will set all controls with the same ID. Depending on the application this feature may be useful or not. When you give an "input control" the same ID as another control, the other control will follow the input control even without the script running.

Label
This property is used to give the control a name. This name is used to label the current scale in the history graph.

Label Color
This value is used to color the data in the history graph. This way you can distinguish which line belongs to which control. Of course it colors the label below the control as well.

Label Size
This value is used to change the size of the Label below the Control.

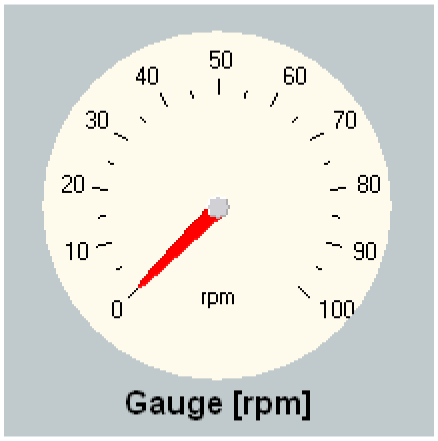
Offset/Factor
When the raw data of the packets need some precalculation before it represents the real value, the calculation can be done by:

```
        RealValue = Offset + (RawValue * Factor)
```

The controls always display the calculated value. The history graph however has an option to "recover" the value as it was set by setValue() and grabValues().

Range Min/Max
These settings are mainly used for adjusting the scale of the history graph. However, this setting is also used to change the scale of the "Gauge" control. Let's say you are "recording" the speed of a motor in RPM, a scale from 0 to 100 would be insufficient. So you would probably set the scale from Range Min = 0 to Range Max = 10000.
(Note: if you change the range for a Gauge you also need to alter the "Scale Step")

Units
As already mentioned above, you may record the speed in RPM. And if so, you would probably label the control "Speed" and use "RPM" as the control's Units. In case of the Gauge it is displayed on the meter itself and on the label.



Alarm
The next block of settings are the Alarm settings. The alarm will alert you when the value of a control exceeds its boundaries.

Alarm Lower/Upper Bound
When the alarm is enabled, these boundaries are visible in the history graph. When a control receives a value from the script with a value lower than the lower bound or higher than the upper bound, the alarm is triggered. The label of the Control will flash and when selected a sound is played.

Alarm Enabled
Whether or not the alarm function will trigger when the boundaries are exceeded.

Alarm Latched
As soon as the value of the control is within the alarm boundaries, the alarm stops. For some quick events you may not want this behavior, but want to be notified if an alarm event has occurred.

Alarm Blink Duration
This value is in milliseconds and holds the blink duration. This is 1000 divided by the frequency. For example you can give an important alarm a low blink duration so it blinks faster (and attracts more attention).

Alarm Sound
In the application directory there is a folder called sounds, we've selected a few to get you started.

Alarm Sound Loop
Plays the sound in a loop when the alarm is triggered. Note that it is different from Alarm Latched as Alarm Sound Loop will stop looping if the alarm is not latched and the value comes within boundaries.

Appearance
Almost all Controls have individual settings for appearance. Most of them are straightforward and won't need any additional information. The Gauge however has some settings that are less trivial.

Origin
Defines the starting position of the first line of the scale in degrees counter clockwise. So "-90" starts at the top. "0" starts on the right, "90" on the bottom etc.

ScaleArcMin
In degrees, counter clockwise, starting from the "Origin" (mostly 0).

ScaleArcMax
In degrees, counter clockwise, starting from origin. If you know you want 3/4 of a circle you set this value on 270, if you want half a circle you can set this value to 180. Then it is easier to get the scale positioned the way you want by adjusting the Origin.

Scale Major Interval
This value is only valid when "Scale Step" = 0. It defines how many intervals are visible on the scale. If a value would result in a strange scale, it is rounded down to the first value that makes sense. If this doesn't result in the scale that you would have liked to see, it is better to use the Scale Step.

Scale Minor Interval
Specifies the number of intervals between two Major values. This is also rounded down to a value that rounds down to whole numbers on a tick. Most practical values are: 0, 3, 6, 10.

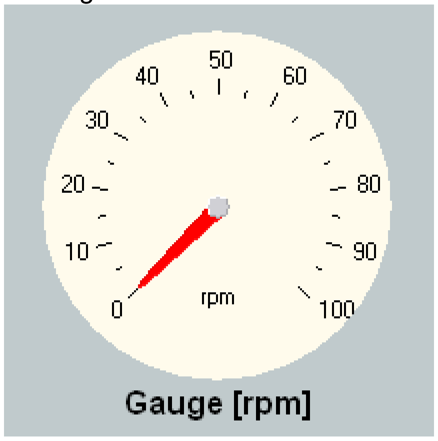
Scale Step
Defines the Step Size on the scale: for example, on a scale from 0 to 100 with a step size of 20 you would see: 0, 20, 40, 60, 80, 100. When the step size is too small compared to the range of the scale it stops looking nice. Or you would have to enlarge the control to make all values fit again.(Note: when set to zero the number of Major Ticks is defined by "Scale Major Interval").

Frame Width
Draws a little border around the meter to give it some 3D effect.

11.3.2. Output Controls

The following controls can be used to display values from a running script.

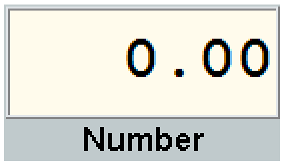
Gauge
A Gauge can be used to show values on a scale that can be defined with the properties dialog.



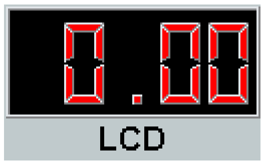
Number
A Number Control can be used to display numeric values. Currently, the following display formats are defined:

- decimal
- hexadecimal
- ASCII

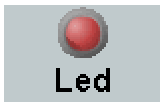
The display format can be set in the properties editor of the Number control.
When using the ASCII format, this indicator shows up to 4 ASCII characters, defined by the four byte values of a 32-bit integer.



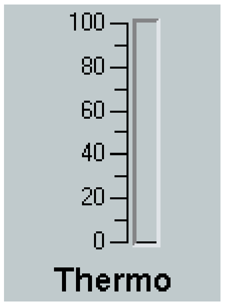
LCD
An LCD Control can be used to display numeric values. Note that this control is slower than the Number control.



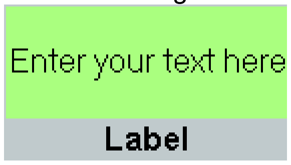
Led
A Led is a control that is either "on" or "off". To turn it off, send a value of '0' to it. All other values turn it on. The color is configurable in the properties dialog.
The Led also has the ability to blink when turned on, this settings is called "Led Blink Duration" and is specified in milliseconds. When you set his value to 1000 it means that the Led will turn on for one second and then off for one second, and so on. Settings this value to 0 means no blinking.



Thermo
A Thermo can be used to show values on a scale that can be defined with the properties dialog.



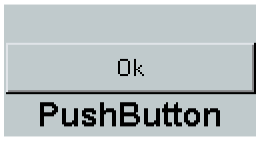
Label
This control shows a design-time defined text on a run-time controlled background color. This is done by associating numeric values to specific colors. When a value is sent to a Label Control, its background color will change accordingly.



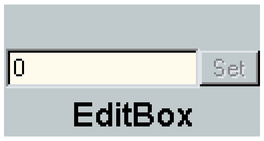
11.3.3. Input Controls

The Control panel also offers controls that act as an input for the script. Input controls can only be controlled when the Control Panel is locked, this is done automatically when the script is started and on load of a script. If the Control Panel is unlocked, Controls cannot be controlled but modified instead. The following input controls are available to the user.

Pushbutton
The Pushbutton is the simplest of the input controls and can only send a '0' or a '1' to the script. When the user presses the button a '1' is send and when it is released a '0' is send. The PushButton can also be controlled from within the script. If you want to disable the button, send a '-1' to it. To show it in a down state, send a '1' and to get it beack to normal send a '0'. Also if you set the button in a disabled state send it a '0' to enable it again.



Editbox
The Editbox can be used to send any number to the scriptor in the Analyzer. The value is submitted on two occasions, one way is by clicking the "set" button next to the editbox, and the other way it by selecting another control. You can see that it has submitted its value because the "set" button gets disabled. Values can be entered in hexadecimal format by typing 0x first and then your number in hex format.



Slider
The Slider makes it possible to quickly send multiple values one after another to the script. However it isn't as accurate as the editbox.
(Tip: If you want to see which value is send to the script, place a number or lcd display next to the slider and assign it the same ID as the slider.)

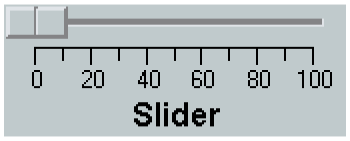
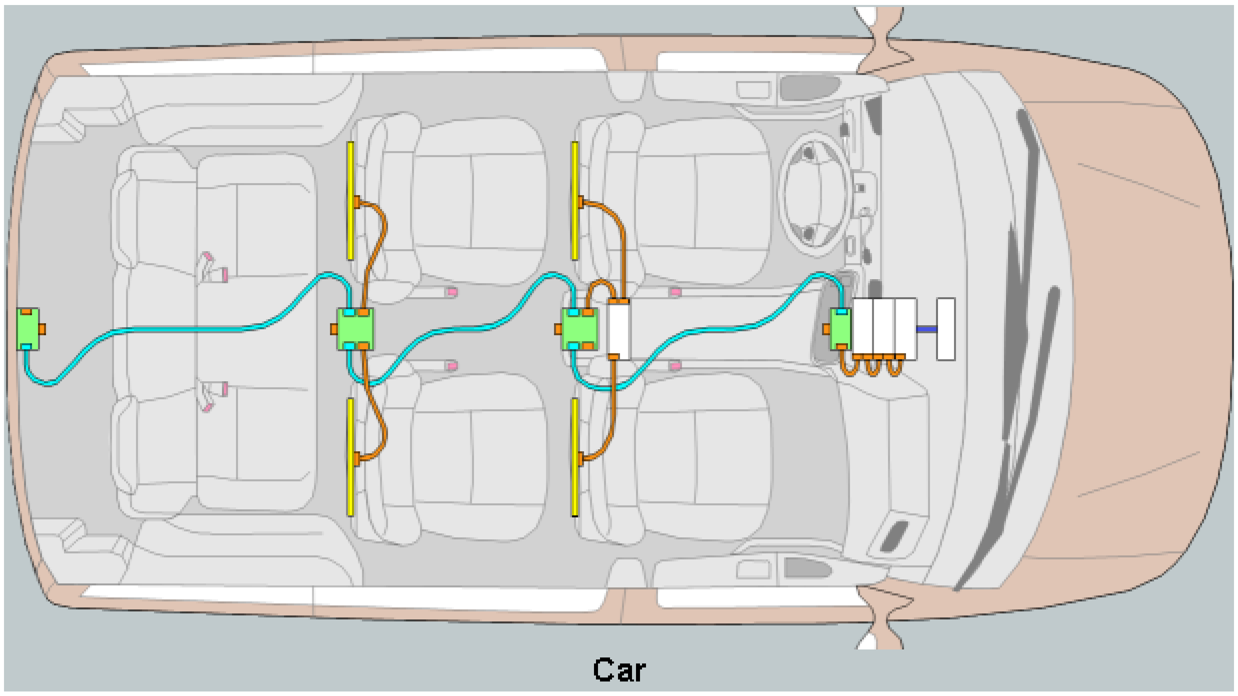
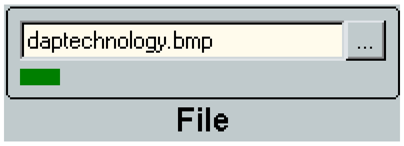


Image
The image control makes it possible to ad an image to the control panel with clickable regions. These regions are defined as HTML image maps and each clickable section has a numeric value associated with it. When the user clicks in a region the corresponding value is sent to the control ID of the image control. An example of a car with some clickable IDB 1394 nodes is shown in the folling picture.



File Browser
The File browser control makes it possible to open a file dialog and select a file for input or output from a running script. The file can be referred to in the script by using the control ID of the file browser. When the

script opens the specified file, the green LED will turn on. When this LED is on, it is not possible to change the filename.



11.3.4. Layout Controls

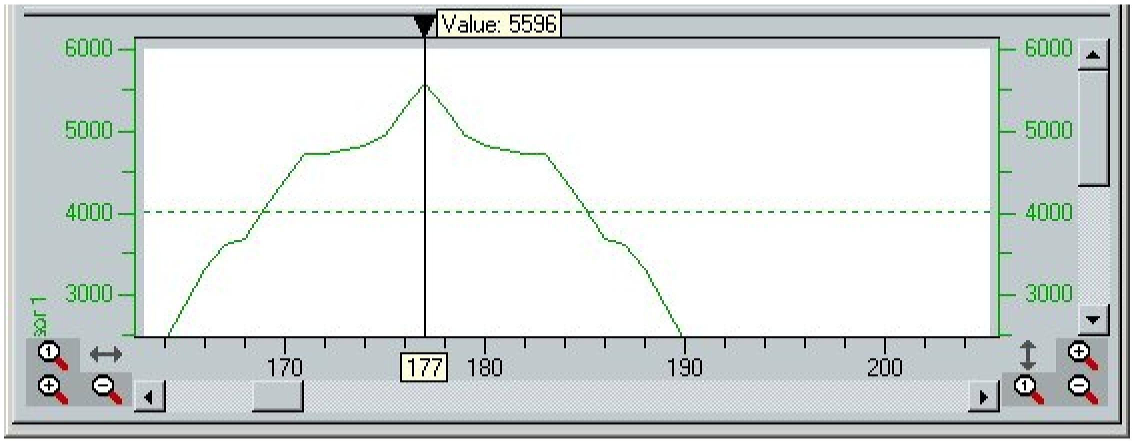
Layout controls donot have any form of dynamic behavior. They can be used to beautify or clarify the control panel. The following layout controls are available to the user:

Text
Text controls can be used to place a text panel on the control panel. The panel frame can be flat, raised or sunken.

Lines
Lines can be used to visually seperate parts of the control panel. Both horizontal as vertical lines are available.

11.3.5. History Graph

The history graph offers a tool to examine script values as a function of time. Scriptor provides a function GrabValues(time) which grabs the current values from the controls and uses it to plot the data. You can hide the graphs from showing on the History Graph by disabling the option "Show History" for every control you don't want to see. It does get recorded, if you want to inspect it afterward you are able to turn it on again. This way the history graph stays "readable".



Each graph can be identified by its color. This color can be changed by changing the "Label Color" of its control (see control properties above).

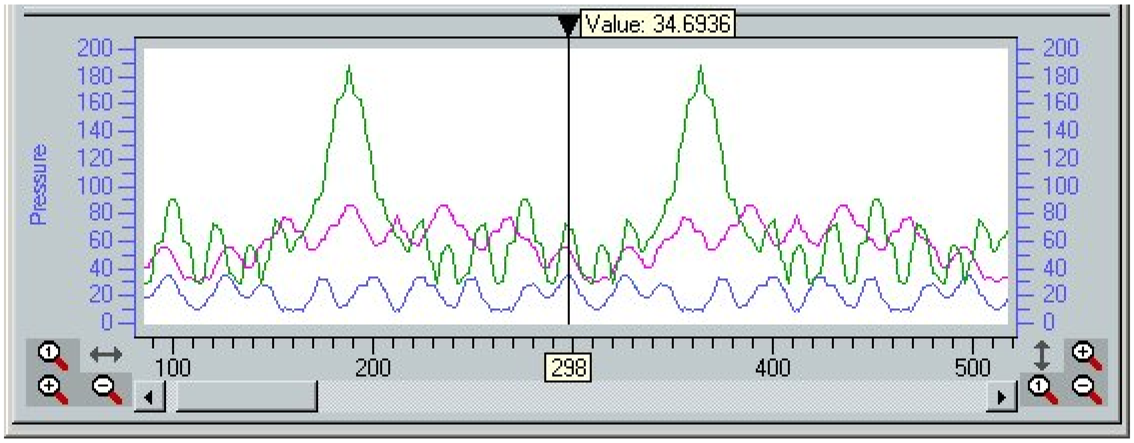
Each graph uses its own scale, so a graph with a range from -10 to 10 can be drawn over a graph with range 100 to 10000. To select which scale is visible for the left and right side of the plot, just click on its control.

The Graph also contains a cursor, which can be used to retrieve the value at the selected time. Move the cursor to position it exactly where you want it, this way it is easy to retrieve a maximum or minimum value. The graphs in the plot represent the calculated values in accordance to the description of the Offset and Factor properties. If you are interested in the "Raw" value of some point , right click on the cursor. A popup menu enables you to select "Raw values". When enabled, the plot still draws the calculated values but the cursor displays the "Raw" values.

When new values are added to the graph the plot normally scrolls the view automaticaly.This can be stopped in two ways. One is by placing the cursor somewhere in the "past". The other way is by scrolling the scrollbar to the past. Automatic scrolling is enabled again by scrolling all the way to the right or by clicking somewhere in the "future".

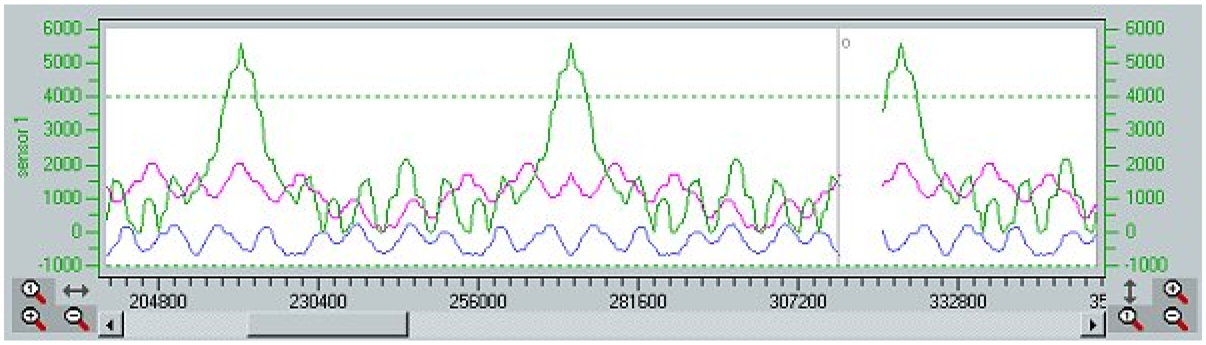
When controls are added to the Control Panel, their data also becomes visible in the plot. If, at some point, you don't want to see all graphs, right click the control you are not interested in and disable "Show History". Note that the data is still being recorded and can be made visible again by enabling "Show History".

The magnifying glasses on the left and right side of the bottom scrollbar enable you to zoom horizontally and vertically. The magnify glass with the '1' inside resets the zooming to default.



The vertical-zoom function can only zoom in, but the horizontal-zoom function also enables you to zoom out to get more data in the same plot, which makes it possible to overview data for a wider time range.

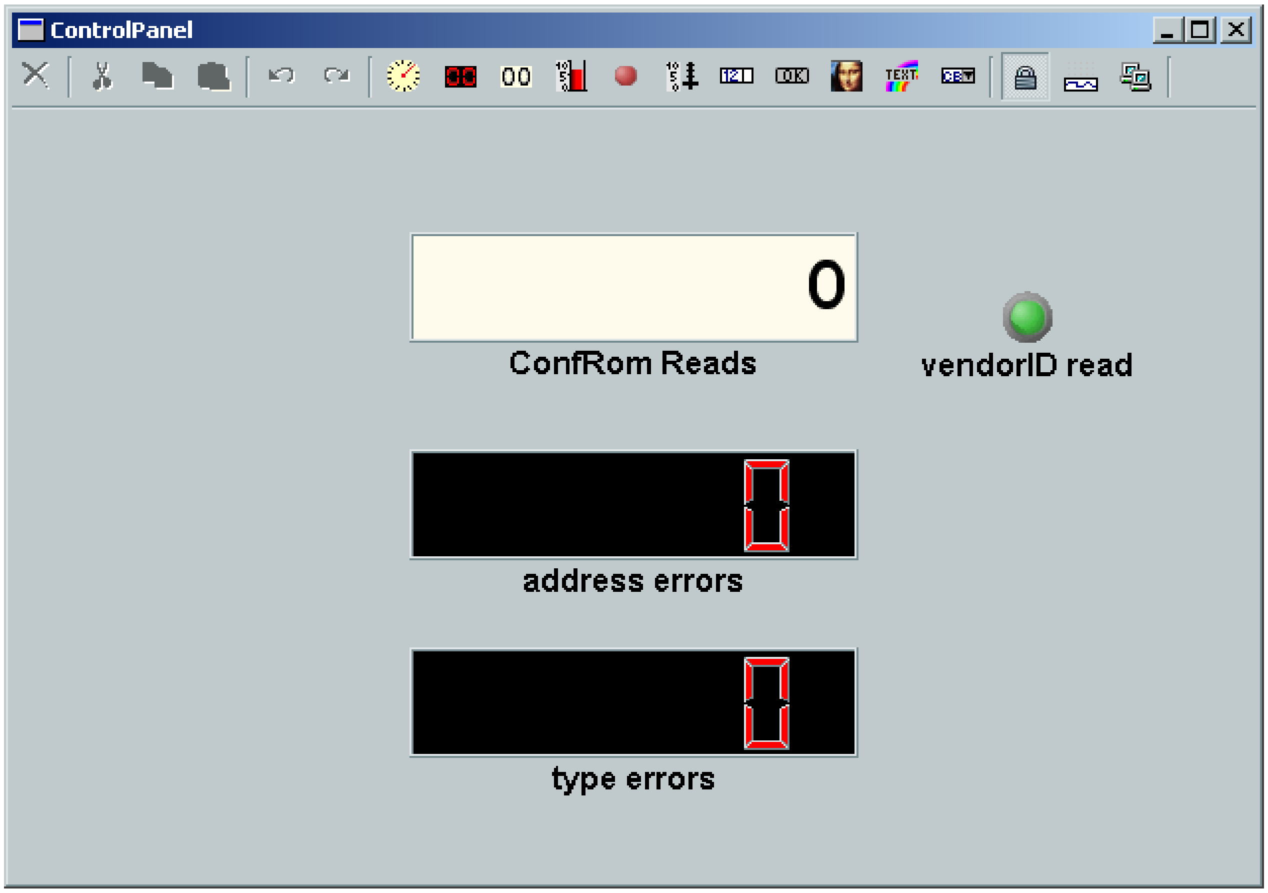
As the Analyzer stores its data in an internal buffer and the Analyzer software reads from this buffer it might happen that overflow occurs. This is indicated in the history graph by gaps labeled with an 'O' as can be seen in the picture below. The graph continues to plot data when the overflow situation has ended. To prevent overflow, increase the Scriptor memory on the Memory tab of the Settings dialog.



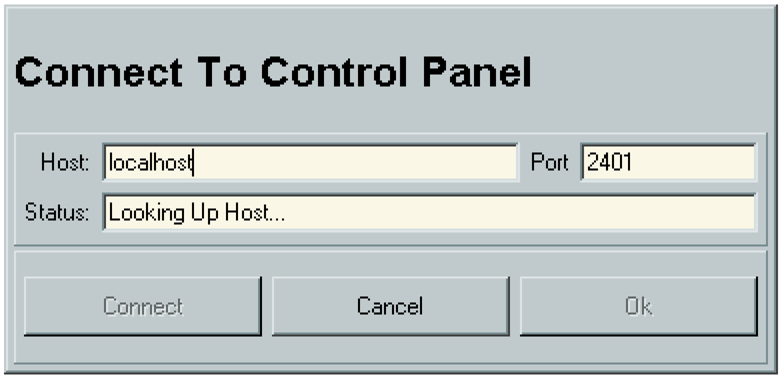
11.3.6. Client

The Analyzer Application has support for a remote client that can view the Control Panel that is controlled by the scriptor. The remote client can be found in the start menu.
When started it will look the same as the control panel in the Analyzer application but without the scriptor

attached. You can place the same controls on it as with the application. There is only one control that is not supported in this client and that is the FileBrowser.
If you created a control panel in the Analyzer application than you can export this control panel to file in the file menu if the scriptor. This file can then be opened by right clicking somewhere on the control panel and choose "Load Control Panel". As an example the control panel from the "SimConfRom.fss" script is exported and loaded in the client.



In the right top corner you can see an icon representing two computers, this icon will show the connection dialog like below.

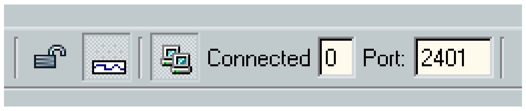

Connect To Control Panel

Host: localhost

Port 2401

Status: Looking Up Host...

This dialog has two input fields, the host it must try to connect to and the port number. Because the communication between the Analyzer application and the client uses TCP/IP you need an IP address or a computername and a port number. The Analyzer application only accepts connection when the same icon is selected.

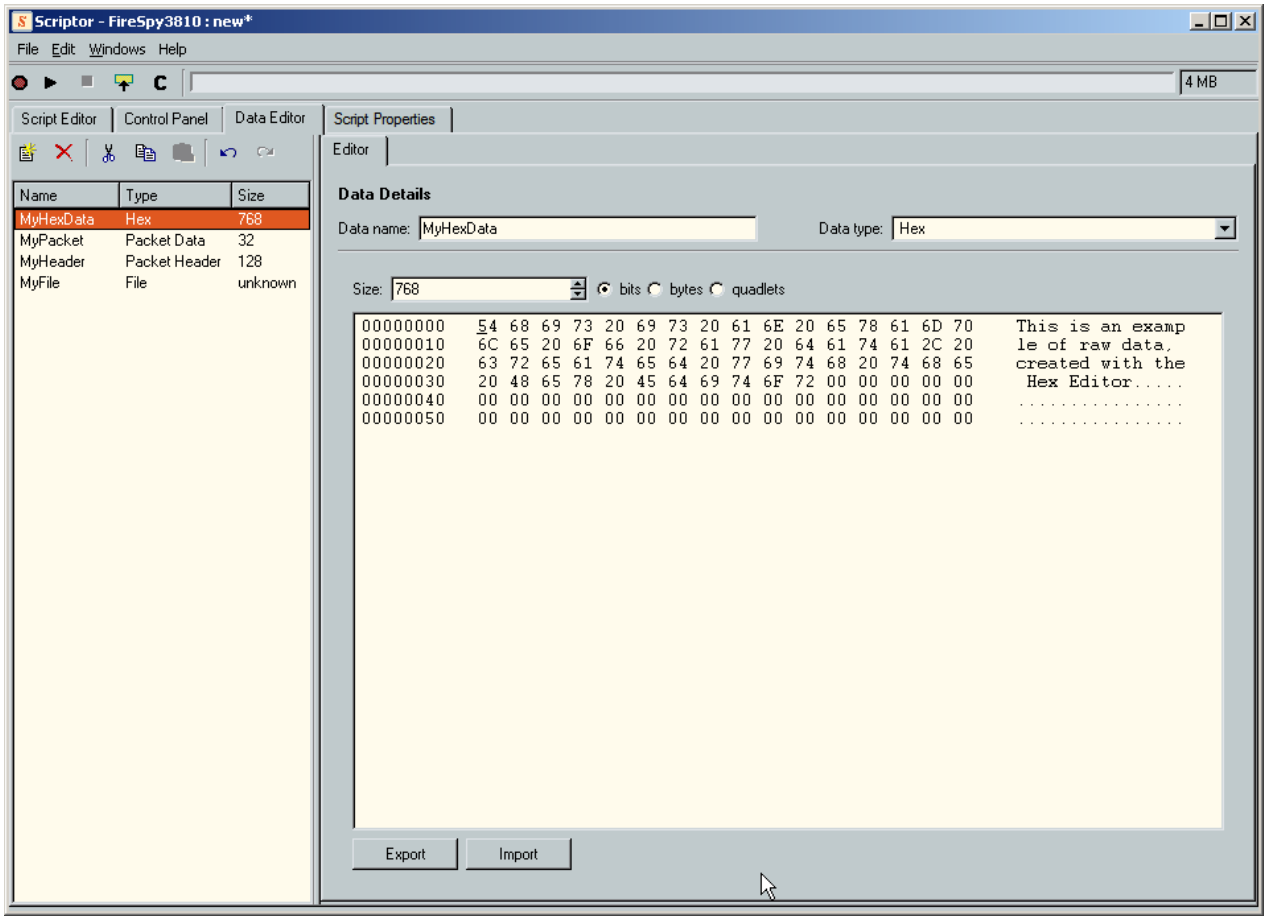


Next to this icon the application shows the number of clients connected. The port number of the application and the client can be freely chosen so that it is possible to have multiple applications running a server.

When the client is connected all control values that are send from the script are also send to the connected remote clients. This can be used to create one large control panel that is divided between multiple computers and/or monitors.

11.4. Data Editor

The data tab is used for editing data which can be used to set as packet data or as the header of a packet. The Data Editor is shown in the picture below. The left side shows a table with all the data objects that are defined, together with their properties. The name will be associated with a constant number by a a define statement in de current script. Therefore, when filling packets, the name of a data object can be used to refer to it. For data objects that contain more than 1 data block, two define statements (`<NAME>_FIRST` and `<NAME>_LAST`) will be created which you can use in the script to refer to your data.



Name	Type	Size
MyHexData	Hex	768
MyPacket	Packet Data	32
MyHeader	Packet Header	128
MyFile	File	unknown

Data Details

Data name: MyHexData

Data type: Hex

Size: 768 (● bits ○ bytes ○ quadlets)

```
00000000   54 68 69 73 20 69 73 20 61 6E 20 65 78 61 6D 70   This is an examp
00000010   6C 65 20 6F 66 20 72 61 77 20 64 61 74 61 2C 20   le of raw data,
00000020   63 72 65 61 74 65 64 20 77 69 74 68 20 74 68 65   created with the
00000030   20 48 65 78 20 45 64 69 74 6F 72 00 00 00 00 00    Hex Editor.....
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

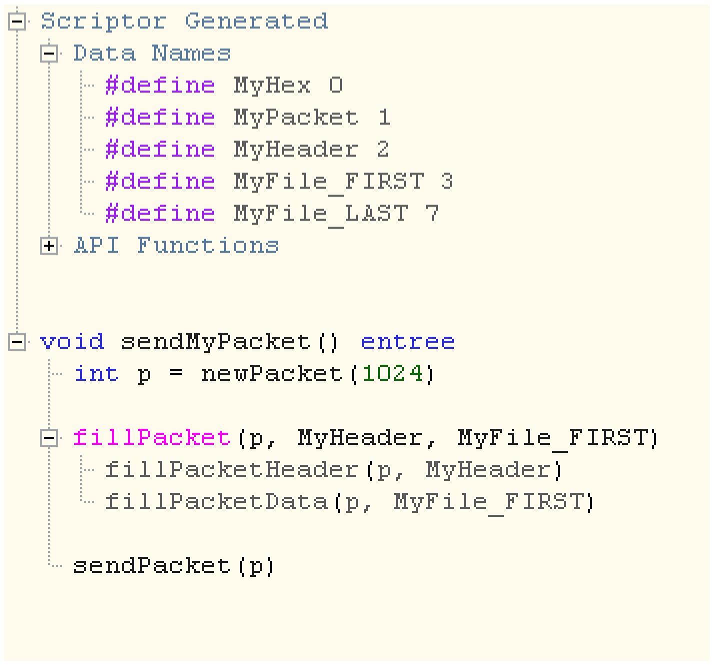
Adding a new data object

To add a new data object, click on the "New" button int the toolbar. By default an empty data object is created and its type is set to "None". Now, first select the type of data you would like to define by selecting the correct item in the "Data type" combo box. The corresponding data editor will be shown and depending on the type of data one may set the size in the "Size" spinbox. The number of bytes will automatically be updated accordingly. The different data editors will be explained below.

Removing a data object

To remove a data object from the script, click on the "Delete" button in the toolbar.

Using data objects in the script
In the picture below, a script is shown that contains some data blocks. The script only uses the first file data object as input for a packet. The packet header is also set from a data object. After the packet is filled with a header and some data, it is send once. The data object contents are described in the next section.



```
Scriptor Generated
  Data Names
    #define MyHex 0
    #define MyPacket 1
    #define MyHeader 2
    #define MyFile_FIRST 3
    #define MyFile_LAST 7
  API Functions


void sendMyPacket() entree
  int p = newPacket(1024)

  fillPacket(p, MyHeader, MyFile_FIRST)
    fillPacketHeader(p, MyHeader)
    fillPacketData(p, MyFile_FIRST)

  sendPacket(p)
```

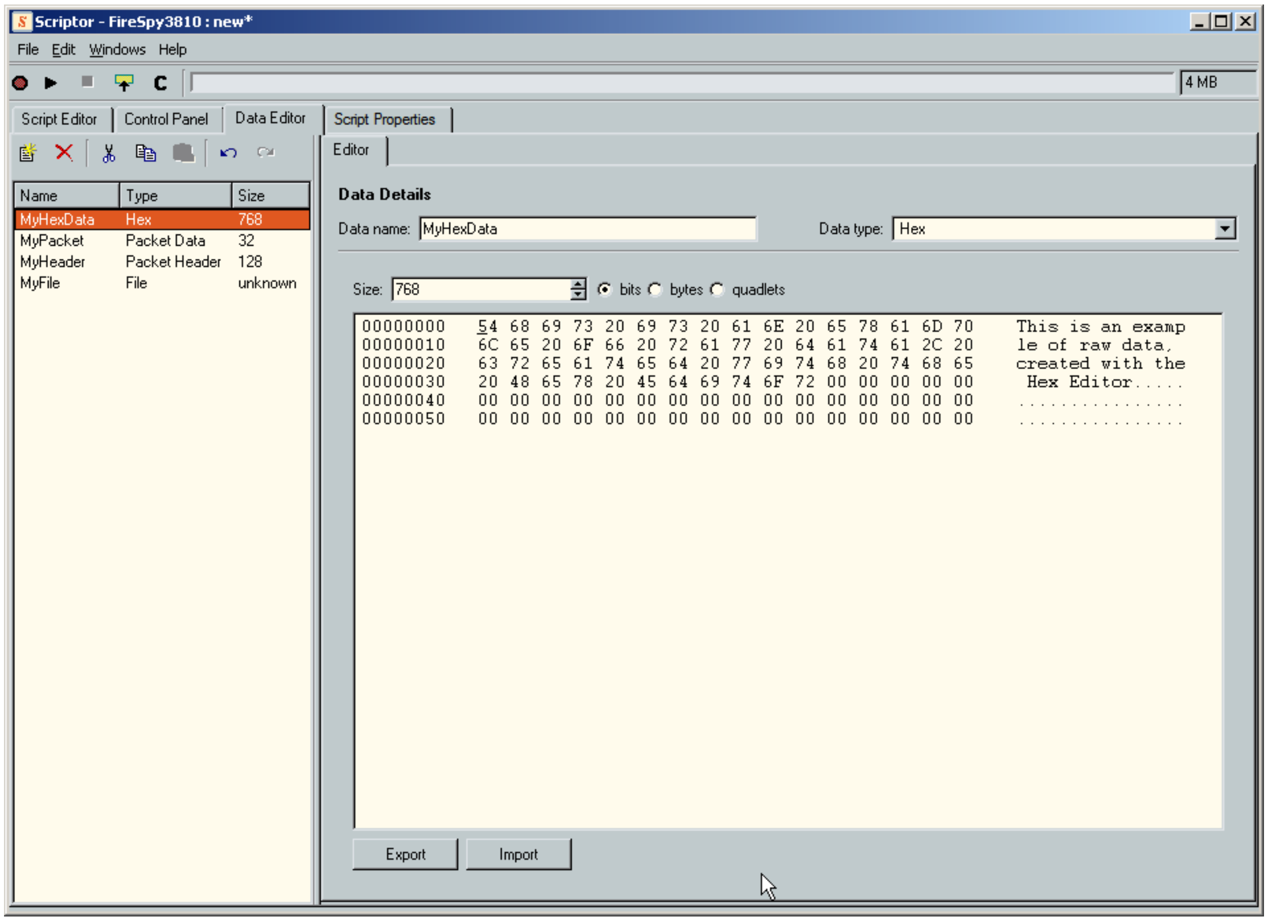
11.4.1. Toolbar

From the left to the right, the toolbar contains the following items:
- New Data Object
- Delete selected data object
- Cut selected data object
- Copy selected data object
- Paste copied data object
- Undo last action
- Redo last undone action

11.4.2. Hex Data Editor

This type of data editor is shown in the picture below. The editor is divided in three columns. The first column displays the hexadecimal index in the data of the first quadlet of that row. The second column displays the data in Hex format, grouped by Quadlets. The third column is the ascii representation of the quadlet values. The data can be edited in both the second and the third column. Note that the last quadlet is masked if the number of bits is not equal to an exact number of quadlets.

Data in this editor can be imported and exported from/to files by clicking on the "Import" and "Export" buttons. Make sure to first set the size of the data before loading from file.



Name	Type	Size
MyHexData	Hex	768
MyPacket	Packet Data	32
MyHeader	Packet Header	128
MyFile	File	unknown

Data Details

Data name: MyHexData

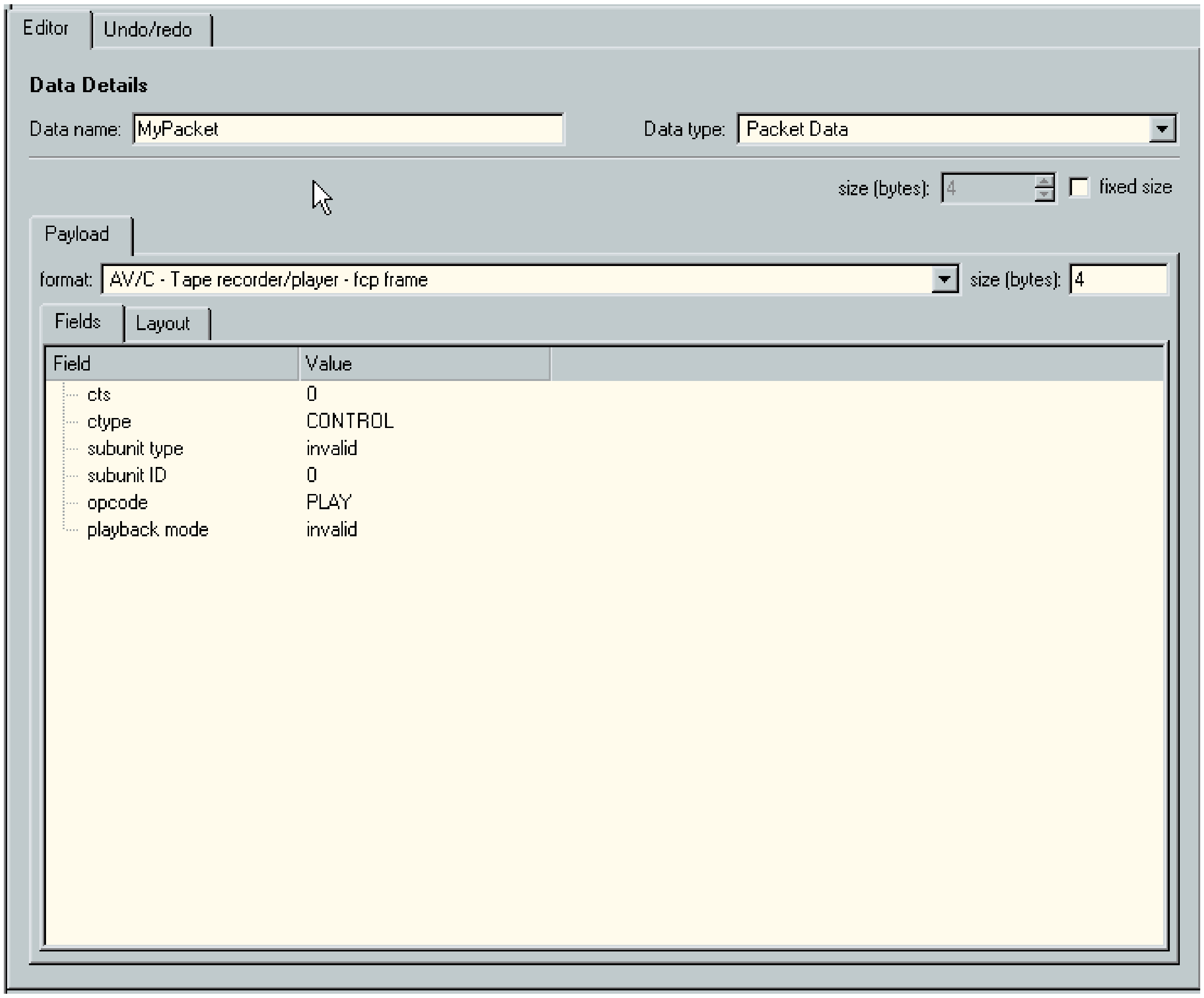
Data type: Hex

Size: 768 [✓ bits] bytes quadlets

```
00000000   54 68 69 73 20 69 73 20 61 6E 20 65 78 61 6D 70    This is an examp
00000010   6C 65 20 6F 66 20 72 61 77 20 64 61 74 61 2C 20    le of raw data,
00000020   63 72 65 61 74 65 64 20 77 69 74 68 20 74 68 65    created with the
00000030   20 48 65 78 20 45 64 69 74 6F 72 00 00 00 00 00     Hex Editor.....
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

11.4.3. Packet Data Editor

When the data type is set to "Packet data", a data editor will appear as shown in the picture below. The "format" combo can be used to set the packet format. Selecting a format will update the number of bits to the default value for the selected format and display the corresponding field lay out. For some formats, changing the value of some field will also change the size of the format. The packet editor will reflect this. If you want to use a different packet size than the automatically calculated size, check the box "fixed size" and set the size to an arbitraty number. Field values can be set by clicking in the "Value" column and by providing a value.



Data Details

Data name: MyPacket

Data type: Packet Data

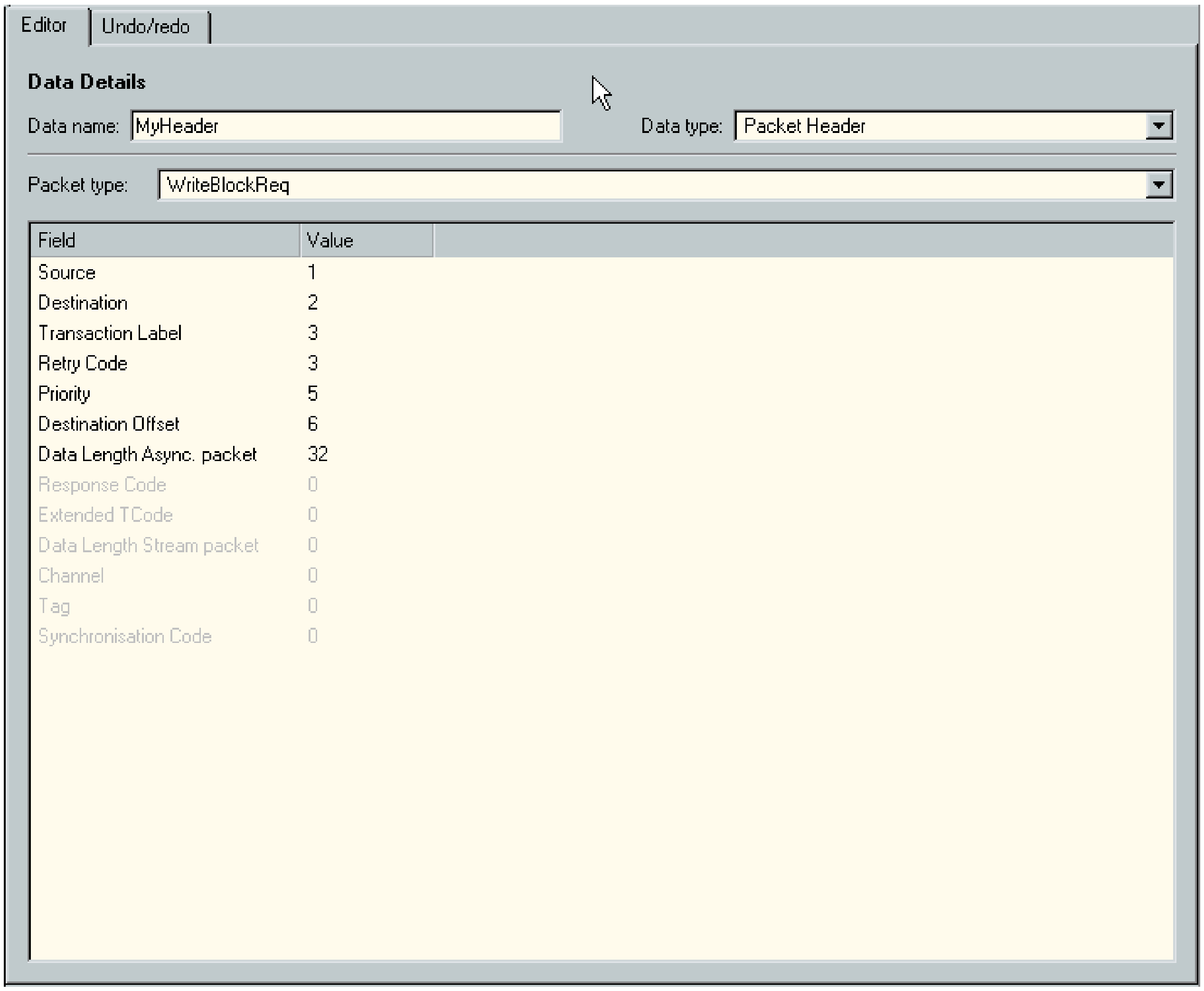
size (bytes): 4 ☐ fixed size

format: AV/C - Tape recorder/player - fcp frame size (bytes): 4

Field	Value
cts	0
ctype	CONTROL
subunit type	invalid
subunit ID	0
opcode	PLAY
playback mode	invalid

11.4.4. Packet Header Editor

When the data type is set to "Packet Header", a data editor will appear as shown in the picture below. For this type of data editor, the number of bits is determined by the selected packet type. Also, the available header fields are determined by the selected packet type. A header field can be changed by clicking in the "Value" column.



Data Details

Data name: MyHeader

Data type: Packet Header

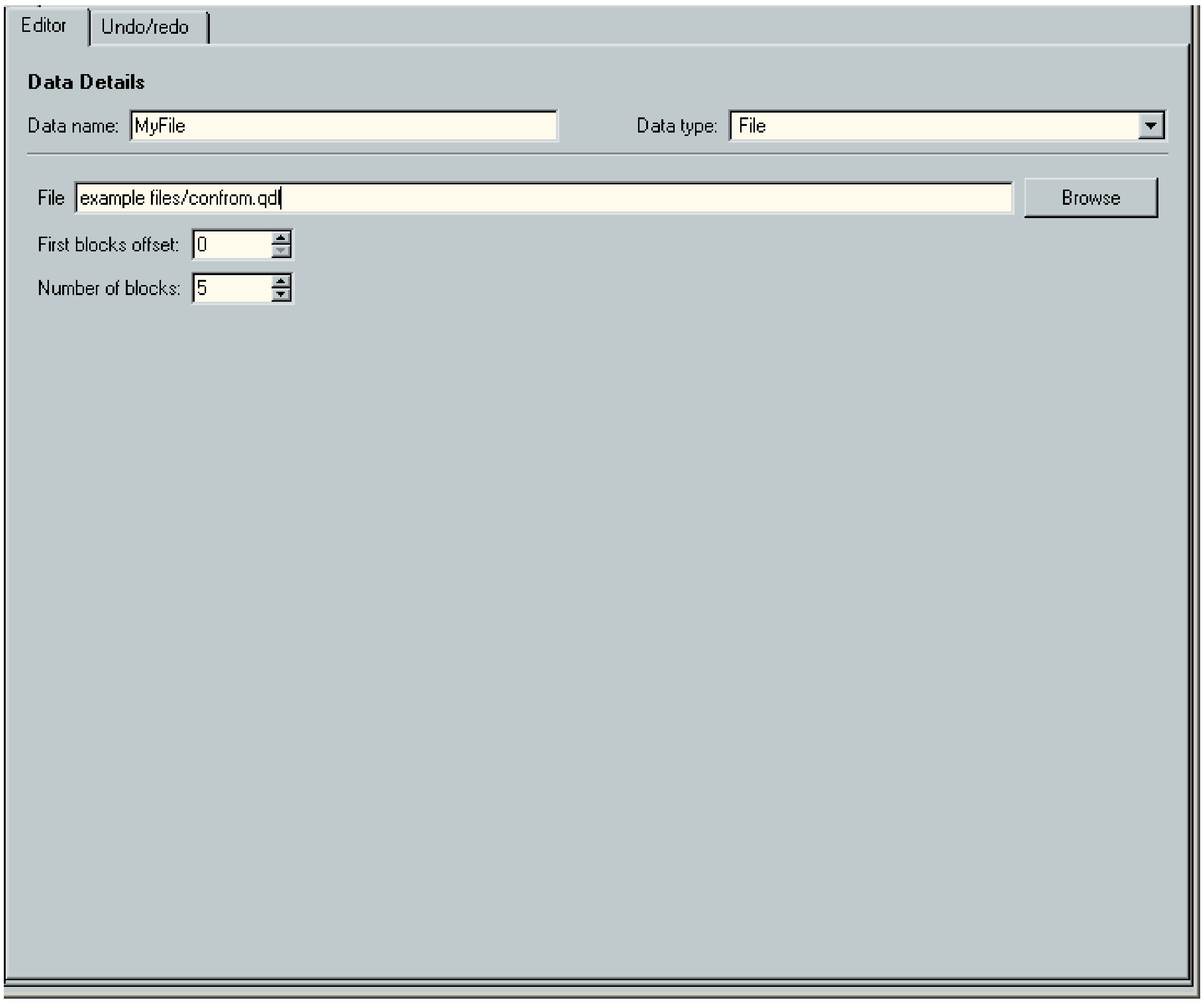
Packet type: WriteBlockReq

Field	Value
Source	1
Destination	2
Transaction Label	3
Retry Code	3
Priority	5
Destination Offset	6
Data Length Async. packet	32
Response Code	0
Extended TCode	0
Data Length Stream packet	0
Channel	0
Tag	0
Synchronisation Code	0

11.4.5. File Data Editor

When the data type is set to "File", a data editor will appear as shown in the picture below. To select the file, click on the "Browse" button on the right. It is possible to load from Quadlet files "*.qdl" and from Hex data files "*.hex". File names can have absolute paths like in the picture below and also relative paths. A relative path uses the path where the script is saved as its offset.

A file may contain an arbitrary number of data blocks. Use the "First block offset" to select at what block index in the file the editor should start reading. Set "Number of blocks" to the number of consecutive blocks you want to load from the file. Note that an error will occur when trying to read beyond the file limits. This error will be thrown when uploading the script to the Analyzer.



Data Details

Data name: MyFile

Data type: File

File: example files/confrom.qdl

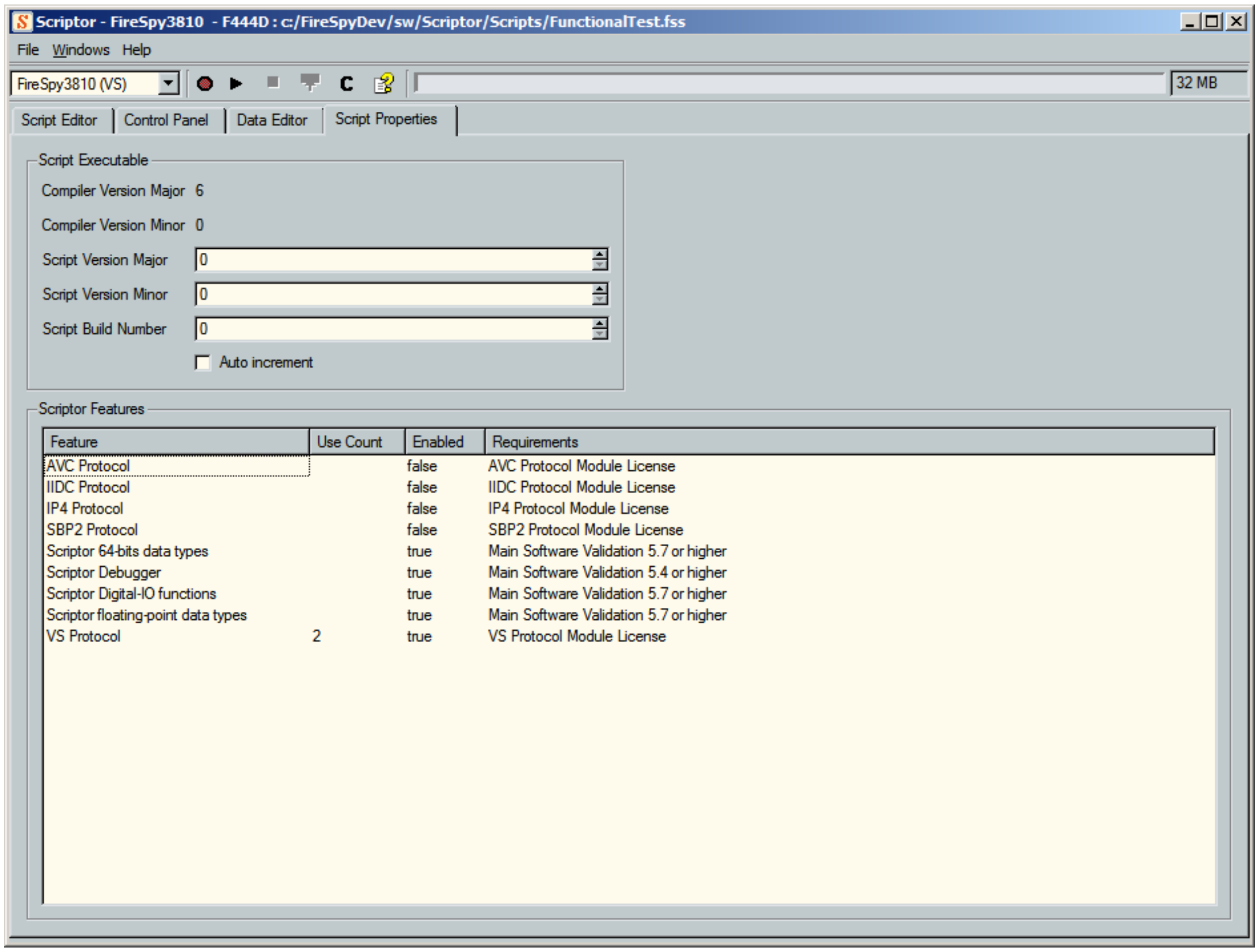
First blocks offset: 0

Number of blocks: 5

11.5. Script Properties

The Script Properties tab provides information about the current script. All values defined here are saved both in the script source and the generated executable for later use. These values could for example be used to verify if an executable and source file match and which features are required to run the compiled script.

The following picture shows the Script Properties tab.


Scriptor - FireSpy3810 - F444D : c:/FireSpyDev/sw/Scriptor/Scripts/FunctionalTest.fss

Script Executable

Compiler Version Major 6

Compiler Version Minor 0

Script Version Major 0

Script Version Minor 0

Script Build Number 0

☐ Auto increment

Scriptor Features

Feature	Use Count	Enabled	Requirements
AVC Protocol		false	AVC Protocol Module License
IIDC Protocol		false	IIDC Protocol Module License
IP4 Protocol		false	IP4 Protocol Module License
SBP2 Protocol		false	SBP2 Protocol Module License
Scriptor 64-bits data types		true	Main Software Validation 5.7 or higher
Scriptor Debugger		true	Main Software Validation 5.4 or higher
Scriptor Digital-IO functions		true	Main Software Validation 5.7 or higher
Scriptor floating-point data types		true	Main Software Validation 5.7 or higher
VS Protocol	2	true	VS Protocol Module License

11.5.1. Script Executable

The Script Executable group of controls on the Script Properties tab page contains different kinds of version numbers. Some of the values will be generated at compile time while others are user defined. Whenever the script is compiled, all values will be stored in the executable file. By using the API it is possible to extract these values from the executable file to verify them.

The following version numbers are available:

- Compiler Version (major and minor): Will be filled in by the Analyzer software and represents the internal version of the compiler used to compile the script.
- Script Version (major and minor): This is a user-definable value that can later on be used to check the version of a script.
- Script Build Number: When Auto Increment is enabled, this number will be incremented each time the script is compiled. The incremented value will be stored in the Script executable file.

11.5.2. Scriptor Features

The Scriptor Features table contains an overview of all Analyzer features that are under license control with information about use and availability. Whenever a script is compiled and some of the licensed features are used in the script, the person who wants to run the executable will also need sufficient license validation for those functions.

Note that these values are only updated at the moment the script is compiled.

The table contains the following fields:

- Feature: The description of the feature defined in a row
- Used Count: The number of times this specific feature is used by the current compiled script.

- Enabled: This field indicates whether the feature is currently enabled or not
- Requirements: This field describes the (license/validation) requirements for this feature.

11.6. Examples

11.6.1. Simple Sending

This example shows how to define a packet using the build-in data editor and how to send a multiple of this packet each time a bus reset is detected. There is no particular reason to wait for a bus reset before starting to send the packets. It is implemented this way to have more interactivity with the script. Sending packets can now be controlled by externally applying bus resets.

11.6.1.1. How to use it

After loading the file "simpleSend.fss", a script will be shown as in the picture below. This script contains a couple of defined data blocks and one main function.

```
// Main function
void main () entry
        int p = newPacket ( 1000 )
        fillPacket ( p, DATA_ID_1, DATA_ID_2 )
                fillPacketHeader ( p, DATA_ID_1, 0 , - 1 )
                fillPacketData ( p, DATA_ID_2, 0 , - 1 )


            while true
                waitBusReset ()

                repeat 100
                        sendPacket ( p )
```

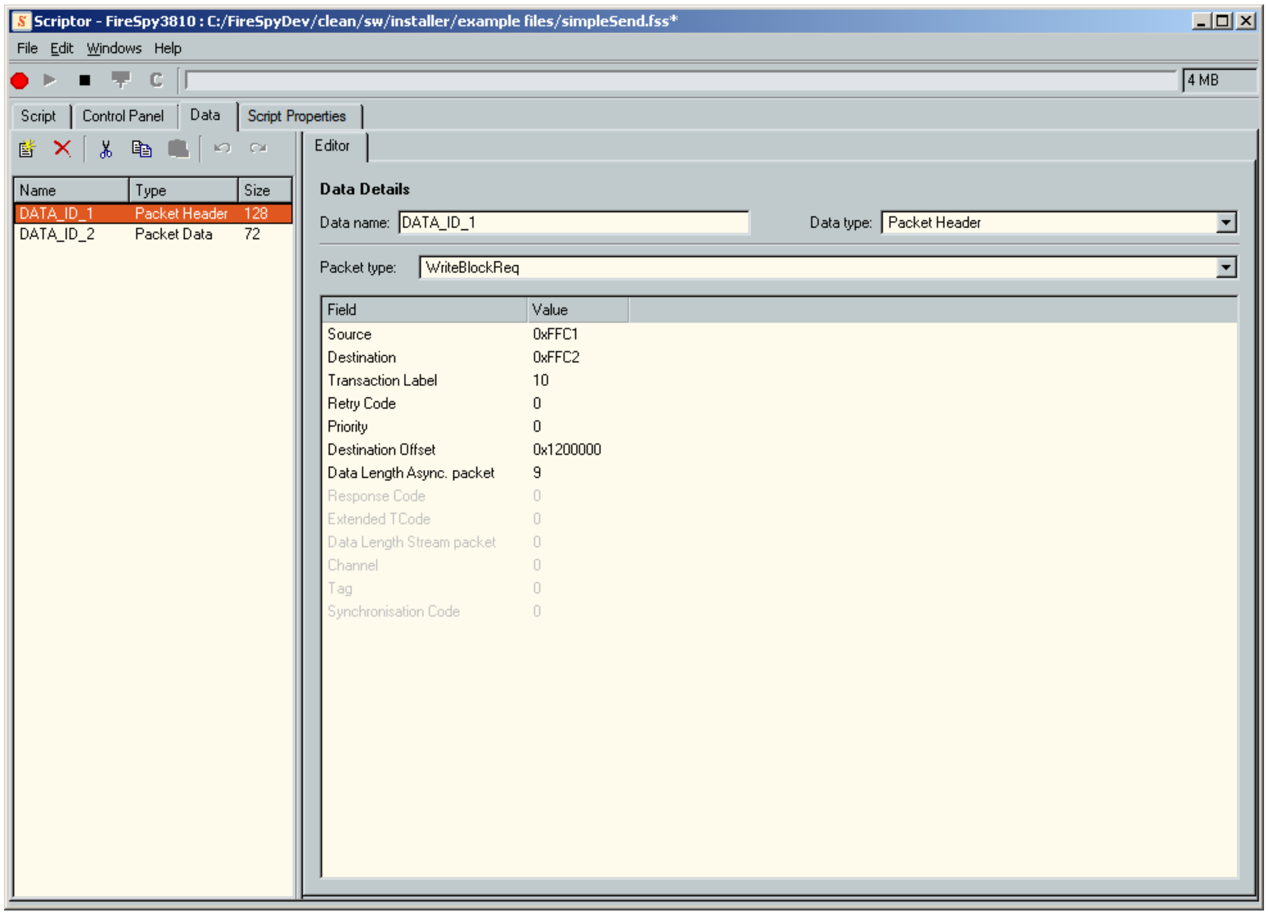
To see the script in action, proceed with the following steps:

1. Open the recorder and start a new recording
2. Upload and start the script from the Scriptor
3. Open the Commander and generate a bus reset with the [R] button
4. Stop the running script from the Scriptor
5. Stop the recorder and download the recording

Now if you examine the data recorded with the recorder, you can see that first a bus reset is detected and then, the same packet is sent a hundred times in a row.

11.6.1.2. Details

The function main first creates a packet with a maximum size of 1000 quadlets. Then the macro "fillPacket" is called to fill the packet with the header data and packet data which have been defined on the data tab. The corresponding data definitions on the data tab can be seen in the following two pictures:


Scriptor - FireSpy3810 : C:/FireSpyDev/clean/sw/installer/example files/simpleSend.fss*
File Edit Windows Help
4 MB
Script
Control Panel
Data
Script Properties
Editor

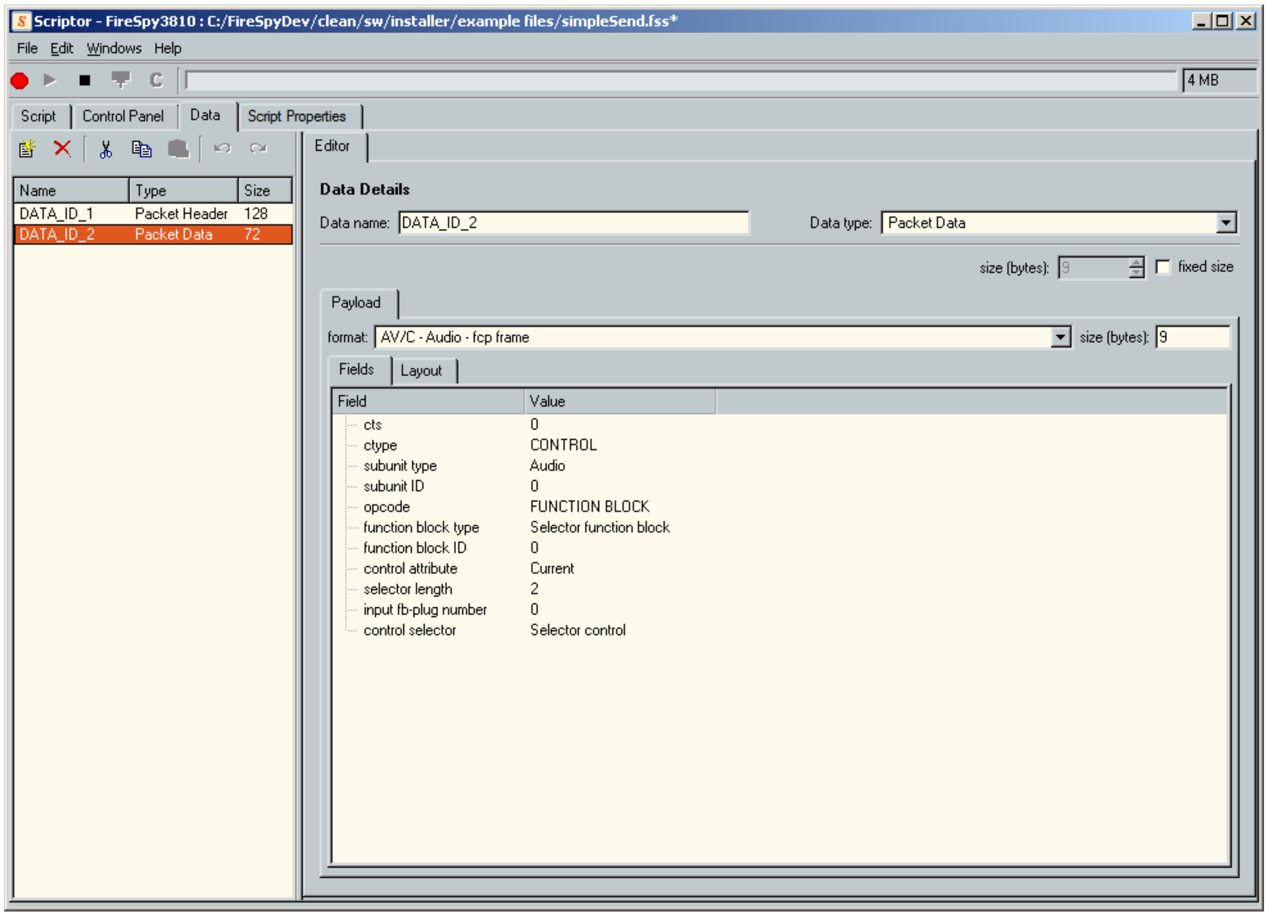
Name	Type	Size
DATA_ID_1	Packet Header	128
DATA_ID_2	Packet Data	72

Data Details

Data name: DATA_ID_1 Data type: Packet Header

Packet type: WriteBlockReq

Field	Value
Source	0xFFC1
Destination	0xFFC2
Transaction Label	10
Retry Code	0
Priority	0
Destination Offset	0x1200000
Data Length Async. packet	9
Response Code	0
Extended TCode	0
Data Length Stream packet	0
Channel	0
Tag	0
Synchronisation Code	0



Name	Type	Size
DATA_ID_1	Packet Header	128
DATA_ID_2	Packet Data	72

Data Details

Data name: DATA_ID_2 Data type: Packet Data

size (bytes): 9 fixed size

format: AV/C - Audio - fcp frame size (bytes): 9

Field	Value
cts	0
ctype	CONTROL
subunit type	Audio
subunit ID	0
opcode	FUNCTION BLOCK
function block type	Selector function block
function block ID	0
control attribute	Current
selector length	2
input fb-plug number	0
control selector	Selector control

11.6.2. SimConfRom

When running this script, all reads of the Analyzer configuration Rom will be catched and a corresponding value from a quadlet file will be returned. This quadlet file was created by using the Memory R/W function of the Commander of the Analyzer. The configuration Rom of a desktop computer (with FireWire card) was read and stored to a quadlet file. As a result, when running this script and connecting the Analyzer to a computer, the computer reads the configuration Rom of the Analyzer and will think that another computer was connected (which actually is the Analyzer) and it probably will display some message that a network connection has been added because in the simulated configuration Rom defines support for IP.

11.6.2.1. How to use it

This script waits for asynchronous packets and checks if they are configuration-rom read requests. If so, it checks for a specific address range and when valid it reads the data from a file and sends the file contents as its response. In addition it also sends information, like how many reads have occured, to the control panel, which displays these values in a nice graphical way.The script is shown in the picture below.

To see the script in action, carry out the following steps:

1. Make sure the Analyzer is only connected to the computer by its USB cable
2. Upload and start the script
3. Select the "Control Panel" to see the scripts output
4. Connect the Analyzer to the computer by using a FireWire cable
5. The Control Panel should now show some activity
6. Possibly the connected computer indicates that a new network connection is available

```
void main () entry
        #define RESPCODE_COMPLETE 0
        #define RESPCODE_ADDR_ERROR 7
        #define RESPCODE_TYPE_ERROR 6
        #define TCODE_READREQ 4
        int b
        int cnt = 0 , cnt_addr_errors = 0 , cnt_type_errors = 0

        int req = newPacket ( 100 )
        int resp = newPacket ( 100 )

        setPacketHeaderSize ( resp, 4 )
        setPacketHeaderQuadlet ( resp, 0 , 0 )
        setPacketHeaderQuadlet ( resp, 1 , 0 )
        setPacketHeaderQuadlet ( resp, 2 , 0 )
        setPacketHeaderField ( resp, 0 , 24 , 4 , 6 )

        setControlValue ( 1 , 0 )
        setControlValue ( 2 , 0 )
        setControlValue ( 3 , 0 )
        setControlValue ( 10 , 0 )

        while true
                int requestor, node, label
                int tcode = recvAsyncPacket ( req, - 1 , - 1 , 1 , 0 )

                getPacketHeaderFields ( req, requestor, node, label )
                        requestor = getPacketHeaderField ( req, 1 , 0 , 16 )
                        node = getPacketHeaderField ( req, 0 , 0 , 16 )
                        label = getPacketHeaderField ( req, 0 , 16 , 6 )

                    setPacketHeaderFields ( resp, node, requestor, label )
                        setPacketHeaderField ( resp, 1 , 0 , 16 , node )
                        setPacketHeaderField ( resp, 0 , 0 , 16 , requestor )
                        setPacketHeaderField ( resp, 0 , 16 , 6 , label )
```

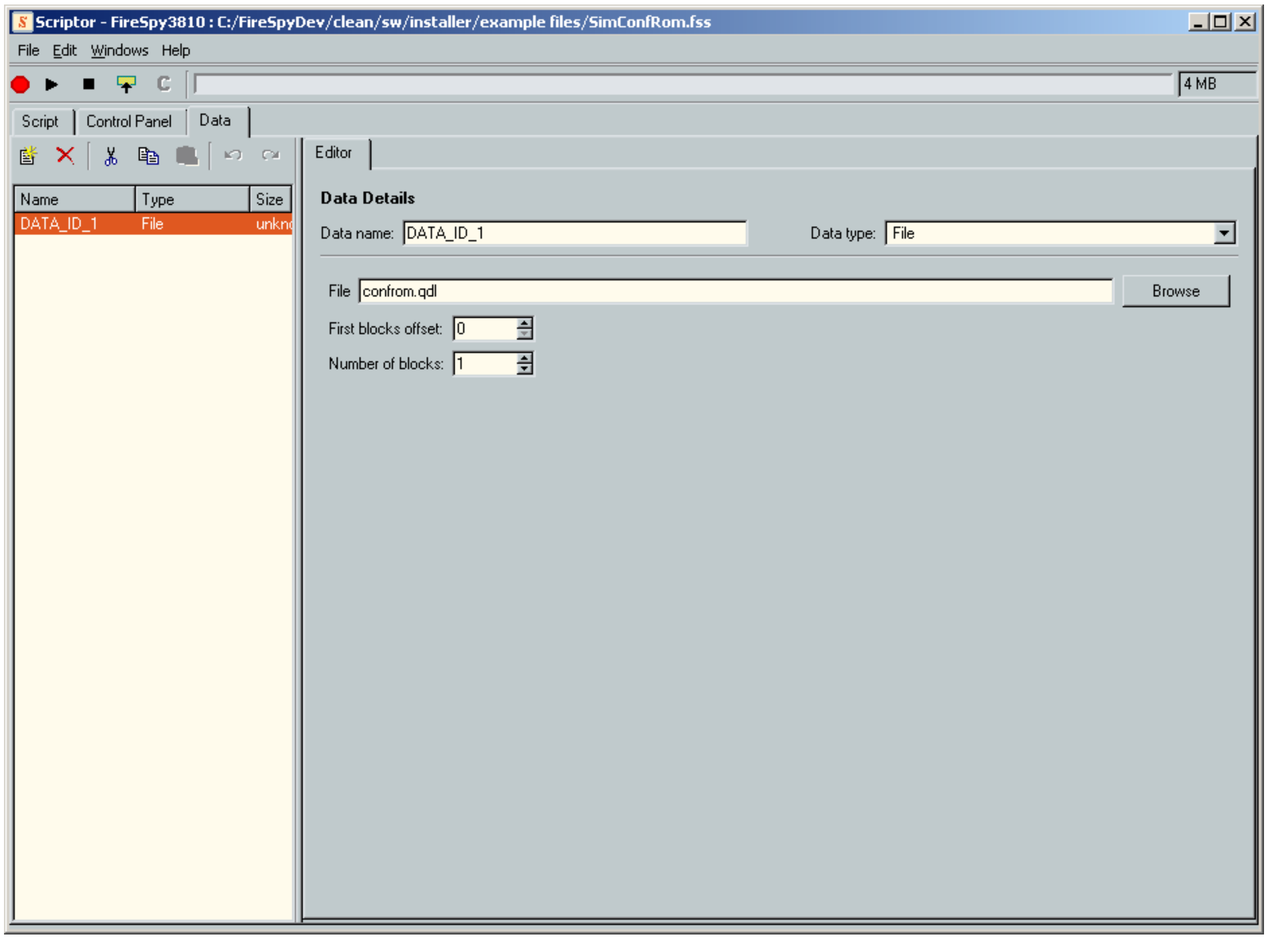
```
if tcode == TCODE_READREQ
    int q1 = getPacketHeaderField ( req, 1 , 16 , 16 )
    int q2 = getPacketHeaderQuadlet ( req, 2 )

    if q1 == 0xFFFF && ( q2 & 0xFFFFFC00 )== 0xF0000400
            int addr = ( q2 & 0x3FF )>> 2

            setPacketHeaderFields ( resp, RESPCODE_COMPLETE )
                    setPacketHeaderField ( resp, 1 , 16 , 4 ,
                    RESPCODE_COMPLETE )

                writePacketHeader ( resp, 3 ,DATA_ID_1_FIRST ,addr, 1 )
                sendPacket ( resp )
                cnt = cnt + 1
                setControlValue ( 1 ,cnt )

                if addr == 3
                    setControlValue ( 10 , 1 )


        else
            setPacketHeaderFields ( resp, RESPCODE_ADDR_ERROR )
                    setPacketHeaderField ( resp, 1 , 16 , 4 ,
                    RESPCODE_ADDR_ERROR )

                sendPacket ( resp )
                cnt_addr_errors = cnt_addr_errors + 1
                setControlValue ( 2 ,cnt_addr_errors )


else
    if tcode >= 0
            setPacketHeaderFields ( resp, RESPCODE_TYPE_ERROR )
                    setPacketHeaderField ( resp, 1 , 16 , 4 ,
                    RESPCODE_TYPE_ERROR )

                sendPacket ( resp )
                cnt_type_errors = cnt_type_errors + 1
                setControlValue ( 3 ,cnt_type_errors )
```

11.6.2.2. Details

This script contains one data definition as can be seen by the only child of the "Data defines" node at the top. The script again contains one function called main. This function starts by defining a couple of constants by using "#define" statements. Then it declares and initializes a couple of integer variables of which "req" and "resp" are initialized to id numbers of new packets. This is done with the "newPacket" API function.

Now that two packets are setup, their header fields are set to some initial values by a couple of API calls. (setPacketHeaderSize, setPacketHeaderQuadlet and setPacketHeaderField).

Then a couple of calls to the API function "setControlValue" are made to give the controls their initial values. The control panel the controls belong to can be seen in the picture below.

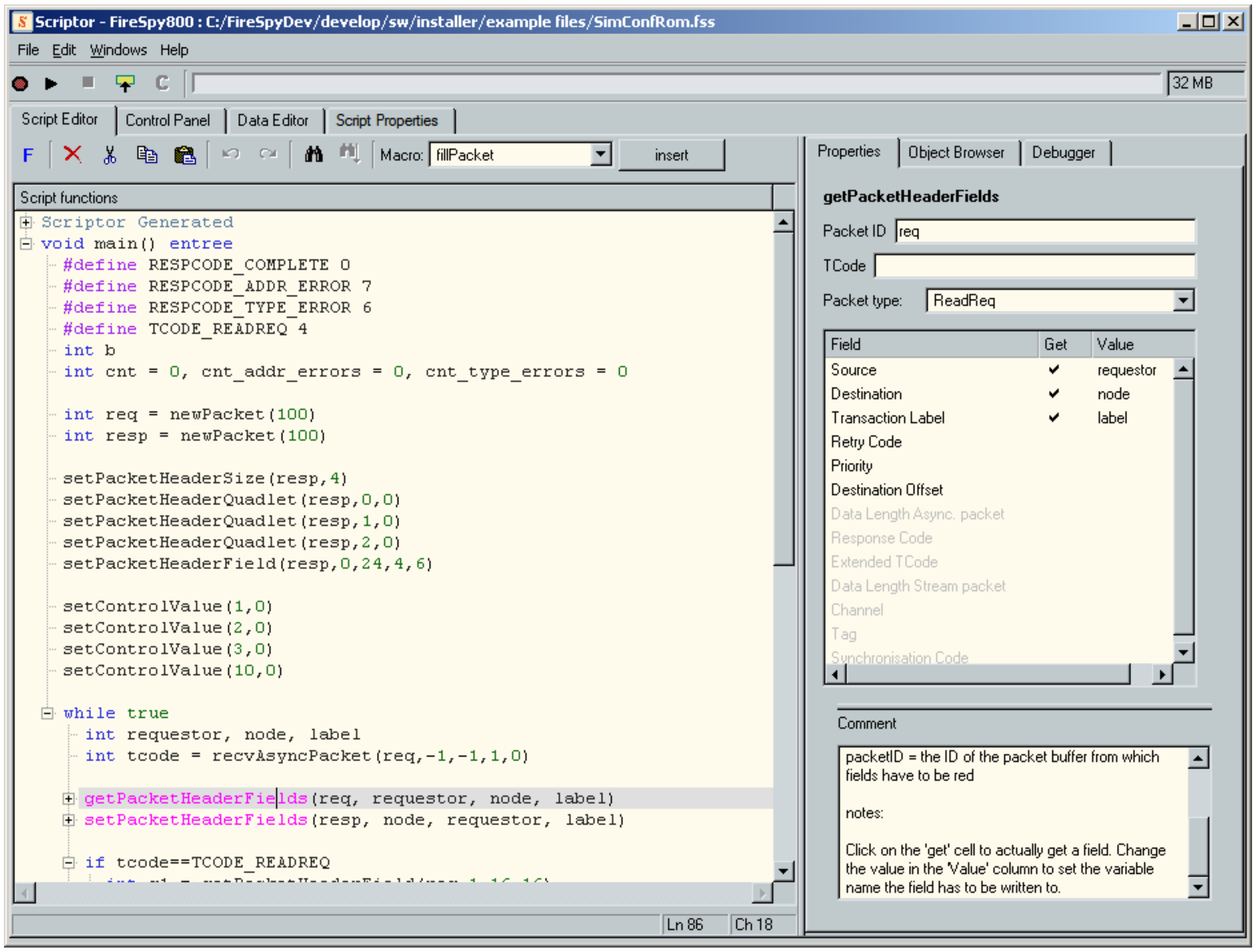

Scriptor - FireSpy3810 : C:/FireSpyDev/clean/sw/installer/example files/SimConfRom.fss
File Edit Windows Help
4 MB
Script
Control Panel
Data
Editor

Name	Type	Size
DATA_ID_1	File	unkn

Data Details

Data name: DATA_ID_1
Data type: File
File: confrom.qdl
First blocks offset: 0
Number of blocks: 1

Now that all has been initialized, the script enters an indefinite loop. (while 1) A loop iteration starts by waiting for an asynchronous packet. When a packet is received, a macro is encountered with the name "getPacketHeaderFields". See the picture below.



```
Scriptor Generated
void main() entree
  #define RESPCODE_COMPLETE 0
  #define RESPCODE_ADDR_ERROR 7
  #define RESPCODE_TYPE_ERROR 6
  #define TCODE_READREQ 4
  int b
  int cnt = 0, cnt_addr_errors = 0, cnt_type_errors = 0

  int req = newPacket(100)
  int resp = newPacket(100)

  setPacketHeaderSize(resp,4)
  setPacketHeaderQuadlet(resp,0,0)
  setPacketHeaderQuadlet(resp,1,0)
  setPacketHeaderQuadlet(resp,2,0)
  setPacketHeaderField(resp,0,24,4,6)

  setControlValue(1,0)
  setControlValue(2,0)
  setControlValue(3,0)
  setControlValue(10,0)

  while true
    int requestor, node, label
    int tcode = recvAsyncPacket(req,-1,-1,1,0)

    getPacketHeaderFields(req, requestor, node, label)
    setPacketHeaderFields(resp, node, requestor, label)

    if tcode==TCODE_READREQ
```

getPacketHeaderFields

Packet ID: req

TCode:

Packet type: ReadReq

Field	Get	Value
Source	✓	requestor
Destination	✓	node
Transaction Label	✓	label
Retry Code		
Priority		
Destination Offset		
Data Length Async. packet		
Response Code		
Extended TCode		
Data Length Stream packet		
Channel		
Tag		
Synchronisation Code		

Comment

packetID = the ID of the packet buffer from which fields have to be red

notes:

Click on the 'get' cell to actually get a field. Change the value in the 'Value' column to set the variable name the field has to be written to.

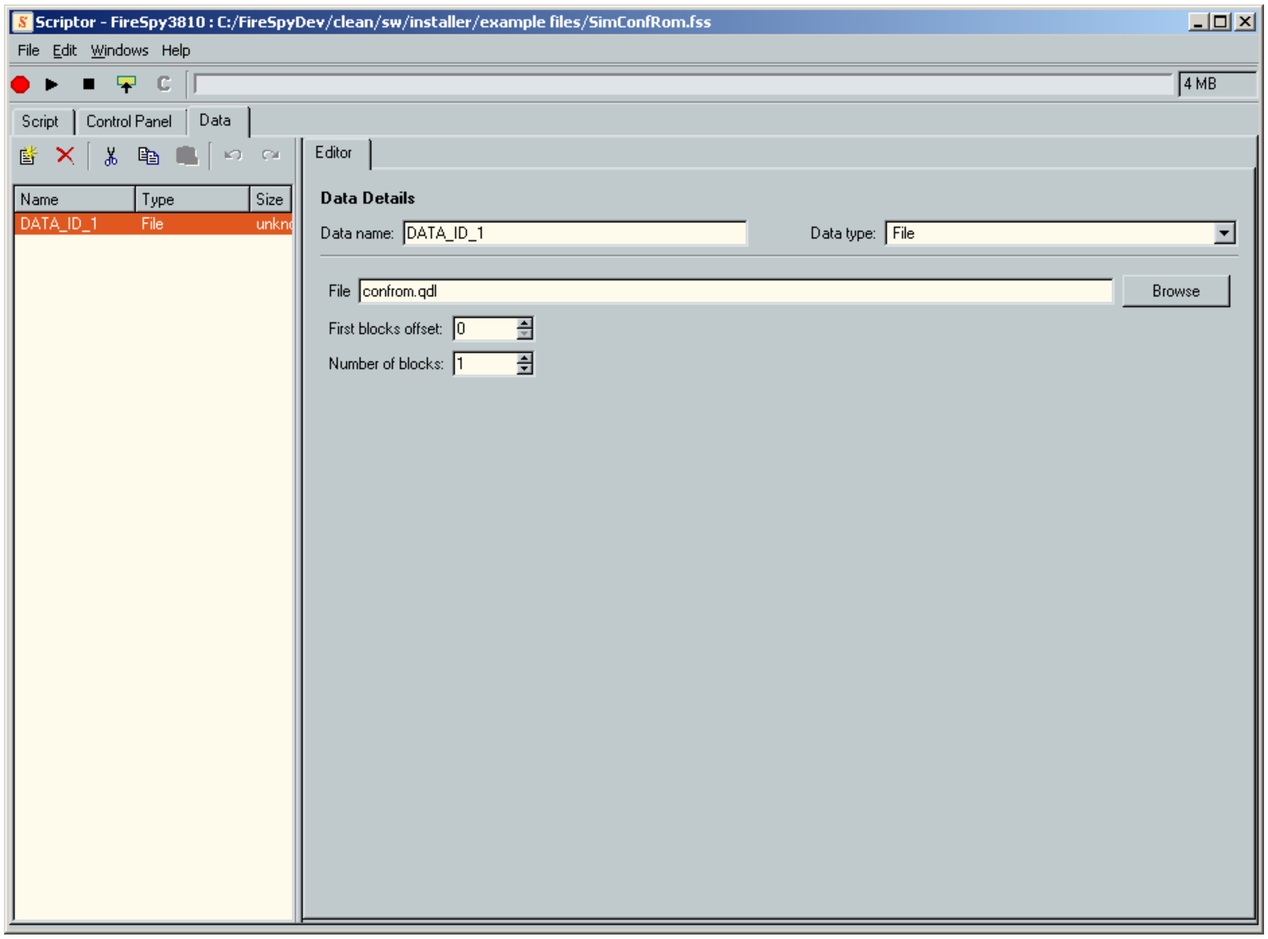
This macro is filled in to get the values of "Source", "Destination" and "Transaction label" for a "ReadReq" packet. The values should respectively be set to "requestor", "node" and "label". As can be seen in the script, the macro accomplishes this by generating three calls to API functions. These calls to "getPacketHeaderFields" are indicated with a gray color, which means that they are read-only. This is because those statements are generated by the macro and if the user would like to change them, the user should change the properties in the corresponding macro editor.

After reading the header fields from the "req" packet, another macro call fills the "resp" packet with the correct header values.

Then the value of tcode is checked. If the value is a read request, the script continues on the line below, otherwise it jumps to the else clause, which contains some constructs to create and return an error packet. The control "type errors" on the control panel is also updated.

If the tcode was a read request, the script retreives two values "q1" and "q2" from the header. If these values are within a specific range, the script continues on the next line. Otherwise a jump is made to the else clause. The else clause contains some constructs to create and return an error packet. The control "address errors" on the control panel is also updated.

If q1 and q2 are within the correct range, first a macro setPacketHeaderFields is filled such that the response code is set to RESPONSE_COMPLETE. Then the API function "writePacketHeader" is called to set the packet header with data from a file. The file data is referred to by the name "DATA_ID_1_FIRST". The define statement ending in "_FIRST" always refers to the first datablock that is read from a file and one ending in "_LAST" refers to the last datablock which should be read from file. The numbers in between can be used to access the file data blocks in between. The data definitions for this script can be seen in the following picture:


Scriptor - FireSpy3810 : C:/FireSpyDev/clean/sw/installer/example files/SimConfRom.fss
File Edit Windows Help
4 MB
Script
Control Panel
Data
Editor
Name
Type
Size
Browse

Name	Type	Size
DATA_ID_1	File	unkno

Data Details

Data name: DATA_ID_1

Data type: File

File confrom.qdl

First blocks offset: 0

Number of blocks: 1

Note that the data is read from file during script upload from the specified file. When the script is running on the Analyzer, the file data is already in Analyzer memory.

After reading the data from file, the packet is sent and the controls on the control panel are updated.

11.6.3. Erroneous Sending

For testing purposes it sometimes is useful to be able to create and send packets that contain errors. The script explained on this page has precisely this purpose. The script can send packets that contain CRC errors and incorrect data sizes on all Analyzernodes, both in synchronous and asynchronous mode.

11.6.3.1. How to use it

Control Panel Contents

After loading the file "erroneousSend.fss" and clicking on the Control Panel tab, a Control Panel will be shown as in the picture below. The Control Panel contains the following controls:

Buttons Node A, Node B and Node C

These buttons can be pressed and if the script is running, the buttons will remain down if they are enabled and come back up if they are disabled. When one or more of these buttons are enabled, the packets will be sent on the corresponding Analyzer nodes.

Buttons "Data CRC error" and "Incorrect Data length"

These buttons can be pressed and if the script is running, the buttons will remain down if they are enabled and come back up if they are disabled. If the data crc button is enabled, all packets that are sent will have an incorrect data CRC value. If the incorrect data length button is enabled, the header field "Data size Stream Packet" will be set one byte bigger than the size of the actual data.

Number of packets

This is the exact number of packets that will be sent if one of the start buttons is pressed.

Channel
This is the channel the packets will be sent on.

Frame offset
This field is only used if the packets are sent in synchronous mode and sets the time offset within a frame each packet is sent.

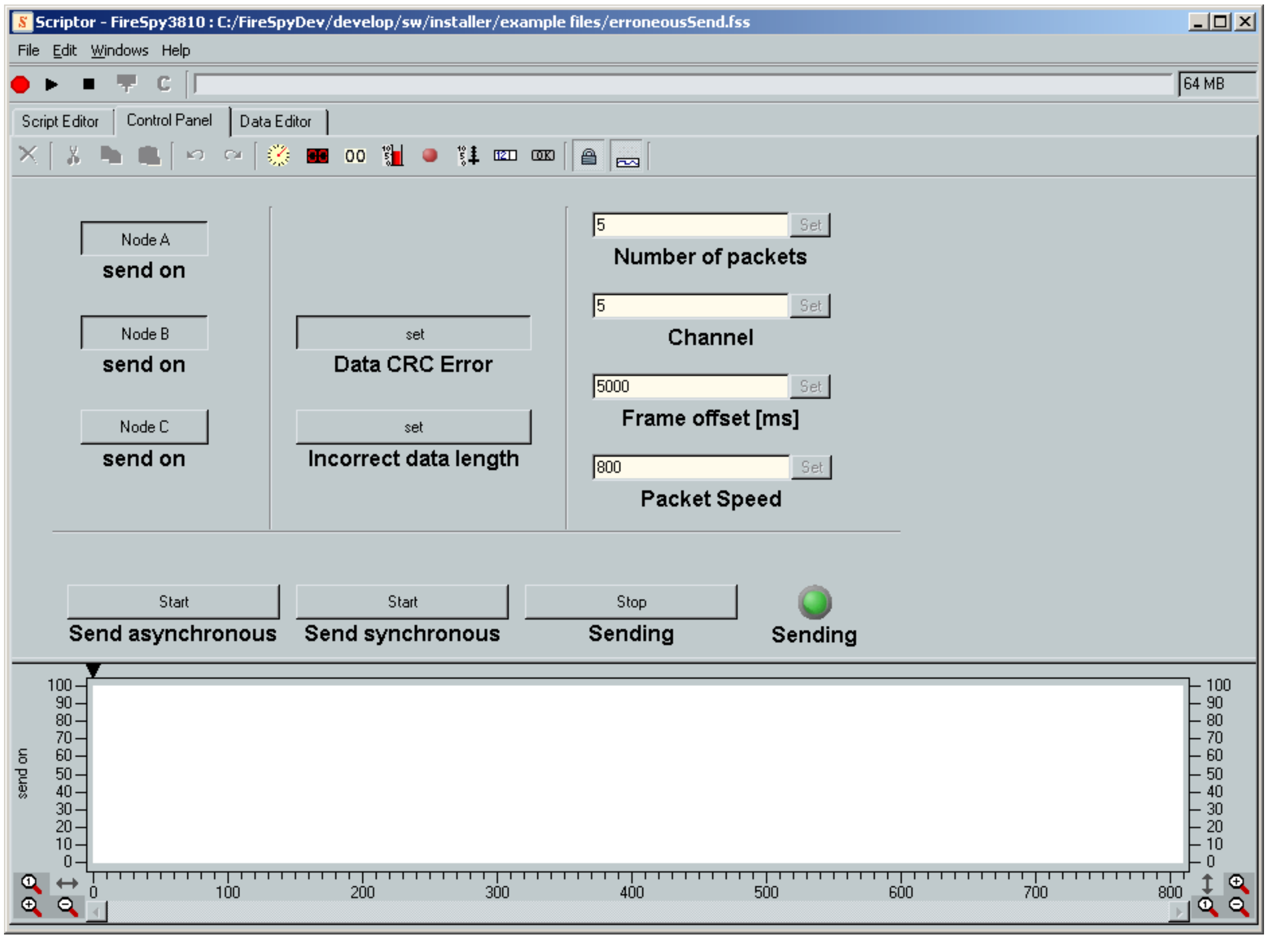
Packet speed
Sets the speed of the packets that are sent.

Send asynchronous
This button will cause the Analyzer to start sending the specified number of packets in asynchronous mode with the settings as set by the controls described above.

Send synchronous
This button will cause the Analyzer to start sending the specified number of packets in synchronous mode with the settings as set by the controls described above.

Stop
Stops sending packets.

Sending
When this led is on, the script is currently busy sending.



Running the script
To run this example, please proceed with the following steps:

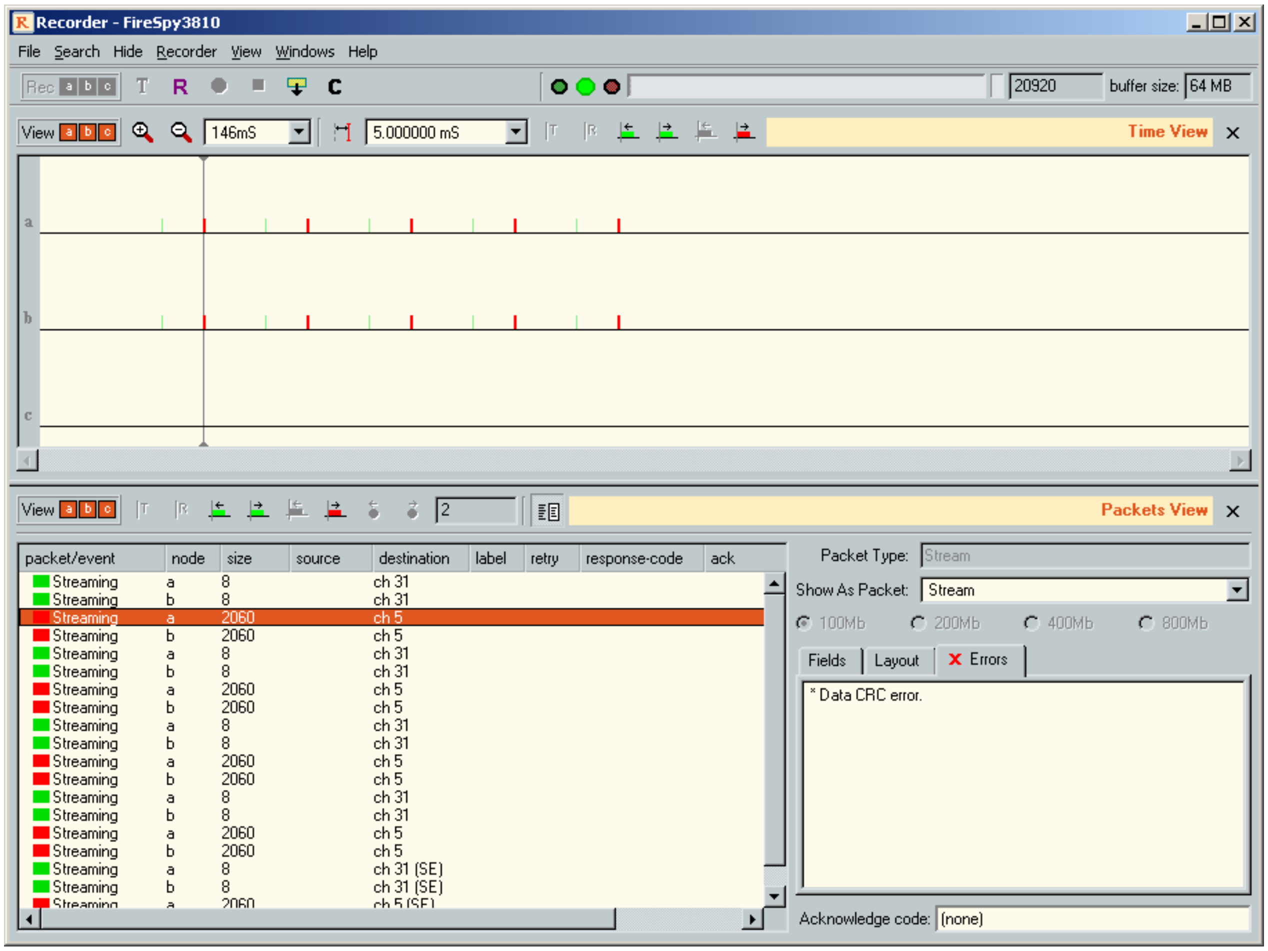
1. Open the recorder and start recording
2. Go back to the scriptor and select the Control Panel
3. Start the script

4. Enable the buttons "Node A" and "Node B"
5. Enable the button "Data CRC Error"
6. Set "Number of packets" to 5 (press set)
7. Set "Channel" to 5
8. Set "Frame Offset" to 5000ms
9. Set "Packet Speed" to 800
10. Start sending by pressing the button: "Send synchronous"
11. Open the recorder window and download the recording
12. In the recorder, open the Packets View (Menu Bar - View - Packets View)

Now that all windows are setup something like the Recorder picture below and the Control Panel picture above, it is time to start looking at what happened.

In the recorder picture, the time view shows 5 green and 5 red packets on nodes A and B. The green ones are the STOF packets that are also sent by the script and they donot contain any errors. The other packets are colored red because they contain errors (as set on the control panel). The distance between a neighbouring red and green packet is exactly the frame offset we specified on the control panel.

By selecting a red packet in the Packet View, we can also see what kind of error the packet contains. In this case we chose for a data CRC error on the Control Panel and the recorder shows that the packet indeed contains a data CRC error.

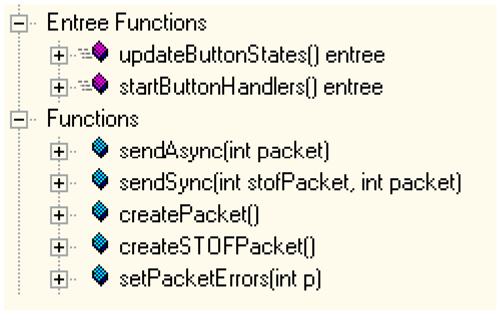


packet/event	node	size	source	destination	label	retry	response-code	ack
🟩 Streaming	a	8		ch 31				
🟩 Streaming	b	8		ch 31				
🟥 Streaming	a	2060		ch 5				
🟥 Streaming	b	2060		ch 5				
🟩 Streaming	a	8		ch 31				
🟩 Streaming	b	8		ch 31				
🟥 Streaming	a	2060		ch 5				
🟥 Streaming	b	2060		ch 5				
🟩 Streaming	a	8		ch 31				
🟩 Streaming	b	8		ch 31				
🟥 Streaming	a	2060		ch 5				
🟥 Streaming	b	2060		ch 5				
🟩 Streaming	a	8		ch 31				
🟩 Streaming	b	8		ch 31				
🟥 Streaming	a	2060		ch 5				
🟥 Streaming	b	2060		ch 5				
🟩 Streaming	a	8		ch 31 (SE)				
🟩 Streaming	b	8		ch 31 (SE)				
🟥 Streaming	a	2060		ch 5 (SE)				

Packet Type: Stream
Show As Packet: Stream
(● 100Mb) (200Mb) (400Mb) (800Mb)
Fields | Layout | Errors
* Data CRC error.
Acknowledge code: (none)

11.6.3.2. Details

The script source code
The picture below shows the Object Browser contents for the current example. The picture shows that the script contains two main entry functions and five regular functions. The purpose of the functions will be explained below: (see also the complete source further down this document)



Entree Functions
updateButtonStates() entree
startButtonHandlers() entree
Functions
sendAsync(int packet)
sendSync(int stofPacket, int packet)
createPacket()
createSTOFPacket()
setPacketErrors(int p)

updateButtonStates
On the control panel, a couple of buttons are used as so-called state buttons. This means that a button should stay down when it is enabled and up when it is disabled. Normally, buttons on the control panel will come back up as soon as they are released after pressing them. This process contains an infinite loop that checks if a button is pressed. If it is, it will set the button to a down state. Also if an incorrect packet speed is chosen, it will set the edit box back to its default value. To prevent the Analyzer from being occupied by this loop all the time, the function "waitStartOfFrame" is called every iteration as an alternative for the "sleep" command. (the sleep command will be implemented in one of the next versions of the software)

startButtonHandlers
This process monitors the start buttons and as soon as one is pressed, it will call one of "sendASync" and "sendSync" functions. Control will be given to that function until it is done sending. If it is done, this function will continue to monitor the start buttons.

sendAsync
This function will send the provided packet exactly the specified number of times on the specified channel(s) in Asynchronous mode. Before it starts sending, the function "setPacketErrors" is called to set whether the packet should contain an error or not. After sending each individual packet, it checks the state of the stop button. If pressed, it will stop sending packets.

sendSync
This function will send the provided packet exactly the specified number of times on the specified channel in Synchronous mode. It sends the packet at the specified frame offset time and it also sends a STOF packet at frame offset time 0 for each frame. Before it starts sending, the function "setPacketErrors" is called to set whether the packet should contain an error or not. After sending each individual packet, it checks the state of the stop button. If pressed, it will stop sending packets.

createPacket
This function is called once when the script is started. It will use the data as defined on the Data Editor to setup a basic packet. This packet will be used to send in both asynchronous and synchronous mode.

createSTOFPacket
This function is called once when the script is started. It creates an empty STOF packet to be used in Sync mode.

setPacketErrors
This function is provided with a packet buffer. It reads the settings from the control panel and sets/resets errors and header fields of the provided packet to the values as specified by the user.

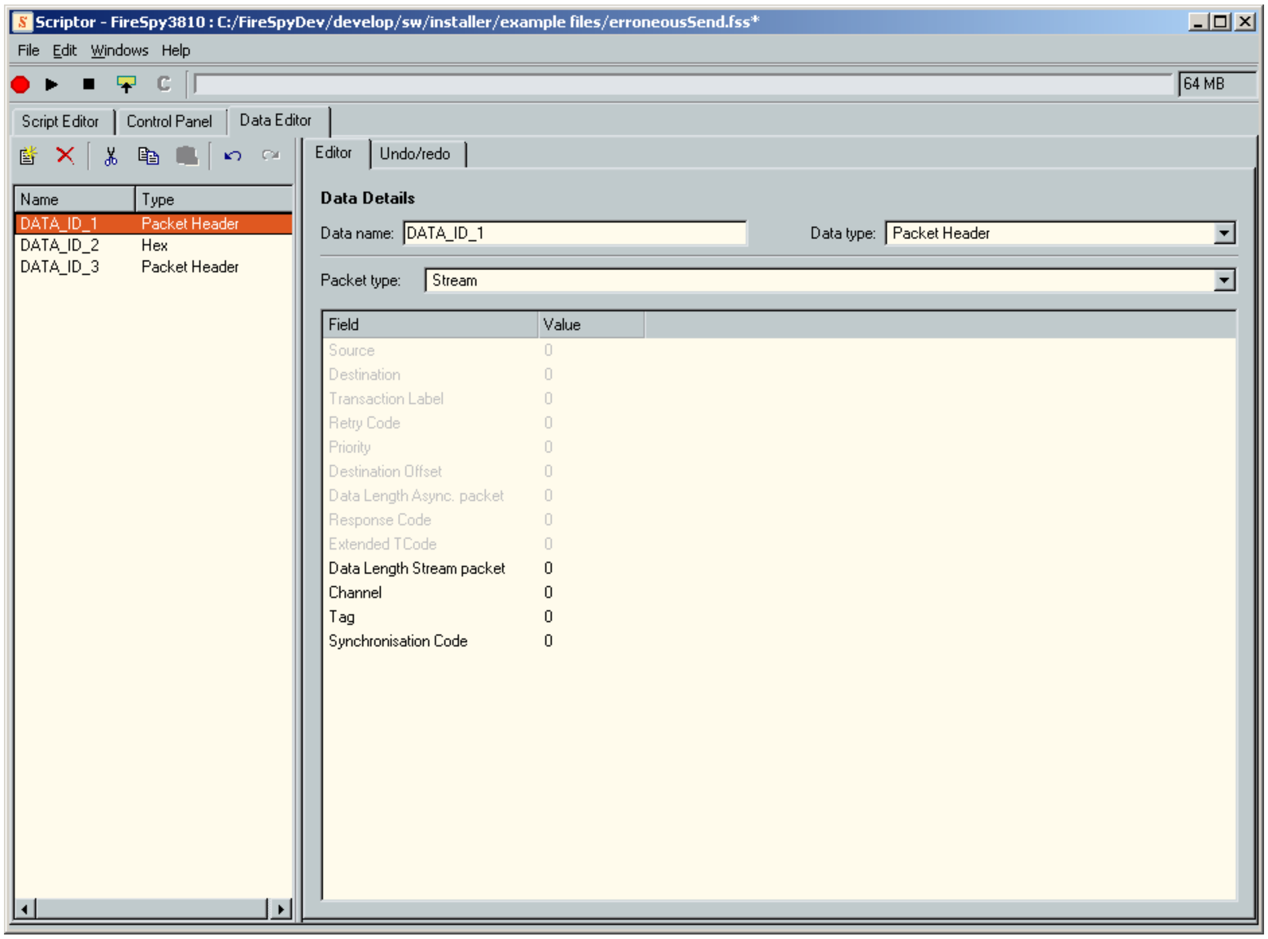
```
//-------------------------------------------------------------------
void updateButtonStates () entry
        int i
        int numButtons = 6

        while true
                int speed = getControlValue ( 34 )
                int packets = getControlValue ( 31 )
                waitStartOfFrame ()

                i = 0
                while i < numButtons
```

```
                    setControlValue ( i + 11 , getControlValue ( i + 11 ) )
                    i = i + 1



                if speed != 100 && speed != 200 && speed != 400 && speed != 800
                    setControlValue ( 34 , 800 )


                if packets < 1
                    setControlValue ( 31 , 1 )


//------------------------------------------------------------------
void startButtonHandlers () entry

    int packet = createPacket ()
    int stofPacket = createSTOFPacket ()

    while true
            setControlValue ( 3 , 0 )
            waitStartOfFrame ()

            if getControlValue ( 1 )
                    setControlValue ( 1 , 0 )
                    setControlValue ( 3 , 1 )
                    sendAsync ( packet )
                    repeat 50
                            waitStartOfFrame ()


                elseif getControlValue ( 2 )
                    setControlValue ( 2 , 0 )
                    setControlValue ( 3 , 1 )
                    sendSync ( stofPacket, packet )
                    repeat 50
                            waitStartOfFrame ()

//------------------------------------------------------------------
void sendAsync ( int packet )
    int i = 0
    int num = getControlValue ( 31 )
    bool sendA = getControlValue ( 11 )
    bool sendB = getControlValue ( 12 )
    bool sendC = getControlValue ( 13 )

    if ! sendA && ! sendB && ! sendC
            return


        setPacketErrors ( packet )

        while i < num
            if getControlValue ( 4 )
                    return


                if sendA
                    selectFireSpyNode ( 0 )
                    sendPacket ( packet )

                if sendB
```

```
                        selectFireSpyNode ( 1 )
                        sendPacket ( packet )

                    if sendC
                        selectFireSpyNode ( 2 )
                        sendPacket ( packet )


                    i = i + 1

//-------------------------------------------------------------------
void sendSync ( int stofPacket, int packet )
    int i = 0
    int num = getControlValue ( 31 )
    int frameOffset = getControlValue ( 33 )
    bool sendA = getControlValue ( 11 )
    bool sendB = getControlValue ( 12 )
    bool sendC = getControlValue ( 13 )

    if ! sendA && ! sendB && ! sendC
            return


        setPacketErrors ( packet )

        while i < num
            if getControlValue ( 4 )
                    return


                waitStartOfFrame ()

                if sendA
                    selectFireSpyNode ( 0 )
                    sendPacketNextFrame ( stofPacket, 0 )
                    sendPacketNextFrame ( packet, frameOffset )

                if sendB
                    selectFireSpyNode ( 1 )
                    sendPacketNextFrame ( stofPacket, 0 )
                    sendPacketNextFrame ( packet, frameOffset )

                if sendC
                    selectFireSpyNode ( 2 )
                    sendPacketNextFrame ( stofPacket, 0 )
                    sendPacketNextFrame ( packet, frameOffset )


                i = i + 1

//-------------------------------------------------------------------
int createPacket ()
    int p = newPacket ( 512 )

    fillPacket ( p, DATA_ID_1, DATA_ID_2 )
            fillPacketHeader ( p, DATA_ID_1 )
            fillPacketData ( p, DATA_ID_2 )


        return p

//-------------------------------------------------------------------
```

```
int createSTOFPacket ()
    int p = newPacket ( 512 )

    fillPacket ( p, DATA_ID_3 )
            fillPacketHeader ( p, DATA_ID_3 )


        return p

//-----------------------------------------------------------------
void setPacketErrors ( int p )
    int channel = getControlValue ( 32 )
    int speed = getControlValue ( 34 )
    bool dataCRC = getControlValue ( 15 )
    bool sizeError = getControlValue ( 16 )
    int dataBytes = getPacketDataSize ( p ) * 4

    // Check if we have to set an incorrect data size in the header
    if sizeError
            dataBytes = dataBytes + 1


        // Set the header fields
        setPacketHeaderFields ( p, dataBytes, channel )
            setPacketHeaderField ( p, 0 , 0 , 16 , dataBytes )
            setPacketHeaderField ( p, 0 , 18 , 6 , channel )


        // Set the packet speed
        if speed == 100
            setPacketSpeed ( p, 0 )

        elseif speed == 200
            setPacketSpeed ( p, 1 )

        elseif speed == 300
            setPacketSpeed ( p, 2 )

        elseif speed == 400
            setPacketSpeed ( p, 3 )


        // check if we have to set a crc error in the data
        setSendDataCRCError ( p, dataCRC )
```

Data Definitions
Data Objects are used for setting the packet contents to the correct formats. The pictures below show the data definitions used by the script. The first picture shows the header of a stream packet. The second picture shows the packet data contents. Note that all fields are just set to zero and will be set by the script at runtime.


Scriptor - FireSpy3810 : C:/FireSpyDev/develop/sw/installer/example files/erroneousSend.fss*
File Edit Windows Help
64 MB
Script Editor
Control Panel
Data Editor
Editor
Undo/redo

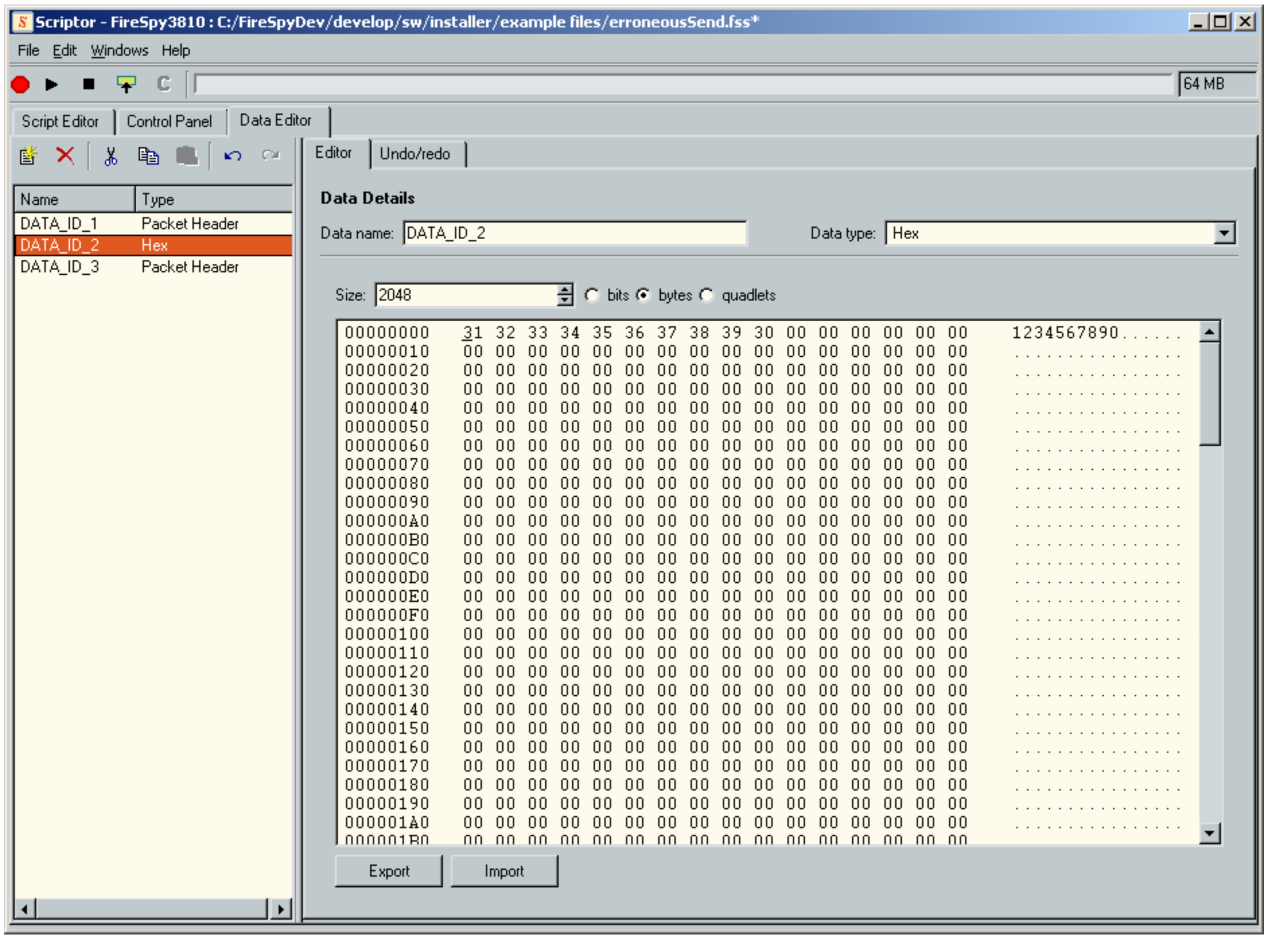
Name	Type
DATA_ID_1	Packet Header
DATA_ID_2	Hex
DATA_ID_3	Packet Header

Data Details

Data name: DATA_ID_1 Data type: Packet Header

Packet type: Stream

Field	Value
Source	0
Destination	0
Transaction Label	0
Retry Code	0
Priority	0
Destination Offset	0
Data Length Async. packet	0
Response Code	0
Extended TCode	0
Data Length Stream packet	0
Channel	0
Tag	0
Synchronisation Code	0


Scriptor - FireSpy3810 : C:/FireSpyDev/develop/sw/installer/example files/erroneousSend.fss*
File Edit Windows Help
64 MB
Script Editor
Control Panel
Data Editor
Editor
Undo/redo
Export
Import

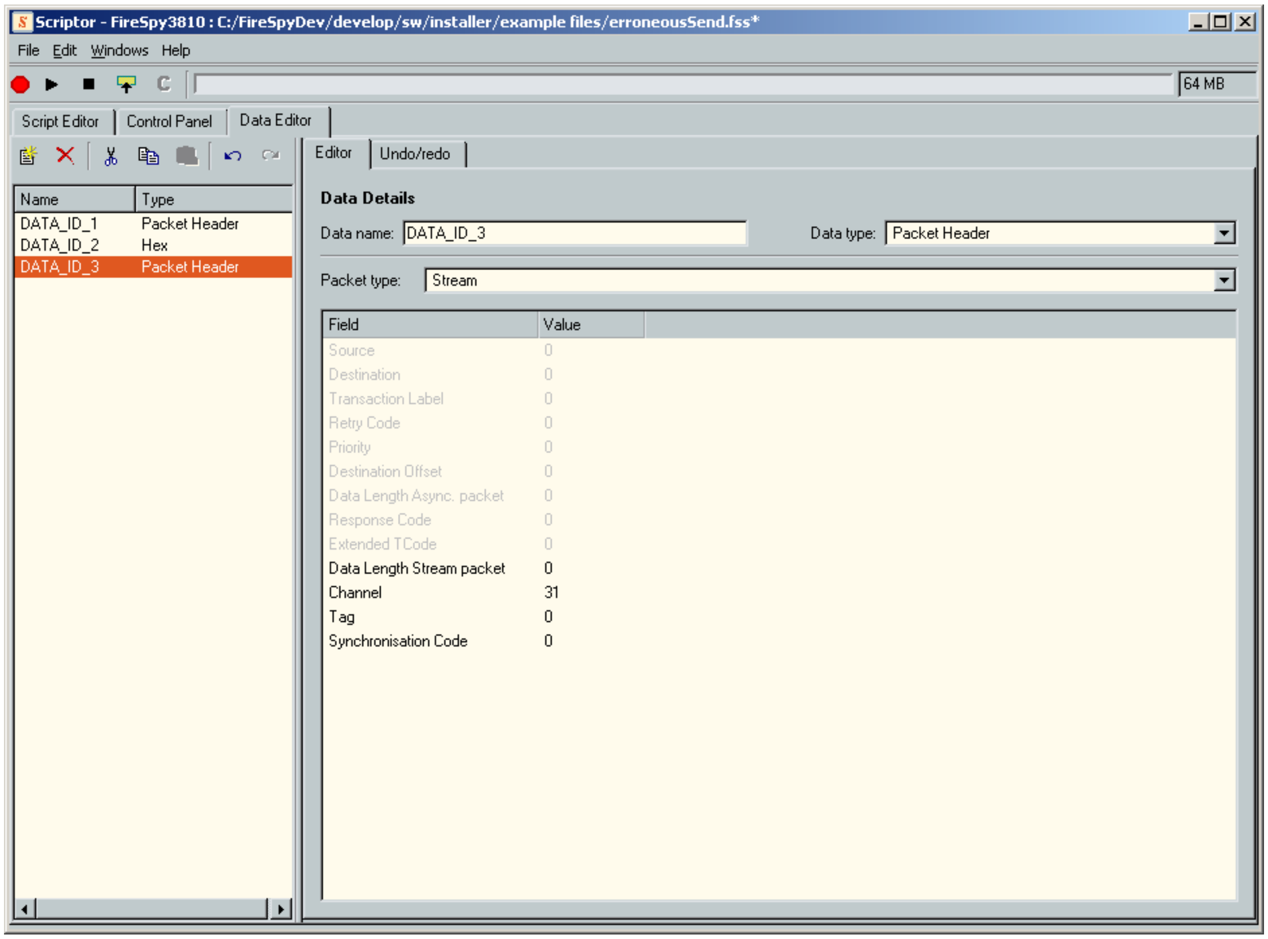
Name	Type
DATA_ID_1	Packet Header
DATA_ID_2	Hex
DATA_ID_3	Packet Header

Data Details

Data name: DATA_ID_2 Data type: Hex

Size: 2048 bits [✓ bytes] quadlets

```
00000000   31 32 33 34 35 36 37 38 39 30 00 00 00 00 00 00   1234567890......
00000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000080   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000090   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000000F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000100   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000120   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000140   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000170   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```


Scriptor - FireSpy3810 : C:/FireSpyDev/develop/sw/installer/example files/erroneousSend.fss*
File Edit Windows Help
64 MB
Script Editor
Control Panel
Data Editor
Editor
Undo/redo

Name	Type
DATA_ID_1	Packet Header
DATA_ID_2	Hex
DATA_ID_3	Packet Header

Data Details

Data name: DATA_ID_3   Data type: Packet Header

Packet type: Stream

Field	Value
Source	0
Destination	0
Transaction Label	0
Retry Code	0
Priority	0
Destination Offset	0
Data Length Async. packet	0
Response Code	0
Extended TCode	0
Data Length Stream packet	0
Channel	31
Tag	0
Synchronisation Code	0

11.6.4. Cable Test

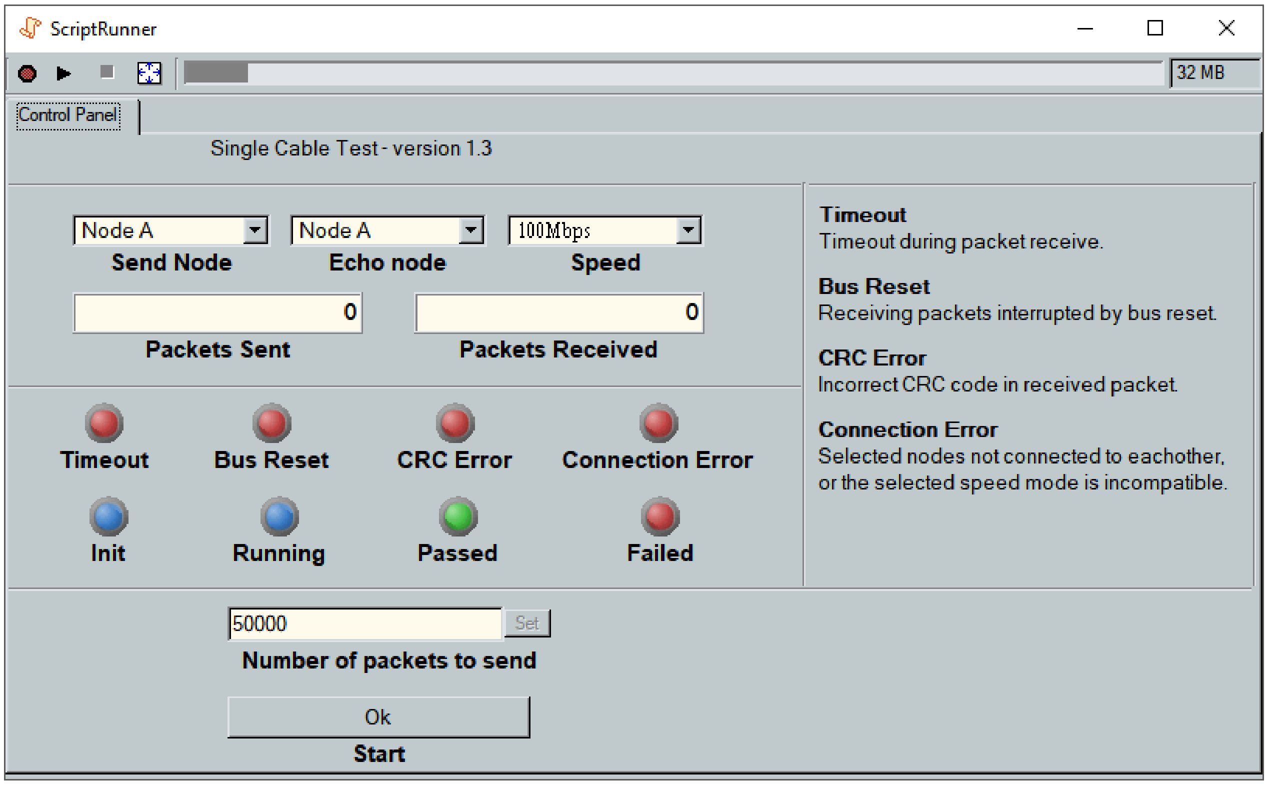
If you only have one triple FireSpy analyzer (FS3810 or FS3850) for the test, you can test one cable at a time. If you have more than one FireSpy® analyzer for the test, you can configure two non-triple FireSpy analyzers as a pair, or configure one triple FireSpy analyzer as a server and up to three FireSpy analyzers as clients.

11.6.4.1. One triple FireSpy analyzer

This is the simplest way to test a cable since it requires only one triple FireSpy analyzer (FS3810 or FS3850) and one script.

Set up the analyzer so that the target cable is connected to two of three ports on the front of the analyzer. Start the FireSpy application, open the Scriptor, and then open SingleCableTest.fss file. The Control Panel pane on the Scriptor window should look similar to the screenshot below.

Select two nodes of which the target cable is connected to and the speed. You might want to change the number of packets (defaulted to 50000) transmitted during the test.



Single Cable Test - version 1.3

Send Node: Node A
Echo node: Node A
Speed: 100Mbps
Packets Sent: 0
Packets Received: 0
Number of packets to send: 50000

Timeout
Timeout during packet receive.

Bus Reset
Receiving packets interrupted by bus reset.

CRC Error
Incorrect CRC code in received packet.

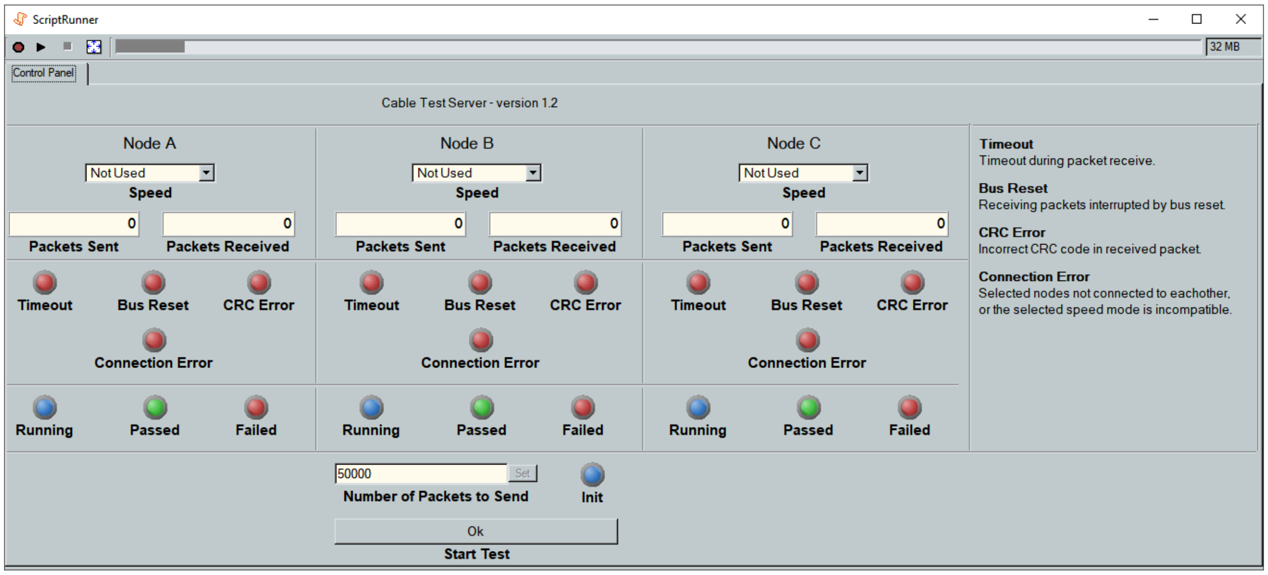
Connection Error
Selected nodes not connected to eachother, or the selected speed mode is incompatible.

11.6.4.2. Two non-triple FireSpy analyzers

You will need to dedicate one analyzer as a server and the other as a client. Start the FireSpy application on both analyzers and load CableTestServer.fss and CableTestClient.fss in the Scriptor accordingly.

The client-side script has no controls: you will simply need to start the script.

On the server side script, however, you will need to setup the controls according to the analyzer type and cable settings you have. Since your server analyzer is a non-triple one, you only need to setup the transmission speed for Node A. After setting the number of packets to send, the script is ready to test the cable.



Cable Test Server - version 1.2

Node A: Speed: Not Used; Packets Sent: 0; Packets Received: 0
Node B: Speed: Not Used; Packets Sent: 0; Packets Received: 0
Node C: Speed: Not Used; Packets Sent: 0; Packets Received: 0
Number of Packets to Send: 50000

Timeout
Timeout during packet receive.

Bus Reset
Receiving packets interrupted by bus reset.

CRC Error
Incorrect CRC code in received packet.

Connection Error
Selected nodes not connected to eachother, or the selected speed mode is incompatible.

11.6.4.3. More than one cable with a triple FireSpy analyzer

You can test more than one cable at the same time by utilizing one triple FireSpy analyzer as a server and up to three client FireSpy analyzers (or another triple, using all three nodes at the same time).

All settings are very similar to step 3 except that you need to enable Node B and/or Node C cable speed depending on the configuration you have. You will also need to load the CableTestClient.fss script file to each client FireSpy® analyzer and CableTestServer.fss to the triple analyzer.

Chapter 12. Filter/Trigger

This section describes the Filter / Trigger functionality. Please take note that not all features described are available on all analyzer models.

With the Recorder Filter one can skip packets or events. Skipped packet or events are not stored in memory by the Recorder. The packets to be skipped can be selected by type, speed, error type or by using any boolean combination of the four packet Sets.

The Recorder can be triggered using extensive hardware trigger logic. You can select simple trigger conditions such as trigger on a particular type of error packet, trigger when a packet fits the description of a packet Set or any boolean combination of the four packet Sets, or trigger on external port inputs. For more advance triggering a powerful trigger sequencer can be used.

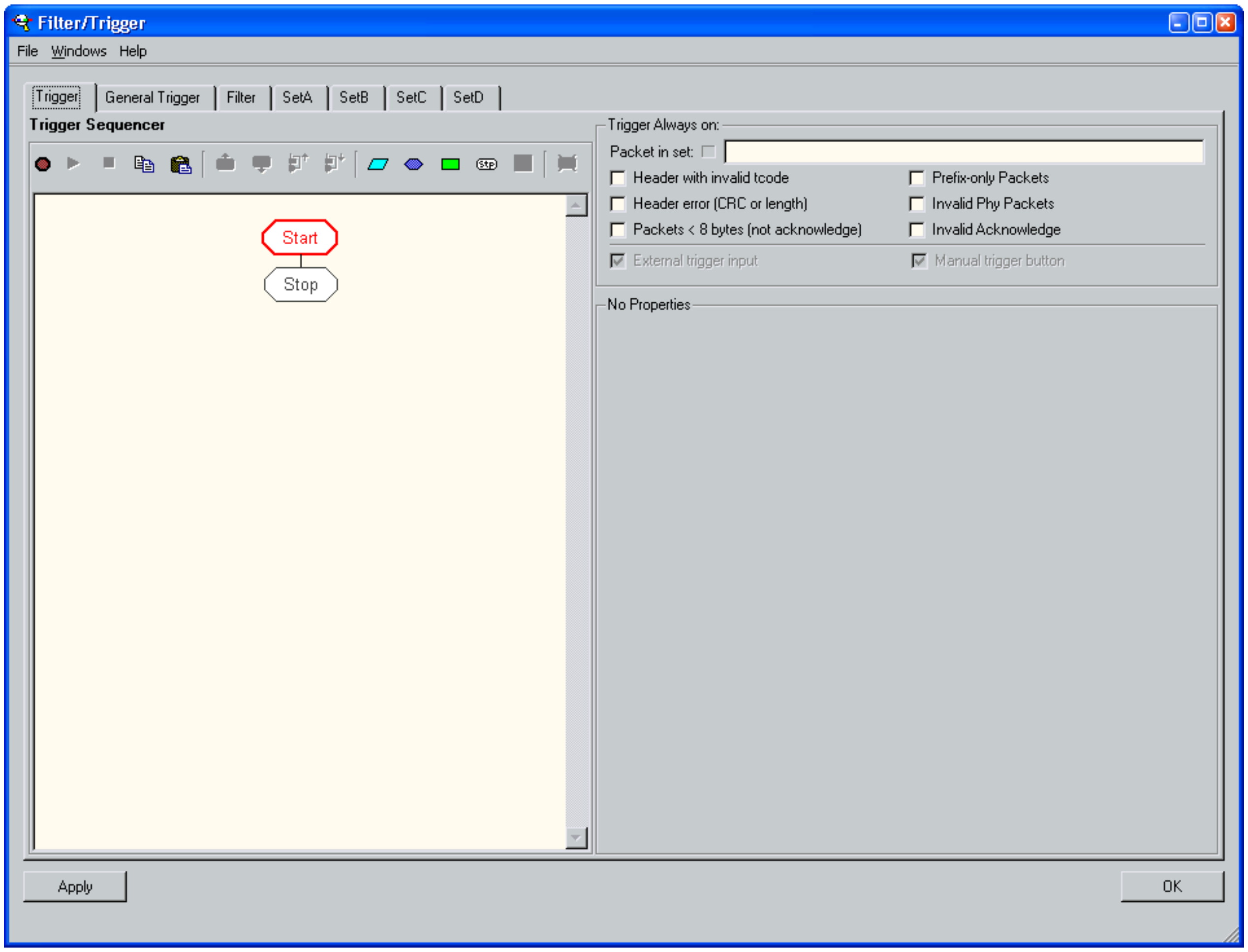
You also can select the amount of data to be recorded maximal after a trigger. And you can select to use a cyclic buffer for the data before the trigger. The size of this cyclic buffer can be changed too.

Both Filter and Trigger logic can use any boolean combination of four packet Sets. These Sets consist of any combination of primary (asynchronous or stream) packets or a phy packet, with optional value checking on packet fields. Value checking can be performed on header fields or data fields. In case of data fields, a powerful packet editor is included to be able to put conditions on data fields of a large variety of packets

12.1. How to use it

You can open the Filter/Trigger Settings window by selecting the Filter/Trigger Settings command from the Analyzer menu at the top of one of the open Analyzer windows.

When you open the Filter/Trigger Settings, the window below will appear.



Trigger Always on:

Packet in set: [ ]

[ ] Header with invalid tcode
[ ] Prefix-only Packets
[ ] Header error (CRC or length)
[ ] Invalid Phy Packets
[ ] Packets < 8 bytes (not acknowledge)
[ ] Invalid Acknowledge
[✓] External trigger input
[✓] Manual trigger button

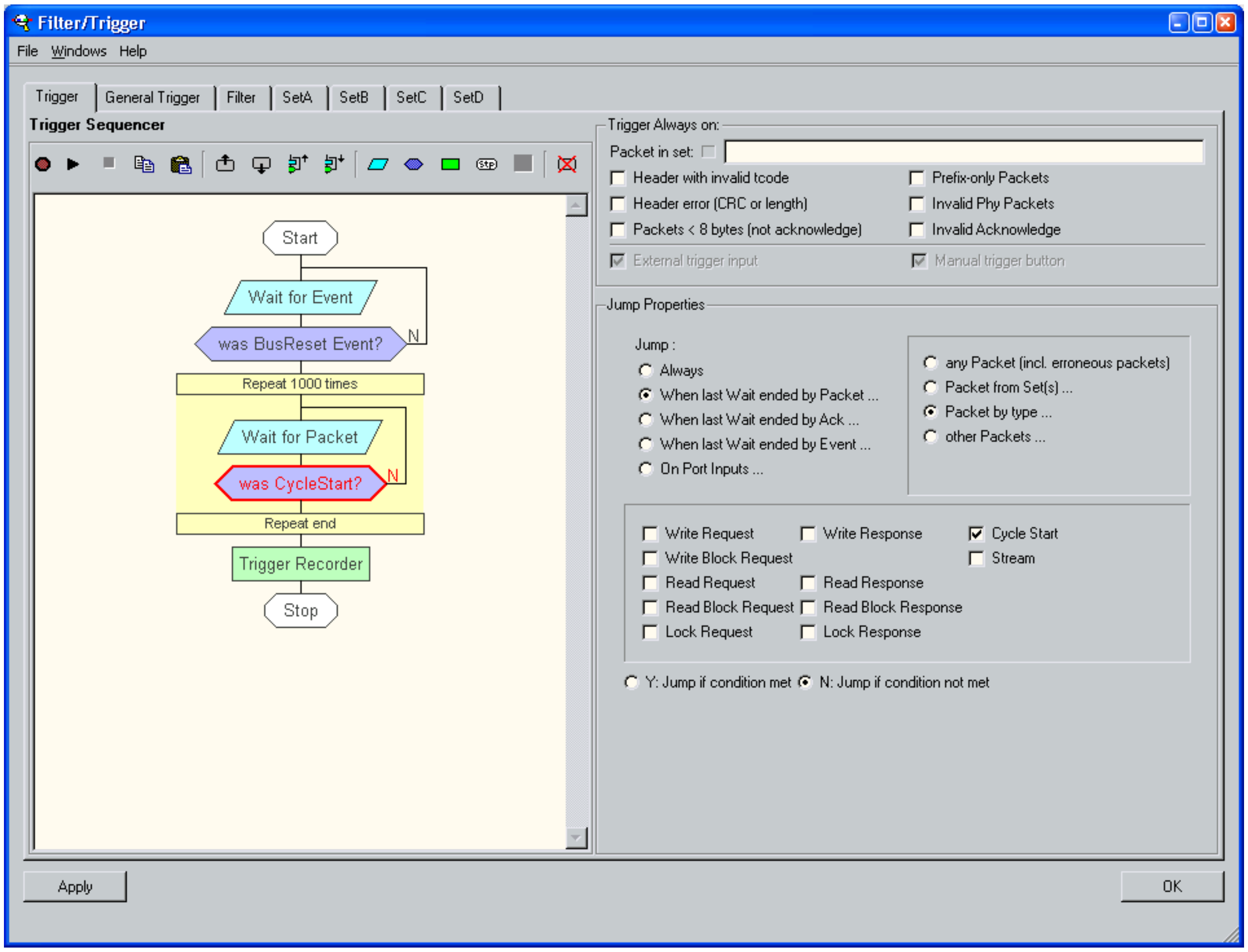
If you want to change the trigger settings, select the Trigger page (like above) or General Trigger page. To change the Filter settings, select the Filter page. If you use one of the four basic packet sets and you want to change one of them, select the corresponding SetA, SetB, SetC or SetD pages.

12.1.1. Trigger

The Trigger Sequencer is not available on

- Stealth Series in Symbol Recorder mode

An example of the Trigger page is shown below.



Trigger Always on:

Packet in set: [ ]

[ ] Header with invalid tcode [ ] Prefix-only Packets
[ ] Header error (CRC or length) [ ] Invalid Phy Packets
[ ] Packets < 8 bytes (not acknowledge) [ ] Invalid Acknowledge
[✓] External trigger input [✓] Manual trigger button

Jump Properties

Jump :
- ( ) Always
- (●) When last Wait ended by Packet ...
- ( ) When last Wait ended by Ack ...
- ( ) When last Wait ended by Event ...
- ( ) On Port Inputs ...

- ( ) any Packet (incl. erroneous packets)
- ( ) Packet from Set(s) ...
- (●) Packet by type ...
- ( ) other Packets ...

[ ] Write Request [ ] Write Response [✓] Cycle Start
[ ] Write Block Request [ ] Stream
[ ] Read Request [ ] Read Response
[ ] Read Block Request [ ] Read Block Response
[ ] Lock Request [ ] Lock Response

( ) Y: Jump if condition met (●) N: Jump if condition not met

The most important part for the trigger settings is the **Trigger Sequencer**, see left pane in example above. Here you can graphically define a trigger program by adding items to the flow and change theire properties. By executing the program you can generate a Recorder trigger when a particular sequence of packets or events has been detected.

The most typical items in this flow are: the Wait, the Jump and the Action item. With the Wait item you can instruct the trigger sequencer to wait for a packet, an acknowledge, an event or any combination of them. The next item will not be executed until the packet, acknowledge or event has been reveived. Such a Wait is typically followed by one or more Jump items that can check what kind of packet, acknowledge or event was received. The Jump can for instance check if a received packet is of some type or if the packet fits the description of a packet Set or fits any boolean combination of the four Sets. With the Action item you can for instancte trigger the Recorder. One or more items can also easily be repeated by putting them in a surrounding repeat box (see yellow box in example above).
In the bottom box to the right of the Trigger Sequencer, the properties of the selected item will be displayed. In the example above it are the properties of the selected Jump item, where the jump condition can be specified. In this example the condition is that the packet most be of type Cycle Start. And the jump is taken if the condition has not been met.
The trigger program in the example above for instance will trigger the Recorder when a bus-reset event has been detected, followed by 1000 cycle-start packets. More examples will be shown below where the details of the Trigger Sequencer are described.
To activate the trigger sequencer press the Start or restart trigger sequencer button in the toolbar, or start the Recorder (a Recorder start always will (re)start the trigger sequencer). The Trigger sequencer active indicator in the toolbar will light as indication that the trigger sequencer is active and a red arrow will be displayed in front of the currently executed item, to help in debugging the trigger program.
Pressing the Stop trigger sequencer button in the toolbar will stop the trigger sequencer.

In the **Trigger Position** box, you can choose the size of the Recorder buffer to be used maximal after a trigger has occured, using the Max post-trigger size slider. If a trigger occurs and the specified amount of data has been recorded after the trigger, the Recorder will automatically stop. If the Stop recording when

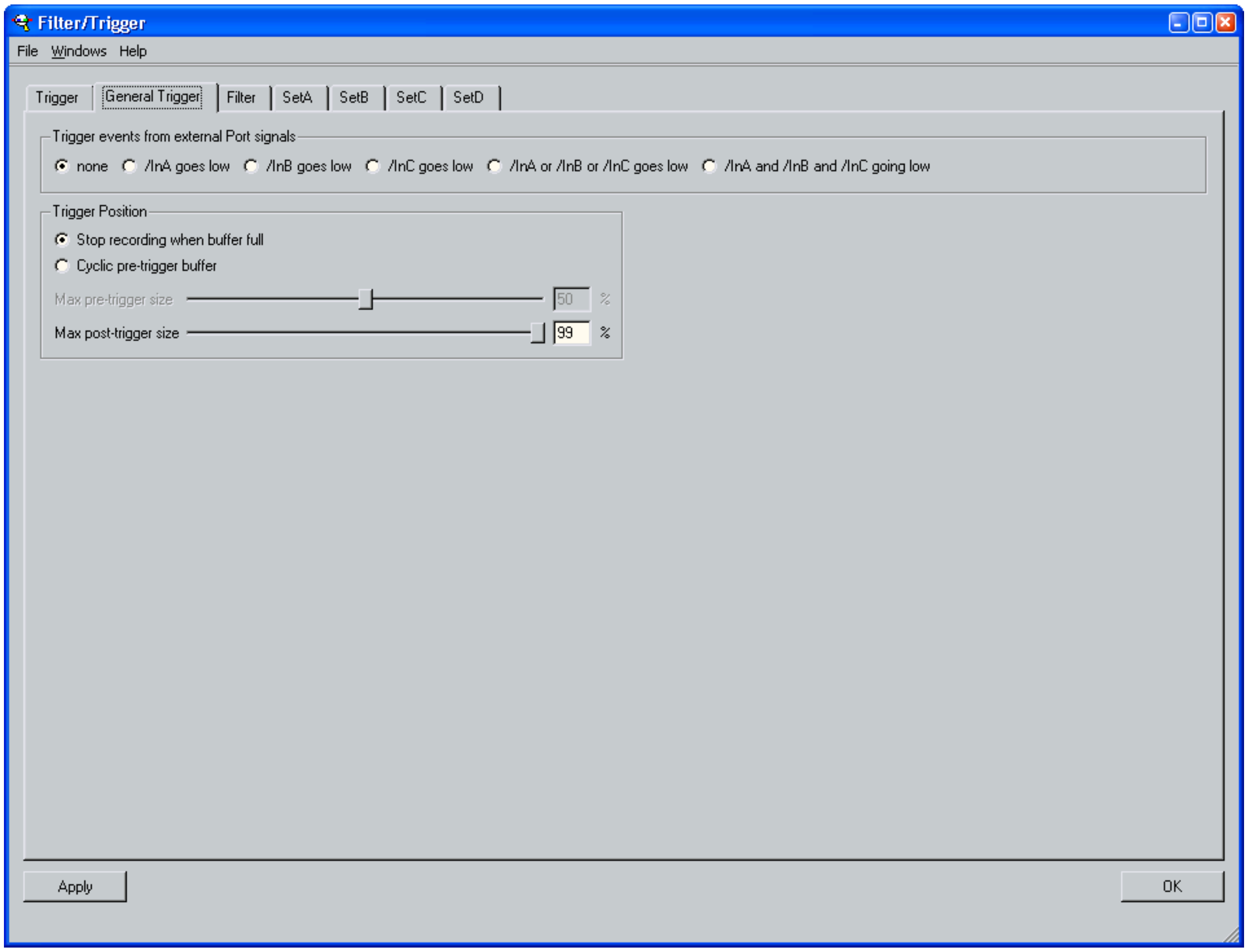
buffer full option is selected (default), the Recorder will also stop when the complete Recorder buffer has been filled with data. In this mode you will never overwrite recorded data.
When you however select the Cyclic pre-trigger buffer option, the Recorder will use a cyclic buffer for the recorded data before a trigger occurs. When this cyclic buffer gets full, the oldest data will be overwritten with the new recorded data. Until a trigger occurs. Then the cyclic buffer will not be overwritten by new data. The new data will be stored in the remaining part of the Recorder buffer until the buffer is completely full or until the Max post-trigger size condition has been met as descibed above. If the cyclic option has been selected, the size of the cyclic buffer can be defined using the Max pre-trigger size slider.

Inside the **Trigger Always on**: box you can specify some conditions on which the Recorder should always trigger (thus independent of the Trigger Sequencer). You can specify to trigger on some type of errors and you can specify to trigger if a packed is detected that fits a packet Set or fits any boolean combination of the four packet Sets. In the example above for instance, the Recorder will always be triggered when a packet is detected with a Header error or when a packet is detected that fits the description of packet SetA, but not that of SetB or SetC. Note that a red cross will be displayed before the boolean expression when there is an error in the expression.
After defining the complete filter/trigger information, you can upload the settings to the Analyzer by pressing the Apply button at the bottom of the window.
If you used one of the four basic sets you should define which packets fits the set. See Packet Sets below.

12.1.2. General Trigger

Cyclic buffer mode is not available on
- Third Generation Analyzers

An example of the General Trigger page is shown below.



Trigger events from external Port signals
[✓ none] /InA goes low /InB goes low /InC goes low /InA or /InB or /InC goes low /InA and /InB and /InC going low

Trigger Position
[✓ Stop recording when buffer full]
Cyclic pre-trigger buffer
Max pre-trigger size 50 %
Max post-trigger size 99 %

12.1.3. Filter

The Filter Tab is not available on

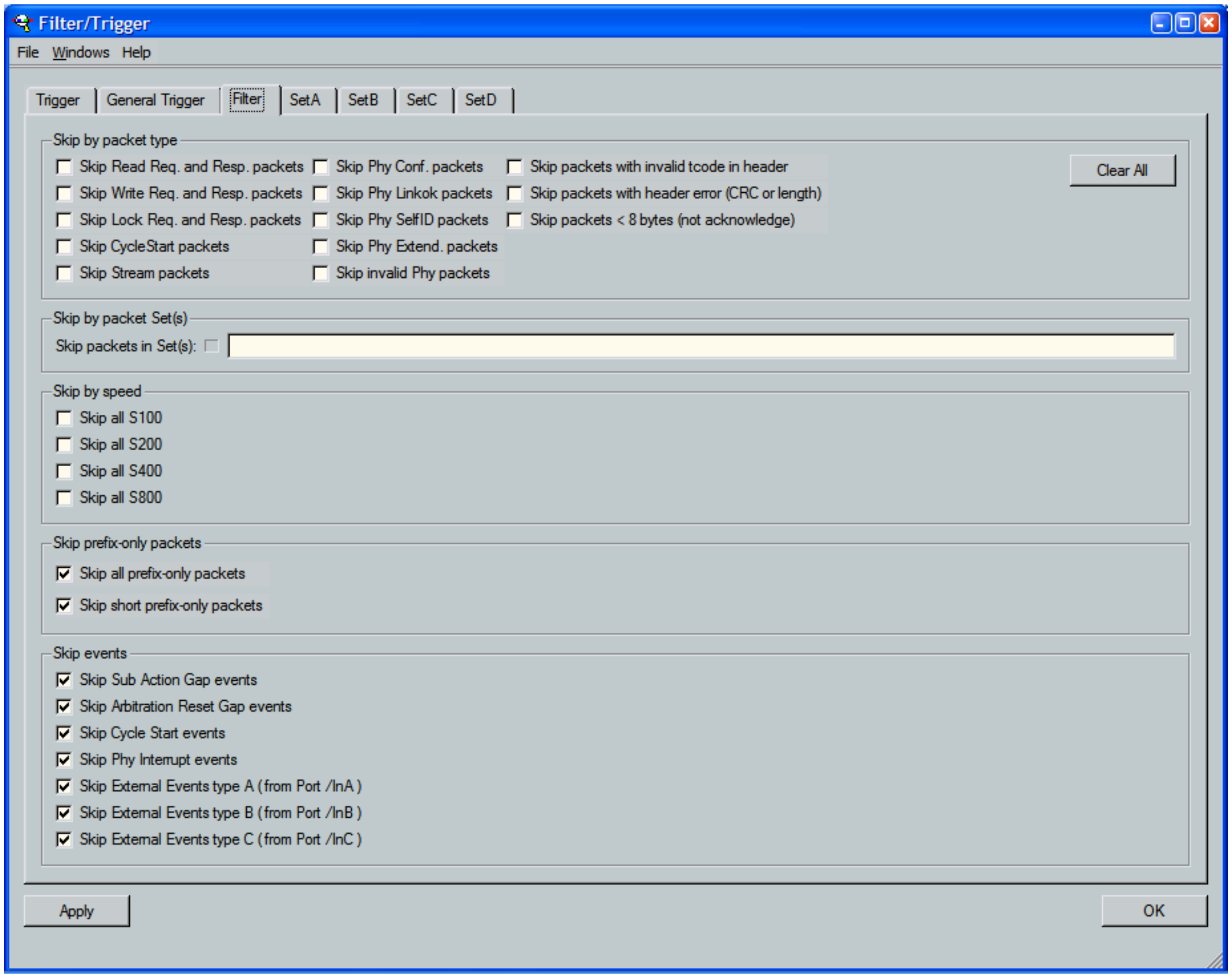
- Stealth Series in Symbol Recorder mode

Actual available options for filtering depend on the analyzer model used.
An example of the Filter page is shown below.

You can select on this page which packets and events should be skipped.

In the **Skip by packet type** box you can select packet types to be skipped. You can also select some error types to skip packets with the selected error types. In the example below, all Cycle Start packets will be skipped. Note that you can even skip the Phy-SelfID packets. When you do this, the software will not be able to show a Topology for the corresponding (skipped) Phy-SelfID packets.

In the **Skip by packet Set(s)** box you can specify a packet Set or any boolean combination of the four packet Sets. All packet that fits the described Sets will be skipped. In the example below, all packets that fits both packet SetA and SetB will be skipped. Note that a red cross will be displayed before the boolean expression when there is an error in the expression



Skip by packet type
- [ ] Skip Read Req. and Resp. packets
- [ ] Skip Write Req. and Resp. packets
- [ ] Skip Lock Req. and Resp. packets
- [ ] Skip CycleStart packets
- [ ] Skip Stream packets
- [ ] Skip Phy Conf. packets
- [ ] Skip Phy Linkok packets
- [ ] Skip Phy SelfID packets
- [ ] Skip Phy Extend. packets
- [ ] Skip invalid Phy packets
- [ ] Skip packets with invalid tcode in header
- [ ] Skip packets with header error (CRC or length)
- [ ] Skip packets < 8 bytes (not acknowledge)

Skip by packet Set(s)
- [ ] Skip packets in Set(s):

Skip by speed
- [ ] Skip all S100
- [ ] Skip all S200
- [ ] Skip all S400
- [ ] Skip all S800

Skip prefix-only packets
- [✓] Skip all prefix-only packets
- [✓] Skip short prefix-only packets

Skip events
- [✓] Skip Sub Action Gap events
- [✓] Skip Arbitration Reset Gap events
- [✓] Skip Cycle Start events
- [✓] Skip Phy Interrupt events
- [✓] Skip External Events type A (from Port /InA)
- [✓] Skip External Events type B (from Port /InB)
- [✓] Skip External Events type C (from Port /InC)

Using the **Skip by speed** box you can skip all packets of particular speed. When checking the Skip all S800 for instance, no S800 packets will be recorded.

In the **Skip prefix-only packets** box you can specify to skip prefix-only packets. When you check the Skip all prefix-only packets checkbox, no prefix-only packets will be recorded. When this checkbox is not checked and the Skip short prefix_only packets checkbox is checked, only the short prefix_only packets will be skipped. This option is by default on. It is used to skip the short prefix-only packets that are sometimes generated by the 1394b phy interface when sending a bus status event.

In the **Skip events** box you can select which events should be skipped by checking the corresponding checkboxes. By default all the 'Bus Reset' events will be skipped.

Note that Bus-Reset events can not be skipped.

12.1.4. Packet Sets

The Packet Sets tab is not available on

- Stealth Series in Symbol Recorder mode

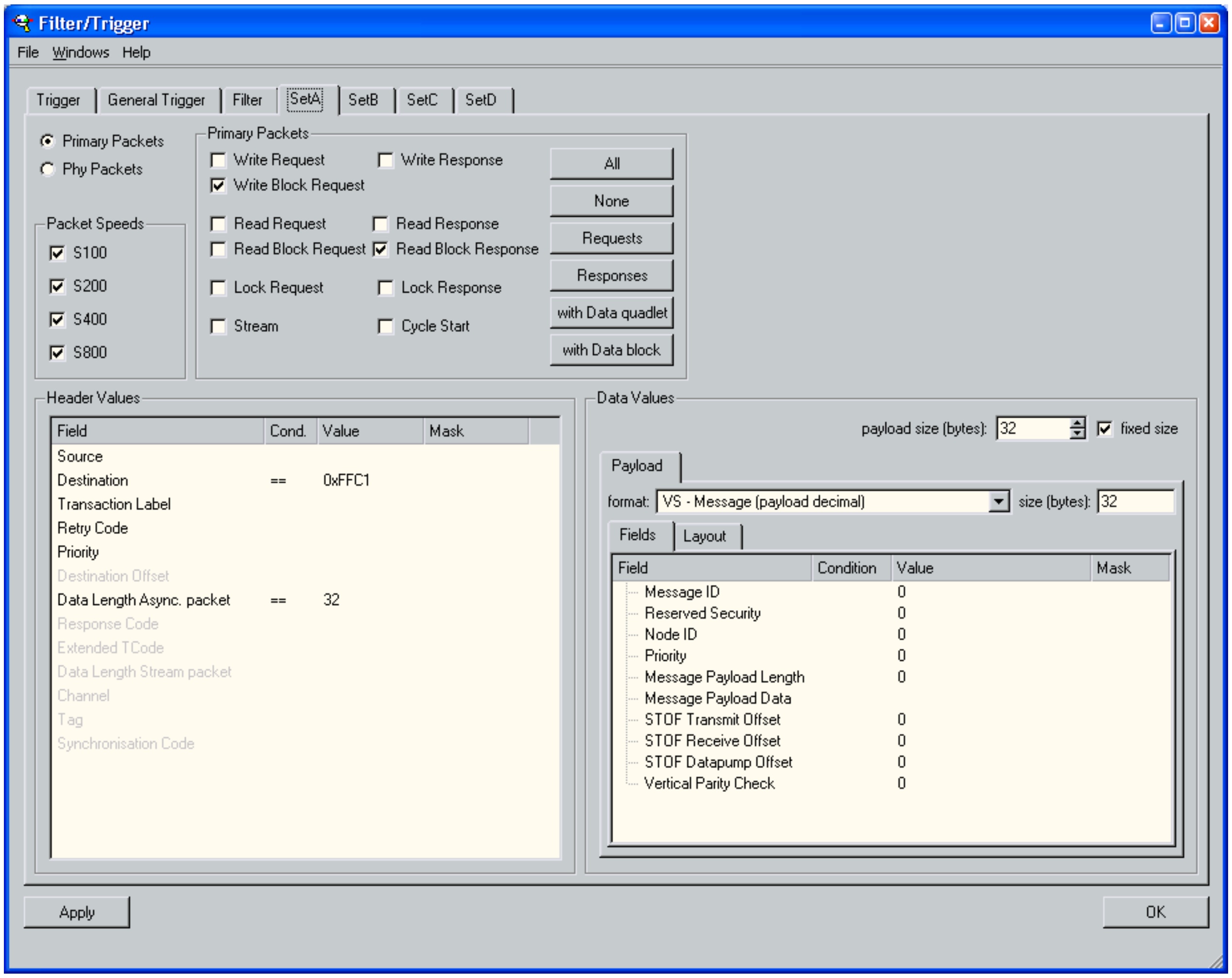
Header and data fields are limited to the "==" operation for the condition field on the following device types:

- Third Generation Analyzers

An example of a page of one of the four basic packet sets is shown below. Here you can define the condition of the packets that fits the set.

First of all, you can select between Primary Packets or Phy Packets. When Primary Packets are selected, you can select which primary types may fit the set by checking the corresponding checkboxes inside the **Primary Packets** box. In the example below only Write Block Request and Read Block Response packets will fit the set. Inside this box there are also some buttons to make a quick selection of some types. Using the with Data block button for instance you can select the packet types that have an associated data block.

Using the checkboxes inside the **Packet Speeds** box, you can include or exclude packets by their packet speed. By default all speeds will fit the set



[● Primary Packets] ○ Phy Packets

Primary Packets:
☐ Write Request ☐ Write Response
☑ Write Block Request
☐ Read Request ☐ Read Response
☐ Read Block Request ☑ Read Block Response
☐ Lock Request ☐ Lock Response
☐ Stream ☐ Cycle Start

Packet Speeds:
☑ S100
☑ S200
☑ S400
☑ S800

Header Values

Field	Cond.	Value	Mask
Source			
Destination	==	0xFFC1	
Transaction Label			
Retry Code			
Priority			
Destination Offset			
Data Length Async. packet	==	32	
Response Code			
Extended TCode			
Data Length Stream packet			
Channel			
Tag			
Synchronisation Code			

Data Values

payload size (bytes): 32 ☑ fixed size

Payload

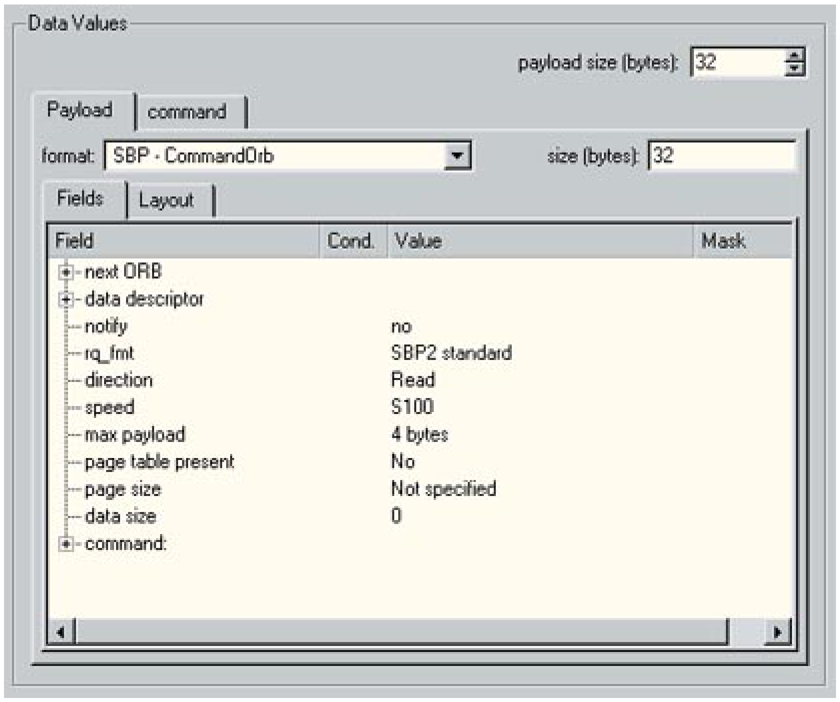
format: VS - Message (payload decimal) size (bytes): 32

Fields | Layout

Field	Condition	Value	Mask
Message ID		0	
Reserved Security		0	
Node ID		0	
Priority		0	
Message Payload Length		0	
Message Payload Data			
STOF Transmit Offset		0	
STOF Receive Offset		0	
STOF Datapump Offset		0	
Vertical Parity Check		0	

In the **Header Values** box, you can specify conditions for header fields. Only packets for which the specified conditions are met will fit the set. The table in this box shows all possible fields that can be present in a 1394 packet header. Only those fields that are present in the selected packet types will be enabled. You can only specify conditions for those enabled fields. If more packet types are selected, only those fields will be enalbed that are present in all the selected types.
In the example above for instance, the Write Block Request and Read Block Response packet types are selected and thus only those fields in the Header Values table are enabled that are present in both of these packet types.
For each enabled field you can specify a value and you can select if the value in the packet should be equal (==), not equal (!=), less than or equal (<=) or greater than or equal (>=) the value specified in tha table. In the example above for instance the packet will only fit the set if its Destination field is 0xFFC1 and the Data Length field is 32.
Additionaly you can specify a Mask value per field. This is a bit mask that (when defined) will be applied to the packet field value and table field value before the field will be compared. For instance if you are only interrested in the lower 6 bits of the Destination field you could specify a Mask value of 0x003F.

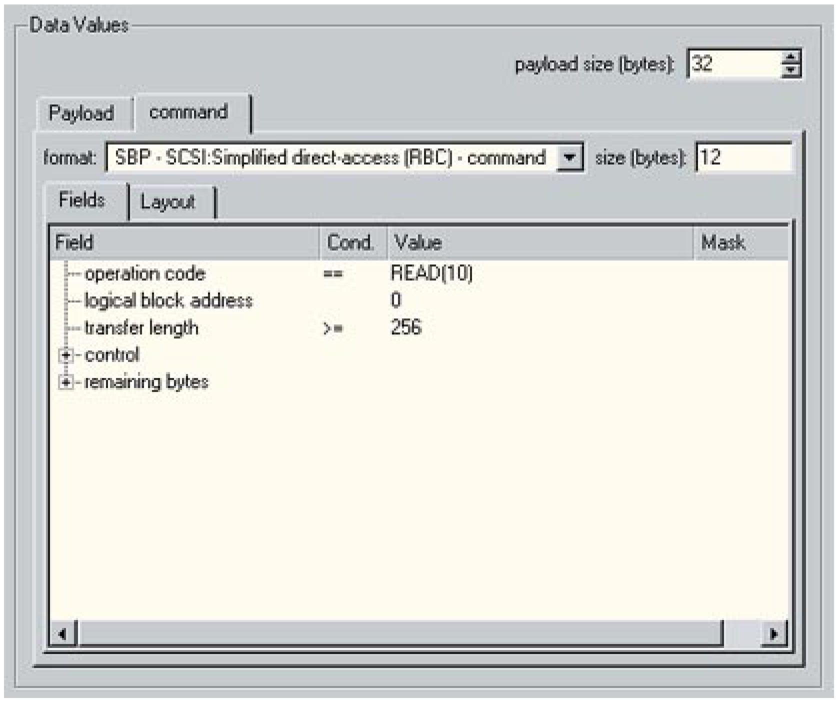
Using the **Data Values** box, you can specify condtions for data fields. Only packets for which the specified conditions are met will fit the set. A powerfull packet editor is used to be able to specify conditions for a large number of possible packets. You have to specify a format for the packet payload data in the Payload tab. In the example below the 'SBP - CommandOrb' format has been selected. This is a packet format used for the SBP2 protocol.



payload size (bytes): 32
format: SBP - CommandOrb
size (bytes): 32

Field	Cond.	Value	Mask
next ORB			
data descriptor			
notify		no	
rq_fmt		SBP2 standard	
direction		Read	
speed		S100	
max payload		4 bytes	
page table present		No	
page size		Not specified	
data size		0	
command:			

The fields in the table show the fields for the payload of this packet format. You can put conditions on these values just as could be done with the header fields. If the format of a packet depends on the value of one of its fields, the format will automatically be adjusted if you change the value of that field.
The last field in the example table above is 'command:'. Each SBP CommandOrb packet has a 12 byte command. There are a lot of possible command formats, and therefore the format of this command can be selected by a sub format. For parts in a format that are formatted by a sub format, a tab will be added. In the example above we see the command tab to the right of the Payload tab.
Selecting this tab will show the formatted command, see the example below. Here you can select a sub format for the command. In the example below the 'SBP - SCSI:Simplified direct-access (RBC) - command' format is selected.



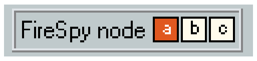
Field	Cond.	Value	Mask
operation code	==	READ(10)	
logical block address		0	
transfer length	>=	256	
control			
remaining bytes			

The selected format describes the commands for a 'Simplified direct-access (RBC)' unit. By choosing a value for the 'operation code' field, the format will automatically adjust to the format of the selected command.

In the example conditions are set on some fields such that the packet will only fit the set if the 'operation code' equals the 'READ(10)' command with a 'transfer length' or 256 or more.

12.2. Details

Triple Analyzers
For triple Analyzers only, the toolbar contains the following control to select the active Analyzer node.



This control can be used to select the Analyzer node to set Filter/Trigger conditions for. By clicking on the "a", "b" or "c", the corresponding node can be selected. A red background indicates the node is shown. A white background indicates a node is hidden.

12.2.1. Menus

File
The following menu items can be selected:

Open
With this command you can open an existing Analyzer filter/trigger settings file. By default these files have an extension of .fsf.
In the Settings Dialog you can select a file that should be opened automatically when the Analyzer application is started.

Save
With this command you can save the current filter/trigger settings as a Analyzer filter/trigger settings file. By default these files have an extension of .fsf.
Filter/trigger settings saved this way, can later be viewed again by using the 'Open' command of the 'File' menu.
If no file name was given before, a file dialog is displayed to enter the file name and path.

Save As

This commands does the same as the 'Save' command, except that it will always display the file dialog, so that you can specify a file name and path.

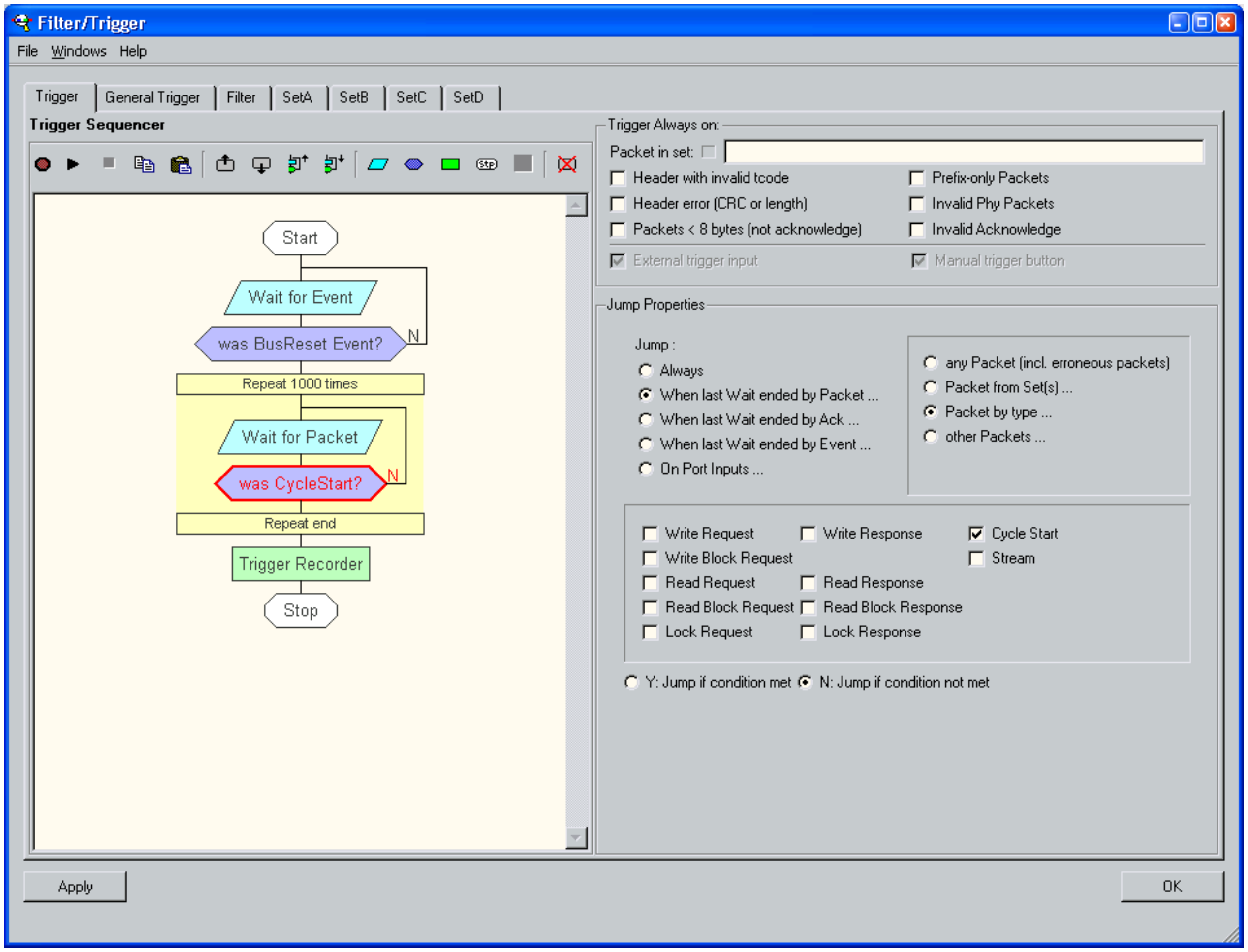
Windows
From the 'Windows' menu you can open one of the other windows of the Analyzer.

12.2.2. Trigger

The Trigger Sequencer is not available on

- Stealth Series in Symbol Recorder mode

An example of the Trigger page is shown below.



Trigger Always on:
Packet in set: ☐
☐ Header with invalid tcode ☐ Prefix-only Packets
☐ Header error (CRC or length) ☐ Invalid Phy Packets
☐ Packets < 8 bytes (not acknowledge) ☐ Invalid Acknowledge
☑ External trigger input ☑ Manual trigger button

Jump Properties
Jump :
○ Always
● When last Wait ended by Packet ...
○ When last Wait ended by Ack ...
○ When last Wait ended by Event ...
○ On Port Inputs ...

○ any Packet (incl. erroneous packets)
○ Packet from Set(s) ...
● Packet by type ...
○ other Packets ...

☐ Write Request ☐ Write Response ☑ Cycle Start
☐ Write Block Request ☐ Stream
☐ Read Request ☐ Read Response
☐ Read Block Request ☐ Read Block Response
☐ Lock Request ☐ Lock Response

○ Y: Jump if condition met ● N: Jump if condition not met

In this example a little more complex Trigger Sequencer program flow is used. And a cyclic pre-trigger buffer has been enabled that will never use more than 25% of the total Recorder memory. We will discuss the different part of this page in more details below.

Trigger Sequencer
At the top of the Trigger Sequencer box there are some indicators and tool buttons. In the remaining area of the box the Trigger Sequencer program flow diagram is displayed. A flow can be defined by adding items to the program flow and changing their properties.
We will first discuss the indicators and tool buttons and after that the program flow diagram area:

Trigger sequencer active indicator
This indicator will light when the trigger sequencer is active. The sequencer will be active when it is started and has not been stopped yet. An active sequencer will always be executing one of the items in the trigger sequencer program flow. The item currently being executed is indicated with a red arrow (see example above).
If a 'Stop' item is executed, the sequencer will stay executing this 'Stop' item and as a result of this, it stays active but it does not execute any other item anymore.

button: Start or restart trigger sequencer
Clicking this button will activate the sequencer (if it wasn't already active) and the sequencer will start executing the first item in the trigger sequencer program flow. When the filter/trigger settings need to be uploaded (Apply button enabled), the user will be asked if the settings must be uploaded before (re)starting the sequencer. The user can choose to upload, cancel the (re)start or ignore it. If ignored, the sequencer will run with the old settings information previously loaded into the Analyzer.

button: Copy sequence
Copy sequence to the clipboard.

button: Paste sequence
Paste sequence from the clipboard..

button: Stop trigger sequencer
Clicking this button will deactivate the sequencer. The button will be enabled if the sequencer is active.

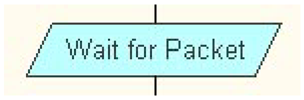
button: Move item up
With this button the currently selected item can be moved up. If is was a Jump item its jump target will be unchanged. If you move the Repeat-start item up, the item previously above the Repeat-start item will be included in the repeat block. If you move a Repeat-end item up, the item previously above the Repeat-end item (which was included in the repeat block) will be moved out the repeat block. Note that you can not move Repeat items up passing other Repeat items. For more information about Repeat items, see below. You can not move items above the firs item (the Start item).

button: Move item down
With this button the currently selected item can be moved down. If is was a Jump item its jump target will be unchanged. If you move the Repeat-start item down, the item previously below the Repeat-start item (which was included in the repeat block) will be moved out the repeat block. If you move a Repeat-end item down, the item previously below the Repeat-end item will be included in the repeat block. Note that you can not move Repeat items down passing other Repeat items. For more information about Repeat items, see below. You can not move items below the last item.

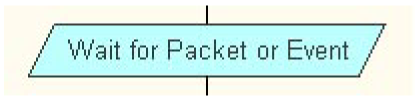
button: Move jump target up
When a Jump item is selected, you can move the jump target for this jump one item up. The button will be disabled if the jump target is already the highest possible target.

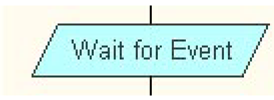
button: Move jump target down
When a Jump item is selected, you can move the jump target for this jump one item down. The button will be disabled if the jump target is already the lowest possible target.

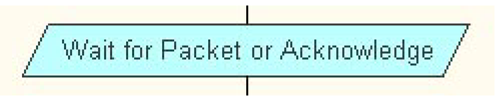
button: Add a Wait item
When clicking this button, a new Wait item will be inserted in the flow below the selected item. Initially the Wait item looks as follow:



When this Wait item is executed, the sequencer will wait for a packet. Any packet received will end the execution of it and the sequencer continues with the execution of the next item in the flow.
When the Wait item is selected, you can change its properties in the Wait Properties box. See below for the description of these properties. When the properties are changed, the text in the Wait item will reflect the new properties. Here are some examples:

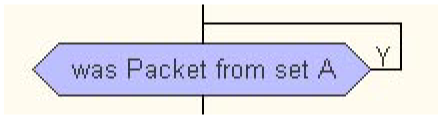




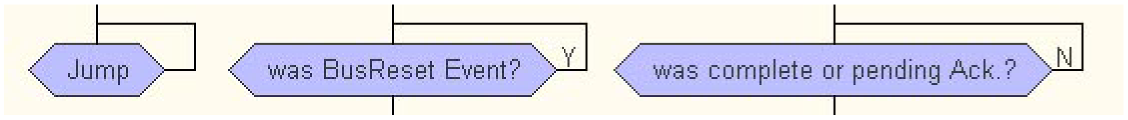


Below the Wait item will be described in more details.

button: Add a Jump item
When clicking this button, a new Jump item will be inserted in the flow below the selected item. Initially the Jump item looks as follow:

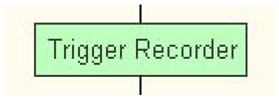


When this Jump item is executed, the sequencer will jump if the last Wait was ended by the reception of a packet and the received packet fits the description of setA. The sequencer will jump to the item as indicated by the jump line which starts from the right of the Jump item. In this case it jumps to itself. You should change the jump target by using the Move jump target up or Move jump target down button as described above.
When the Jump item is selected, you can change its properties in the Jump Properties box. See below for the description of these properties. When the properties are changed, the text in the Jump item will reflect the new properties. Here are some examples:



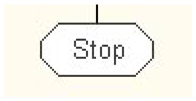
Below the Jump item will be described in more details.

button: Add an Action item
When clicking this button, a new Action item will be inserted in the flow below the selected item. The Action item looks as follow:



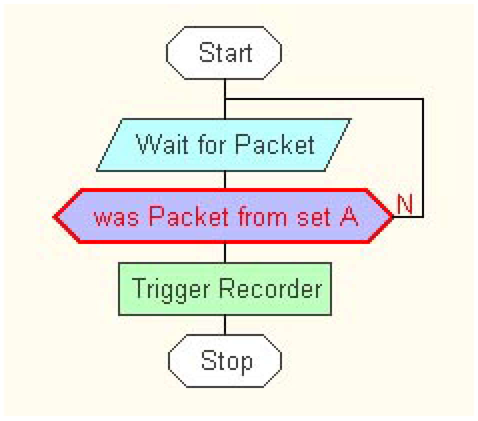
When this Action item is executed, the Recorder will be triggered.
When the Action item is selected, you can change its properties in the Action Properties box. At this moment the only option is to Trigger the Recorder.

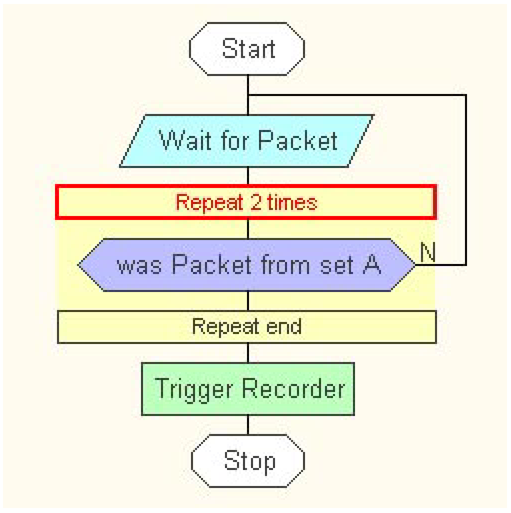
button: Add a Stop item
When clicking this button, a new Stop item will be inserted in the flow below the selected item. The Stop item looks as follow:



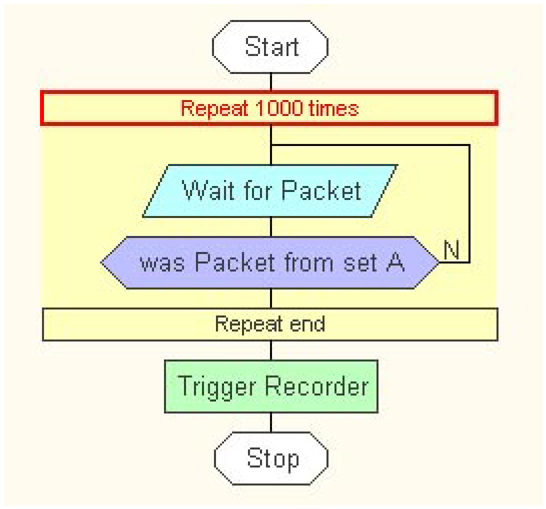
When the Stop item is executed, the sequencer will keep executing the Stop item forever. This effectively stops executing the program. Note that the last item in the flow must be a Stop item and that more Stop items may exist in the flow.

button: Add Repeat
When this button is clicked, the selected item will be enclosed in a repeat box. This button will only be enabled if the selected item can be enclosed in a Repeat box. Repeats can not be nested, thus when an item already is enclosed in a Repeat box, the button will be disabled. The two flow examples below are examples of the situation before the Add Repeat button has been clicked (left) and after the button have been clicked (right).





The result is that the Jump item is enclosed in a Repeat-start and a Repeat-end item. Executing this flow would repeat the Jump item 2 times, which does not make any sense. You can expand the repeat block by using the Move item up button when the Repeat-start item is selected, or using the Move item down button when the Repeat-end item is selected. In the example below, the Repeat-start item has been moved one item higher. Also the number of times the repeat box should be executed has been increased to 1000. This can be done in the Repeat Properties box, when the Repeat-start item is selected.



Now the flow does something meaningful: the Recorder will be triggered when 1000 packets have been detected that fits the description for packet setA.
Below the Repeat item will be described in more details.

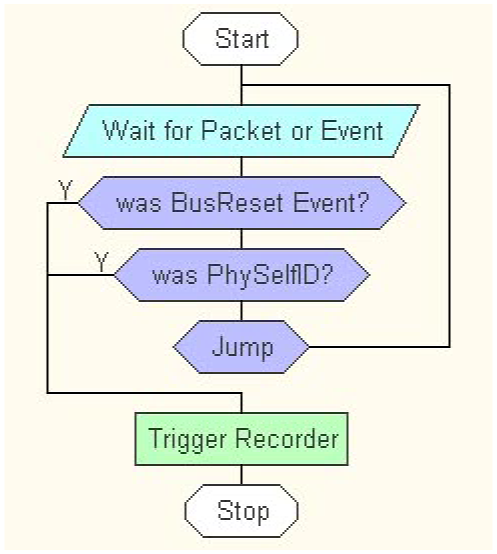
button: Delete Item
Clicking this button will delete the selected item.

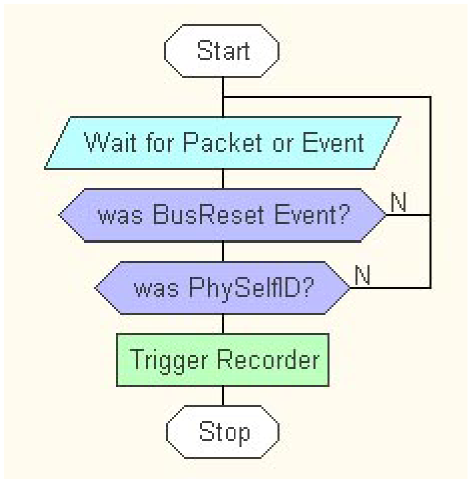
flow diagram area
In this area the flow diagram will be drawn. The flow always start with a Start item and ends with a Stop item. Inbetween one or more Wait, Jump, Action or Stop items may exist and groups of items can be enclosed in a Repeat box.
When the sequencer is started, the execution starts after the Start item and continues until a Stop item is executed. A red arrow will indicate the currently executed item (see above).
Items can be selected by clicking with the mouse on an item. New items can be inserted and existing ones can be removed using the buttons above the flow diagram area. Items can also be moved up and down in the flow and for Jump items the jump target can be moved up and down using buttons above the flow diagram area. Groups of items can be enclosed in a Repeat box by using the Add Repeat button and the number of items enclosed in the Repeat box can be changed by moving the Repeat-start or Repeat-end items up or down.
When an item is selected, the Properties pane will show the properties of the item. The properties can be

changed and the text in the item will indicate the new properties of the item. The properties of all items are described in more details below.

Now we will discuss the different types of items and describe what happens when they are executed.

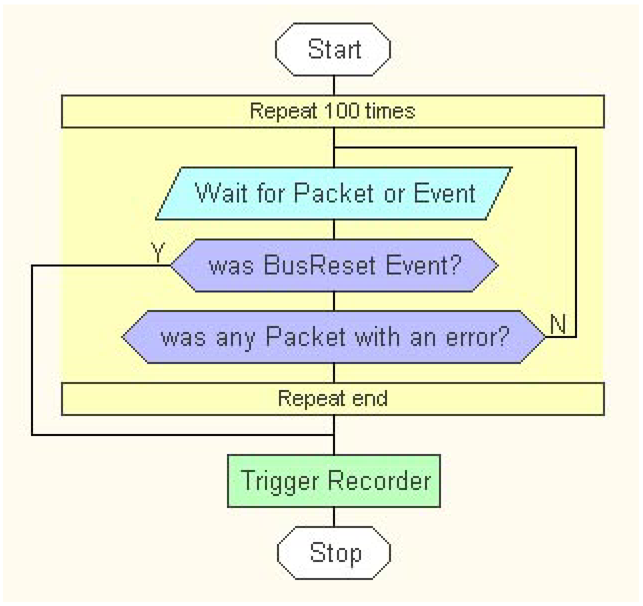
- *Start:* The Start item is the starting point of the trigger program. The item after the Start item will be executed if the Trigger Sequencer is (re)started. It has no properties and can not be deleted.
- *Wait:* A Wait item is used to make the sequencer wait for a packet and/or acknowledge and/or event. When a Wait item is selected, you can select where to wait for in the properties box. You can select to wait for a packet, an acknowledge or an event, or any combination of them. When a Wait item is executed, the program does not continue until the packet, acknowledge or event is detected.
  For instance if an Wait item is executed that waits for a packet or event, then the item after the Wait is executed when a packet or an event is detected on the bus. It does not matter what kind of packet or event. You can use the Jump item to check what was detected (Packet or Event). If a packet was detected, the Jump item can also be used to check the packet type, content or errors. And if an event was detected, the Jump can be used to differentiate on event type.
- *Jump:* When a Jump item is selected, you specify a jump-condition in the properties box. The specified condition is indicated by the text in the Jump item. You can also specify if the sequencer should jump to the target if the jump-condition is met (true) or if the jump-condition is not met (not true) (indicated with a Y or N at the jump line). If the jump is taken, the item executed next is the item where the jump line points to (the line going to the right or left from the Jump item). If the jump is not taken, the item after the Jump item will be executed next.
  The jump-condition refers to the result of the last executed Wait item. So if the last executed Wait item stopped executing because a CycleStart packet was detected (note that this only can happen if this Wait item has the 'wait for packet' property included), then the following condition are for instance true:
  'was Packet?'
  'was CycleStart?'
  'was WrBlockReq or RdBlockReq or CycleStart?'
  'was Packet from set A or B?' assuming that a CycleStart packet fits set A or set B
  And the following are for instance not true:
  'was Event?'
  'was BusReset Event?'
  'was Ack.?'
  'was complete or pending Ack.?'
  'was WrBlockReq or RdBlockReq?'
  'was Packet from set D?' assuming that a CycleStart packet does not fit set D
  'was PhySelfID?'
  'was any Packet with an error?'
  You can use more than one Jump item after a Wait item. For example the execution of the left flow below will result in a Recorder trigger as soon as a BusReset or PhySelfID packet has been detected. The Wait will execute until any event or packet has been detected and the two first Jump items will jump if a BusReset or PhySelfID was detected. It not, the third Jump will make sure the Wait is executed again.





Note that an Acknowledge will not end the Wait execution and thus is not detected. Note also that the flow on the right does not do the same thing! This is because the first Jump item after the Wait item will jump if the Wait was not ended because of a BusReset event. If the Wait was ended because of the detection of packet, including a PhySelfID packet, then this Jump item will result in a jump to the Wait again. And thus the Recorder will not be triggered if a PhySelfID has been detected.

- *Action:* When an Action item is executed, the indicated action will be performed and the next item will be executed. When an Action item is selected, the actions to be performed can be selected in the properties box.
- *Stop:* When the Stop item is executed, the program will continuously execute the Stop item, effectively stopping the program. The Sequencer will stay active and the red arrow indicating the item being executed will continuously point to the Stop item.
  The Stop item has no properties and the last Stop item can not be deleted. The last item of the flow always should be the Stop item and thus you can not insert new items after the last Stop item. But Stop items may be added in the middle of a flow to stop the Sequencer at any point. These Stop items can be deleted and new items can be inserted after such a Stop item.
- *Repeat:* All items enclosed in a Repeat-start and Repeat-end item are repeated the number of times indicated in the Repeat-start item. This number can be changed in the properties box when the Repeat-start is selected. The complete group of items to be repeated are inside the yellow colored box. A repeat loop can be aborted by jumping out the loop. The Repeat-start will set a repeat-counter with the number of times to executed the block. The Repeat-end will decrement the repeat-counter and will jump to the item after the Repeat-start item when this count is not 0.
  For example the following simple example will Trigger the Recorder when 100 Packets with an error are detected or when one BusReset is detected.



Trigger Position
In this box you can specify the trigger position in the Recorder memory and you can enable a cyclic buffer for the data before the trigger.

Stop recording when buffer full
Selecting this option (default) will cause the Recorder to stop when the Recorder buffer is completely filled. The Recorder will also stop when the data after the trigger reached the the amount indicated by the Max post-trigger size slider.

Cyclic pre-trigger buffer
Selecting this option will enable a cyclic buffer for the data recorded before the Recorder has been triggered. The size of this cyclic pre-trigger buffer is indicated by the Max pre-trigger size slider. If the recorded data reaches the amount indicated by this slider, the oldest data in the pre-trigger buffer will be overwritten by the new data.
Before the Recorder is triggered, the total used Recorder memory will never be more than indicated by

the Max pre-trigger size slider. As soon as the Recorder is triggered, the data in the pre-trigger buffer will not be overwritten anymore and the remaining Recorder memory will be filled. The Recorder will stop when the complete Recorder memory has been filled or when the data after the trigger reached the the amount indicated by the Max post-trigger size slider.

Max pre-trigger size
This slider is only enabled when the cyclic pre-trigger buffer option is enabled. With this slider you can specify the size of this pre-trigger buffer expressed in % of the whole Recorder memory. You can also type a value in the text box at the right from the slider.

Max post-trigger size
With this slider you can specify the amount of Recorder memory to be used maximal for data after the Recorder has been triggered. If this amount have been reached, the Recorder will be stopped. Note that the Recorder will also stop when the complete buffer has been filled. You can also type a value in the text box at the right from the slider.

Trigger Always on
Inside this box you can select packets on which the Recorder should always be triggered. This works independent form the Trigger Sequencer. Both the Trigger Sequencer and the detection of packets specified here can trigger the Recorder.

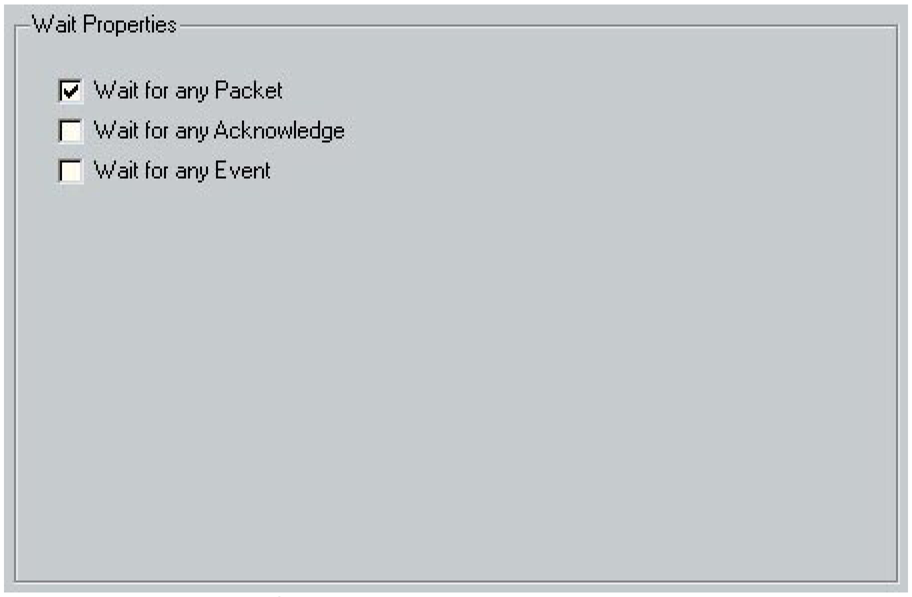
Packet in set:
Here you can specify a Set or any boolean combination of the four packet Sets. If a packet is detected that fits the Set or combination of Sets, the Recorder will be triggered.
The set names that can be used are 'seta', 'setb', 'setc' and 'setc'. The names are not case sensitive, thus 'SetA' is valid too. You also can use just 'a', 'b', 'c' or 'd' (or 'A', 'B' etc.) for the set names. The Boolean operators are 'not', 'and' and 'or'. But they may be substituted by the characters '!', '&' and '|' respectively. Parenthesis may be used to group the Boolean operation.
Some examples of valid expressions are:

```
setc
NOT SETB
(seta or setb) and not setd
!SetA & (SetC | SetB)
a
C or D
```

Note that a red cross will be displayed before the boolean expression when there is an error in the expression.

Properties
In this box the properties of the flow item selected in the Trigger Sequencer will be displayed. For each item type there is a separate property page. The name of the properties box will indicate this type. The following property pages can be displayed:

- **No Properties:** If no flow item is selected or an item is selected with no properties then this page is displayed. Items with no properties are the Start item, Stop item and Repeat-end item. It contains no properties.
- **Wait Properties:** When a Wait item is selected, this properties page will be displayed

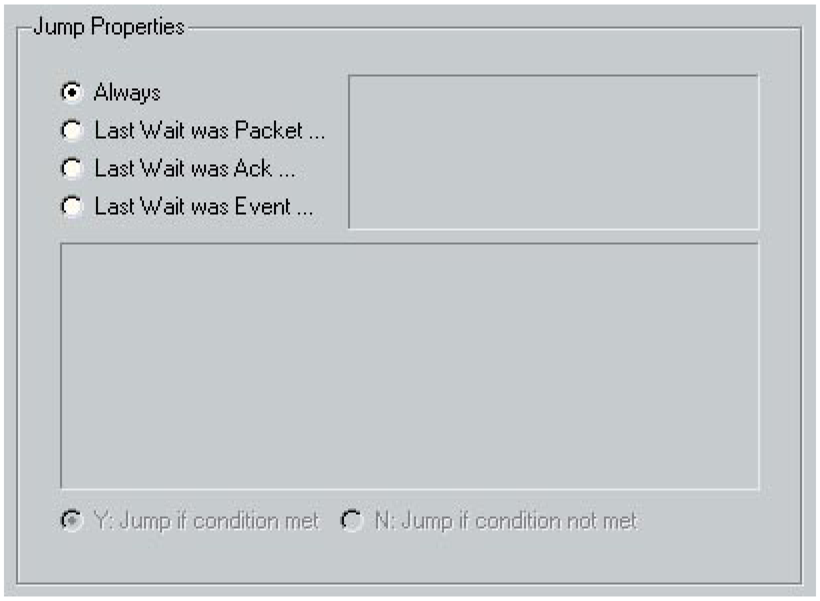


Wait Properties

[✓] Wait for any Packet
[ ] Wait for any Acknowledge
[ ] Wait for any Event

You can select to wait for a Packet, an Acknowledge, an Event, or any combination of them. See flow diagram area above for a description of what happens when a Wait item is executed. The text inside the Wait item will indicate the selected properties.

- **Jump Properties:** Using this properties, you can specify the jump-condition and if the sequencer should jump if the jump-condition is met or jump if the condition has not been met. See flow diagram area above for a description of what happens when a Jump item is executed. The text inside the Jump item will indicate the selected jump-condition.
  First of all you must specify if the condition specifies a Packet, Acknowledge or Event or if the there is no condition at all. This selection is made with the radio buttons at the top left of the Jump Properties box. See example below.



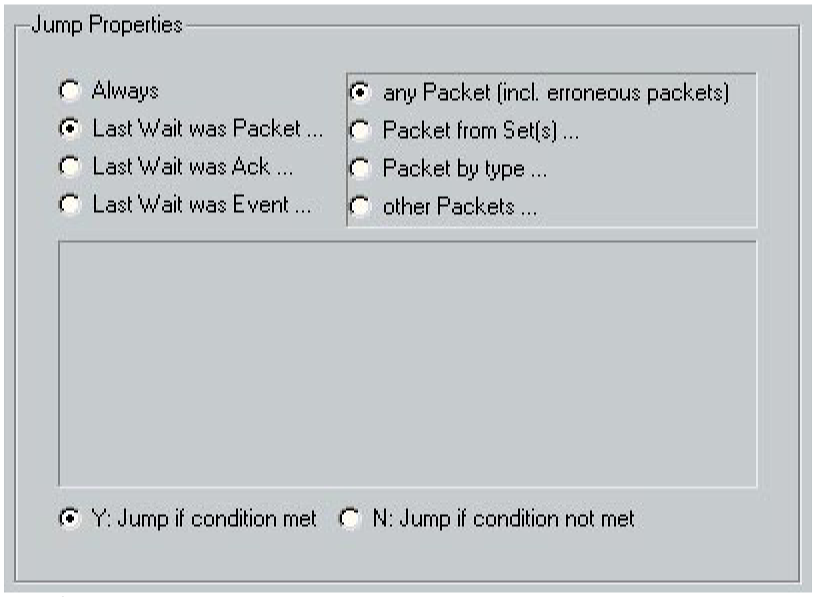
Jump Properties

[✓ Always]
Last Wait was Packet ...
Last Wait was Ack ...
Last Wait was Event ...

[✓ Y: Jump if condition met] N: Jump if condition not met

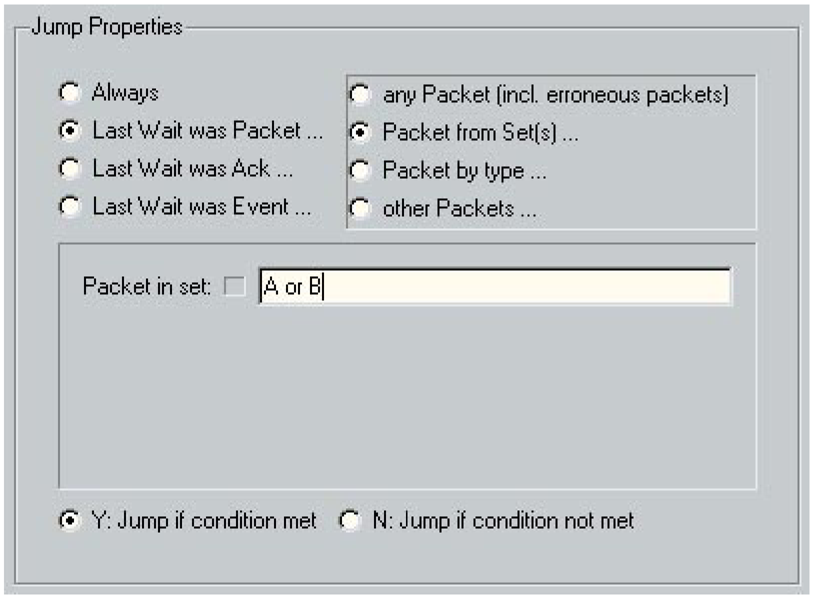
The following options are possible:

  - *Always:* If the Always option is selected, the jump will always be taken. There is no additional selection needed.
  - *Last Wait was Packet ....:* This option is used to specify a Packet condition. If the last Wait was ended because of the detection of an Acknowledge or Event, then the result of the jump-condition is always false. The three ellipses indicate that additional sub-selections are possible. When selecting this option, the additional sub-selection options appear at the right top of the Jump Properties box. See example below.



The following sub-selections are possible:

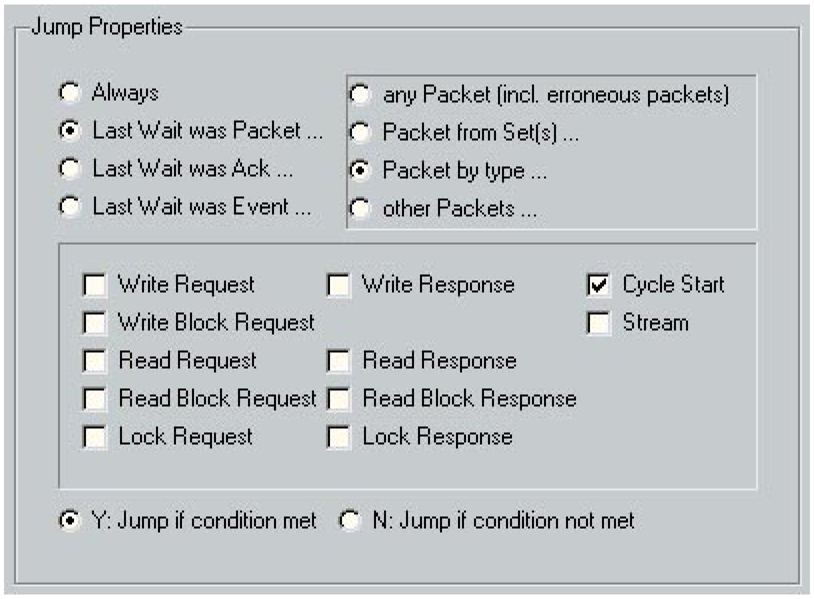
- *Any Packet (incl. erroneous packets):* If this option is selected, the jump-condition will be true for every packet that is detected, including packets with an error or invalid packets.
- *Packet from Set(s)...:* If this option is selected, you can specify a packet Set or a boolean combination of packet Sets. For every packet that fits the set or combination of sets, the jump-condition will be true.



Packet in set: A or B

In the example above, the jump-condition is true for each packet that fits SetA or SetB.

- *Packet by type ...:* If this option is selected, additional check boxes appear below where you can specify one or more packet types. If a packet was detected with a type for which the corresponding checkbox is checked, the jump-condition is true.



Jump Properties

- ( ) Always
- (•) Last Wait was Packet ...
- ( ) Last Wait was Ack ...
- ( ) Last Wait was Event ...

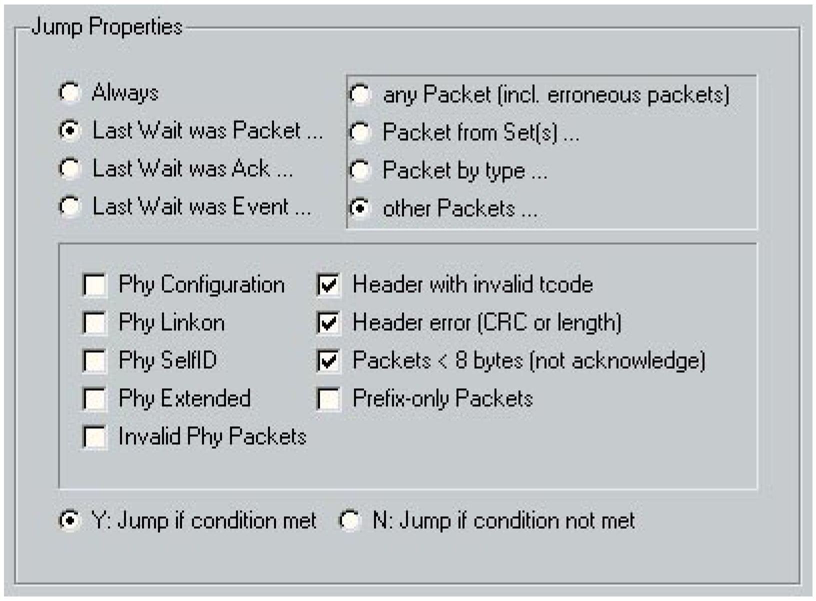
- ( ) any Packet (incl. erroneous packets)
- ( ) Packet from Set(s) ...
- (•) Packet by type ...
- ( ) other Packets ...

- [ ] Write Request
- [ ] Write Block Request
- [ ] Read Request
- [ ] Read Block Request
- [ ] Lock Request
- [ ] Write Response
- [ ] Read Response
- [ ] Read Block Response
- [ ] Lock Response
- [✓] Cycle Start
- [ ] Stream

(•) Y: Jump if condition met ( ) N: Jump if condition not met

In the example above, the jump-condition is only true for CycleStart packets.

- *other Packets ....:* If this option is selected, additional check boxes appear where you can specify some special packet types or some packet errors. If a packet was detected with a type for which the corresponding checkbox is checked or with an error for which the corresponding checkbox is checked, the jump-condition is true.



Jump Properties

- ( ) Always
- (•) Last Wait was Packet ...
- ( ) Last Wait was Ack ...
- ( ) Last Wait was Event ...

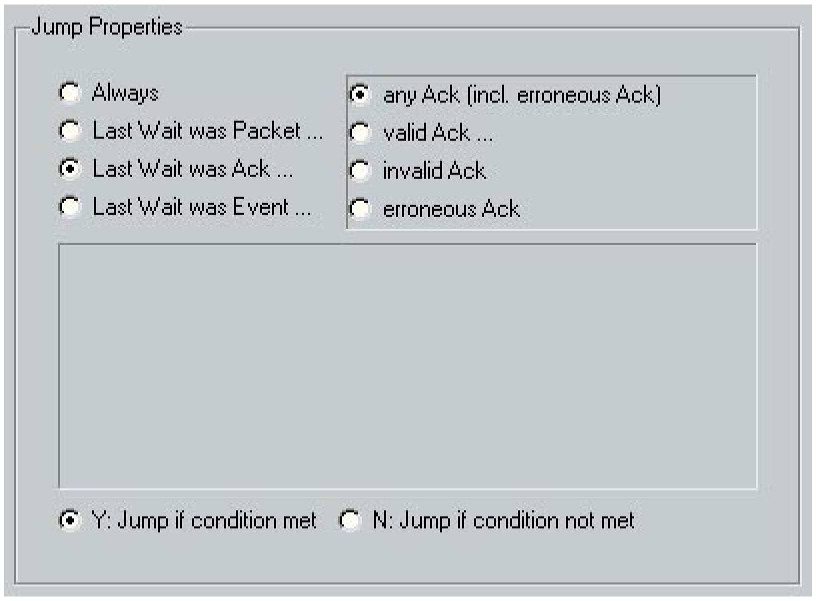
- ( ) any Packet (incl. erroneous packets)
- ( ) Packet from Set(s) ...
- ( ) Packet by type ...
- (•) other Packets ...

- [ ] Phy Configuration
- [ ] Phy Linkon
- [ ] Phy SelfID
- [ ] Phy Extended
- [ ] Invalid Phy Packets
- [✓] Header with invalid tcode
- [✓] Header error (CRC or length)
- [✓] Packets < 8 bytes (not acknowledge)
- [ ] Prefix-only Packets

(•) Y: Jump if condition met ( ) N: Jump if condition not met

In the example above, the jump-condition is true for packets with an error.

- *Last Wait was Ack ....:* This option is used to specify an Acknowledge condition. If the last Wait was ended because of the detection of an Packet or Event, then the result of the jump-condition is always false. The three elipses indicate that additional sub-selections are possible. When selecting this option, the additional sub-selection options appear at the right top of the Jump Properties box. See example below.



Jump Properties

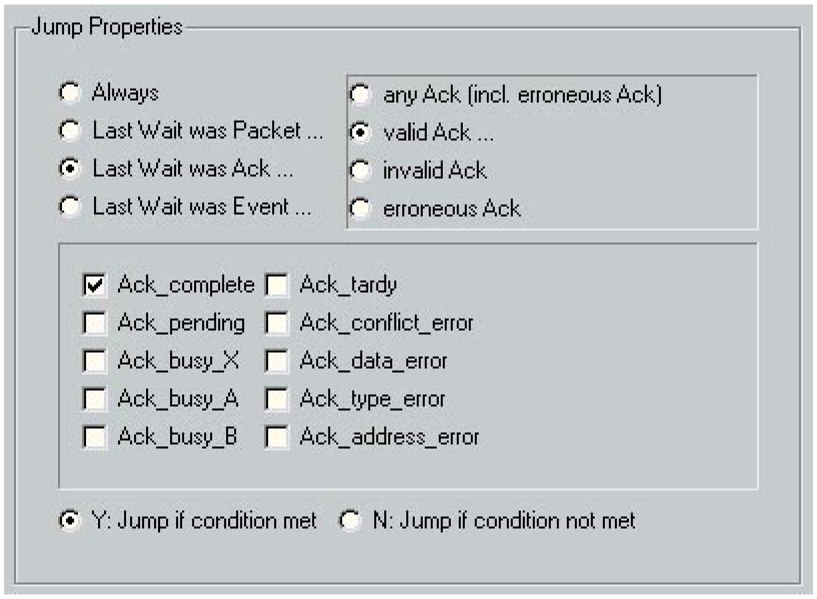
- ( ) Always
- ( ) Last Wait was Packet ...
- (●) Last Wait was Ack ...
- ( ) Last Wait was Event ...

- (●) any Ack (incl. erroneous Ack)
- ( ) valid Ack ...
- ( ) invalid Ack
- ( ) erroneous Ack

(●) Y: Jump if condition met ( ) N: Jump if condition not met

The following sub-selections are possible:

- *Any Ack (incl. erroneous Ack):* If this option is selected, the jump-condition will be true for every acknowledge that is detected, including erroneous acknowledges or invalid acknowledges.
  Erroneous acknowledges are 8 bits long (which identifies an acknowledge), for which the first 4 bits are not the inverse of the last 4 bits. Invalid acknowledges are 8 bits long for which the first 4 bits correctly are the inverse of the last 4 bits, but with an invalid acknowledge code.
- *valid Ack ...:* If this option is selected, additional check boxes appear where you can specify one or more valid acknowledge codes. If a valid acknowledge was detected with a code for which the corresponding checkbox is checked, the jump-condition is true.

  Jump Properties

  - ( ) Always
  - ( ) Last Wait was Packet ...
  - (●) Last Wait was Ack ...
  - ( ) Last Wait was Event ...

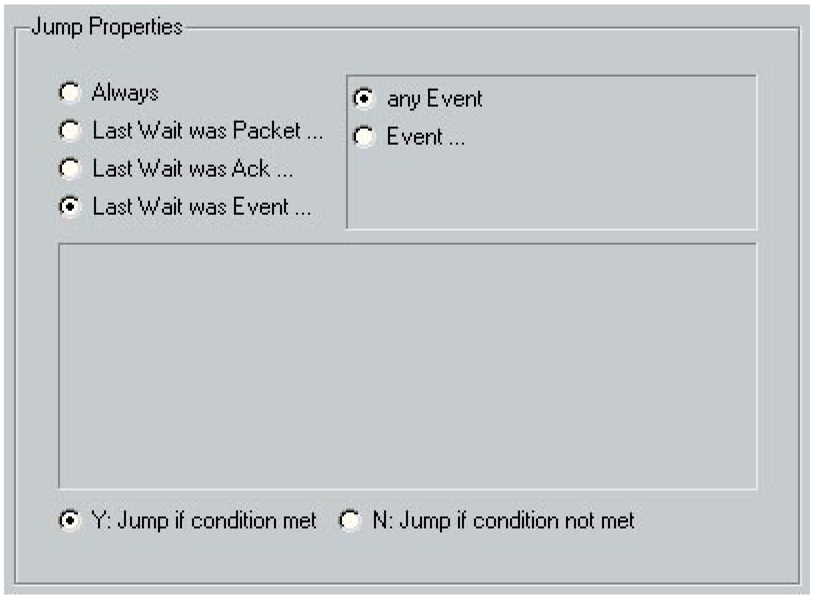
  - ( ) any Ack (incl. erroneous Ack)
  - (●) valid Ack ...
  - ( ) invalid Ack
  - ( ) erroneous Ack

☑ Ack_complete	☐ Ack_tardy
☐ Ack_pending	☐ Ack_conflict_error
☐ Ack_busy_X	☐ Ack_data_error
☐ Ack_busy_A	☐ Ack_type_error
☐ Ack_busy_B	☐ Ack_address_error

  (●) Y: Jump if condition met ( ) N: Jump if condition not met

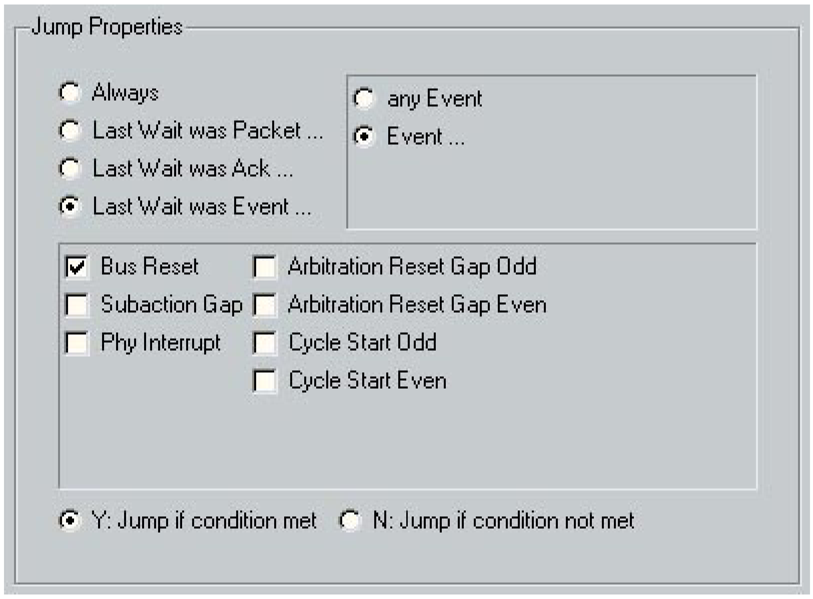
  In the example above, the jump-condition is only true for an acknowledge with code 'Ack_complete'.
- *invalid Ack:* If this option is selected, the jump-condition is true if an invalid acknowledge was detected. An invalid acknowledge is 8 bits long and the first 4 bits are the inverse of the last 4 bits, but with an invalid acknowledge code.
- *erroneous Ack:* If this option is selected, the jump-condition is true if an erroneous acknowledge was detected. An erroneous acknowledge is 8 bits for which the first 4 bits are not the inverse of the last 4 bits.

- *Last Wait was Event ...:* This option is used to specify a Event condition. If the last Wait was ended because of the detection of Packet or Acknowledge, then the result of the jump-condition is always false. The three elipses indicate that additional sub-selections are possible. When selecting this option, the additional sub-selection options appear at the right top of the Jump Properties box. See example below.



The following sub-selections are possible:

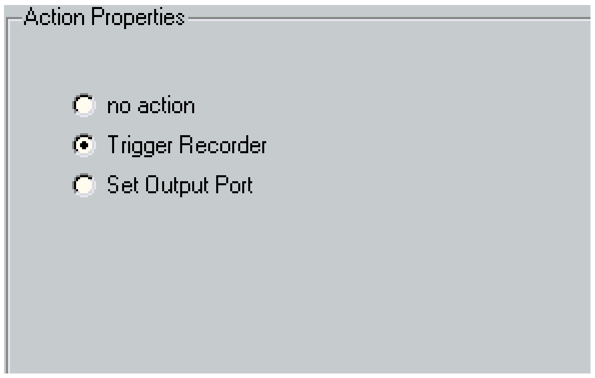
- *Any Event:* If this option is selected, the jump-condition will be true for every event that is detected.
- *Event ...:* If this option is selected, additional check boxes appear where you can specify one or more events. If an event was detected for which the corresponding checkbox is checked, the jump-condition is true.



In the example above, the jump-condition is only true for a BusReset Event.

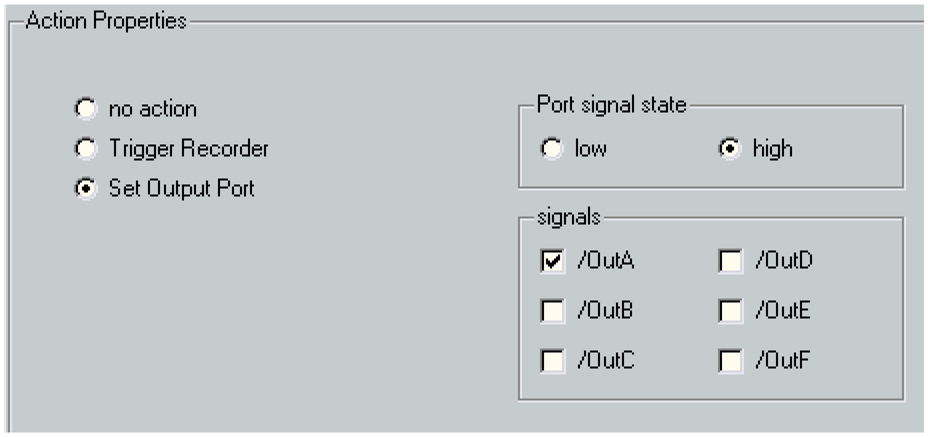
At the bottom of the Jump Properties page you can select if the jump should be taken if the specified jump-condition is true (radio button Y: Jump if condition met) or if the jump-condition is false (radio button N: Jump if condition not met).
The letter 'Y' or 'N' drawn on the jump line left or right of the Jump item (in the flow diagram) indicates this selection.

- **Action Properties:** Using this properties page you can select the actions to be performed for an Action item. The page is shown below.



Action Properties

- no action
- [✓ Trigger Recorder]
- Set Output Port



Action Properties

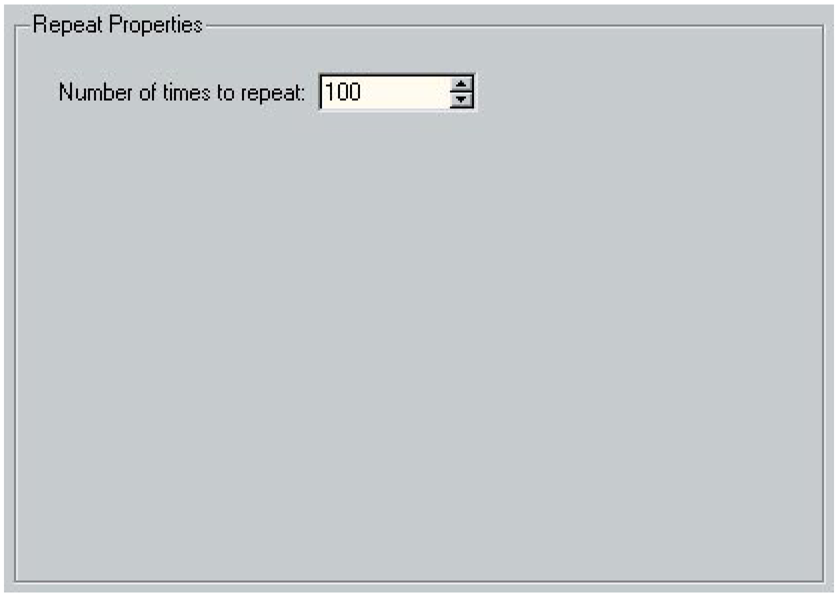
- no action
- Trigger Recorder
- [✓ Set Output Port]

Port signal state: low [✓ high]

signals: [✓ /OutA] /OutB /OutC /OutD /OutE /OutF

The second action, Trigger Recorder will trigger the Recorder.
The third action, Set Output Port will set the specified port to the specified state.
The text inside the Action item will indicate the selected properties.

- **Repeat Properties:** If a Repeat-start item is selected, this properties page will be displayed. Here you can specify the number of repeats. See example below.



Repeat Properties

Number of times to repeat: 100

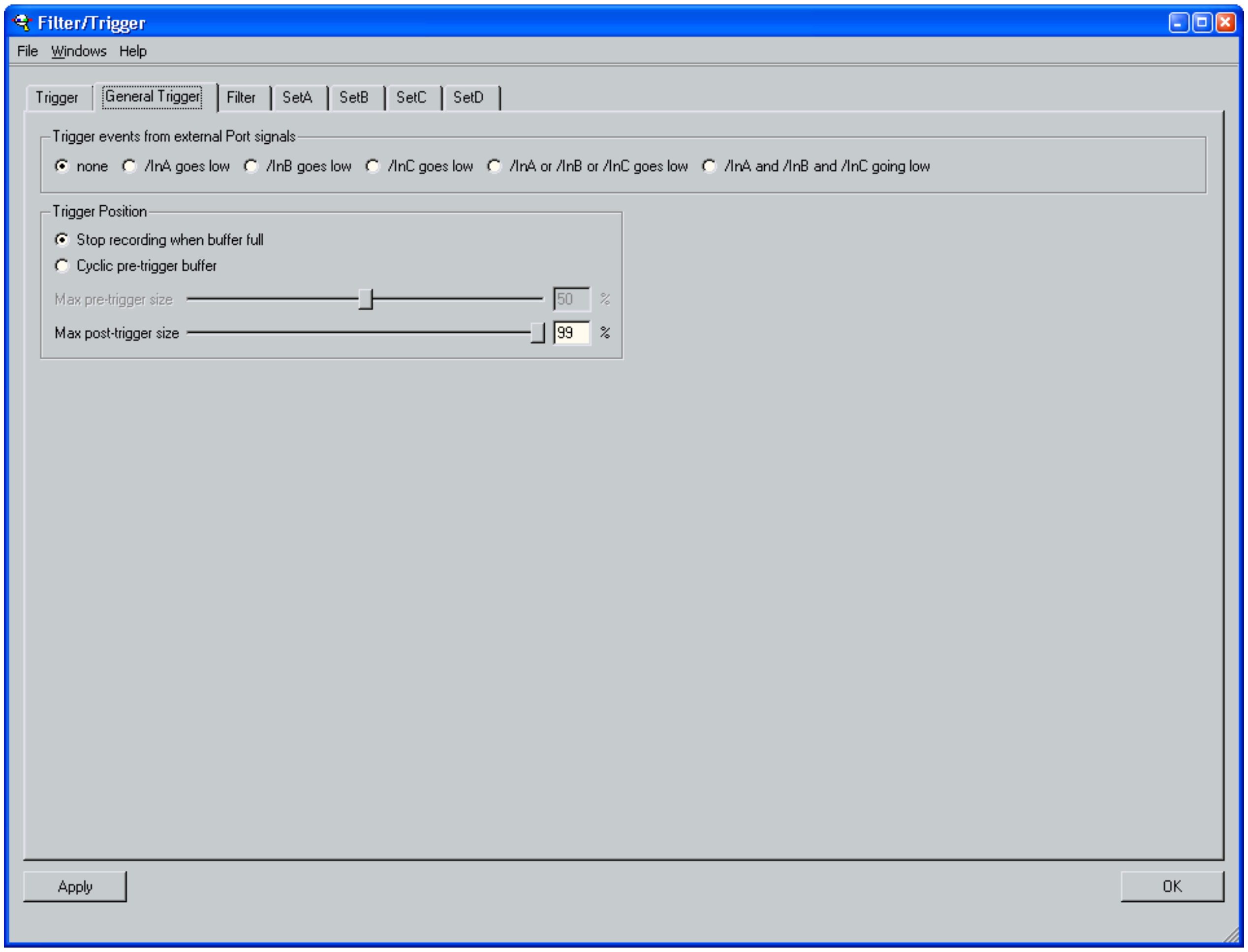
See flow diagram area above for a description of how a Repeat box is executed.

12.2.3. General Trigger

Cyclic buffer mode is not available on

- Third Generation Analyzers

An example of the General Trigger page is shown below.



Trigger events from external Port signals
[✓ none] /InA goes low /InB goes low /InC goes low /InA or /InB or /InC goes low /InA and /InB and /InC going low

Trigger Position
[✓ Stop recording when buffer full]
Cyclic pre-trigger buffer
Max pre-trigger size 50 %
Max post-trigger size 99 %

Trigger events from external Port signals
In this box you can specify the external ports input which will cause the trigger. For detailed pins/ports settings, please refer to External Port Settings.

none
No trigger occurs.

/InA goes low
A trigger occurs when a signal on port InA becomes low.

/InB goes low
A trigger occurs when a signal on port InB becomes low.

/InC goes low
A trigger occurs when a signal on port InC becomes low.

/InA or /InB or /InC goes low
A trigger occurs when a signal on port InA or InB or InC become low.

/InA and /InB and /InC going low
A trigger occurs when all signals on port InA and InB and InC become low.

Trigger Position
In this box you can specify the trigger position in the Recorder memory and you can enable a cyclic buffer for the data before the trigger.

Stop recording when buffer full
Selecting this option (default) will cause the Recorder to stop when the Recorder buffer is completely filled. The Recorder will also stop when the data after the trigger reached the the amount indicated by the Max post-trigger size slider.

Cyclic pre-trigger buffer
Selecting this option will enable a cyclic buffer for the data recorded before the Recorder has been triggered. The size of this cyclic pre-trigger buffer is indicated by the Max pre-trigger size slider. If the recorded data reaches the amount indicated by this slider, the oldest data in the pre-trigger buffer will be overwritten by the new data.
Before the Recorder is triggered, the total used Recorder memory will never be more than indicated by the Max pre-trigger size slider. As soon as the Recorder is triggered, the data in the pre-trigger buffer will not be overwritten anymore and the remaining Recorder memory will be filled. The Recorder will stop when the complete Recorder memory has been filled or when the data after the trigger reached the the amount indicated by the Max post-trigger size slider.

Max pre-trigger size
This slider is only enabled when the cyclic pre-trigger buffer option is enabled. With this slider you can specify the size of this pre-trigger buffer expressed in % of the whole Recorder memory. You can also type a value in the text box at the right from the slider.

Max post-trigger size
With this slider you can specify the amount of Recorder memory to be used maximal for data after the Recorder has been triggered. If this amount have been reached, the Recorder will be stopped. Note that the Recorder will also stop when the complete buffer has been filled. You can also type a value in the text box at the right from the slider.

Cyclic Buffer sizes Example

Suppose you have some bug, on which you could trigger, and after a while the communication stalls. So you would like to trigger and record all data until no packets are present anymore. Suppose you want a minimum of 50% pre-trigger data, but more if possible. And you want as much as pre-trigger data as possible. Then you can put the pre-trigger at maximum, and the post-trigger to 50%.
By doing so, you'll get as much pre-trigger as possible and the amount of post-trigger depends on the amount of post-trigger data present, but will never be more than 50%.

12.2.4. Filter

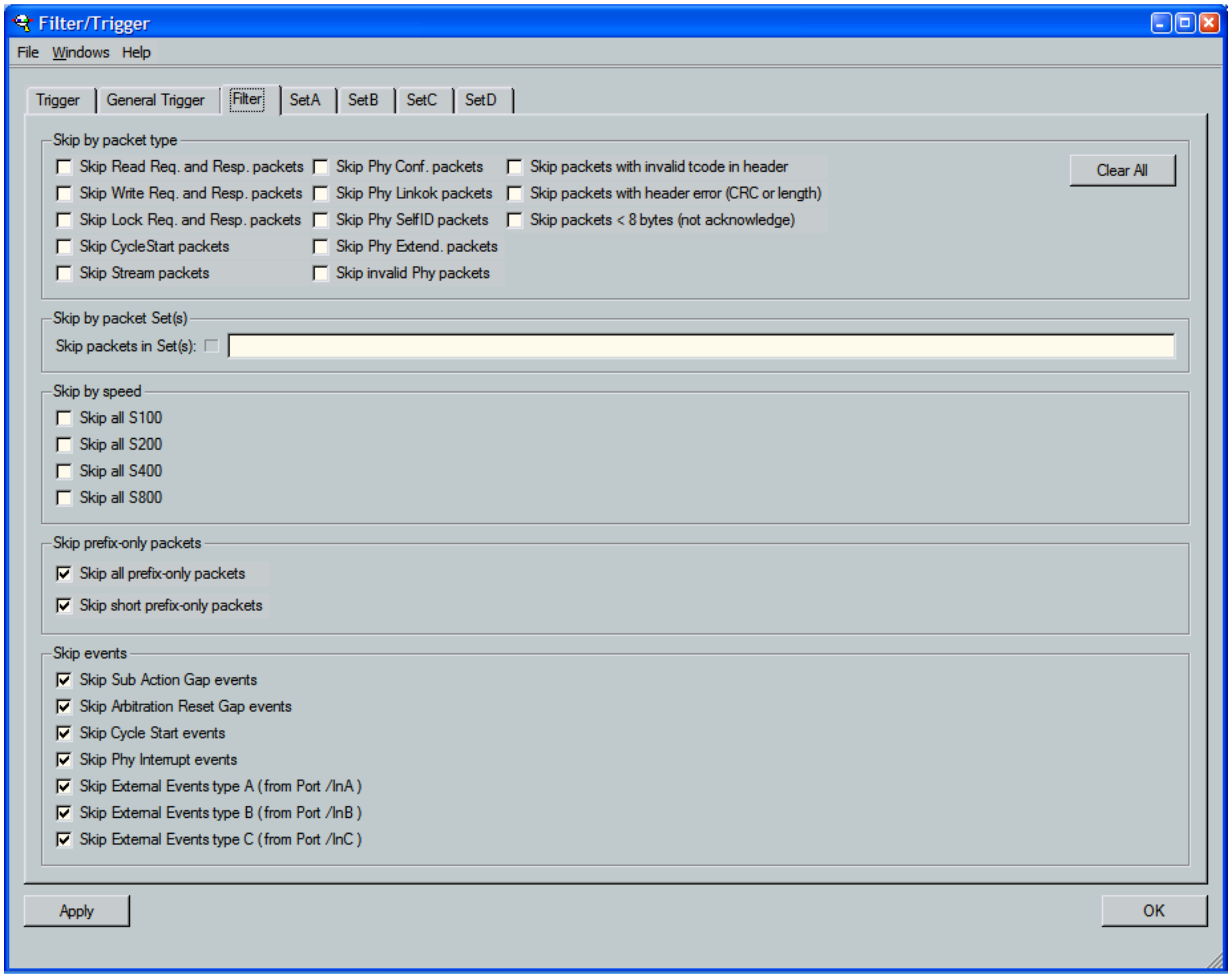
The Filter Tab is not available on
- Stealth Series in Symbol Recorder mode

Actual available options for filtering depend on the specific analyzer model used.

An example of the Filter page is shown below. It has a number of boxes to select the packet and or event that you want to skip. The skipped packets are not stored in the Recorder memory.

It exists of the following boxes:

- 'Skip by packet type' box
- 'Skip by packet Set(s)' box
- 'Skip by speed' box
- 'Skip prefix-only packets' box
- 'Skip events' box



Skip by packet type

- [ ] Skip Read Req. and Resp. packets
- [ ] Skip Write Req. and Resp. packets
- [ ] Skip Lock Req. and Resp. packets
- [ ] Skip CycleStart packets
- [ ] Skip Stream packets
- [ ] Skip Phy Conf. packets
- [ ] Skip Phy Linkok packets
- [ ] Skip Phy SelfID packets
- [ ] Skip Phy Extend. packets
- [ ] Skip invalid Phy packets
- [ ] Skip packets with invalid tcode in header
- [ ] Skip packets with header error (CRC or length)
- [ ] Skip packets < 8 bytes (not acknowledge)

Skip by packet Set(s)

Skip packets in Set(s): [ ]

Skip by speed

- [ ] Skip all S100
- [ ] Skip all S200
- [ ] Skip all S400
- [ ] Skip all S800

Skip prefix-only packets

- [✓] Skip all prefix-only packets
- [✓] Skip short prefix-only packets

Skip events

- [✓] Skip Sub Action Gap events
- [✓] Skip Arbitration Reset Gap events
- [✓] Skip Cycle Start events
- [✓] Skip Phy Interrupt events
- [✓] Skip External Events type A (from Port /InA)
- [✓] Skip External Events type B (from Port /InB)
- [✓] Skip External Events type C (from Port /InC)

Skip by packet type
Using this box, you can select which packet types need to be skipped by the Recorder. You can select some packet types (like Cycle packet or Stream packet), you can select some packet type combinations (like Write Request and Response packets) and you can select packet with some kind of error (like packets with invalid tcode in header).
In the example above, all Cycle Start packets will be skipped.

Skip by packet Set(s)
Here you can type in a set name or any boolean combination of the four sets. All packets that fit the Set or combination of Sets, will be skipped. In the example above, all packets that fit the description of SetA or SetB will be skipped.
The set names that can be used are 'seta', 'setb', 'setc' and 'setc'. The names are not case sensitive, thus 'SetA' is valid too. You also can use just 'a', 'b', 'c' or 'd' (or 'A', 'B' etc.) for the set names. The Boolean operators are 'not', 'and' and 'or'. But they may be substituted by the characters '!', '&' and '|' respectively. Parenthesis may be used to group the Boolean operation.

Some examples of valid expressions are:

```
setc
NOT SETB
(seta or setb) and not setd
!SetA & (SetC | SetB)
a
C or D
```

Note that a red cross will be displayed before the boolean expression when there is an error in the expression.

Skip by speed

Here you can select packet speeds. All packets with a speed of one of the selected speeds will be skipped. Note that when the S100 speed is selected, the Phy SelfID packets will be skipped too. This means that the topology can not be drawn in the Topology View of the Recorder, because the Phy SelfID packets are needed for that.

Skip prefix-only packets
Here you can select to skip all prefix-only packets. And if you do not skip all, you can select to skip only short prefixc-only packets.
Prefix-only packets are packets for which the Analyzer analyzer did not see any data. It only sees a prefix. There are two situations this can happen:

- *speed limitation*
  If for example some node is transmitting a packet at 400Mb/s and this node is connected to the Analyzer with a slower speed node in between, then this 400Mb/s packet can not be repeated by this slower node to the Analyzer. The Analyzer sees a packet but not the data. During the duration of the whole packet a prefix will be detected.
  These are mostly long prefixes, because all data is detected as prefix. To skip these kind of prefix-only packets, you should skip all (short and long) prefix-only packets.
- *1394b status signalling*
  If all nodes are 1394b nodes, it is possible that a status event is embedded in a null-packet. These null packets may by detected by the phy layer and the phy can signal a prefix to the link layer, without any data. The result will be a short prefix-only packet.
  This phy behaviour can not be disabled. Therefore, we added on option to be able to skip short prefix-only packets. Because the status is embedded in nul-packets, those prefixes are usually very short.

Skip events
In this box you can specify to skip some bus-events, the phy-interrupt event and external events. By default these events will all be skipped.
When Arbitration Reset Gaps, Cycle Starts and especially Subaction Gaps are not skipped, be aware that a lot of these events may be stored. This can increase the download time considerably.
Note that when these event are not skipped, they will show up in the Packet View and also in the Time View if you zoom far enough.

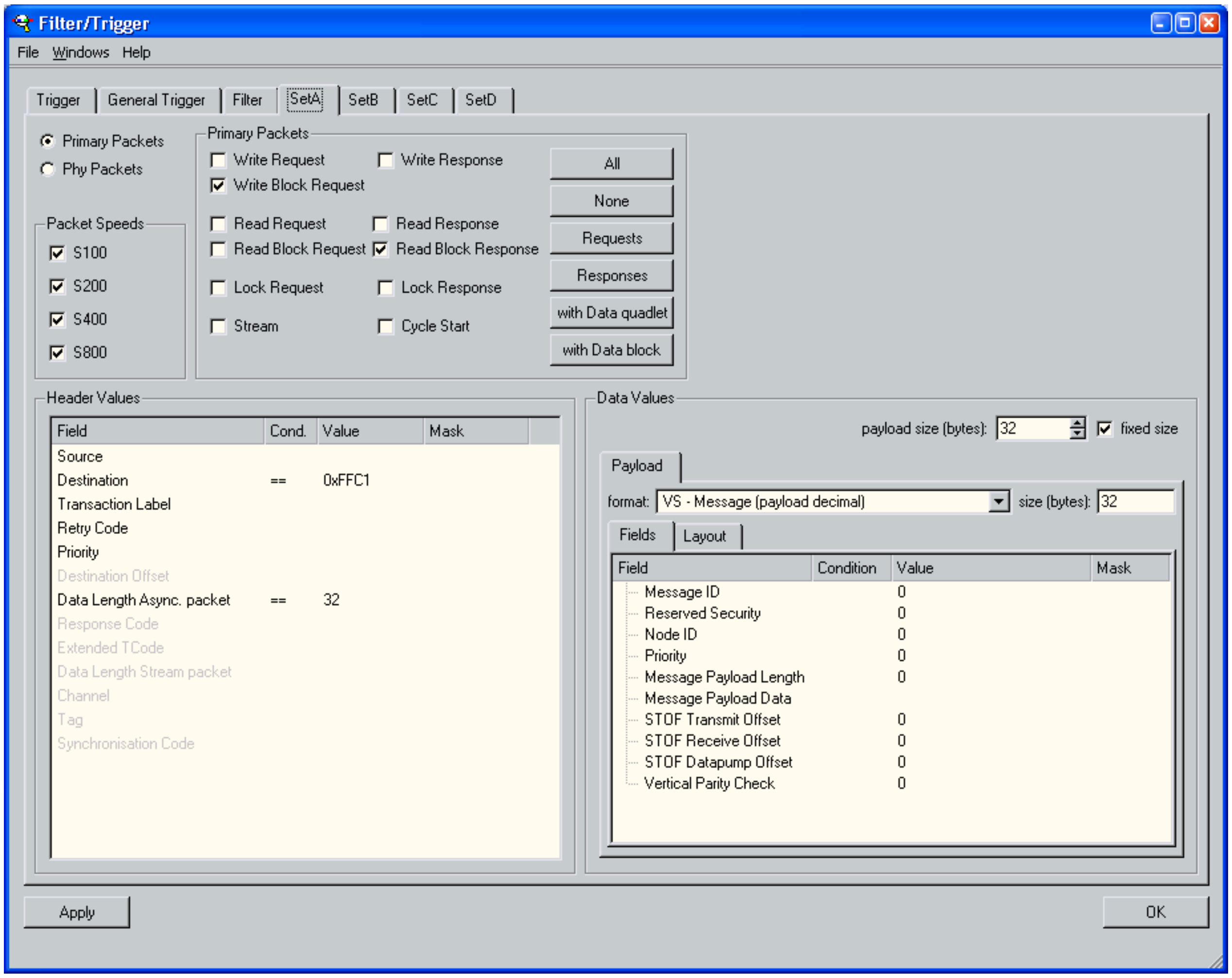
12.2.5. Packet sets

The Packet Sets tab is not available on
- Stealth Series in Symbol Recorder mode

Each of the four basic sets (SetA, SetB, SetC and SetD) can be used for the record filter and or trigger logic as described above. A Set defines a subset of all asynchronous and isochronous packets by specifying packet types, packet speeds and optionally value conditions for one or more header and or data fields.
A packet is said to 'fit' the Set if all specified conditions are met. Thus if the packet is of the specified type, and the packet is of the specified speed, and the optional header and data fields conditions are met, then the packet 'fits' the Set.

A Set page has the following parts:

- 'Primary Packets' or 'Phy Pakets' selection
- 'Primary Packets' box to select primary types
- 'Packet Speeds' box to select which packet speeds are included
- 'Header Values' box to specify optional header field conditions
- 'Data Values' box to specify optional data field conditions



(●) Primary Packets
( ) Phy Packets

Primary Packets

☐ Write Request	☐ Write Response
☑ Write Block Request	
☐ Read Request	☐ Read Response
☐ Read Block Request	☑ Read Block Response
☐ Lock Request	☐ Lock Response
☐ Stream	☐ Cycle Start

Packet Speeds
☑ S100
☑ S200
☑ S400
☑ S800

Header Values

Field	Cond.	Value	Mask
Source			
Destination	==	0xFFC1	
Transaction Label			
Retry Code			
Priority			
Destination Offset			
Data Length Async. packet	==	32	
Response Code			
Extended TCode			
Data Length Stream packet			
Channel			
Tag			
Synchronisation Code			

Data Values
payload size (bytes): 32 ☑ fixed size
format: VS - Message (payload decimal) size (bytes): 32

Field	Condition	Value	Mask
Message ID		0	
Reserved Security		0	
Node ID		0	
Priority		0	
Message Payload Length		0	
Message Payload Data			
STOF Transmit Offset		0	
STOF Receive Offset		0	
STOF Datapump Offset		0	
Vertical Parity Check		0	

Primary Packets or Phy Packets
Here you can select if this set specifies primary packets or a phy packets. Primary packets are Request packets, Response packets, Cycle Start packets or Stream (isochronous or asynchronous) packets.
If you select Phy Packets, then you can specify optional phy packet field value conditions using the Data Values box. The Header Values box will be disabled for the Phy Packets.
If you select Primary Packets, then you can select packet types in the Primary Packets box, you can specify optional header value conditions in the Header Values box and optional data value conditions in the Data Values box.

Packet Speeds
Here you can select which speed a packet should have to fit the Set. If no speed is selected at all, no packet will fit the Set at all. If you select all speeds, the speed of a packet doesn't matter. In the example above, packets of speed 100Mb/s or 200Mb/s do not fit SetA.

Primary Packets
Here you can specify which primary packet types may fit the Set. All types for which the corresponding checkbox is not checked, do not fit the Set. So in the example above, only ReadBlockResponse packets may fit SetA.
If one or more types are selected, then only those lines in the Header Values table will be enabled for which the corresponding field is present in all the selected types. If you select for instance only request types, then the Destination Offset field will be enabled. you can put a condition on this field. But if you also select a Response, this field will be disabled because it is not present in a Response packet.

There are also some buttons to quickly select multiple types at once:

All
Clicking this button will select all types.

None
Clicking this button will deselect all types.

Requests
Clicking this button will select all request types.

Responses
Clicking this button will select all response types.

with Data quadlet
Clicking this button will select all packet types, for which a data quadlet is present in the header.

with Data block
Clicking this button will select all packet types, for which a data block may follow the header.

Header Values
Here you can specify additional header field conditions. As mentioned above, only those fields will be enabled that are present in all packet types selected in the Primary Packets box. For each enabled field, a condition may be entered.
A condition can be set by specifying a condition and a value. In the example above for instance, a condition is set on the Source field.
A packet only does fit the Set if all the conditions specified (and that are not disabled) are true for the packet.
In the example above for instance, a packet does not fit the Set if the Source field is not equal to 0xFFC1.

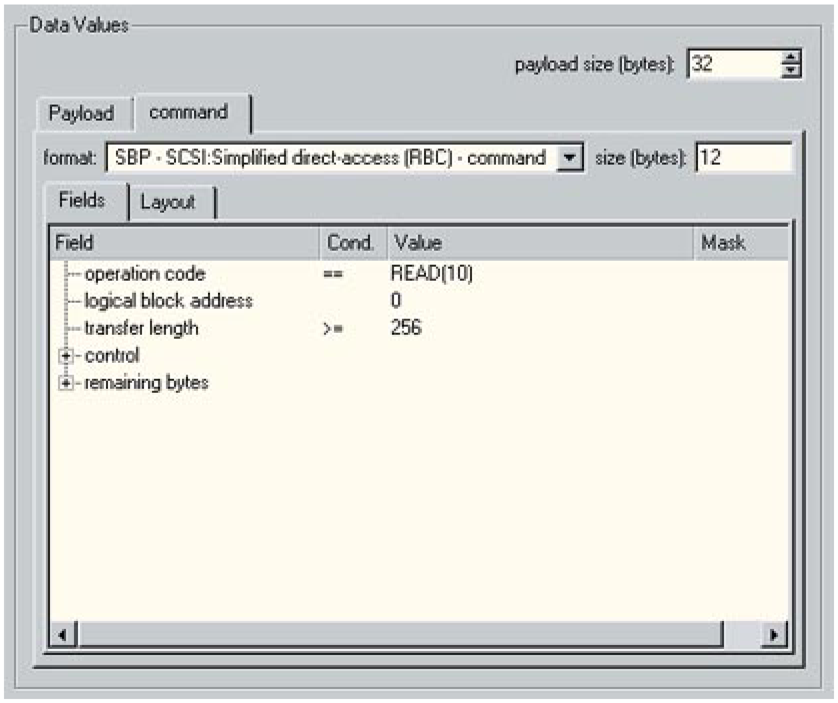
The Header Values table has the following columns:

- *Field:* The Field column displays the header field name. Note that only those fields are enabled that are present in all the selected types in the Primary Packets box. Fields not enabled are displayed with a gray color.
- *Cond:* Here a compare type can be selected. You define a compare type by clicking in this field and select the type from the combo box. You can select the following types:
  - == condition met if the packet field value equals the specified Value.
  - != condition met if the packet field value does not equal the specified Value.
  - <= condition met if the packet field value is less than or equal to the specified Value.
  - >= condition met if the packet field value is greater than or equal to the specified Value.

  You also can select an 'empty' operation meaning that the field condition has to be removed. Selecting this will also remove the field Value.
  Note that for each byte in the packet, only one condition may exist, unless the condition is '=='. You can for instance not put a '<=' condition on the Transaction Label field if there is also a condition other then '==' on the Retry Code field. If both conditions are '==', there is no problem. The software will inform you about possible conflicts as soon as you try to Apply the settings.
- *Value:* Here you can specify the value for the condition. You can enter values in decimal or hexadecimal notation. If you clear this field, the compare type in the Cond. columns will be removed too.
- Together with the compare type from the Cond. column it specifies the condition the packet field should have to fit the Set.
- *Mask:* Before the packet field is compared with the Value, a logical-and is performed with this optionally Mask value. It can be used to mask one or more bits before the compare is done. For each bit in the mask with value 0, the corresponding bit in the packet field and the corresponding bit in the specified Value will cleared before they are compared.
  If in the example above a mask value of 0x003F was specified for the Source field, then effectively only the lower 6 bits are compared because all other bits will be zero-ed by the mask operation.

Data Values
Here you can specify additional data field conditions. Basically it works the same as the header field conditions, but because the format and size of data fields are not fixed, a powerful packet data editor has been added. Here you can select the format for all kind of data payload types.
In the example above for instance, the CommandOrb format (part of the SBP2 support) has been selected as the packet payload format. This gives you some fields on which conditions can be set. In the example above the conditions are such that a packet will fit the set only if the rq_fmt field equals "SBP2 standard", the direction field equals "Read", the speed field equals "S100" and the 'data size' field is

greater than or equal to 1024. Note that some values can be selected by selecting a symbolic value from a combo box, so you do not have to look up these values. If you want to know the values of these symbolic names, you can look inside the 'Layout' tab, or you can use a right-mouse click and select the 'Decimal' values option for instance.
For some formats, there may be a data field for which you can define a separate sub-format in an additional tab. Here you can also add additional conditions for the fields in that sub-format. In the CommandOrb format example above for instance, the 'command' field can be sub-formatted in a additional tab. In the example below this additional 'command' tab has been selected. The 12 bytes command is shown here. Now you can select a separate format for this 12 bytes command. In the example we chose the 'Simplified direct-access' command set, which is part of the SCSI specifications. This format can show all SCSI commands that are part of that set. When you select an other operation code, the format will adapt to the format for that particular command. This way it is very easy to add additional conditions on fields that are specific for a command set and specific to a command from that set. In the example below for instance additional conditions are added so that only a READ(10) command that has also a 'transfer length' field value of 1024 or more does fit the set.



Field	Cond.	Value	Mask
operation code	==	READ(10)	
logical block address		0	
transfer length	>=	256	
control			
remaining bytes			

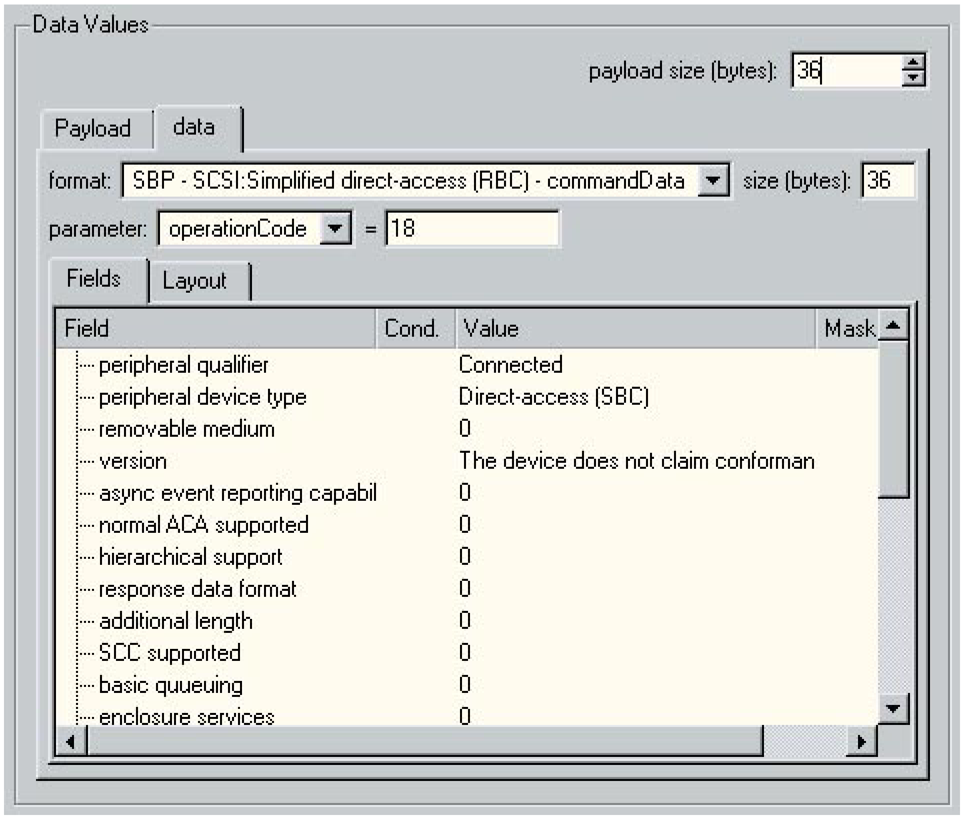
Depending on the license keys that are installed, you can select all kind of formats and optionally sub-formats where needed. There are formats for the SBP protocol, IIDC protocol, IP4 protocol and AV/C protocol. You also can select the format 'unformatted' to be able to set any kind of condition on any bits. Now we will describe the different parts of the Data Values box:

payload size (bytes):
Here you can enter the payload size of the packet, expressed in number of bytes. It will specify the size of the data shown in the 'Payload' tab. Changing the value will also change the size (bytes) value in the 'Payload; tab. This value will automatically adapted if you specify a new value for the 'Data Length Async. packet' field in the Header Values box.

Format tabs
The 'Payload' tab will always be present. It describes the whole payload. If a field can be described with a sub-format, an additional tab will be present for the format of that field. See 'command' tab in example above. All tabs have the same parts in it:

- *format:* With this combo box you can select the format of the packet or sub-format of the field. Note that the formats you can select from depend on the license keys that are installed for the currently connected Analyzer. If no Analyzer is connected, there will only be the unformatted option.
- *size (bytes):* Here the number of bytes will be displayed that correspond to the data in the selected tab. For the 'Payload' tab this will be the size of the complete payload as indicated in the payload size (bytes): above it.
- *parameter:* Sometimes a format may need extra parameters to be able to format the data correctly. In

the example below for instance the selected format is one of the command sets for the command data for the SBP2 protocol. Command data is returned as a reaction on a command. In this case, the command has an operation code that determines the format of the command data. Thus to be able to format a command data packet, you will need to enter an operation code. In such situations a parameter combo box will appear. Here you can select a parameter and enter the value for it in the box right from it.



payload size (bytes): 36

format: SBP - SCSI:Simplified direct-access (RBC) - commandData size (bytes): 36

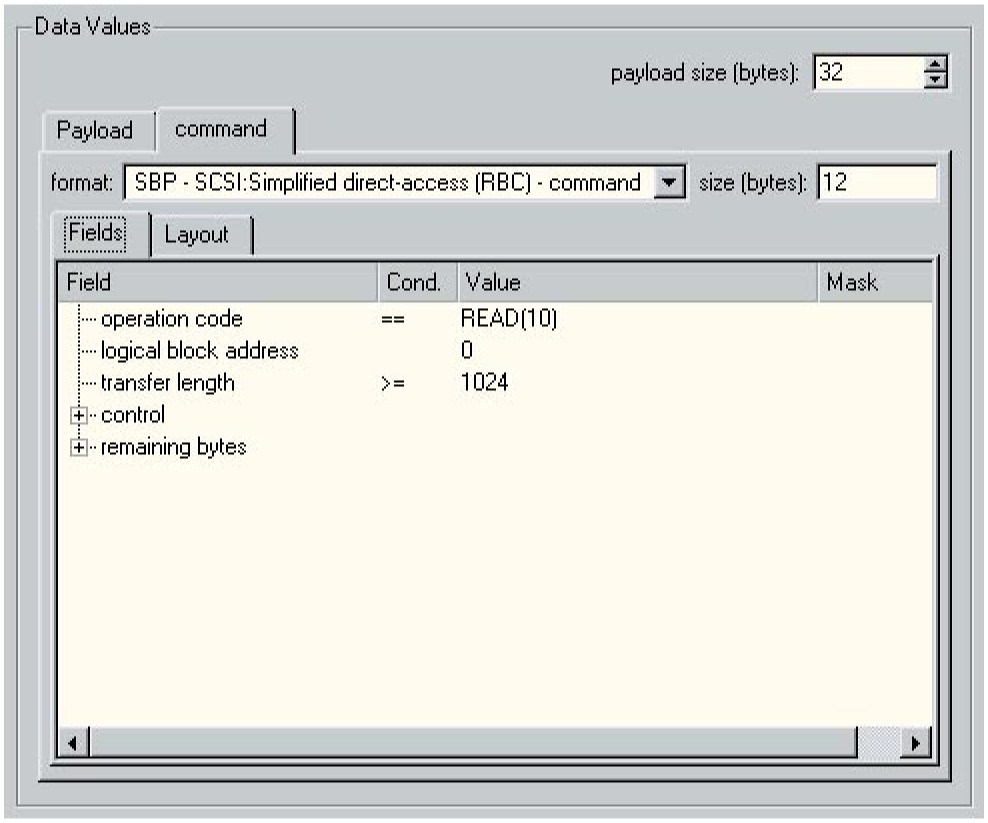
parameter: operationCode = 18

Field	Cond.	Value	Mask
peripheral qualifier		Connected	
peripheral device type		Direct-access (SBC)	
removable medium		0	
version		The device does not claim conforman	
async event reporting capabil		0	
normal ACA supported		0	
hierarchical support		0	
response data format		0	
additional length		0	
SCC supported		0	
basic quueuing		0	
enclosure services		0	

For some parameters, the value can be entered by selecting a symbolic value from a list of symbolic values in a combo box.

- *Fields:* The formatted payload or field can be shown as a table or a layout picture. The Fields tab shows the table. Here you can enter the data field conditions just as you could do with the header fields. The table has the following columns:
  - *Field:* The Field column displays the data field name.
  - *Cond.:* Here a compare type can be selected. You define a compare type by clicking in this field and select the type from the combo box. You can select the following types:
    - == condition met if the packet field value equals the specified Value.
    - != condition met if the packet field value does not equal the specified Value.
    - <= condition met if the packet field value is less than or equal to the specified Value.
    - >= condition met if the packet field value is greater than or equal to the specified Value.

    You also can select an 'empty' operation meaning that the field condition has to be removed. Selecting this will also remove the field Value.
    Note that for each byte in the packet, only one condition may exist, unless the condition is '=='. The software will inform you about possible conflicts as soon as you try to Apply the settings.
  - *Value:* Here you can specify the value for the condition. Note that the values are displayed in a form depending on the field type. For instance the 'transfer length' field in the example below is displayed as a decimal number. The 'logical block address' field is displayed as hexadecimal number and the 'operation code' field is shown using symbolic values.



Field	Cond.	Value	Mask
operation code	==	READ(10)	
logical block address		0	
transfer length	>=	1024	
control			
remaining bytes			

Decimal and hexadecimal numbers can be entered decimal or hexadecimal. For fields with symbolic values, a combo box will be displayed when clicking on the value so that a value can be chosen. You can change this behaviour by selecting an other display form (overruling the automatic form) using the right mouse button as described below.
If you clear this field, the compare type in the Cond. columns will be removed too.
Together with the compare type from the Cond. column it specifies the condition the packet field should have to fit the set.

- *Mask:* Before the packet field is compared with the Value, a logical-and is performed with this Mask value. It can be used to mask one or more bits before the compare is done. For each bit in the mask with value 0, the corresponding bit in the packet field and the corresponding bit in the specified Value will be cleared before they are compared.

Using a right mouse click in the Field table you can popup a menu to select one of the following options:

- *display form:* You can select if the value of the fields should be displayed Hexadecimal, Decimal or Automatic. Automatic (default) means that the format will determine the display form for each field separately. One of the automatic forms can be a symbolic value.
- *Select All:* With this option you can quickly select all fields in the table.
- *Import and Export:* With this option you can import or export the selected field(s) or all fields.
  If you choose to import, a file dialog is presented. You can select to import from a hex data file (*.hex) file or quadlet data file *.qdl file. See File Formats for more information on these formats.
  If you choose to export, you can select the type of export in a little dialog and then with a file dialog you can select a file to export to.
- *Hex/Ascii edit:* With this option you can edit the selected field(s) or all fields using a hex/ascii editor.

- *Layout:* The formatted payload or field can be shown as a table or a layout picture. The Layout tab shows the layout picture. The same fields are presented as in the 'Fields' tab, but now you can see the sizes of the fields and how they are positioned in the data block. Note that reserved fields are shown here too. Reserved fields are not shown in the 'Fields' table.

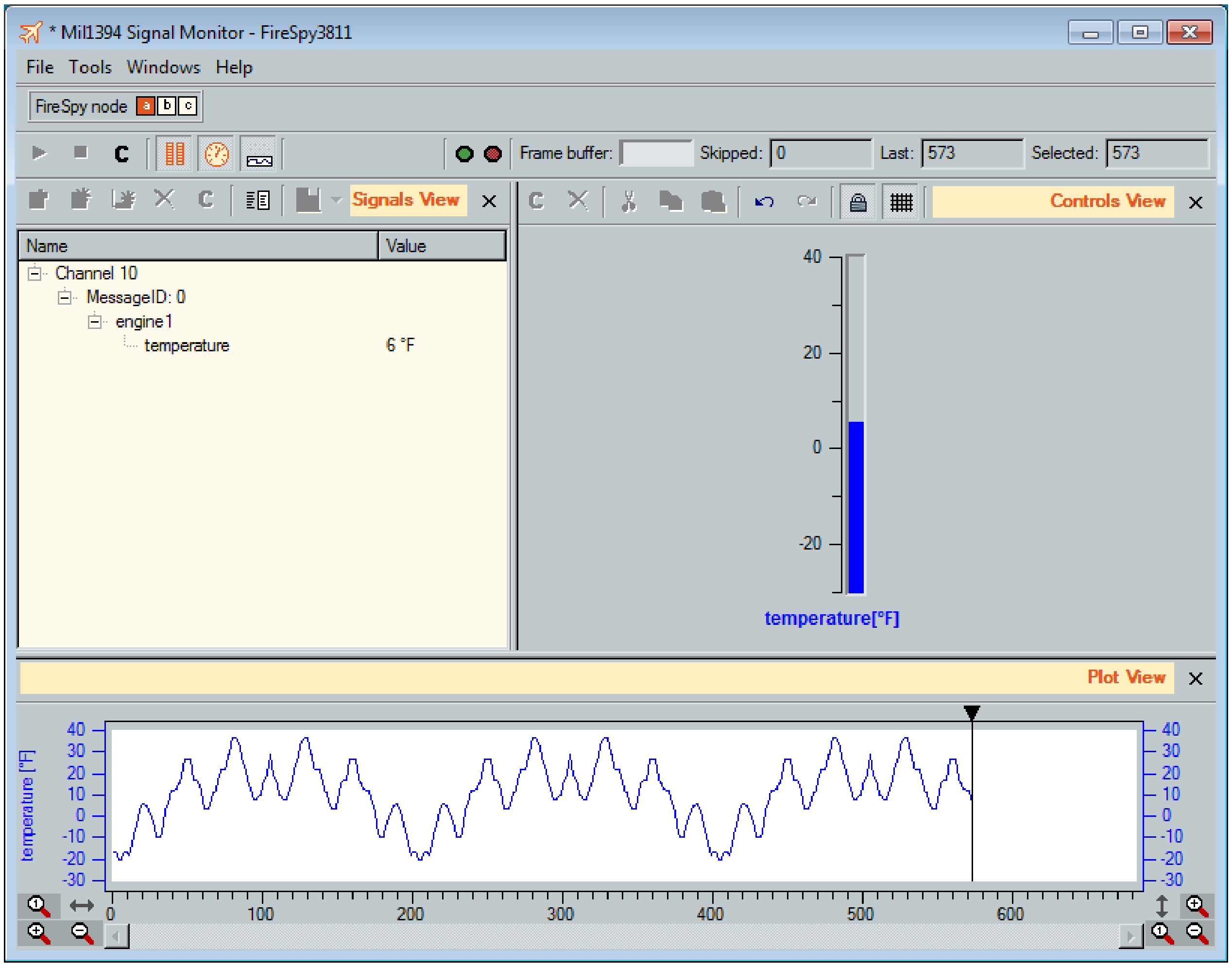
Chapter 13. Mil1394 Protocol support

Mil1394 Protocol Support is a combination of features dedicated to supporting the SAE AS5643 protocol within the FireDiagnostics Suite software. This includes a special version of the Generator that supports timed transmission of ASM Messages based on STOF, packet decoding and encoding according to the AS5643 packet format throughout the suite (Generator, Scriptor, Recorder) and a Mil1394 Signal Monitor application that allows monitoring sensor/actuator data without going into the details of 1394 packet formatting and transmission.

13.1. Mil1394 Signal Monitor

The AS5643 specification defines so called Asynchronous Subscriber Messages (ASM). Remote nodes will listen on (subscribe to) a specific 1394 channel and look at the MessageID field in each packet received on that channel to determine the format of the message data and decode for example sensor values. The Mil1394 Signal Monitor allows one to define the location and format of such values within the 1394 traffic and then monitor their interpreted values over time and leaving out all the 1394 specifics.

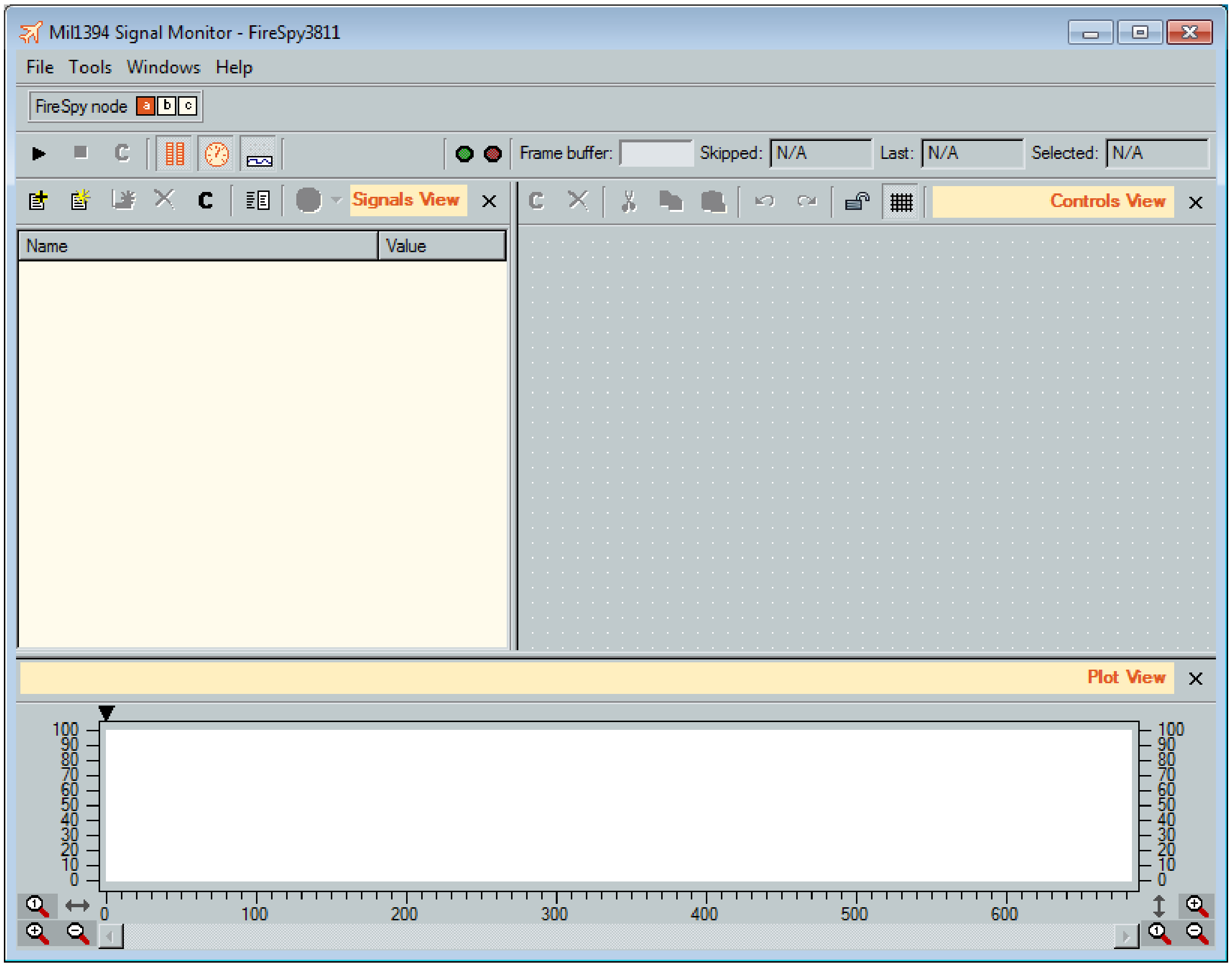
For example the picture below shows a screenshot of a configuration where a temperature sensor is monitored over time. A new value with be added to the plot view as shown on the bottom for each frame a value is received. Then for current values a nice representation can be defined using the built-in Control Panel as shown on the right. The tree representation on the left provides an overview of signals selected/ defined for monitoring and their current values.



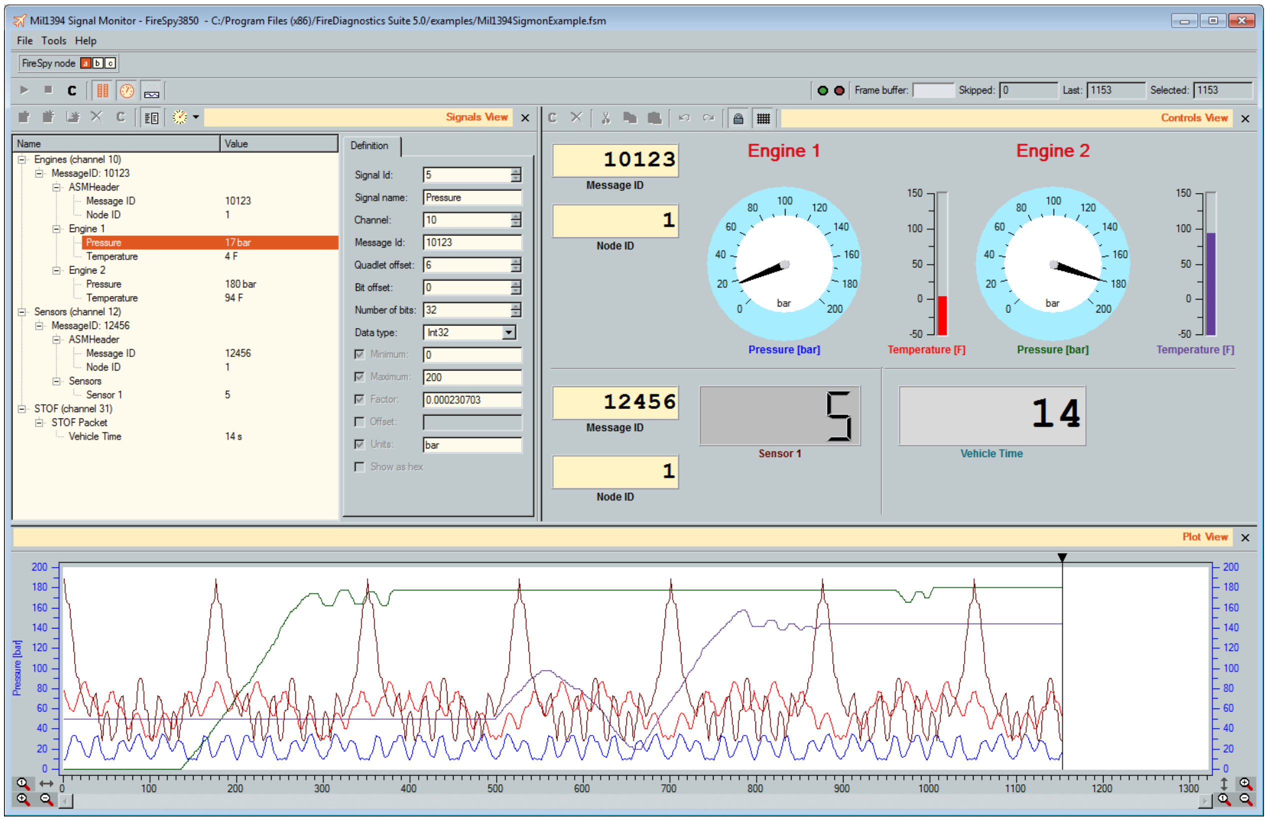
Note: the Mil1394 Signal Monitor requires a Mil1394 protocol license. Refer to the License Manager to learn how to view your license and how to get one.

13.1.1. How to use it

The Mil1394 Signal Monitor can be started through the Windows Start menu as a standalone application or from the Main Window of the FireSpy application. After starting the Mil1394 Signal Monitor, an empty view will be shown as can be seen in the picture below.



The easiest way to get started is to load the example file "Mil1394SigmonExample" that comes with the application. It already contains a set of signal definitions with some captured values also. After opening the file, the Mil1394 Signal Monitor will look like the following picture.


Mil1394 Signal Monitor - FireSpy3850 - C:/Program Files (x86)/FireDiagnostics Suite 5.0/examples/Mil1394SigmonExample.fsm
File Tools Help
FireSpy node a b c
Frame buffer: Skipped: 0 Last: 1153 Selected: 1153
Signals View
Controls View
Plot View
Engine 1
Engine 2
Message ID
Node ID
Pressure [bar]
Temperature [F]
Sensor 1
Vehicle Time
Pressure [bar]

```
Name                          Value
⊟ Engines (channel 10)
  ⊟ MessageID: 10123
    ⊟ ASMHeader
        Message ID            10123
        Node ID               1
    ⊟ Engine 1
        Pressure              17 bar
        Temperature           4 F
    ⊟ Engine 2
        Pressure              180 bar
        Temperature           94 F
⊟ Sensors (channel 12)
  ⊟ MessageID: 12456
    ⊟ ASMHeader
        Message ID            12456
        Node ID               1
    ⊟ Sensors
        Sensor 1              5
⊟ STOF (channel 31)
  ⊟ STOF Packet
      Vehicle Time            14 s
```

```
Definition
Signal Id:        5
Signal name:      Pressure
Channel:          10
Message Id:       10123
Quadlet offset:   6
Bit offset:       0
Number of bits:   32
Data type:        Int32
[✓] Minimum:      0
[✓] Maximum:      200
[✓] Factor:       0.000230703
[ ] Offset:
[✓] Units:        bar
[ ] Show as hex
```

Using the cursor line in the plot view at the bottom you can navigate through all the frames and show the signal values captured in a specific frame number.

The next step would be to actually capture some data. This could easily be done in either of two ways:

- Connect another FireSpy and run the example Script called "Mil1394ScriptExample.fss" it will transmit some values from file and some other values can be controlled on its control panel
- On a triple FireSpy, connect nodes A and B. Open the same script as in the above step but change the FIRESPY_NODE macro at the top of the script from 0 to 1 and then start the script.

In the Mil1394 Signal monitor press the "Clear" button in the main toolbar of the Mil1394 Signal Monitor to clear all captured data and prepare for the next run and then press start.

13.1.2. Details

13.1.2.1. Menu

File

New
With this command you can clear the current contents and start working on a new and empty document.

Open
With this command you can open an existing Mil1394 Signal Monitor file. By default these files have an extension of .fsm.

Save
With this command you can store the current contents of the Mil1394 Signal Monitor to file. This includes both the signal definitions as any recorded data.

Save as...
Same as the above, however, always asks for a file name even if the data was already stored to file before.

Import

- *Control Panel*: Allows loading controls from a text file, replacing the current Control Panel contents.

These files could be created with the Control Panel of the Mil1394 Signal Monitor itself or with another tool with Control Panel like the Scriptor.
- *Signal Definitions*: Allows loading a set of signals from a .csv file, overwriting the current signal set. This function was added to support loading signal definition files as used in previous versions of the application.

Export
- *Signal log file:* Allows writing the recorded signal values to a text file in almost the same format as was used by older versions of the signal monitor. Due to changes in the way the STOF packet is monitored, the file format no longer has a dedicated section for STOF packet contents. Instead, STOF packet contents can be monitored in the same way as other values.
- *Control Panel:* Allows exporting the current Control Panel to a text file. The text file can then be loaded into any other tool that contains a Control Panel like the Scriptor.

Tools

Settings
This command will bring up the Settings Dialog.

Windows

Monitor
This command will bring up the Monitor window.

Recorder
This command will bring up the Recorder window.

Commander
This command will bring up the Commander window.

Generator
This command will bring up the Generator window.

Scriptor
This command will bringup the Scriptor window.

Main
This command will bring up the main window.

Help

Manual
This command will bring up the help dialog and navigates to the first page.

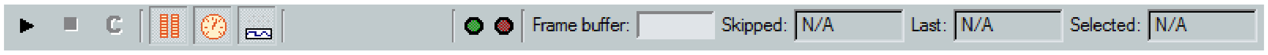
Mil1394 Signal Monitor
This command will bring up the help dialog and navigates to the first page of the Mil1394 Signal Monitor section.

Analyzer Info
This command will bring up a dialog that shows information about the analyzer currently in use.

About
This command will bring up the About... dialog.

13.1.2.2. Toolbar

The Mil1394 Signal Monitor toolbar is shown below and contains respectively the following items:



Controls

Start
This button can be used to start monitoring the currently defined signals. The button will be disabled (grayed out) whenever the Mil1394 Signal Monitor is already active.

Stop
This button can be used to stop monitoring currently defined signals. The button will be disabled (grayed out) when the Mil1394 Signal Monitor is not active. After stopping the Mil1394 Signal Monitor the captured data will remain loaded.

Clear
This button can be used to clear the captured signal values (data). This button will not clear the Signal Definitions. This button will be disabled (grayed out) when the Mil1394 Signal Monitor is active.

Views

Signals View
This is a toggle button that can be used to show or hide the Signals View. The Signals View is shown when the button is in lowered state.

Controls View
This is a toggle button that can be used to show or hide the Controls View. The Controls View is shown when the button is in lowered state.

Plot View
This is a toggle button that can be used to show or hide the Plot View. The Plot View is shown when the button is in lowered state.

Indicators

Active
This indicator will be on when the Mil1394 Signal Monitor is active and monitors signals.

Frames skipped
This indicator will be on when the Mil1394 Signal Monitor detected one or more skipped frames during the currenly loaded capture.

Frame buffer
The Mil1394 Signal Monitor contains a memory buffer in FireSpy internal memory. The host fetches frames of data from the FireSpy as quickly as possible. This indicator shows the fill level of the buffer in FireSpy memory. When this indicator shows a value close to 100% then the host has problems keeping up with fetching the data from FireSpy. You can then either reduce the number of signals being monitored or try on a faster pc.

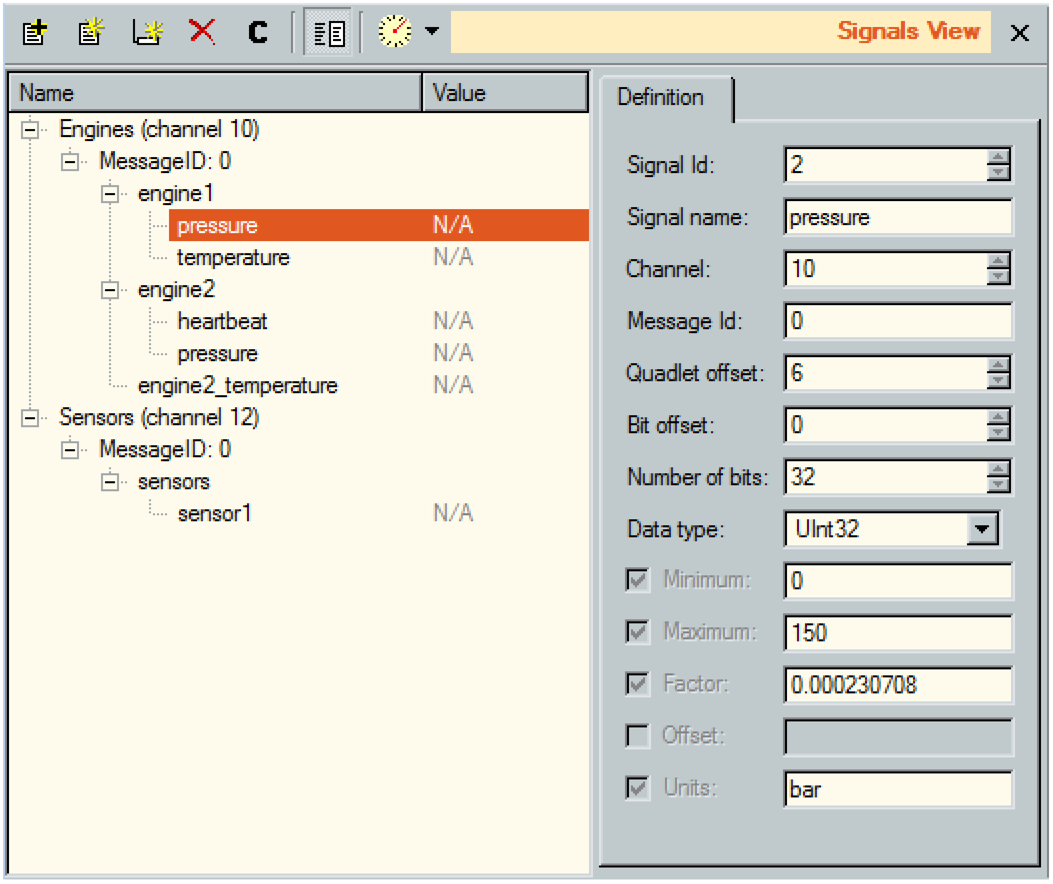
Skipped
This counter indicates how many frames were skipped during the current capture.

Last
This counter indicates how many frames were captured.

Selected
This counter shows the frame number currently selected. The plot view will show the cursor at this frame number on the X-axis and the Signals View and Controls Views will show the signal values captured in the selected frame number.

13.1.2.3. Signals View

The Signals View can be used to select signals to be monitored from a set of predefined signals or define signals to be monitored manually. It also shows current signal values when the Mil1394 Signal Monitor is active. Below a picture of the Signals View is shown. It contains the following components.



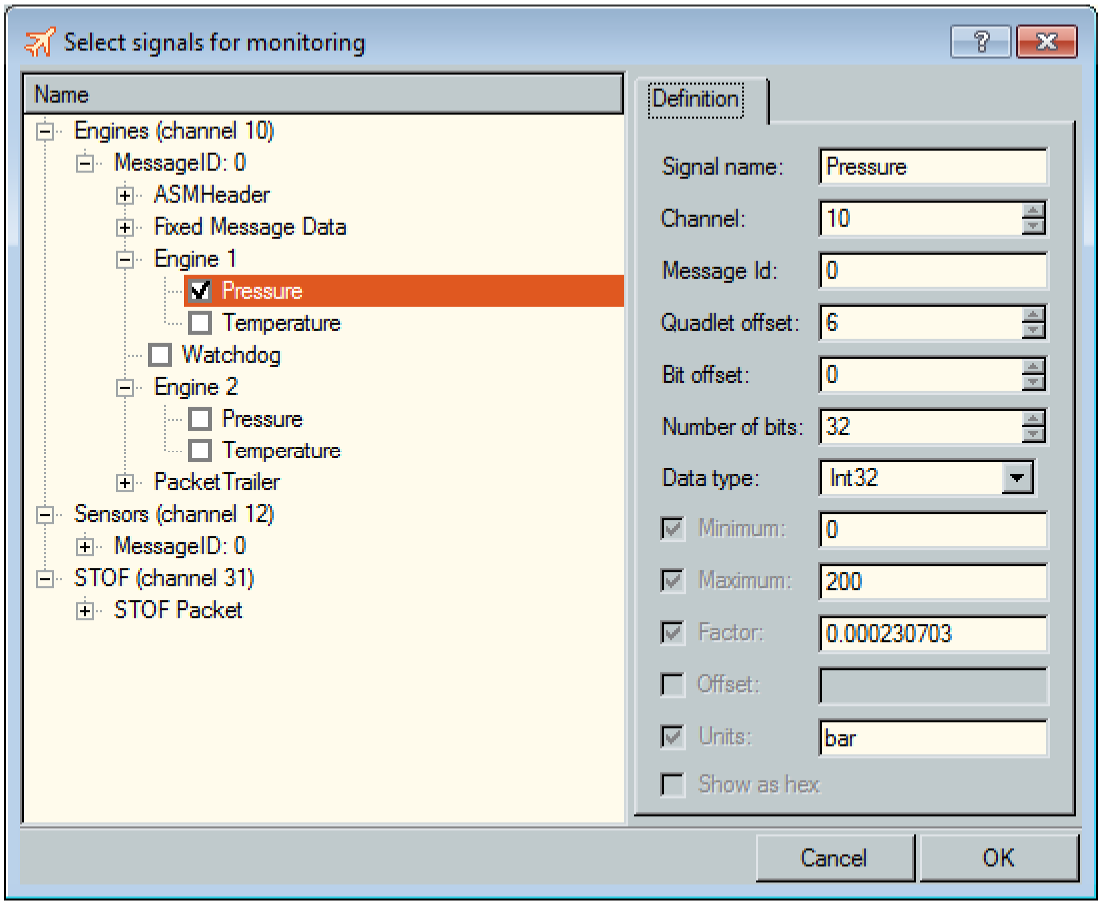
Engines (channel 10)
MessageID: 0
engine1
pressure N/A
temperature N/A
engine2
heartbeat N/A
pressure N/A
engine2_temperature N/A
Sensors (channel 12)
MessageID: 0
sensors
sensor1 N/A

Signal Id:	2
Signal name:	pressure
Channel:	10
Message Id:	0
Quadlet offset:	6
Bit offset:	0
Number of bits:	32
Data type:	UInt32
✓ Minimum:	0
✓ Maximum:	150
✓ Factor:	0.000230708
☐ Offset:	
✓ Units:	bar

Toolbar

The top of the view contains a toolbar which is labeled "Signals View". It contains respectively the following items:

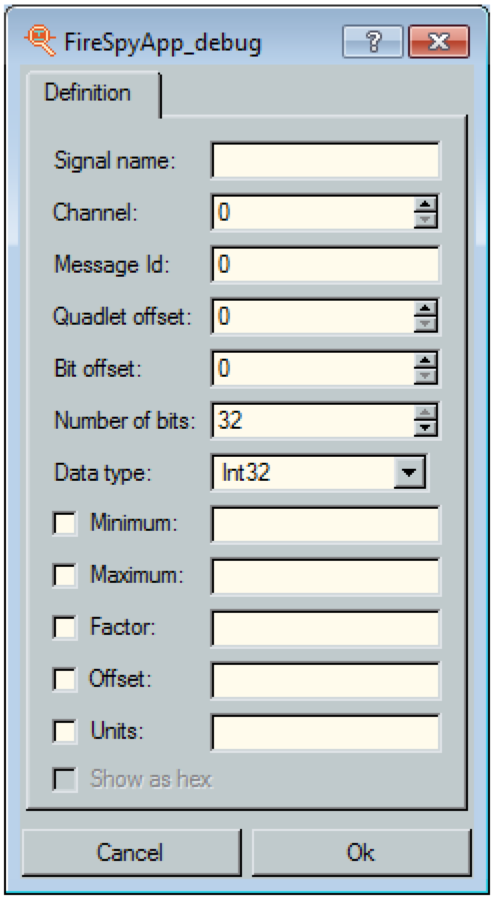
Add predefined
This button can be used to select signals from a predefined set of signals. The pre-defined set either originates from an XML file that defines several AS5643 slash sheet defined configuration items or it originates from a .csv file with signal definitions. Please refer to the Mil1394 Settings documentation for how to choose and define these files. When the Mil1394 XML-based Settings are enabled and configured this makes it really easy to select what to monitor. When the button is pressed, the following dialog will be shown:



Signal name: Pressure
Channel: 10
Message Id: 0
Quadlet offset: 6
Bit offset: 0
Number of bits: 32
Data type: Int32
Minimum: 0
Maximum: 200
Factor: 0.000230703
Offset:
Units: bar
Show as hex

The dialog contains a similar representation of signals as the Signals View itself, however, no values column and the details pane can not be hidden. You can now simply check the checkbox in front of signals you are interested in and after pressing "OK" these will be added to the active set. Pressing "Cancel" will close the dialog and keep the active set as it was before the dialog opened. A detailed description of all the fields in the "Definition" part will be given further down below. Whenever a signal is checked, its parent items will also be added to the current group in order to maintain the same grouping as in the selection dialog. Whenever an item is added to the active set, it will no longer be shown the next time the dialog is opened. This button will be disabled (grayed out) when the Mil1394 Signal Monitor is active.

Manually add
This button can be used to manuall add a new signal to the active set. After pressing the button, the following dialog will be shown.



Signal name:	
Channel:	0
Message Id:	0
Quadlet offset:	0
Bit offset:	0
Number of bits:	32
Data type:	Int32
☐ Minimum:	
☐ Maximum:	
☐ Factor:	
☐ Offset:	
☐ Units:	
☐ Show as hex	

You can now manually define all signal details and after pressing "OK" the signal will be added to the active set. Pressing "Cancel" will ignore the contents of the dialog and the active set will remain the same as just before opening this dialog. When a signal definition is equal to an already monitored signal, it will not be added. A detailed description of all the fields in the "Definition" section of the dialog will be given further below. This button will be disabled (grayed out) when the Mil1394 Signal Monitor is active.

Manually add child
This button behaves the same as the button above except that the signal will be added as child of the currently selected signal. When no signal is selected, this button will be disabled (grayed out). The "Channel" and "Message Id" will be fixed in this case as they need to be the same as the parent item. This button will be disabled (grayed out) when the Mil1394 Signal Monitor is active.

Delete
When this button is pressed, the currently selected signal will be removed from the active set. This button will be disabled (grayed out) when the Mil1394 Signal Monitor is active.

Clear
When this button is pressed, all signals will be removed from the active set. This button will be disabled (grayed out) when the Mil1394 Signal Monitor is active.

Show details
This button can be used to show/hide the "Definition" panel on the right side of the view. When the button state is lowered, the panel is shown.

Add Control
This button can be used to add a Control to the Control Panel. When pressing the little arrow on the right side of this button, a pop-up menu is shown that allows you to choose which control to add. The selected control type will be added to the control panel at the first free location it finds. The Control will automatically be matched to the Signal Id of the currently selected signal and all relevant properties like units, factor, e.o. are automatically sent to the control.

Close

This button can be used to hide the Signals View.

Signals Table

The signals table is shown on the left side (or using the complete width if the "Details" panel is hidden) and displays the active set of signals. Signals are grouped by 1394 channel number and Message ID. Optionally, signals can also be grouped into signal groups. Within a group, signals are sorted in the order they were added to the active set.

The first column contains the Name of the signal and the second column displays the current value. Current values are calculated from the raw captured bits using the signal definition fields like data type, factor, offset e.o. Values can be shown in the following colors:

- Black: The signal occurred exactly once in the selected frame
- Red: The signal occurred more than once in the selected frame. The last value received is shown.
- Gray: The signal did not occur in the selected frame. The last value received before the selected frame is shown.

As the tree view contains different kinds of items, only the items that are actual signals will have text in the values column. Signal items that have their number of bits field set to zero will be considered group items and will show nothing in the Values column.

The currently selected signal is indicated by the selection bar. In the picture above the bar is placed at the signal labeled Pressure. When you select a signal through the signals view, the Control(s) on the Control Panel that are linked to the selected signal will also be selected.

Details Panel

The "Details" panel shows all fields that together form the definition of a signal. The following fields are shown:

Signal Id
This is a number that is automatically assigned by the Mil1394 Signal Monitor when the signal is added to the active set. These numbers are unique per signal and are used to tie Controls on the Controls View to signals.

Signal name
User-defined name for the signal.

Channel
The 1394 channel number this signal is transmitted on.

Message Id
The Mil1394 Message ID this signal is defined in.

Quadlet offset
The offset in quadlets relative to the 1394 data section of the packet. Offset zero points to the Message ID field.

Bit offset
The offset in bits relative to the the quadlet offset defined above.

Number of bits
The number of bits occupied by the signal.

Data type
The data type to store the signal in after extracting the bits. When a signed type is chose, the value is sign-extended. Can be one of the following types:
- Bool: Zero means false, non-zero means true
- Uint8: 8 bits unsigned integer
- Uint16: 16 bits unsigned integer
- Uint32: 32 bits unsigned integer

- Int8: 8 bits signed integer
- Int16: 16 bits signed integer
- Int32: 32 bits signed integer
- Float32: 32 bits floating point value
- Enumeration: associate texts with numeric values. When this data type is chosen, a button will be shown to open the value-text definitions dialog.

Minimum
When the check box is checked, this defines the minimum allowed interpreted value.

Maximum
When the check box is checked, this defines the maximum allowed interpreted value.

Factor
When the check box is checked the extracted value will be multiplied by this value and the Offset field will then be added. The result is the interpreted value.

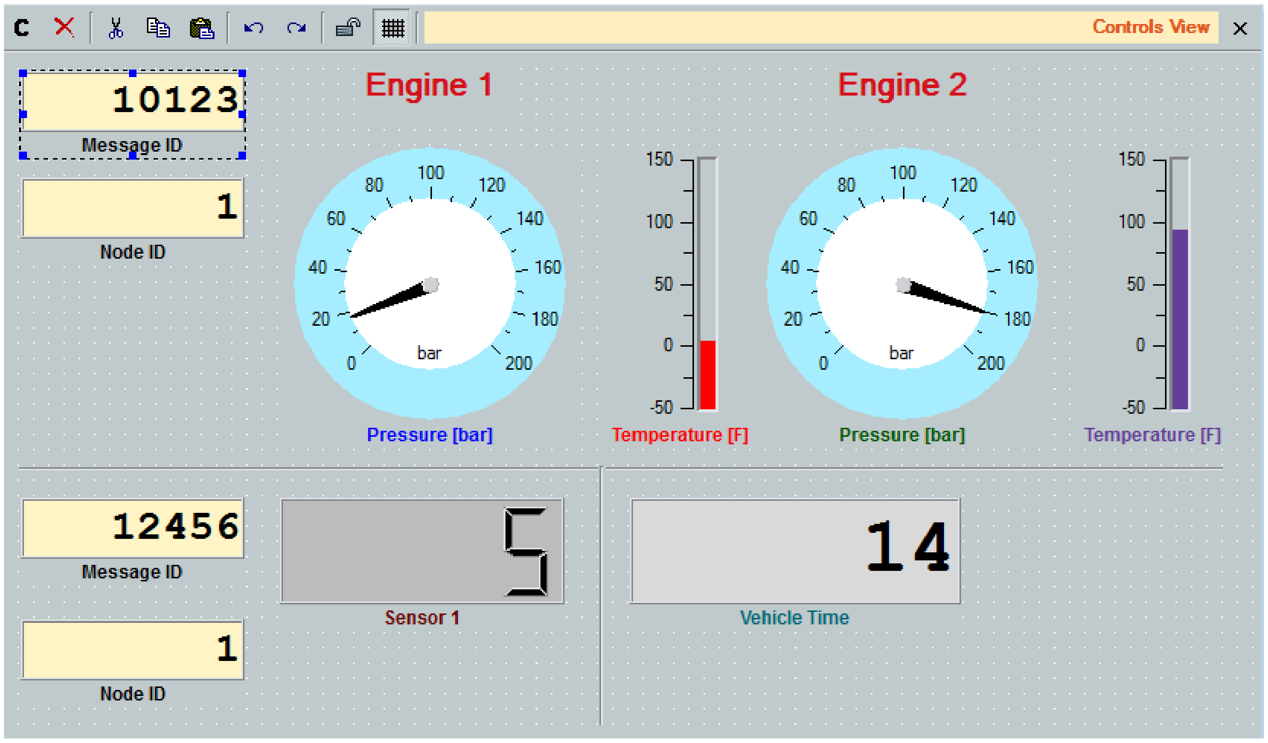
Offset
When the check box is checked the extracted value will be multiplied by the Factor defined above and then this value will be added to form the interpreted value.

Units
When the check box is checked this field shows the units of measure for this signal. This will shown behind the interpreted value.

Show as Hex
When an unsigned integer data type is chosen the value can be shown in hexadecimal notation by checking this box.

13.1.2.4. Controls View

The Controls View offers a nice customizable graphical representation for monitoring signal values. The Control Panel as it exists within the FireSpy Scriptor was used for this view in a slightly altered form. For detailed documentation for the Control Panel, please refer to the Control Panel section within the Scriptor chapter. This section is more focused on how to use a Control Panel to monitor signal values. Readers already familiar with the FireSpy Scriptor will recognize the Control Panel and should be able to quickly learn how to use it in the Mil1394 Signal Monitor. The picture below shows the Controls View and its contents will be explained below.



Toolbar

Clear
This button can be used to clear the complete contents of the current Control Panel. This button will be disabled (grayed out) when the Control Panel is in Locked state.

Delete
This button can be used to delete all controls that are currently selected.This button will be disabled (grayed out) when the Control Panel is in Locked state or when no controls are currently selected.

Cut
This button can be used to cut all controls that are currently selected.This button will be disabled (grayed out) when the Control Panel is in Locked state or when no controls are currently selected.

Copy
This button can be used to copy all controls that are currently selected.This button will be disabled (grayed out) when the Control Panel is in Locked state or when no controls are currently selected.

Paste
This button can be used to paste current contents of the clipboard when the clipboard contains control panel data.This button will be disabled (grayed out) when the Control Panel is in Locked state or when the clipboard does not contain any data.

Undo
This button can be used to undo the previous action. This button will be disabled (grayed out) when the Control Panel is in Locked state or when there is no previous action that can be undone.

Redo
This button can be used to redo the last action that was undone using the above button. This button will be disabled (grayed out) when the Control Panel is in Locked state or when there is no previous undone action that can be redone.

Lock
This button can be used to enable/disable the Locked state of the Control Panel. When the Mil1394 Signal Monitor is started, the Control Panel will always be set to Locked state. Control Panel contents can only be edited in unlocked state.

Grid
This button can be used to show or hide the grid.

Close
This button can be used to close the view.

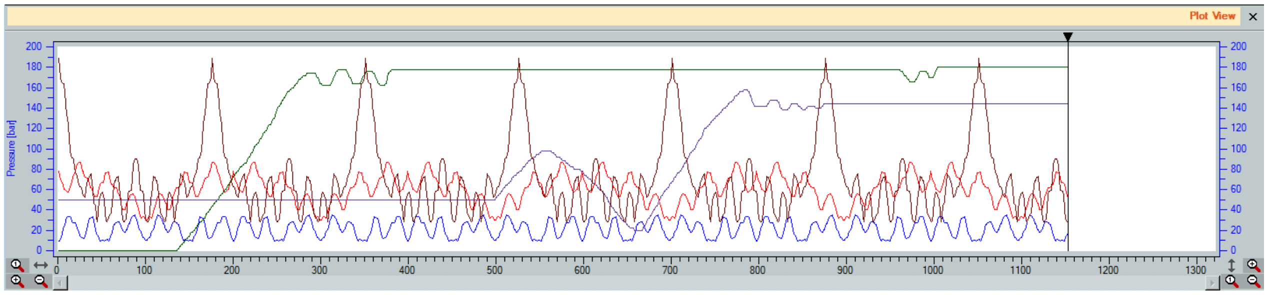
Control Panel

The remaining area below the Toolbar is taken up by the Control Panel. In this area the user can place several kinds of controls as documented in the Control Panel section. Each Control has an ID which does not have to be unique. When the ID matches one of the Signal IDs in the currently active signal set, its values will be monitored.

When right-clicking on the Control Panel, a popup menu will be shown. It is a reduced version of the popup-menu of the Scriptor Control panel. An important aspect is that whenever you add a control to the control panel while a signal is currently selected in the Signals View, the new Control will automatically be tied to that signal by setting the same ID. In addition, important signal properties like factor, offset, minimum and maximum will also be copied from the signal definition into the properties of the new control. This makes it very easy to add controls to the Control Panel to monitor certain signals.

13.1.2.5. Plot View

The plot view is used to display the history of signal values over time. Just like the Control Panel, the Plot View is documented in detail in the Scriptor documentation here. This section is focused on specific functionality for the Mil1394 Signal Monitor. The following picture shows a screenshot of the Plot View and its contents will be explained in detail below.



Toolbar

Close
The close button can be used to hide this view.

Plot View

Contrary to the Scriptor Control Panel, a history is kept for all signals being monitored, even if they are not shown on the Control Panel. Even after a signal capture has been made, Controls can still be added and previously captured history data will then be shown in the plot view. The Control label color determines the color of the data series in the plot view.

To add a Control to the Control Panel without showing it on the history plot, uncheck the "show histort" property in the properties dialog for that control.

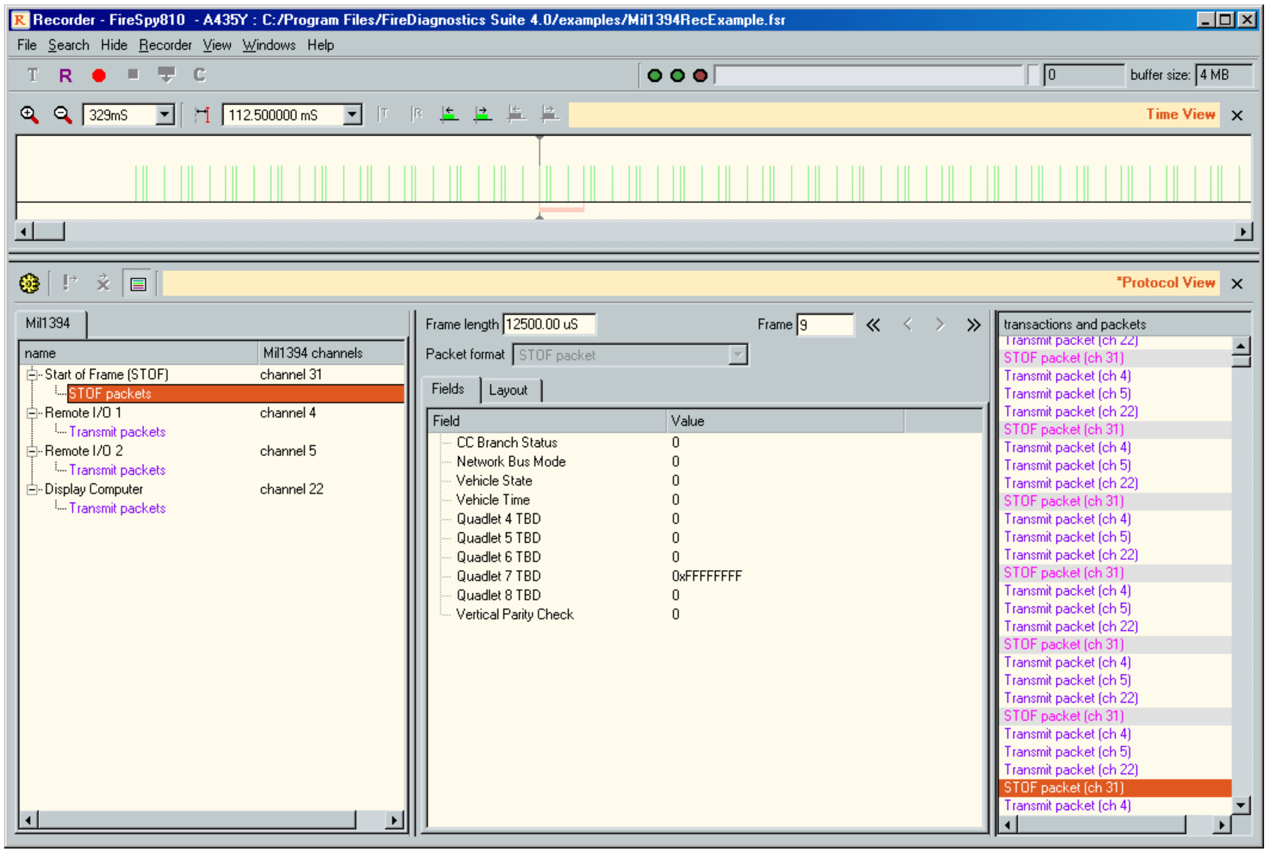
The X-axis contains the frame number and the Y-axis scale is taken from the Control that was last selected.

13.1.2.6. Settings

Settings specific to the Mil1394 Signal Monitor can be found here. However, at least as important are the general Mil1394 settings as they determine how all the predefined signals are put together.

13.2. Recorder Protocol View

This chapter describes the Mil1394 Protocol of the Protocol View. When a Recorder file is opened or when a new recording is downloaded from the Analyzer, or when the Protocol View is opened, the Mil1394 Protocol analyzer scans through the recorded data in the Recorder and displays the found result in the Protocol View. An example is shown below.


Recorder - FireSpy810 - A435Y : C:/Program Files/FireDiagnostics Suite 4.0/examples/Mil1394RecExample.fsr
File Search Hide Recorder View Windows Help
buffer size: 4 MB
329mS
112.500000 mS
Time View
*Protocol View
Mil1394
Fields
Layout

Frame length 12500.00 uS
Frame 9
Packet format STOF packet

name	Mil1394 channels
Start of Frame (STOF)	channel 31
STOF packets	
Remote I/O 1	channel 4
Transmit packets	
Remote I/O 2	channel 5
Transmit packets	
Display Computer	channel 22
Transmit packets	

Field	Value
CC Branch Status	0
Network Bus Mode	0
Vehicle State	0
Vehicle Time	0
Quadlet 4 TBD	0
Quadlet 5 TBD	0
Quadlet 6 TBD	0
Quadlet 7 TBD	0xFFFFFFFF
Quadlet 8 TBD	0
Vertical Parity Check	0

transactions and packets
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)
Transmit packet (ch 5)
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)
Transmit packet (ch 5)
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)
Transmit packet (ch 5)
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)
Transmit packet (ch 5)
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)
Transmit packet (ch 5)
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)
Transmit packet (ch 5)
Transmit packet (ch 22)
STOF packet (ch 31)
Transmit packet (ch 4)

13.2.1. What is it

The result of the Mil1394 Protocol analyzer is displayed in 3 different panes. In the right pane (transactions / packets) all found Mil1394 packets of the involved nodes and bus resets are displayed. The 'transactions / packets' pane lists the Mil1394 packets and bus resets in a chronological order. The left pane (relation tree) shows the result of the Mil1394 Protocol analyzer in a hierarchical tree. The hierarchy of the tree shows the nodes as root items and the found Mil1394 packets of the nodes as child items. An exception is the first root item in the tree. The child item of the first root item represents the found STOF packets. The tree shows a node by its name and assigned channel number. It shows the found Mil1394 packets of a node by their type. The Mil1394 Protocol defines the following 5 type of Mil1394 packets: STOF packet, Transmit packet, Receive packet, DataPump packet and Unknown packet. A Mil1394 packet is an Unknown packet if it can't be identified as one of the other 4 Mil1394 packet types.

If you click an item in the tree, the item will highlight. If one or more Mil1394 packets in the 'transactions / packets' pane correspond to the highlighted item, then the first one of these Mil1394 packets will highlight. The remaining Mil1394 packets will be indicated with a gray background.

Conversely, if you click a Mil1394 packet in the 'transactions / packets' pane, the Mil1394 packet will highlight. The item in the tree of the 'relation tree' pane to which the highlighted Mil1394 packet corresponds will also highlight. The remaining Mil1394 packets, which also correspond to the highlighted item, will be indicated with a gray background color.

The details of the highlighted Mil1394 packet in the 'transactions / packets' pane are displayed in the middle pane (detail) of the Protocol View. The details of a highlighted Mil1394 packet are its fields and their values. The 'detail' pane displays the fields and their values in two different forms. Each form has a tab page. The 'Field' tab page shows the fields and their values in a tree and the 'Layout' tab page shows the fields and their values in the layout of the highlighted Mil1394 packet.

13.2.2. How to use it

To show the Protocol View click menu item 'View | Protocol View' of the Recorder. The Protocol View will be displayed in the lower part of the Recorder.

IMPORTANT:
To enable the Mil1394 Protocol analyzer a valid license key has to be installed. An exception is made for Recorder files, which were made with an Analyzer, with a Mil1394 license key installed.

The Mil1394 Protocol analyzer is able to analyze recorded data in the Recorder if it has knowledge of the nodes on the bus, which support the Mil1394 Protocol. The Mil1394 Protocol analyzer has to know the following information of a node, which support the Mil1394 Protocol: its name, the assigned channel number, the heartbeat and the time slots. The time slots are necessary to identify the Transmit packets of the node, Receive packets of the node and DataPump packets of the node.
This information is to be provided manually in the 'Protocol Analyzer Settings' dialog or via an import file. You can load an import file via the 'Protocol Analyzer Settings' dialog. See the description of the 'Protocol Analyzer Settings' below for more information.

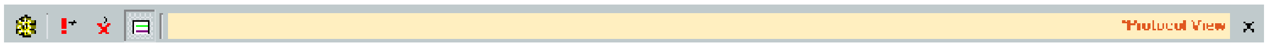
A default import file can be set in tab page protocols of dialog 'Settings'. You can open dialog 'Settings' via menu item 'Analyzer | Settings'. The default import file will be loaded during start up of the Analyzer application.

13.2.3. Details

The Protocol View consists of a toolbar positioned at the top, with three panes below it: 'relation tree' pane, 'detail' pane and 'transactions / packets' pane.

13.2.3.1. Toolbar

At the top of the Protocol View you will find a toolbar with several tool-buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol-analysis settings have been modified.

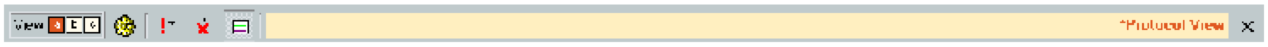
Settings
When clicking this button, the 'Protocol Analyzer Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See 'Protocol Analyzer Settings' below for more information.

Go to next relation tree item with warning(s)
When clicking this button, the relation tree item that contains one or more warnings will be selected. If multiple protocols have been analyzed, clicking this button while pressing the control key will jump to the next protocol tab that has any items with a warning.

Go to next relation tree item with error(s)
When clicking this button, the next relation tree item that contains one or more errors will be selected. If multiple protocols have been analyzed, clicking this button while pressing the control key will jump to the next protocol tab that has any item with an error.

Show transactions sequence
This button toggles the display of the 'transactions / packets' pane at the right side of the Protocol View.

Triple Analyzers
For triple Analyzers the toolbar looks as follows:

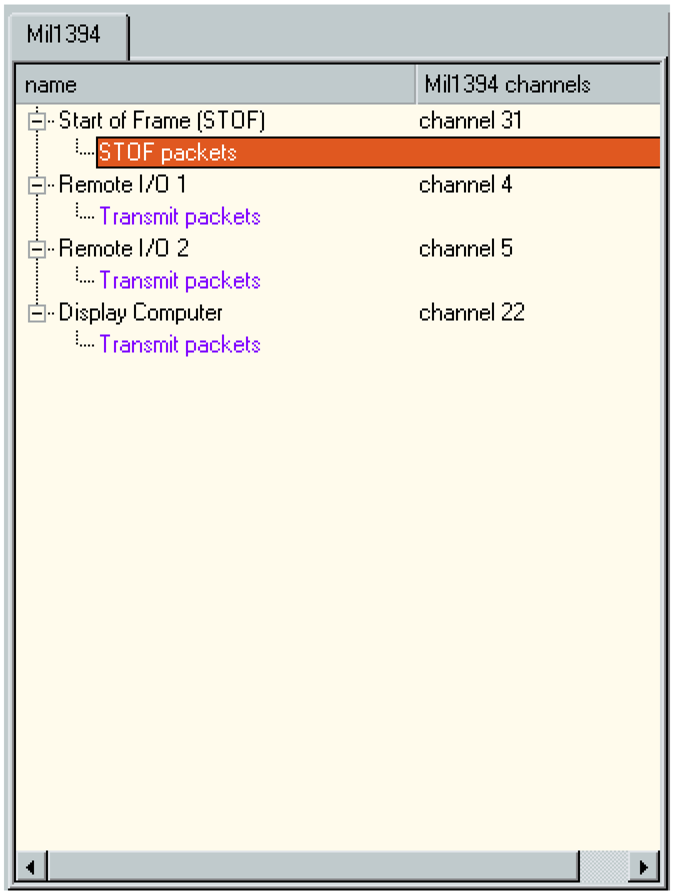


It contains the following additional control:

Analyzer node select control
With this control you are able to select the protocol analysis result of each supported Analyzer node.

13.2.3.2. Relation Tree Pane

The 'relation tree' pane shows the result of the Mil1394 Protocol analyzer in a hierarchical tree. The hierarchy of the tree shows the nodes as root items and the found Mil1394 packets of a node as child items. The tree shows a node by its name and assigned channel number and it shows the found Mil1394 packets of a node by their type. An exception is the first root item its child item represents the found STOF packets, see also figure below.



name	Mil1394 channels
Start of Frame (STOF)	channel 31
STOF packets	
Remote I/O 1	channel 4
Transmit packets	
Remote I/O 2	channel 5
Transmit packets	
Display Computer	channel 22
Transmit packets	

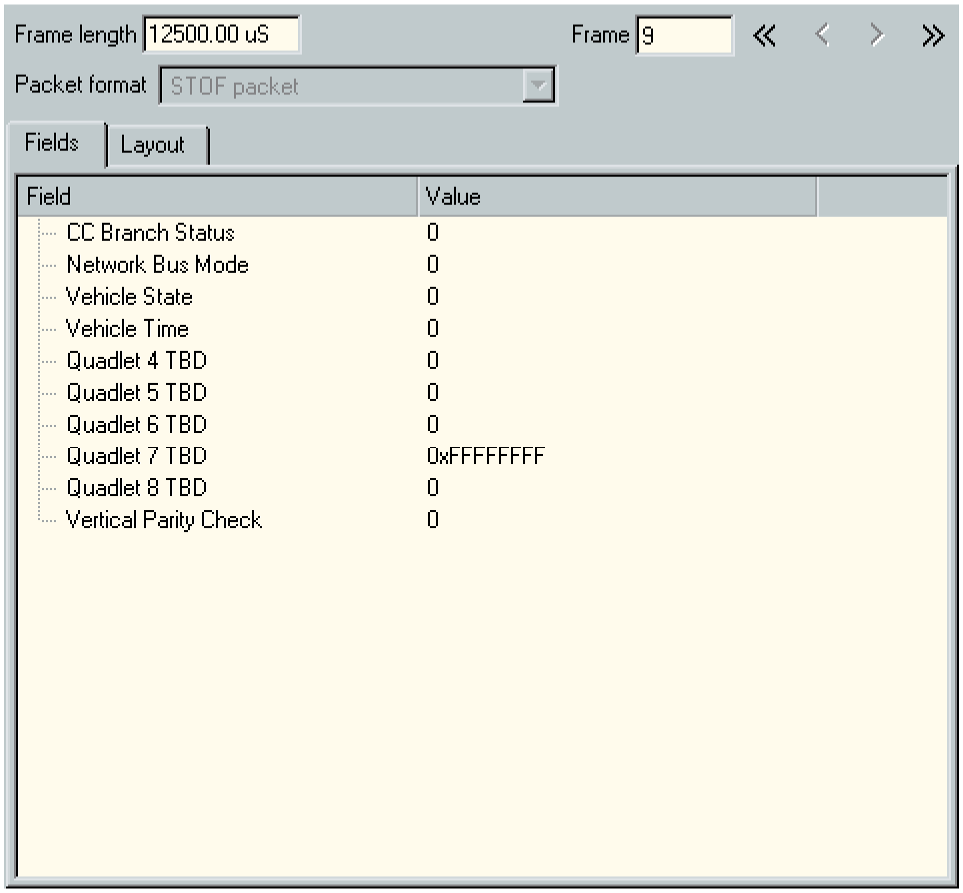
Root items (except the first root item) have four child items maximum. The child items represent the Transmit packets of the node, the Receive packets of the node, the DataPump packets of the node and the Unknown packets of the node. A packet is an Unknown packet if it can't be identified as one of the three Mil1394 packet types mentioned above. A child item of a root item is not displayed if not any Mil1394 packet in the 'transactions / packets' pane corresponds to the child item.

A red cross in front of a child item indicates one or more Mil1394 packets in the 'transactions / packets' pane, which correspond to the child item, contains one or more errors.

Click an item in the tree to highlight it. If one or more Mil1394 packets in the 'transactions / packets' pane correspond to the highlighted item, the first of these Mil1394 packet will highlight and the remaining of these Mil1394 packets will be indicated with a gray background color.

13.2.3.3. Details Pane

The 'detail' pane of the Protocol View shows the details of the highlighted Mil1394 packet in the 'transactions / packets' pane. An example that shows the details of a STOF packet is depicted below.



Frame length: 12500.00 uS
Frame: 9
Packet format: STOF packet

Field	Value
CC Branch Status	0
Network Bus Mode	0
Vehicle State	0
Vehicle Time	0
Quadlet 4 TBD	0
Quadlet 5 TBD	0
Quadlet 6 TBD	0
Quadlet 7 TBD	0xFFFFFFFF
Quadlet 8 TBD	0
Vertical Parity Check	0

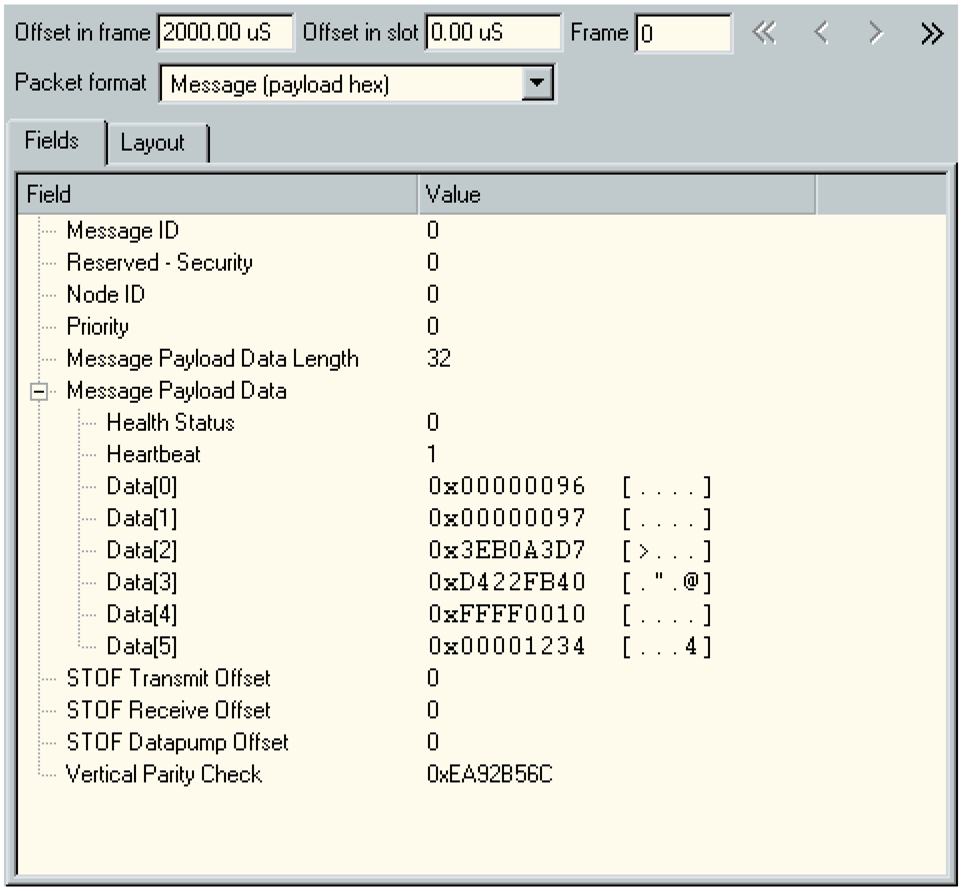
The different parts of the 'detail' pane are described below.

Frame Length
As STOF packets by definition are sent at frame offset time 0, the "Offset in Frame" and "Offset in Slot" values are not shown. A field named "Frame Length" is shown instead and displays the length of the frame started by the selected STOF packet.

Offset in slot
The 'Offset in slot' box shows the offset time of the highlighted Mil1394 packet relative to the time slot, to which the highlighted Mil1394 packet belongs.

Frame
The 'Frame' box shows the frame number to which the highlighted Mil1394 packet belongs. With button ‹ the previous Mil1394 packet in the time slot of the highlighted Mil1394 packet is selected. However if the highlighted Mil1394 packet is the first packet in the time slot, then the last packet of the same time slot in the previous frame is selected. With button › the next Mil1394 packet in the time slot of the highlighted Mil1394 packet is selected. However if the highlighted Mil1394 packet is the last packet in the time slot, then the first packet in the same time slot in the next frame is selected. Mil1394 packets in the same time slot are shown in the 'transactions / packets' pane with a gray background.

The picture below shows the details pane for a regular Mil1394 ASM message


Offset in frame 2000.00 uS Offset in slot 0.00 uS Frame 0

Packet format Message (payload hex)

Fields | Layout

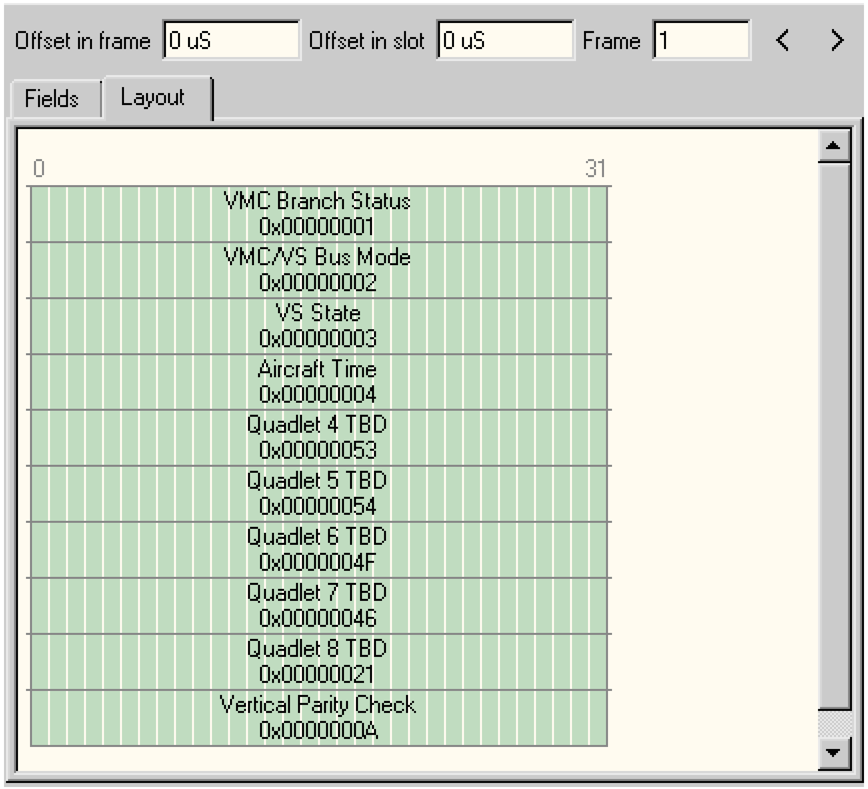
Field	Value
Message ID	0
Reserved - Security	0
Node ID	0
Priority	0
Message Payload Data Length	32
Message Payload Data	
Health Status	0
Heartbeat	1
Data[0]	0x00000096 [....]
Data[1]	0x00000097 [....]
Data[2]	0x3EB0A3D7 [>...]
Data[3]	0xD422FB40 [.".@]
Data[4]	0xFFFF0010 [....]
Data[5]	0x00001234 [...4]
STOF Transmit Offset	0
STOF Receive Offset	0
STOF Datapump Offset	0
Vertical Parity Check	0xEA92B56C

Offset in frame
The 'Offset in frame' box shows the offset time of the highlighted Mil1394 packet relative to the frame, to which the highlighted Mil1394 packet belongs. Each STOF packet starts a new frame. This also means that the offset time of a STOF packet is always 0 microseconds.

Offset in slot
The 'Offset in slot' box shows the offset time of the highlighted Mil1394 packet relative to the time slot, to which the highlighted Mil1394 packet belongs.

Fields
The 'Fields' tab page shows the fields and their values of the highlighted Mil1394 packet, see also figure above.

Layout
The 'Layout' tab page shows the layout of the highlighted Mil1394 packet. The figure below shows the layout of a STOF packet.



Offset in frame: 0 uS · Offset in slot: 0 uS · Frame: 1

Fields | Layout

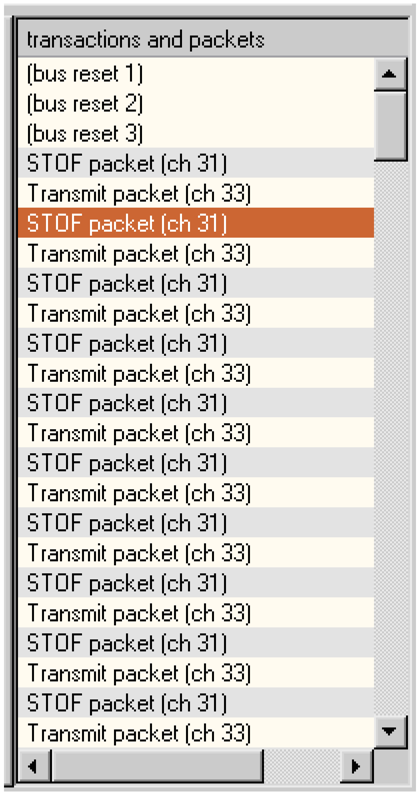
0 … 31

VMC Branch Status 0x00000001
VMC/VS Bus Mode 0x00000002
VS State 0x00000003
Aircraft Time 0x00000004
Quadlet 4 TBD 0x00000053
Quadlet 5 TBD 0x00000054
Quadlet 6 TBD 0x0000004F
Quadlet 7 TBD 0x00000046
Quadlet 8 TBD 0x00000021
Vertical Parity Check 0x0000000A

Like the 'Fields' tab page it shows the fields and their values of the highlighted Mil1394 packet. But now the fields and their values are displayed graphically inside a data block, which represents the highlighted Mil1394 packet.

'transactions / packets' pane

The 'transaction / packets' pane of the Protocol View shows all bus resets and Mil1394 packets found by the Mil1394 Protocol analyzer, see the figure below.



transactions and packets

(bus reset 1)
(bus reset 2)
(bus reset 3)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)
STOF packet (ch 31)
Transmit packet (ch 33)

The 'transactions / packets' pane lists the bus resets and Mil1394 packets in a chronological order.

A red cross in front of a Mil1394 packet in the 'transactions / packets' pane indicates the Mil1394 packet contains one or more errors.

If you click a Mil1394 packet in the 'transactions / packets' pane, it will highlight. The item in the tree of the 'relation tree' pane, to which the highlighted Mil1394 packet corresponds, will also highlight. The remaining Mil1394 packets, which also correspond to the highlighted item, will be indicated with a gray background color.

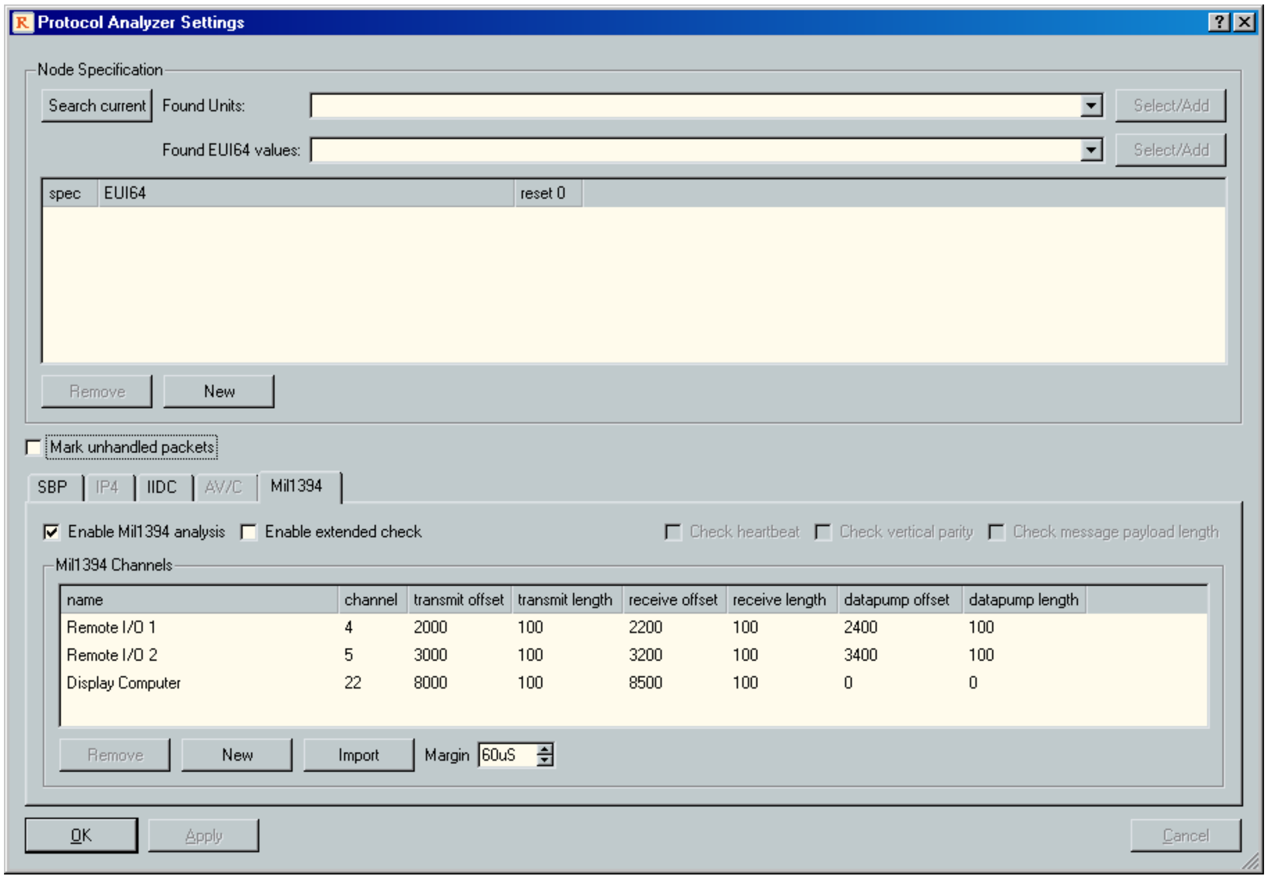
13.2.3.4. Mil1394 Protocol Settings

The Mil1394 Protocol analyzer has to have knowledge of the nodes on the bus, which support the Mil1394 Protocol. The Mil1394 Protocol analyzer has to know the following information of a node, which support the Mil1394 Protocol:

- its name,
- the assigned channel number,
- the transmit offset and length, this specify the time slot the node may transmit asynchronous stream packets,
- the receive offset and length, this specify the time slot the node can aspect to receive asynchronous stream packets,
- and the datapump offset and length, this specifies the time slot the node may transmit datapump packets.

This information of a node is to be provided manually in the 'Protocol Analyzer Settings' dialog or via an import file.

Open 'Protocol Analyzer Settings' dialog
Click the toolbar button 'Settings' to open the 'Protocol Analyzer Settings' dialog, see figure below.



[✓ Enable Mil1394 analysis]

name	channel	transmit offset	transmit length	receive offset	receive length	datapump offset	datapump length
Remote I/O 1	4	2000	100	2200	100	2400	100
Remote I/O 2	5	3000	100	3200	100	3400	100
Display Computer	22	8000	100	8500	100	0	0

Margin 60uS

The information of the nodes, which support the Mil1394 Protocol, is to be added in the table 'Mil1394 Channels' of the Mil1394 tab page. Click on the Mil1394 tab to show the Mil1394 tab page on top.

table Mil1394 Channels
Each row in table 'Mil1394 Channels' represents a node. The table shows in the columns the specific information of a node. This specific information of a node is: its name, the channel number, the heartbeat and the time slots to transmit an asynchronous stream packet, to receive an asynchronous stream packet and to transmit a datapump packet.

There are 2 procedures defined to add information of one or more nodes into table 'Mil1394 Channels'. The first procedure uses button 'New' to add information into the table 'Mil1394 Channels' and the second procedure uses an import file to load information into the table 'Mil1394 Channels', see as an example import file Mil1394Definitions.csv.

add information of one or more nodes

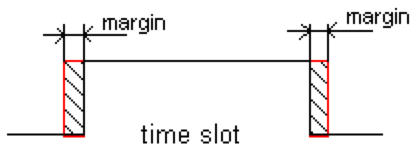
manually
Click button 'New' to add a row to table 'Mil1394 Channels' and provide for each column the specific information.

Import file
Click button 'Import' to open the dialog 'Open' and select the import file (e.g. Mil1394Definitions.csv). The information included in the import file will be loaded into table 'Mil1394 Channel'.

The extra features of the Mil1394 tab page are described below.

remove information of a node
Click the row in table 'Mil1394 Channel' to select it. This will also enable button 'Remove'. Click button 'Remove' to remove the selected row from table 'Mil1394 Channel'.

margin of the time slots
The margin of the time slots is adjustable via spin box 'Margin'. The margin has effect on the start time and the end time of a time slot, see figure below.



The default value of margin is 60 microseconds.

extended error checking
Click the check box 'Enable extended check' to enable the check boxes: 'Check heartbeat', 'Check vertical parity' and 'Check message payload length'.

Click the check box 'Check heartbeat' to enable error checking of the heartbeat value of the nodes, which support the Mil1394 Protocol. During analysis of a Mil1394 packet the Mil1394 Protocol analyzer will first check if the heartbeat value of the node, which generated the Mil1394 packet, has increased. The field 'Heartbeat' of the Mil1394 packet contains the heartbeat value of the node. If the heartbeat value has increased the heartbeat value of the node is valid. If the heartbeat value of the node didn't increased the Mil1394 Protocol analyzer checks if the frequency by which the heartbeat value increase doesn't fall below the minimum heartbeat frequency. If this is the case, a heartbeat error occurred.

Click the check box 'Check vertical parity' to enable error checking of the field 'Vertical Parity Check' of the Mil1394 packets.

Click the check box 'Check message payload length' to enable error checking of the field 'Message Payload Length' of the Mil1394 packets.

13.3. Scriptor

13.3.1. Timed Sending

This example shows some of the timed sending capabilities of the Scriptor. Each frame, two packets are sent. One is a STOF packet that is transmitted at frame offset time 0 and the other packet is a data packet using the Mil1394 Protocol. By using a slider on the Control Panel, the frame offset of the second packet can be controlled.

The next section describes how to run the example and how to use the recorder to verify that the packet's offset actually changes when moving the slider.

13.3.1.1. How to use it

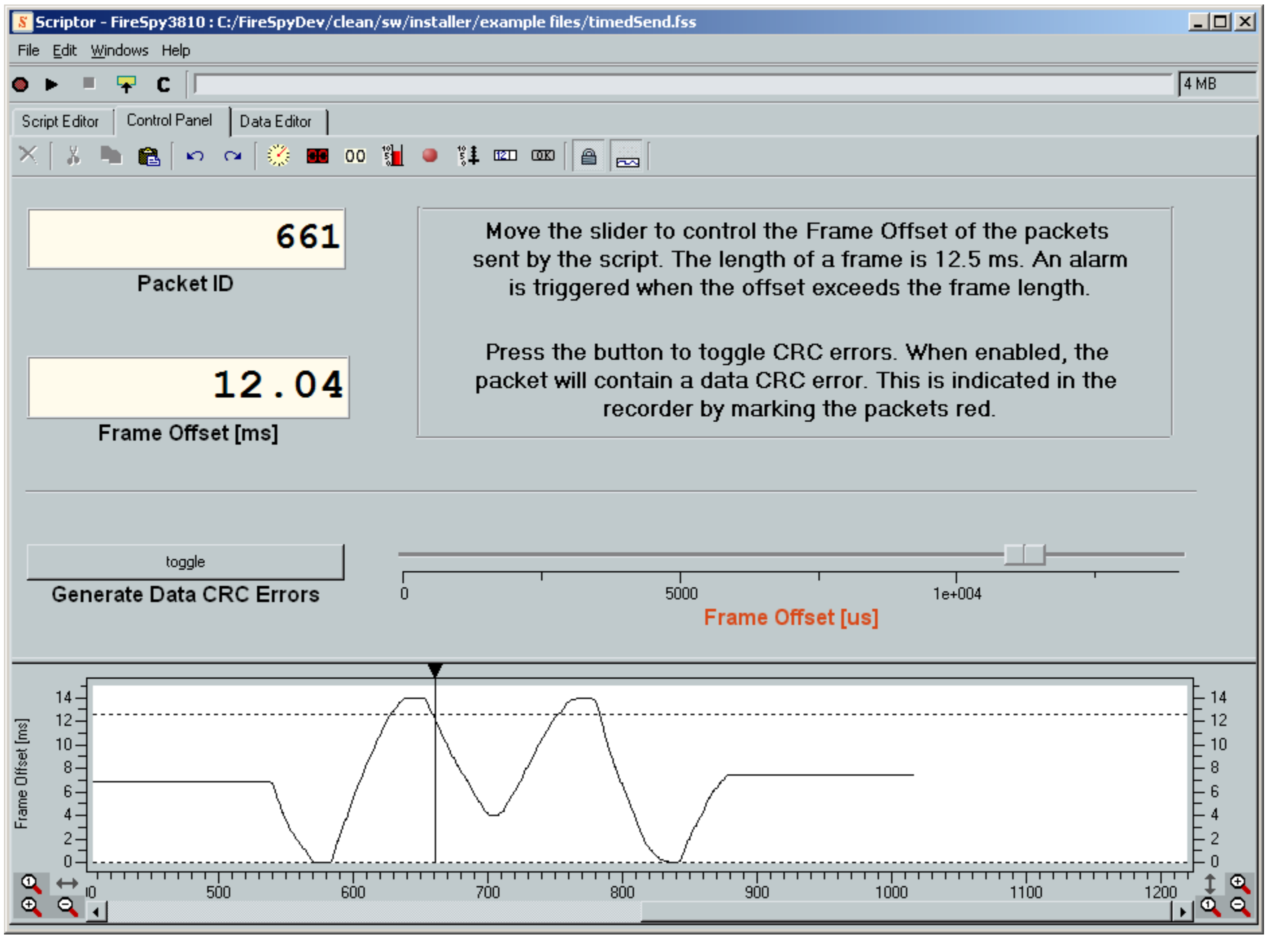
Control Panel contents
After loading the file "timedSend.fss" and clicking on the Control Panel tab, a Control Panel will be shown as in the picture below. The Control Panel contains the following controls:

Packet ID
This control displays the ID of the current packet. The ID of a packet is also written to the packet before it is sent. This way, it is possible to find a packet in the recorder that corresponds to a packet that is selected on the Control Panel.

Frame Offset
This control displays the offset of the current packet in the current frame. The frame length is set to its default value of 12.5 ms, and therefore, this offset should have a value between 0 and 12.5 ms. The Frame Offset control also contains an alarm that is triggered if the value exceeds the frame length. This is also shown in the history graph at the bottom as a dotted line.

Generate Data CRC Errors
This button can be toggled to enable/disable the CRC error on that Mil1394 packet with the user controlled frame offset. This feature makes it easier to distinguish the STOF packets from the Mil1394 data packets in the recorder. (the ones with a CRC error will be displayed in red)

Frame Offset
This slider controls the frame offset of the Mil1394 data packet. By sliding the slider bar along its horizontal axis, it is possible to set the frame offset to a value between 0 and 15ms. As the frame length is only 12.5 ms, setting the offset to a value greater than 12.5 will prefend the packets from being sent. When this happens, the "Frame Offset" control will start blinking.



Move the slider to control the Frame Offset of the packets sent by the script. The length of a frame is 12.5 ms. An alarm is triggered when the offset exceeds the frame length.

Press the button to toggle CRC errors. When enabled, the packet will contain a data CRC error. This is indicated in the recorder by marking the packets red.

Running the script
To run this example, please proceed with the following steps:

1. Open the recorder and start recording
2. Go back to the scriptor and select the Control Panel
3. Start the script
4. Press the button "Generate Data CRC Errors"
5. Move the slider back and forth such that it shows a curve similar to the picture above
6. Stop the script
7. Open the recorder window and download the recording
8. In the recorder, open the Protocol View (Menu Bar - View - Protocol View)
9. Select the Mil1394 tab in the Protocol View
10. Click on the "Protocol Settings" button in the toolbar of the Protocol View
11. Select the Mil1394 tab
12. Click on new and set the channel field to 11
13. Click on "ok"

Now that all windows are setup something like the Recorder picture below and the Control Panel picture above, it is time to start looking at what happened.

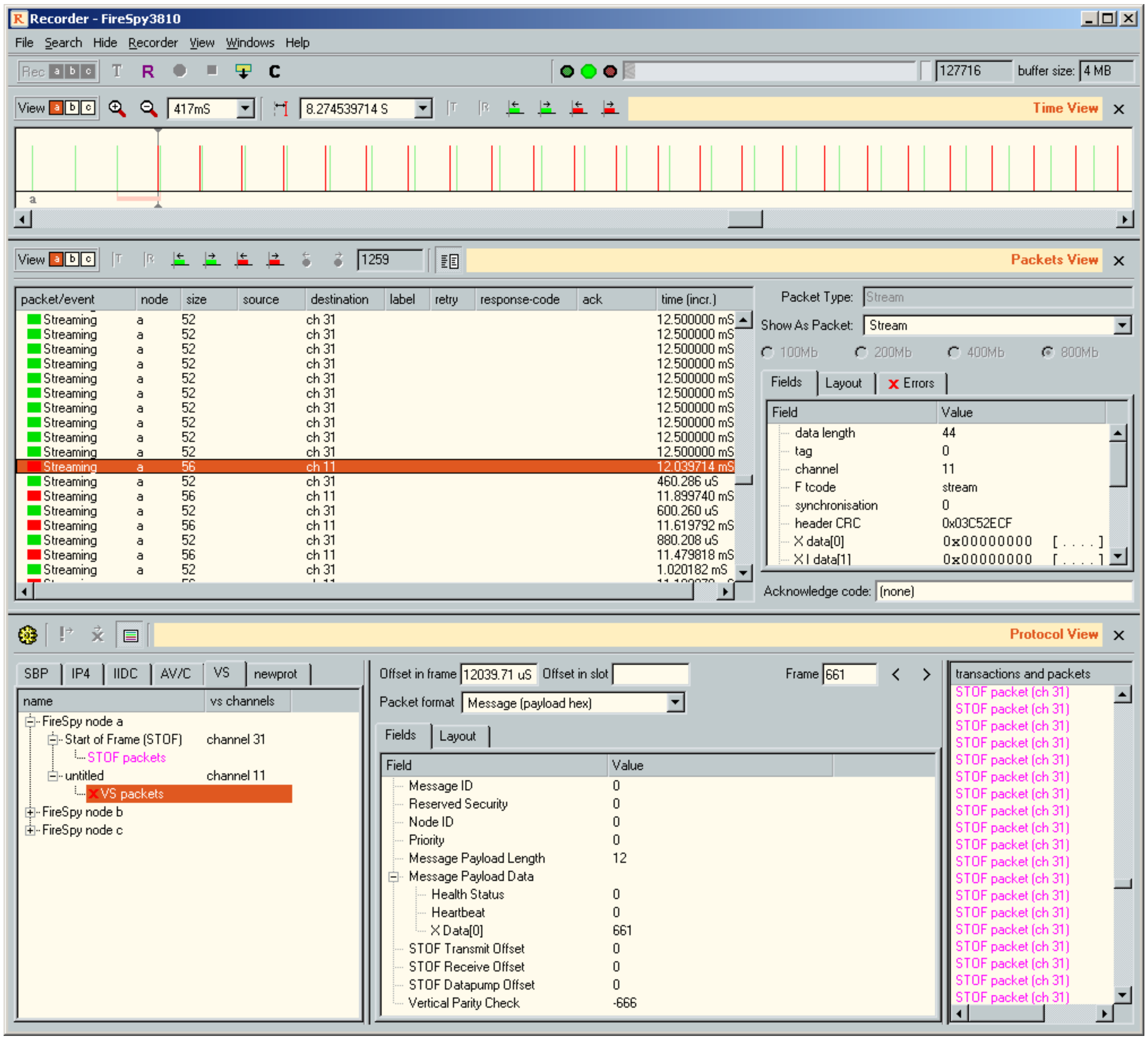
In the Control Panel window, the History graph displays the frame offset versus the packet number. In the picture above, a packet is selected with ID 661, just after the graph descended below the dotted line. The dotted line represents the frame length and, therefore, the selected packet is the first packet in a while that was sent at a frame offset that fits in the frame length. According to the Control Panel, the packet's frame offset is 12.04ms. The Recorder can be used to verify that this is correct.

To find the same packet in the Recorder, the packet ID is written to each packet by the script. The location of the packet ID in the packet is in between the "Heartbeat" and the "STOF Transmit Offset" fields as indicated in the picture below. By walking through the packets using the "Packet View" and looking at the "X Data[0]" field in the "Protocol View" it should be fairly easy to find the packet with ID 661. In the picture below, the packet was found and the picture shows that the "Offset in Frame" was 12039 us, which corresponds to the value indicated by the Control Panel. So, the frame offset was indeed user controllable by the slider on the Control Panel.

The "Time View" of the recorder is also interesting to look at. The first think that you will probably notice is that some of the packets are painted red. This is due to the fact that when running this example, the "Generate Data CRC errors" button was pressed. As the STOF packets are painted green, it is easy to distinguish them. In the picture below, the Packet with ID 661 is selected and the frame it belongs to is also marked below it with a red horizontal bar.

The frames left of the selected packet donot contain a red packet. This is because the selected packet was the first packet in a while with a frame offset smaller than the frame length. The packets before this packet had a frame offset exceeding the frame length and could therefore not be sent.

The selected packet has a frame offset of 12.04 ms and is therefore painted almost at the right side of the frame. When looking at the Time View from the left to the right, starting at the selected packet, it can be seen that the frame offsets are decreasing. This is exactly what we've alread seen in the History Graph of the Control Panel.



packet/event	node	size	source	destination	label	retry	response-code	ack	time (incr.)
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟩 Streaming	a	52		ch 31					12.500000 mS
🟥 Streaming	a	56		ch 11					12.039714 mS
🟩 Streaming	a	52		ch 31					460.286 uS
🟥 Streaming	a	56		ch 11					11.899740 mS
🟩 Streaming	a	52		ch 31					600.260 uS
🟥 Streaming	a	56		ch 11					11.619792 mS
🟩 Streaming	a	52		ch 31					880.208 uS
🟥 Streaming	a	56		ch 11					11.479818 mS
🟩 Streaming	a	52		ch 31					1.020182 mS

Packet Type: Stream
Show As Packet: Stream
100Mb 200Mb 400Mb [✓ 800Mb]
Fields Layout Errors

Field	Value
data length	44
tag	0
channel	11
F tcode	stream
synchronisation	0
header CRC	0x03C52ECF
X data[0]	0x00000000 [. . . .]
X I data[1]	0x00000000 [. . . .]

Acknowledge code: (none)

SBP IP4 IIDC AV/C VS newprot

name	vs channels
FireSpy node a	
Start of Frame (STOF)	channel 31
STOF packets	
untitled	channel 11
VS packets	
FireSpy node b	
FireSpy node c	

Offset in frame 12039.71 uS Offset in slot
Frame 661
Packet format Message (payload hex)
Fields Layout

Field	Value
Message ID	0
Reserved Security	0
Node ID	0
Priority	0
Message Payload Length	12
Message Payload Data	
Health Status	0
Heartbeat	0
X Data[0]	661
STOF Transmit Offset	0
STOF Receive Offset	0
STOF Datapump Offset	0
Vertical Parity Check	-666

transactions and packets
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)
STOF packet (ch 31)

13.3.1.2. Details

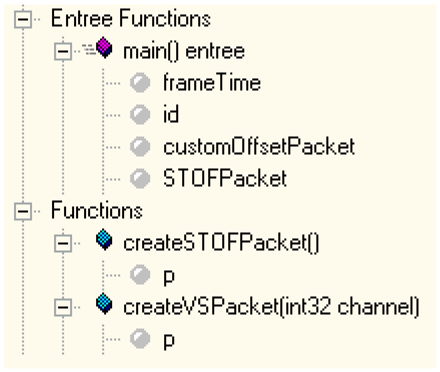
The script source code

The picture below shows the Object Browser contents for the current example. The picture shows that the script contains one main entry function and two regular functions. The purpose of the functions will be explained below:

main
Main is the only process in the script. It sets up the Mil1394 Stream packet and the STOF packet and sends them every frame according to the settings on the Control Panel.

createSTOFPacket
This function creates a STOF packet and returns the ID of the packet buffer the packet is written to. Two data objects from the Data Editor are used. One for the packet header and one for the packet data.

createMil1394Packet
This function creates a Mil1394 packet and returns the ID of the packet buffer the packet is written to. Two data objects from the Data Editor are used. One for the packet header and one for the packet data.

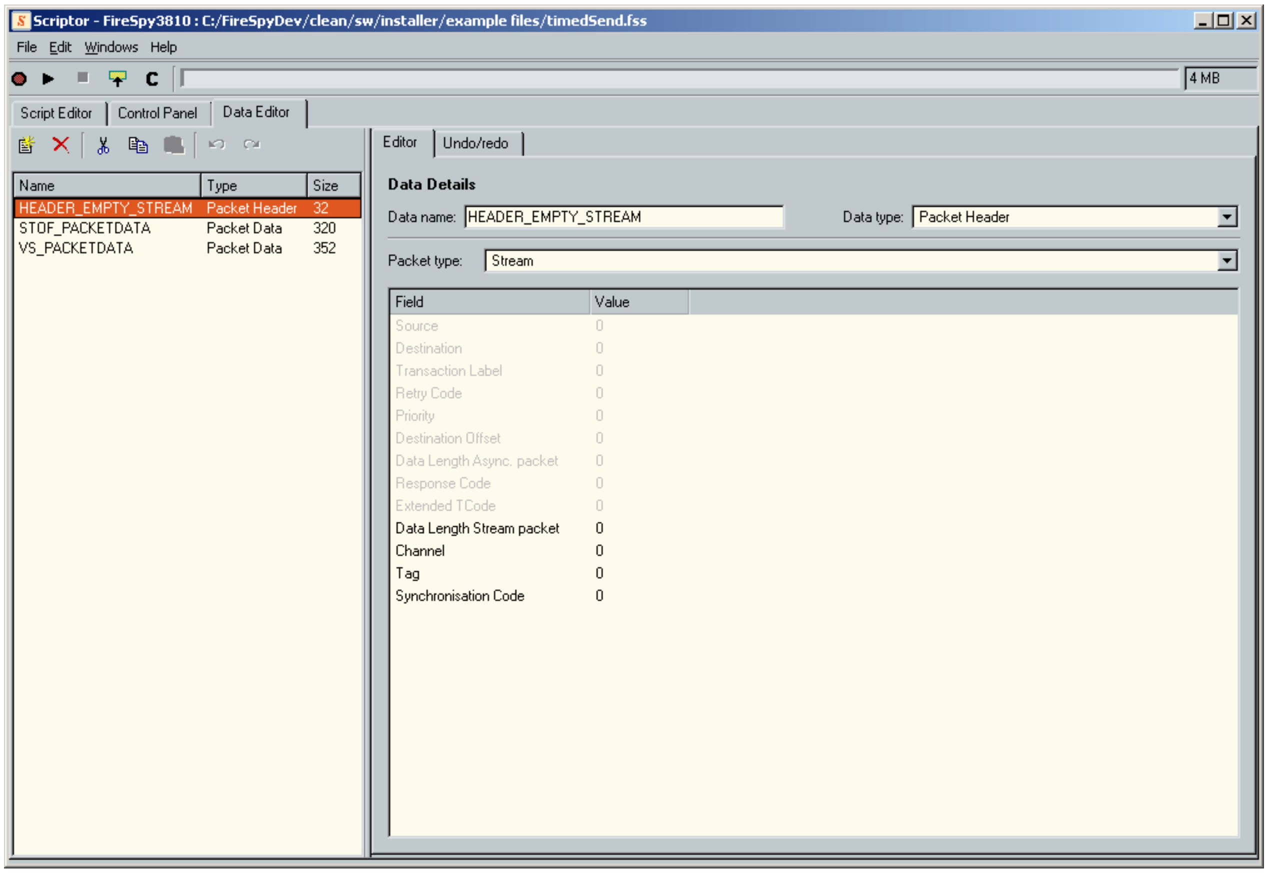

```
Entree Functions
  main() entree
    frameTime
    id
    customOffsetPacket
    STOFPacket
Functions
  createSTOFPacket()
    p
  createVSPacket(int32 channel)
    p
```

The script itself is displayed below. The comments in the script should be rather self-explanatory.

```
//===========================================================================
=====
// Configuration
//===========================================================================
=====

// Control Panel id's
#define CONTROL_FRAMETIME 1
#define CONTROL_ERROR 2
#define CONTROL_ID 3
#define CONTROL_OFFSET 4

// Settings for the packets
#define CHANNEL 11
#define SPEED 3

//===========================================================================
=====
// Processes
//===========================================================================
=====

void main () entry
        // Initialize
        int32 frameTime
        int32 id = 0

        // Create the packets
        int32 customOffsetPacket = createMil1394Packet ( CHANNEL )
        int32 STOFPacket = createSTOFPacket ()

        // Set initial control values
        setControlValue ( CONTROL_FRAMETIME, 11000 )
        // Select Node
        selectFireSpyNode ( 0 )

        // Main loop
        while true
                // Wait for the beginning of a frame and send the STOF packet
                waitStartOfFrame ()
                sendPacketNextFrame ( STOFPacket, 0 )

                // Set a data CRC error to the packet if the control panel button is down
                if getControlValue ( CONTROL_ERROR )
                        setControlValue ( CONTROL_ERROR, 1 )
                        setSendDataCRCError ( customOffsetPacket, true )

                    else
                        setControlValue ( CONTROL_ERROR, 0 )
                        setSendDataCRCError ( customOffsetPacket, false )


                    // Send the packet with custom frame offset
                    frameTime = getControlValue ( CONTROL_FRAMETIME )
                    setPacketDataFields ( customOffsetPacket, id )
                        setPacketDataQuadlet ( customOffsetPacket, 6 , id )

                    sendPacketNextFrame ( customOffsetPacket, frameTime )

                    // Update control panel
                    setControlValue ( CONTROL_OFFSET, frameTime )
                    setControlValue ( CONTROL_ID, id )
                    grabControlValues ( id )

                    // Increment packet id
                    id = id + 1


    //==========================================================================
    ========
    // Functions
    //==========================================================================
    ========

    int32 createSTOFPacket ()
        // Create a packet buffer
        int32 p = newPacket ( 512 )

        // Set the packet data from the Data Editor
        fillPacket ( p, HEADER_EMPTY_STREAM, STOF_PACKETDATA )
                fillPacketHeader ( p, HEADER_EMPTY_STREAM )
                fillPacketData ( p, STOF_PACKETDATA )


            // Set the channel
            setPacketHeaderFields ( p, 31 )
                setPacketHeaderField ( p, 0 , 18 , 6 , 0x1F )


            // Set packet speed
            setPacketSpeed ( p, SPEED )

            // Enable auto VPC calculation
            enableAutoVPC ( p, true )
        return p


    //------------------------------------------------------------------------------
    int32 createMil1394Packet ( int32 channel )
      // Create a packet buffer
      int32 p = newPacket ( 512 )

      // Set the packet data from the Data Editor
      fillPacket ( p, HEADER_EMPTY_STREAM, Mil1394_PACKETDATA )
          fillPacketHeader ( p, HEADER_EMPTY_STREAM )
          fillPacketData ( p, Mil1394_PACKETDATA )


        // Set the channel
        setPacketHeaderFields ( p, channel )
          setPacketHeaderField ( p, 0 , 18 , 6 , channel )


        // Set packet speed
        setPacketSpeed ( p, SPEED )

        // Enable auto VPC calculation
        enableAutoVPC ( p, true )

        return p
```

Data definitions
Data Objects are used for setting the packet contents to the correct formats. The pictures below show the data definitions used by the script. The first picture shows the header of a stream packet. The second picture shows the packet contents of a Mil1394 STOF packet and the third picture shows the packet contents of a standard Mil1394 Stream packet. Note that all fields are just set to zero and that the data objects' only purpose is to set the correct format, not the contents. If you like to send packets that do contain some data values, you can changed the fields in these data definitions.

Stream Header
The channel field will be set by the script code itself. Therefore this header can be used by both the STOF packet and the Mil1394 data packet.


Scriptor - FireSpy3810 : C:/FireSpyDev/clean/sw/installer/example files/timedSend.fss
File Edit Windows Help
4 MB
Script Editor
Control Panel
Data Editor
Editor
Undo/redo

Name	Type	Size
HEADER_EMPTY_STREAM	Packet Header	32
STOF_PACKETDATA	Packet Data	320
VS_PACKETDATA	Packet Data	352

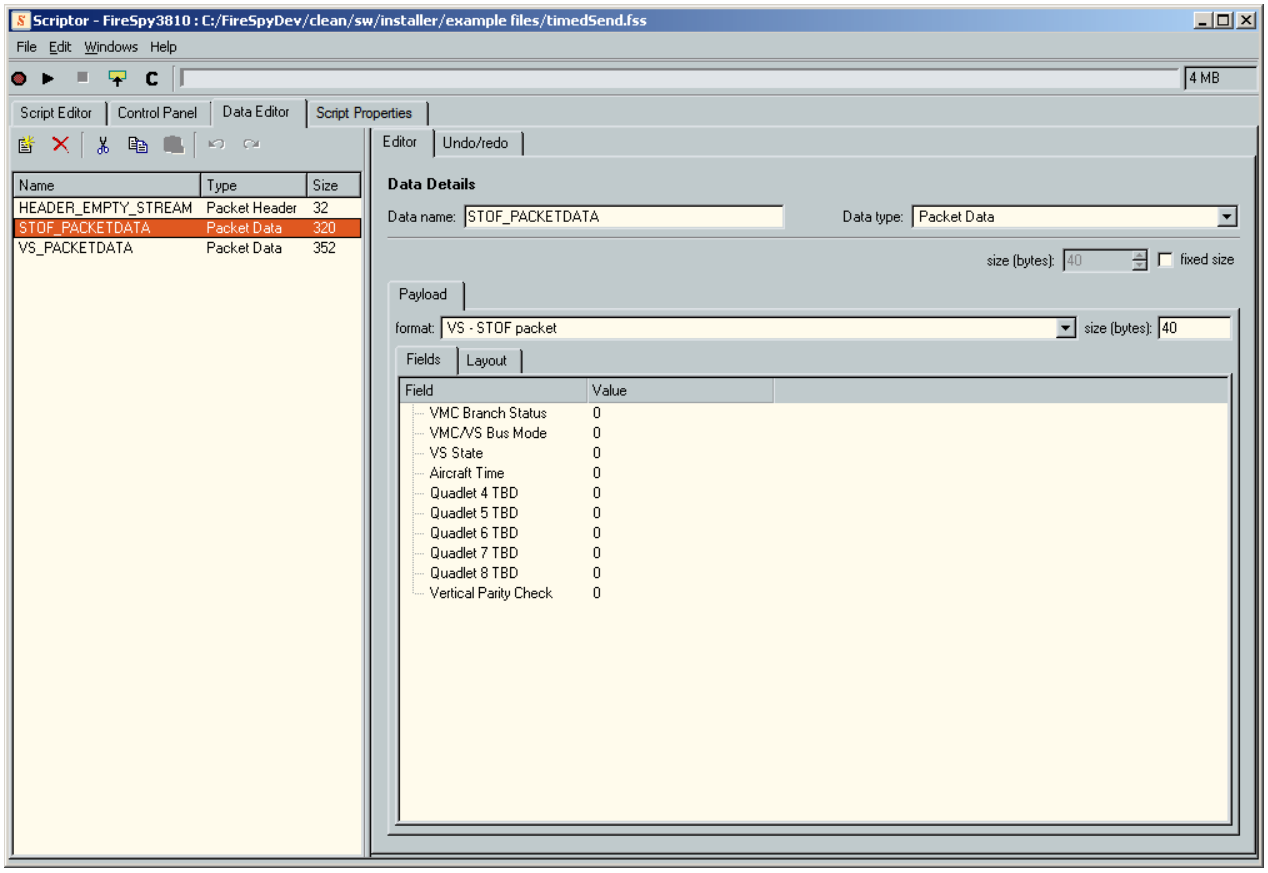
Data Details

Data name: HEADER_EMPTY_STREAM   Data type: Packet Header

Packet type: Stream

Field	Value
Source	0
Destination	0
Transaction Label	0
Retry Code	0
Priority	0
Destination Offset	0
Data Length Async. packet	0
Response Code	0
Extended TCode	0
Data Length Stream packet	0
Channel	0
Tag	0
Synchronisation Code	0

STOF Packet



Name	Type	Size
HEADER_EMPTY_STREAM	Packet Header	32
STOF_PACKETDATA	Packet Data	320
VS_PACKETDATA	Packet Data	352

Data Details

Data name: STOF_PACKETDATA   Data type: Packet Data

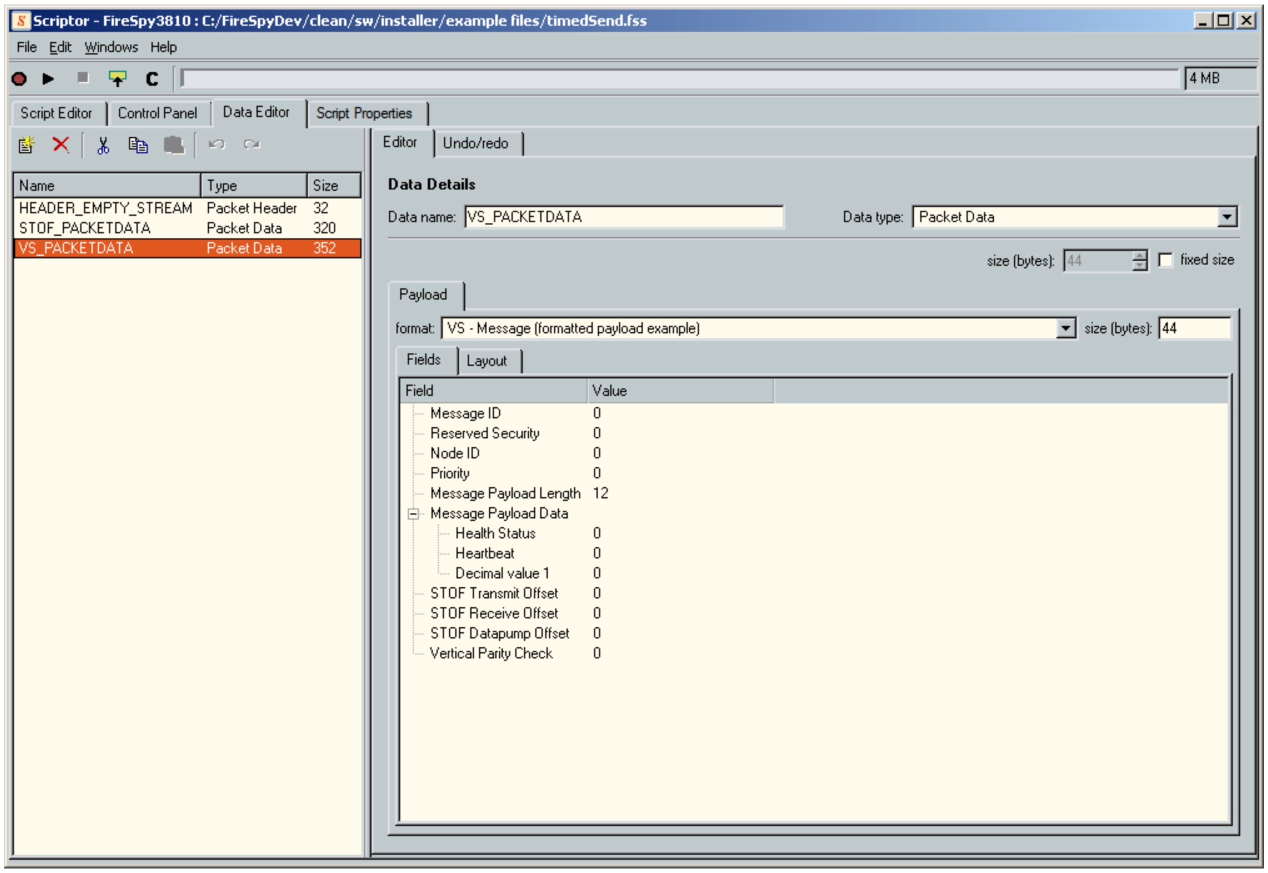
size (bytes): 40   fixed size

format: VS - STOF packet   size (bytes): 40

Field	Value
VMC Branch Status	0
VMC/VS Bus Mode	0
VS State	0
Aircraft Time	0
Quadlet 4 TBD	0
Quadlet 5 TBD	0
Quadlet 6 TBD	0
Quadlet 7 TBD	0
Quadlet 8 TBD	0
Vertical Parity Check	0

Mil1394 Data Packet

The only field that is non-zero is the "Message Payload Length". This field is set to 12 because the script sets the ID of the packet to the "Decimal Value 1" field.



Name	Type	Size
HEADER_EMPTY_STREAM	Packet Header	32
STOF_PACKETDATA	Packet Data	320
VS_PACKETDATA	Packet Data	352

Data name: VS_PACKETDATA
Data type: Packet Data
size (bytes): 44 fixed size
format: VS - Message (formatted payload example) size (bytes): 44

Field	Value
Message ID	0
Reserved Security	0
Node ID	0
Priority	0
Message Payload Length	12
Message Payload Data	
Health Status	0
Heartbeat	0
Decimal value 1	0
STOF Transmit Offset	0
STOF Receive Offset	0
STOF Datapump Offset	0
Vertical Parity Check	0

13.3.2. Mil1394 Signal Monitor

This script can be used to monitor signal values for Mil1394 messages. It uses a channel number and messageID to filter the received messages. Up to 20 signals can be watched simultaneously. Each signal has a configurable location in the message and data type.

13.3.2.1. How to use it

Control Panel contents

After loading the file "Mil1394MessageMonitor.fss" and clicking on the Control Panel tab, a Control Panel will be shown as in the picture below. The Control Panel contains the following controls:

Channel
The channel to receive packets on

MessageID
The MessageID to extract the signals for

Node A, Node B and Node C
Enable (button is down) these buttons to enable receiving packets on the corresponding Analyzer node

Message Count
The number of messages received that matched the specified node, channel and messageID. Note that this counter is only reset when the script is restarted, not when the criteria are changed.

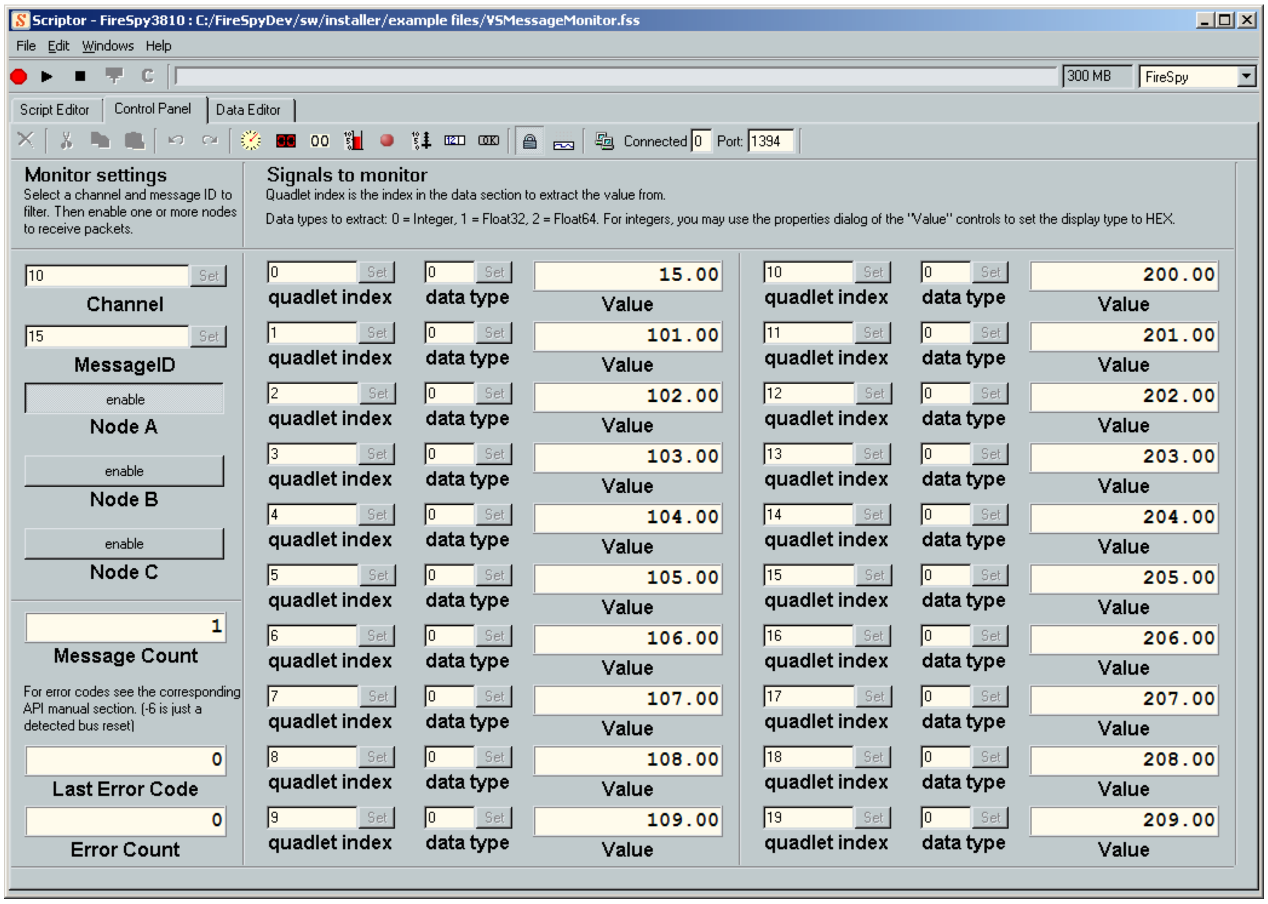
Last Error Code
In case one of the send/receive function calles returned a negative value, this value is displayed in this control. The textual representation of the error codes can be found here: API function error codes

Error Count

This counter counts the number of times an API function returned a negative number
quadlet index
The quadlet offset in the message payload data the signal is located. For Mil1394 Messages, the MessageID is quadlet 0.

Data type
The data type of the signal. The following values are allowed:
0: Integer
1: Float32
2: Float64 (will read two quadlets to get 64 bits)

Value
This indictor displays the last received value for the specified quadlet offset and data type. If you right-click this indicator and open the properties dialog, you may choose to display the value in Hex format. Note that this is only valid if the data type is set to integer.


Scriptor - FireSpy3810 : C:/FireSpyDev/sw/installer/example files/VSMessageMonitor.fss

File Edit Windows Help

300 MB FireSpy

Script Editor | Control Panel | Data Editor

Connected 0 Port: 1394

Monitor settings
Select a channel and message ID to filter. Then enable one or more nodes to receive packets.

Signals to monitor
Quadlet index is the index in the data section to extract the value from.
Data types to extract: 0 = Integer, 1 = Float32, 2 = Float64. For integers, you may use the properties dialog of the "Value" controls to set the display type to HEX.

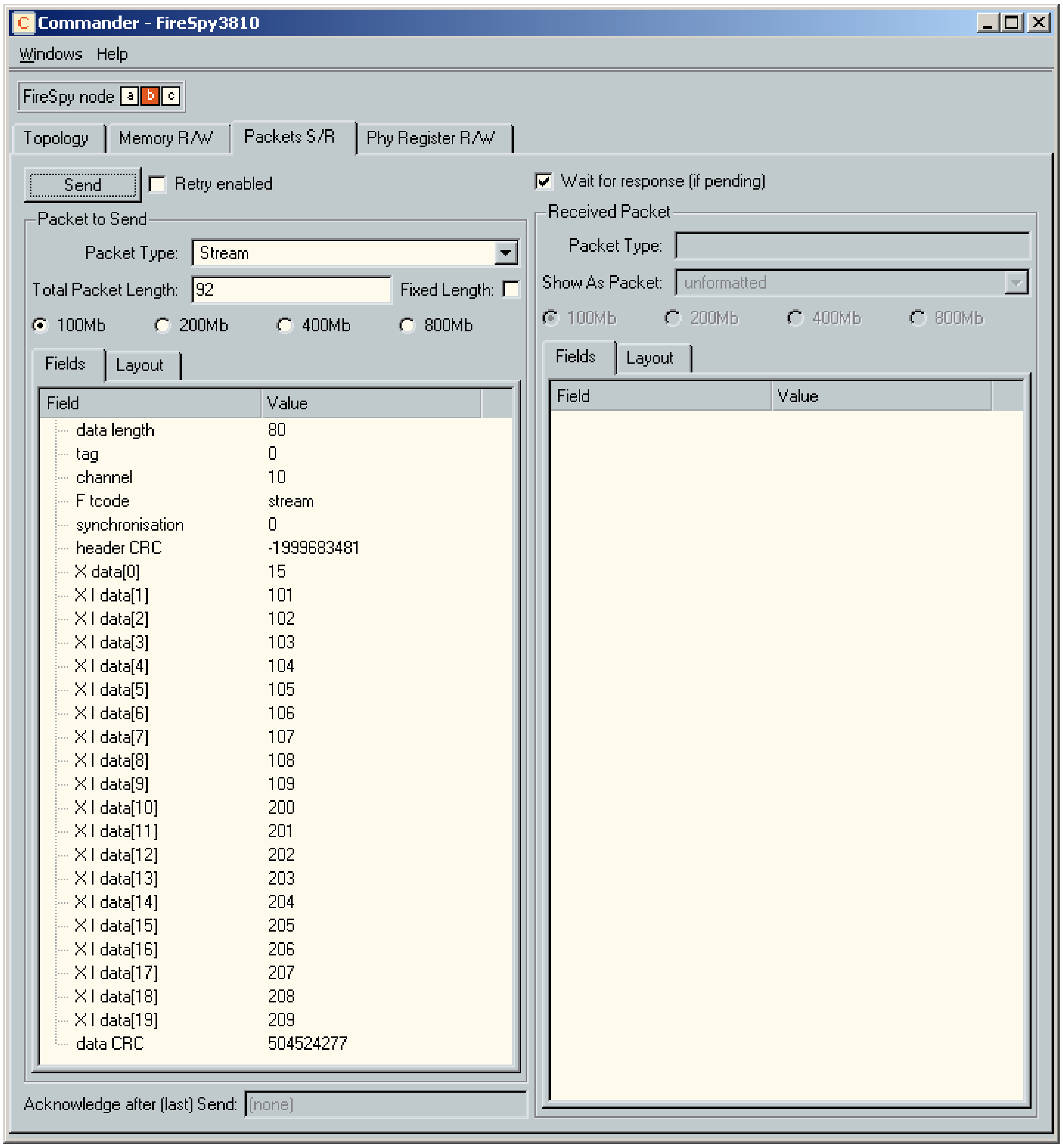
Channel	10
MessageID	15
Node A	enable
Node B	enable
Node C	enable
Message Count	1
Last Error Code	0
Error Count	0

For error codes see the corresponding API manual section. (-6 is just a detected bus reset)

quadlet index	data type	Value	quadlet index	data type	Value
0	0	15.00	10	0	200.00
1	0	101.00	11	0	201.00
2	0	102.00	12	0	202.00
3	0	103.00	13	0	203.00
4	0	104.00	14	0	204.00
5	0	105.00	15	0	205.00
6	0	106.00	16	0	206.00
7	0	107.00	17	0	207.00
8	0	108.00	18	0	208.00
9	0	109.00	19	0	209.00

Running the script
To run this example, please proceed with the following steps:

1. Connect a cable between Analyzer ports A2 and B2
2. Load and start the script
3. Open the commander and set it up like in the picture below
4. Press the send button
5. Open the Scriptor Window and click on the Control Panel tab
6. The Control Panel should now show the values as sent by the commander



[ ] Retry enabled
[✓] Wait for response (if pending)

Packet to Send

Packet Type: Stream
Total Packet Length: 92
Fixed Length: [ ]
[✓ 100Mb] 200Mb 400Mb 800Mb

Field	Value
data length	80
tag	0
channel	10
F tcode	stream
synchronisation	0
header CRC	-1999683481
X data[0]	15
X I data[1]	101
X I data[2]	102
X I data[3]	103
X I data[4]	104
X I data[5]	105
X I data[6]	106
X I data[7]	107
X I data[8]	108
X I data[9]	109
X I data[10]	200
X I data[11]	201
X I data[12]	202
X I data[13]	203
X I data[14]	204
X I data[15]	205
X I data[16]	206
X I data[17]	207
X I data[18]	208
X I data[19]	209
data CRC	504524277

Acknowledge after (last) Send: (none)

Received Packet

Packet Type:
Show As Packet: unformatted
100Mb 200Mb 400Mb 800Mb

Field	Value

13.4. Editing Mil1394 Formats

When defining your own packet formats for the Mil1394 Protocol, you will be able to display the message payload of your Mil1394 stream packets in any format.

You can use these custom formats in the Scriptor, Generator, Mil1394 Protocol analyzer of the Recorder and in the Filter Sets for the Filter/Trigger logic to define field conditions.

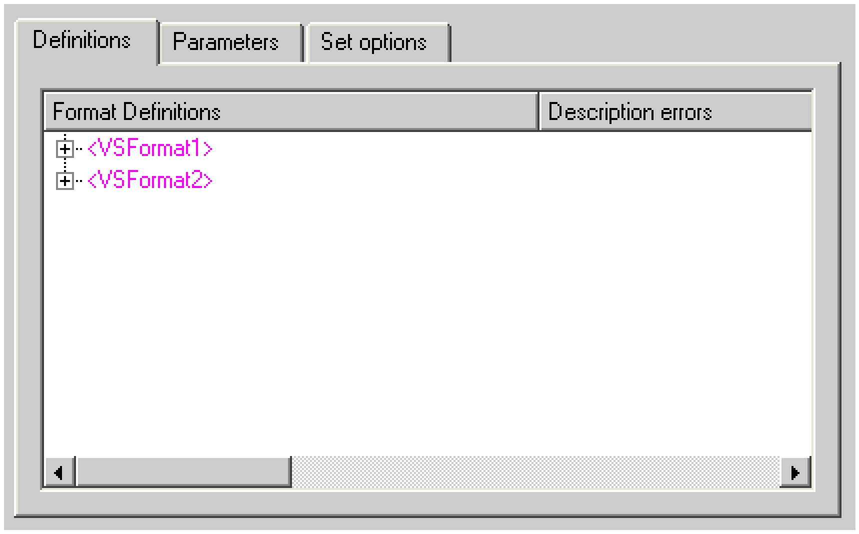
You can create multiple format sets (multiple files), with each file one or more format definitions. Each format definition that you define, will show up in the list of formats to select a format from.

It is also possible that you create a format definition that can be used for more than one channel and that automatically adapts its format to the used channel by using the 'channel' input parameter (see below).

You can create a new format set with the FormatEditor and start building your format defintions from scratch. But the easiest way is to open an existing format file for the Mil1394 Protocol and adapt it to your needs.

13.4.1. Example

The Mil1394FormatExample.dff file is an example file that shows an example of how to define formats for the Mil1394 Protocol. It can be found in the examples folder of your Analyzer install and it contains two format definitions as shown below.

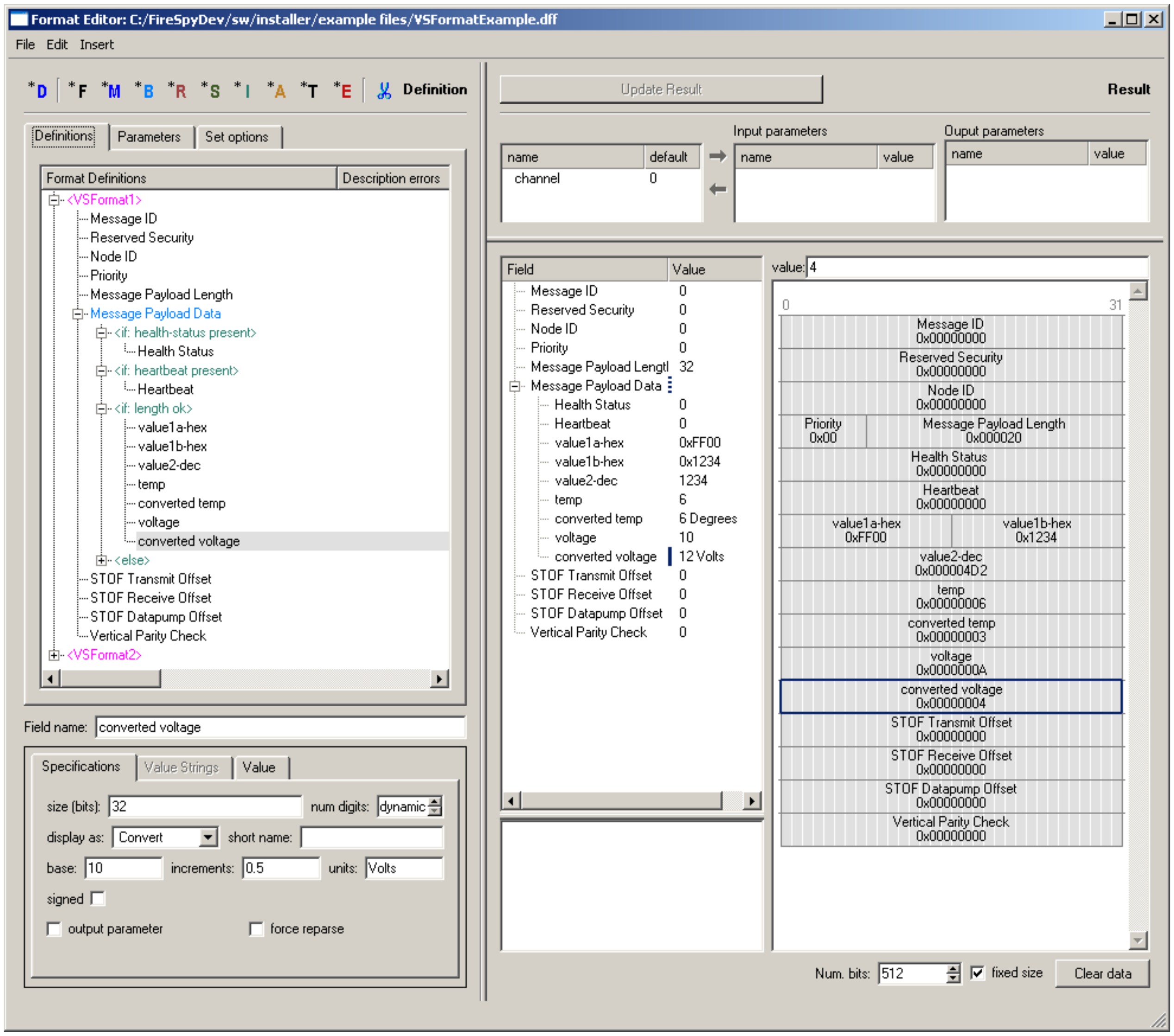


<VSFormat1>
<VSFormat2>

13.4.1.1. Format 1

The format definition 'Mil1394Format1' defines a simple format with a message payload length of 32 bytes. The message payload consists of the standard
'Health-status' and 'Heartbeat' plus 7 example fields.

The following image displaus the 'Mil1394Format1' definition in more detail.


Format Editor: C:/FireSpyDev/sw/installer/example files/VSFormatExample.dff
File Edit Insert
*D *F *M *B *R *S *I *A *T *E Definition
Update Result
Result
Definitions Parameters Set options
Specifications Value Strings Value
Clear data

Format Definitions | Description errors

```
<VSFormat1>
  Message ID
  Reserved Security
  Node ID
  Priority
  Message Payload Length
  Message Payload Data
    <if: health-status present>
      Health Status
    <if: heartbeat present>
      Heartbeat
    <if: length ok>
      value1a-hex
      value1b-hex
      value2-dec
      temp
      converted temp
      voltage
      converted voltage
    <else>
  STOF Transmit Offset
  STOF Receive Offset
  STOF Datapump Offset
  Vertical Parity Check
<VSFormat2>
```

Field name: converted voltage

size (bits): 32   num digits: dynamic
display as: Convert   short name:
base: 10   increments: 0.5   units: Volts
signed ☐
☐ output parameter   ☐ force reparse

Input parameters

name	default
channel	0

name	value

Ouput parameters

name	value

Field	Value
Message ID	0
Reserved Security	0
Node ID	0
Priority	0
Message Payload Lengtl	32
Message Payload Data	
Health Status	0
Heartbeat	0
value1a-hex	0xFF00
value1b-hex	0x1234
value2-dec	1234
temp	6
converted temp	6 Degrees
voltage	10
converted voltage	12 Volts
STOF Transmit Offset	0
STOF Receive Offset	0
STOF Datapump Offset	0
Vertical Parity Check	0

value: 4

```
0                                                   31
Message ID 0x00000000
Reserved Security 0x00000000
Node ID 0x00000000
Priority 0x00 | Message Payload Length 0x000020
Health Status 0x00000000
Heartbeat 0x00000000
value1a-hex 0xFF00 | value1b-hex 0x1234
value2-dec 0x000004D2
temp 0x00000006
converted temp 0x00000003
voltage 0x0000000A
converted voltage 0x00000004
STOF Transmit Offset 0x00000000
STOF Receive Offset 0x00000000
STOF Datapump Offset 0x00000000
Vertical Parity Check 0x00000000
```

Num. bits: 512 ☑ fixed size

Fields 'value1a-hex' and 'value1b-hex' are both 16 bits size and are displayed as hexadecimal numbers. The remaining message payload fields are all 32 bits size. The 'converted temp' and 'converted voltage' fields are of the 'Convert' type. This means that a conversion is done on the raw bits values before it is displayed.

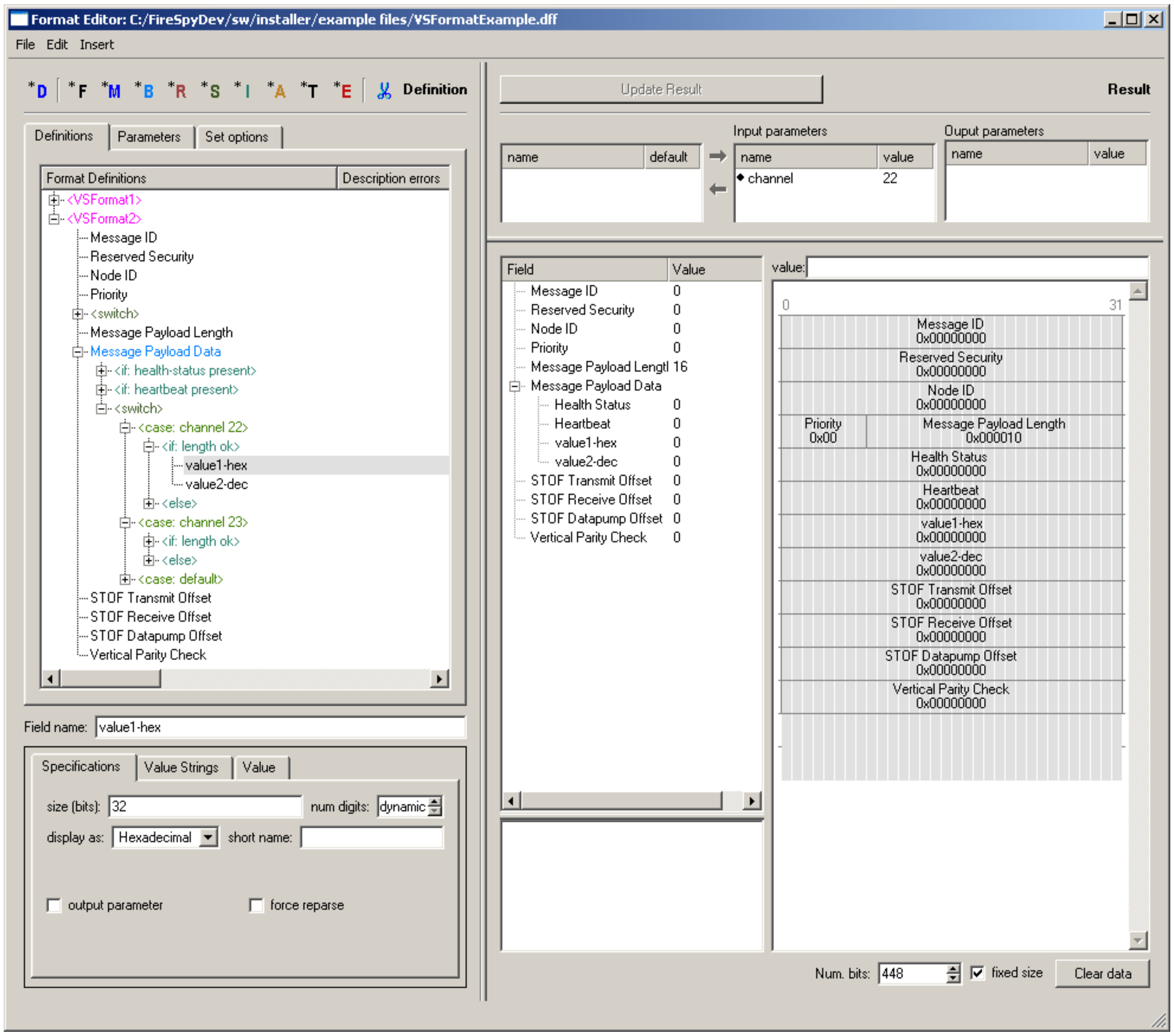
As you can see in the 'Result' pane, the 'converted voltage' field in the table view has value 12 Volts. But the stored value is in fact 4 as you can see in the layout view. This is the result of the automatic conversion. In the 'Definition' pane the associated 'converted voltage' item has been selected and thus the properties of this field are shown below the format defintions tree. There you can see that it is a 'Convert' type field with 'increments' equal to 0.5, which means that when the raw value is incrementd by one, the displayed value is incremented by 0.5. Furthermore, the base is 10, which means that if the raw value is 0, the displayed value is 10.

For the 'converted temp' an increment of 2 has been define, thus a raw value of 3 result in a displayed value of 6. Note that it is also possible to define a 'base' value. The' base' value is always added to the result value.

Note that the defintion checks for the correct 'Message Payload Length. If the length is not 32, then all the payload will be displayed as hexadecimal data values of 32 bits size.

13.4.1.2. Format 2

The format definition 'Mil1394Format2' defines a format for which the message payload format depends on the input parameter 'channel'. This is done using the 'switch' item with a 'case' for each supported channel number (plus a default 'case'). Below the 'Mil1394Format2' is shown in more detail.


Format Editor: C:/FireSpyDev/sw/installer/example files/VSFormatExample.dff
File Edit Insert
*D *F *M *B *R *S *I *A *T *E
Definition
Definitions
Parameters
Set options
Update Result
Result

Format Definitions | Description errors
- <VSFormat1>
- <VSFormat2>
  - Message ID
  - Reserved Security
  - Node ID
  - Priority
  - <switch>
  - Message Payload Length
  - Message Payload Data
    - <if: health-status present>
    - <if: heartbeat present>
    - <switch>
      - <case: channel 22>
        - <if: length ok>
          - value1-hex
          - value2-dec
        - <else>
      - <case: channel 23>
        - <if: length ok>
        - <else>
      - <case: default>
  - STOF Transmit Offset
  - STOF Receive Offset
  - STOF Datapump Offset
  - Vertical Parity Check

Field name: value1-hex

Specifications | Value Strings | Value

size (bits): 32 num digits: dynamic
display as: Hexadecimal short name:
output parameter force reparse

name	default

Input parameters

name	value
channel	22

Ouput parameters

name	value

Field	Value
Message ID	0
Reserved Security	0
Node ID	0
Priority	0
Message Payload Lengtl	16
Message Payload Data	
Health Status	0
Heartbeat	0
value1-hex	0
value2-dec	0
STOF Transmit Offset	0
STOF Receive Offset	0
STOF Datapump Offset	0
Vertical Parity Check	0

value:

```
0                                  31
Message ID 0x00000000
Reserved Security 0x00000000
Node ID 0x00000000
Priority 0x00 | Message Payload Length 0x000010
Health Status 0x00000000
Heartbeat 0x00000000
value1-hex 0x00000000
value2-dec 0x00000000
STOF Transmit Offset 0x00000000
STOF Receive Offset 0x00000000
STOF Datapump Offset 0x00000000
Vertical Parity Check 0x00000000
```

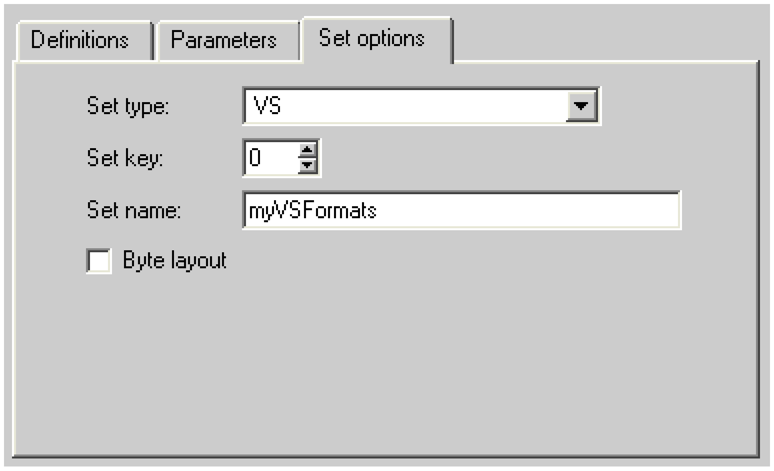
Num. bits: 448 ✓ fixed size Clear data

The 'switch' item is selected and we can see in the properties for this switch that it tests the input parameter 'channel'. If the channel number is 22, two (hexa)decimal fields are added. If the channel number is 23, four floating point values are added to the message payload.

In the 'Result' pane we filled in a value 22 for the 'channel' input parameter. As a result we see the format with two (hexa)decimal fields.

13.4.2. Set Options

To be able to use the format definitions in this format set for the Mil1394 Protocol you need to set the correct 'Set options'. Below the 'Set options' of the example file are shown.



Set type: VS

Set key: 0

Set name: myVSFormats

☐ Byte layout

The 'Set type' should be 'Mil1394'. The 'Set key' must be set to 0. The 'Set name' must be unique for all Mil1394 type files. So you should fill in some other name here.

When the Analyzer application finds two Mil1394 format sets with the same 'Set name' then the second one will be ignored. This is a valid way to overrule an existing format set.

See also Using Format Sets in the Analyzer Application.

13.5. XML Settings

The software can be set to automatically import a Mil1394 slashsheet specification from a single XML file. This way you can generate XML files from your signal specification database. It is also possible to implement security policies where certain people can only view certain parts of the format.

The Recorder, Signal Extractor, Scriptor and Generator automatically list or select the packet definitions derived from these signal definitions.

The XML file also contains settings related to the timing of frames, enumerations used for the signals, ASM header-, trailer- and STOF packet definitions, and the properties for the channels used.

Note: the Mil1394 XML settings import requires a Mil1394 protocol license and a main license version 8.7 or newer. Refer to the License Manager to learn how to view your license and how to get one.

13.5.1. How to use it

And explanation of XML can for instance be found here:

http://www.w3schools.com/xml/default.asp

The FireDiagnostics Suite comes with an example Mil1394 settings XML file. It is advised to edit this file to try out the structures that you wish to generate and familiarize yourself with the structure and possibilities. The format is described in full detail in the XML settings file format section. This section also explains how to validate the file, which should be done before starting the application when the contents have changed.

This feature must be enabled in the settings dialog before it can be used, as described in the Mil1394 settings page documentation.

The following sections describe the different parts in more detail.

13.5.2. Signal Definitions

In the Signal Monitor you can load a list of signals, as shown below.

Each signal is in fact a field within a packet, the ASM packet. Each signal is therefore to be interpreted from a certain number of bits, at a certain offset from the start of the packet; in number of 32 bit blocks called quadlets and possibly a number of additional, individual bits.

It is also linked to a channel and message ID; these are fields from the header of the packet. Their value selects which signal values may be expected to be contained in the packet's payload data, at their respective offsets. The bit values may also be interpreted as being signed or not, floating point or integer, and their value may be linked to an enumeration of strings.

This information can be embedded in the XML file as shown below:

```
<!-- Message Definitions -->
<MessageDefinitions>
        <MessageDefinition Channel="4" MessageID="6000004" PacketSize="320">
                <SignalGroup>
                        <Signal Name="engine1_heartbeat" DataType="udec"  QuadOffset="0"/>
                        <Signal Name="engine1_pressure"  DataType="float" QuadOffset="1"
                                Factor="2.5" Offset="-1.5" MinValue="-50" MaxValue="99.5"
                                Units="bar"/>
                </SignalGroup>
                <Signal Name="engine1_temperature"         DataType="dec"   QuadOffset="2"
                        FirstBit="16" NumBits="16"/>
        </MessageDefinition>

        <MessageDefinition Channel="5" MessageID="0">
                <SignalGroup>
                        <Signal Name="sensors_heartbeat" DataType="udec"  QuadOffset="0"/>
                        <Signal Name="sensors_sensor1"   DataType="dec"   QuadOffset="2"
                                EnumName="switch"/>
                </SignalGroup>
        </MessageDefinition>
</MessageDefinitions>
```

The XML settings file was introduced to allow you to automatically import signal definitions from an existing database, since this format is relatively easy to generate and understand. You may generate multiple files for different usages and security levels. Undefined signals will simply be shown as such, although the values themselves will still be accessible. For instance, the same recording file may show different levels of detail on several computers; each set up with a different Mil1394 XML settings file.

Enumeration strings
The example above contains a signal called "sensors_sensor1", which (for the sake of illustration) refers to the "switch" enumeration. This enumeration, defined below, links the value 0 to "off", 1 to "on", and any value between 2 and 255 (inclusive) to "unspecified". The application uses this hint on several places to translate the signal's value to the corresponding string (using a drop down box), and vice versa.

```
<!-- String Representations -->
<Enums>
        <Enum Name="switch">
                <Item Value="0" Label="off"/>
                <Item Value="1" Label="on"/>
                <Range Start="2" End="255" Label="unspecified"/>
        </Enum>
</Enums>
```

13.5.3. Packet Formats

Besides being able to specify the signal data payload of the ASM message packet, it is also possible to specify how its header and trailer are to be interpreted. And which signals should be included by default, regardless of the channel and message ID. Similarly, the STOF packet can be precisely specified. The syntax of the field definitions is similar to the signal definitions. See the Mil1394 Settings File Format for more information.

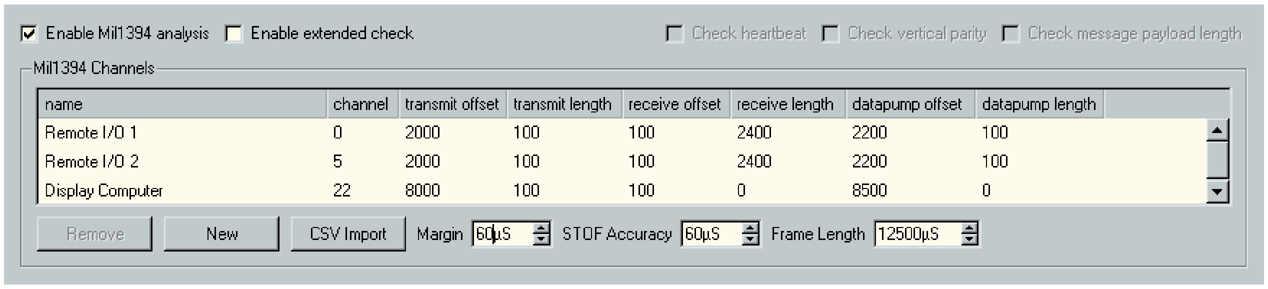
```
<!-- Packet Formats -->
<ASMPacketFormat>
        <ASMHeader>
                <MessageID                QuadOffset="0" FirstBit="0" NumBits="32"
                                          DataType="udec">
                        <DecimalParts>
                                <Branch    NumDigits="2"/>
                                <Channel   NumDigits="2"/>
                                <Message   NumDigits="6"/>
                </DecimalParts>
        </MessageID>

        <Field     Name="Reserved" QuadOffset="1"  FirstBit="0"  NumBits="32"
                                   DataType="hex"/>
        <NodeID                    QuadOffset="2"  FirstBit="0"  NumBits="32"
                                   DataType="udec"/>
        <Priority                  QuadOffset="3"  FirstBit="24" NumBits="8"
                                   DataType="udec"/>
        <MessagePayloadDataLength QuadOffset="3"  FirstBit="0"  NumBits="24"
                                   DataType="udec"/>
    </ASMHeader>

    <PayloadData>
        <HealthStatus              QuadOffset="0"  FirstBit="0"  NumBits="32"
                                   DataType="hex">

                <Port Name="Port 2">
                        <Speed                  FirstBit="2"  NumBits="3"
                                DataType="udec" EnumName="SPEED"/>
                        <BetaMode               FirstBit="5"  NumBits="1"
                                DataType="udec" EnumName="bool"/>
                        <ReceiveOK              FirstBit="6"  NumBits="1"
                                DataType="udec" EnumName="bool"/>
                        <Connected              FirstBit="7"  NumBits="1"
                                DataType="udec" EnumName="bool"/>
                </Port>

                <Port Name="Port 1">
                        <Speed                  FirstBit="10" NumBits="3"
                                DataType="udec" EnumName="SPEED"/>
                        <BetaMode               FirstBit="13" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                        <ReceiveOK              FirstBit="14" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                        <Connected              FirstBit="15" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                </Port>

                <Port Name="Port 0">
                        <Speed                  FirstBit="18" NumBits="3"
                                DataType="udec" EnumName="SPEED"/>
                        <BetaMode               FirstBit="21" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                        <ReceiveOK              FirstBit="22" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                        <Connected              FirstBit="23" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                </Port>

                <NodeError                      FirstBit="29" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                <SubsystemError                 FirstBit="30" NumBits="1"
                                DataType="udec" EnumName="bool"/>
                <PacketError                    FirstBit="31" NumBits="1"
                                DataType="udec" EnumName="bool"/>
        </HealthStatus>
        <Heartbeat                 QuadOffset="1" FirstBit="0"  NumBits="32"
                                   DataType="udec"/>
        <MessageData               QuadOffset="2"/>
    </PayloadData>

    <PacketTrailer>
        <STOFTransmitOffset        QuadOffset="0" FirstBit="0"  NumBits="32"
                                   DataType="udec"/>
        <STOFReceiveOffset         QuadOffset="1" FirstBit="0"  NumBits="32"
                                   DataType="udec"/>
        <STOFDatapumpOffset        QuadOffset="2" FirstBit="0"  NumBits="32"
                                   DataType="udec"/>
        <VerticalParityCheck       QuadOffset="3" FirstBit="0"  NumBits="32"
                                   DataType="hex"/>
    </PacketTrailer>
</ASMPacketFormat>
```

13.5.4. Channel Lists and Frame Timing

Mil1394 may use per-channel properties, such as the heartbeat and a short descriptive label. The transmit, receive and datapump frame start and end times may be defined within which the messages of that channel should be sent or received. These channel definitions are also automatically inserted as (disabled/hidden) slots in a new Generator document.

```
<!-- Pre-Assigned Channels -->
<ChannelList>
        <Channel ID="4" Heartbeat="80" DeviceName="Remote I/O 1">
                <Transmit Offset="2000" Length="100"/>
                <Receive  Offset="2200" Length="100"/>
                <DataPump Offset="2400" Length="100"/>
        </Channel>
        <Channel ID="5" Heartbeat="80" DeviceName="Remote I/O 2">
                <Transmit Offset="2000" Length="100"/>
                <Receive  Offset="2200" Length="100"/>
                <DataPump Offset="2400" Length="100"/>
        </Channel>
        <Channel ID="22" Heartbeat="80" DeviceName="Display Computer">
                <Transmit Offset="8000" Length="100"/>
                <Receive  Offset="8500" Length="100"/>
                <DataPump Offset="0"    Length="0"/>
        </Channel>
</ChannelList>
```

The list loaded from the XML file occurs for instance in the recorder's protocol settings:



[✓] Enable Mil1394 analysis [ ] Enable extended check [ ] Check heartbeat [ ] Check vertical parity [ ] Check message payload length

Mil1394 Channels

name	channel	transmit offset	transmit length	receive offset	receive length	datapump offset	datapump length
Remote I/O 1	0	2000	100	100	2400	2200	100
Remote I/O 2	5	2000	100	100	2400	2200	100
Display Computer	22	8000	100	100	0	8500	0

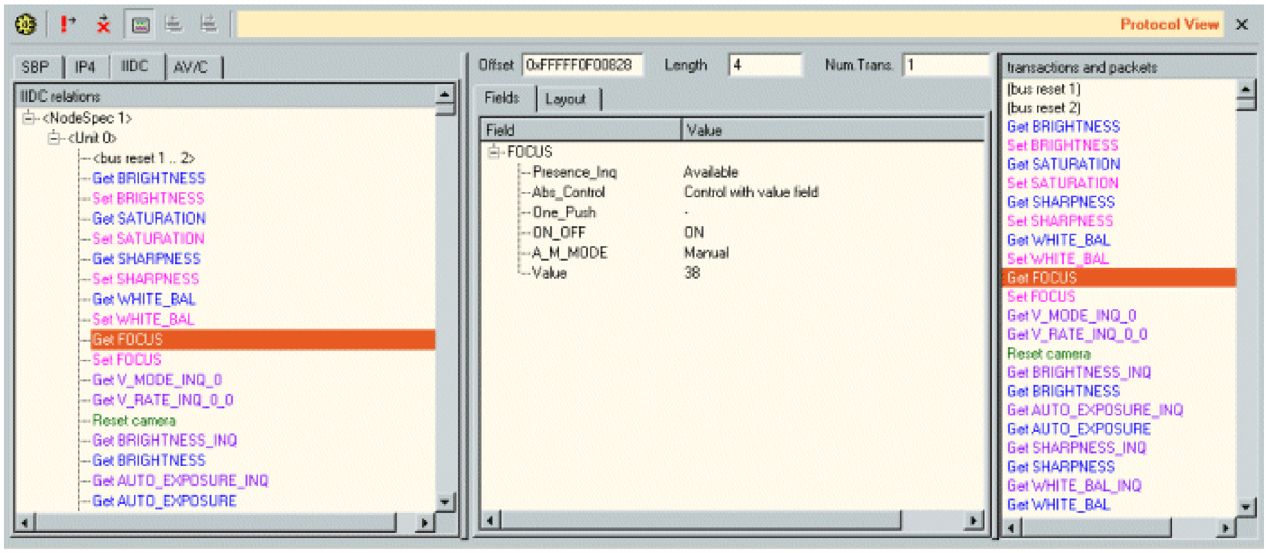
Margin 60µS STOF Accuracy 60µS Frame Length 12500µS

Note that you can also set the frame length, the frame length margin (i.e. how many microseconds the packet may arrive early or late) and the STOF packet accuracy. The latter is the amount of microseconds the next STOF packet may arrive early or late, in order not to be marked as invalid, relative to the prior one. These settings can also be defined in the XML file:

```
<!-- Global Settings -->
<Properties
        Name="SAE Example"

        FrameLength="12500"
        STOFAccuracy="250"
        ASMPacketAccuracy="25"
/>
```

Chapter 14. Industrial / Instrumentation Digital Camera protocol (IIDC)

One of the protocols supported by the Protocol View of the Recorder is the Industrial / Instrumentation Digital Camera protocol (IIDC) protocol. The IIDC-protocol analyzer software will scan through all transactions and packets to find IIDC-compliant Units and it will fill the 'relations' pane in the IIDC tab page with IIDC-related items, as described in Protocol View and Protocol Settings. An example of the resulting Protocol View is displayed below.



IIDC relations

- <NodeSpec 1>
  - <Unit 0>
    - <bus reset 1 .. 2>
    - Get BRIGHTNESS
    - Set BRIGHTNESS
    - Get SATURATION
    - Set SATURATION
    - Get SHARPNESS
    - Set SHARPNESS
    - Get WHITE_BAL
    - Set WHITE_BAL
    - Get FOCUS
    - Set FOCUS
    - Get V_MODE_INQ_0
    - Get V_RATE_INQ_0_0
    - Reset camera
    - Get BRIGHTNESS_INQ
    - Get BRIGHTNESS
    - Get AUTO_EXPOSURE_INQ
    - Get AUTO_EXPOSURE

Offset 0xFFFFF0F00828 Length 4 Num Trans. 1

Field	Value
FOCUS	
Presence_Inq	Available
Abs_Control	Control with value field
One_Push	-
ON_OFF	ON
A_M_MODE	Manual
Value	38

transactions and packets

|bus reset 1|
|bus reset 2|
Get BRIGHTNESS
Set BRIGHTNESS
Get SATURATION
Set SATURATION
Get SHARPNESS
Set SHARPNESS
Get WHITE_BAL
Set WHITE_BAL
Get FOCUS
Set FOCUS
Get V_MODE_INQ_0
Get V_RATE_INQ_0_0
Reset camera
Get BRIGHTNESS_INQ
Get BRIGHTNESS
Get AUTO_EXPOSURE_INQ
Get AUTO_EXPOSURE
Get SHARPNESS_INQ
Get SHARPNESS
Get WHITE_BAL_INQ
Get WHITE_BAL

It displays the result in three different parts. On the right (transactions and packets), all packets and transactions found are displayed in the order they were detected on the bus. The left part (relations) shows the relation between these items in a hierarchical tree. The transactions and packets belonging together are grouped into tree items and these items are displayed hierarchically in the 'relations' pane. In this relations tree you will find for instance:

- Inquiry register reads
- Status-register and control-register reads and writes
- Camera control
- Video frame data
- Bus resets
- Channel allocations and de-allocations

The details of the item selected in the relations tree, are displayed in the middle pane of the Protocol View. It displays the fields defined for the selected item and the values of those fields.

14.1. How to use it

To display the IIDC results, select the IIDC tab page in the Protocol View of the Recorder. The Protocol View can be displayed by selecting it in the 'View' menu of the Recorder.

IMPORTANT:
You will need an IIDC-Protocol license key to be able to select this view. However, if you open a Recorder file that was made with an Analyzer with a valid IIDC license key installed, the Protocol View can be enabled, even if you do not have a valid IIDC license key yourself. For more information about license keys, click here.
The IIDC-protocol analyzer needs some information from the Configuration ROM of a IIDC device, to be able to analyze the IIDC transactions and packets corresponding to this device. It might also need to know some other settings of the IIDC unit to be able to properly analyze IIDC data. There are two ways the analyzer can get this information: automatically or manually.

Automatically finding IIDC information

To make sure the IIDC analyzer finds the IIDC information automatically, you should make sure that the reading of Configuration ROM for the node id of the unit is recorded. The Configuration-ROM information is normally read after a bus reset. One way to do this is to (re)connect the IIDC device while the Recorder is recording data. You should also take care that this recorded information is not removed from the recorder buffer because of recorder-buffer overflow (cyclic recorder buffer). One way to do this is to stop the recorder before the cyclic buffer (the buffer part before the trigger position) fills completely, or by generating a trigger before this part is filled completely (see the Recorder for more information).

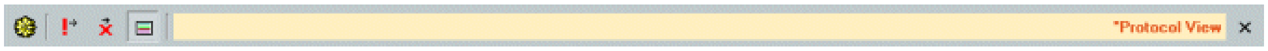
Manually inputting IIDC information
If no Configuration ROM reads are recorded, you will need to input this information manually. Also, if only part of this information can be found automatically, you will need to add information manually. You can do this in the 'Protocol Settings' dialog. It can be opened by clicking the 'Protocol Settings' button on the toolbar (see below). This dialog shows information used by the all-protocol analyzers. It initially shows all information that it found automatically and you can change or add information manually. See the description of the 'IIDC-Protocol Settings' below for more information.

14.2. Details

The Protocol View consists of a toolbar on the top, with three parts below it: the 'relations' pane, the details pane and the 'transactions and packets' pane. For IIDC the details pane can display an extra Tab named 'Video frame'. An extra popup window can also be used to view and play back recorded video frames.

14.2.1. Toolbar

At the top of the Protocol View you will find a toolbar with a few buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol-analysis settings have been modified.

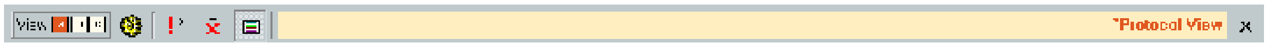
Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See 'IIDC-Protocol Settings' below for more information.

Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane at the right side of the Protocol View.

Triple Analyzer
For triple Analyzers the toolbar looks as follows:



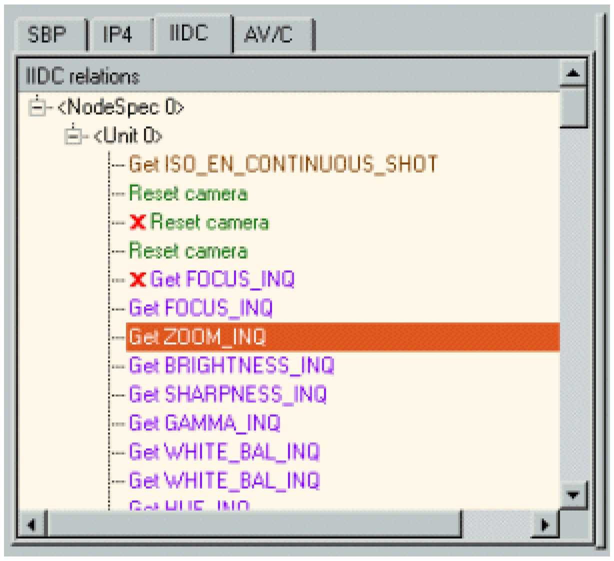
It contains the following additional control:

Analyzer node select control

With this control you are able to select the protocol analysis result of each supported Analyzer node.

14.2.2. Relations Pane

This pane shows all found transactions and packets, grouped into items and displayed in a tree. The tree structure indicates the relation between the items. The items are not necessarily recorded in the same order as displayed in this tree. However, all items inside the same item of the tree (at the same level) are listed in the same order as the first corresponding packet of each item was recorded. To find out the order of corresponding transactions and packets, see the 'transactions and packets' pane or take a look at one of the other views of the Recorder (e.g. the Transactions View).



```
<NodeSpec 0>
  <Unit 0>
    Get ISO_EN_CONTINUOUS_SHOT
    Reset camera
    X Reset camera
    Reset camera
    X Get FOCUS_INQ
    Get FOCUS_INQ
    Get ZOOM_INQ
    Get BRIGHTNESS_INQ
    Get SHARPNESS_INQ
    Get GAMMA_INQ
    Get WHITE_BAL_INQ
    Get WHITE_BAL_INQ
    Get HUE_INQ
```

For each found or manually specified NodeSpec there will be a root item in the tree (NodeSpec 0 above). For information about NodeSpec's, see 'IIDC-Protocol Settings' below. For each found or manually specified Unit, there will be a Unit item inside the corresponding NodeSpec (Unit 0 in NodeSpec 0 above).

A black item indicates that it has no corresponding transactions or packets (e.g. NodeSpec and Unit above). It just is used for proper grouping of the items in the tree. All colored items however correspond to one or more transactions or packets. When selecting such an item, the details of that item are displayed in the 'details pane' and the corresponding transactions and packets are highlighted in the 'transactions and packets' pane (see below). Note that bus resets like '' are also displayed in black, but have a corresponding item in the 'transactions and packets' pane.

The type of items that can be displayed are:

- Node: displays a 1394 node with IIDC units inside that node below it. Each node gets a logical node id, which is fixed for the entire recording. This logical node id is called a NodeSpec and is displayed in the tree.
- Unit: displays a IIDC unit inside a 1394 node. The tree shows the logical unit number.
- Feature register reads or writes.
- Status and control register reads of writes.
- Read of inquiry registers.
- Video frame.
- Channel allocation and de-allocation.
- Bandwidth allocation and de-allocation.

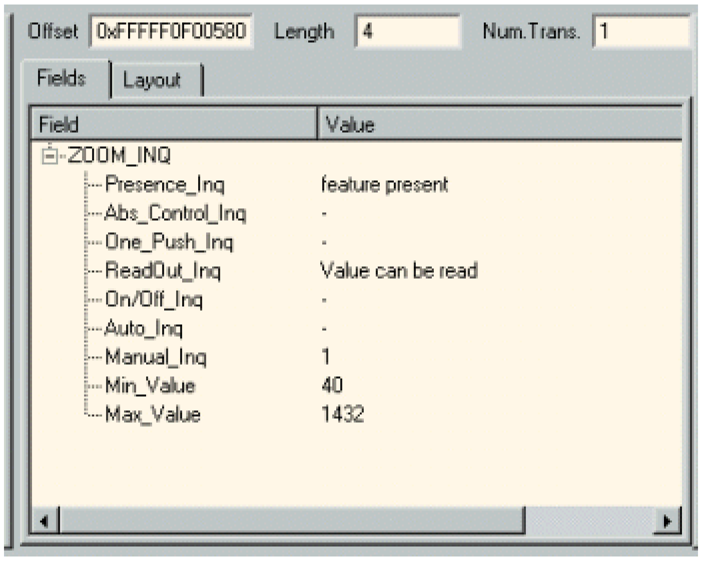
For each item in the tree two extra columns indicate the source and destination NodeSpec ID.
The following color scheme is used for these items:

- item group
- write of a feature
- read of a feature

- write of status and control register
- read of status and control
- read of inquiry
- video frame
- channel allocation and de-allocation
- Bandwidth allocation and de-allocation

14.2.3. Details Pane

The middle part of the Protocol View shows the details of the item that is selected in the 'relations' pane. An example that shows the details of a 'Get ZOOM_INQ' item follows:



Offset 0xFFFFF0F00580 Length 4 Num.Trans. 1

Field	Value
ZOOM_INQ	
Presence_Inq	feature present
Abs_Control_Inq	-
One_Push_Inq	-
ReadOut_Inq	Value can be read
On/Off_Inq	-
Auto_Inq	-
Manual_Inq	1
Min_Value	40
Max_Value	1432

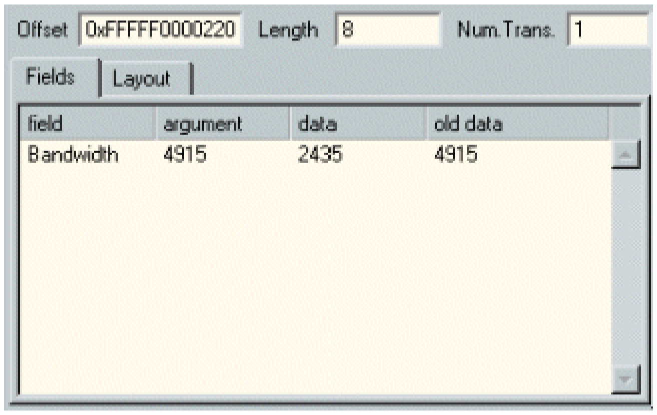
We will describe the different parts below:

Offset
This box displays the address (offset) of the start of the selected item.

Length
The 'Length' box shows the size of the selected item in bytes.

Num. Trans.
This box shows the number of transactions that corresponds to the selected item. For IIDC this will always be 1.

Fields
The 'Fields' table shows the fields that are defined for the selected item and the corresponding values of these fields. See example above.
If a Bandwidth allocation/de-allocation or Channel allocation/de-allocation or Plug register transaction is selected, the 'Fields' page displays the data corresponding to a lock transaction. For lock transactions, the 'Argument', 'Data' and 'Old data' are shown. See below for an example of the fields for a bandwith-allocation lock transaction.



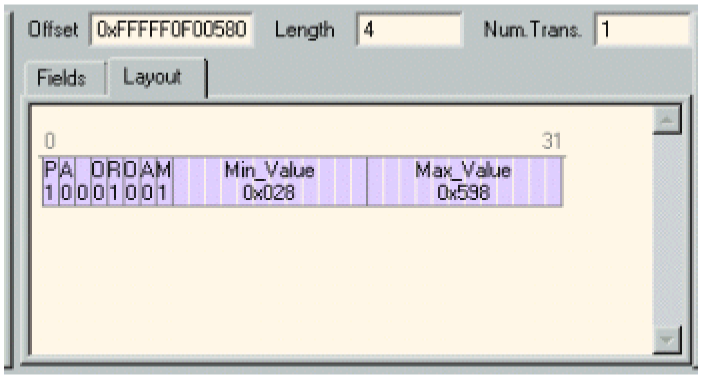
Offset 0xFFFFF0000220 Length 8 Num.Trans. 1

Fields | Layout

field	argument	data	old data
Bandwidth	4915	2435	4915

Layout

The 'Layout' shows the layout of the selected item. Like the 'Fields' table it shows the fields that are defined for the selected item and the corresponding values of these fields. But now the fields are displayed graphically inside the data item. See below for an example of the layout view for the same item as shown in the example above.

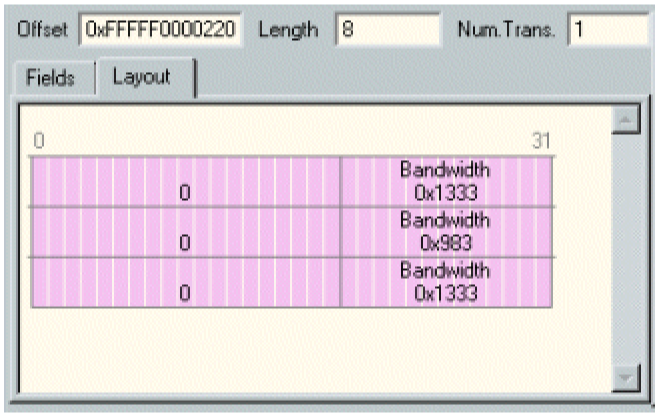


Offset 0xFFFFF0F00580 Length 4 Num.Trans. 1

Fields | Layout

P	A	O	R	O	A	M	Min_Value	Max_Value
1	0	0	0	1	0	0	1	0x028	0x598

If a Bandwidth allocation/de-allocation or Channel allocation/de-allocation or Plug register transaction is selected, the 'Layout' page displays the data corresponding to a lock transaction. For lock transactions, the 'Argument' layout is drawn on the top, the 'Data' layout in the middle, and the 'Old data' layout at the bottom. See below for an example of the layout for a bandwith allocation lock transaction.



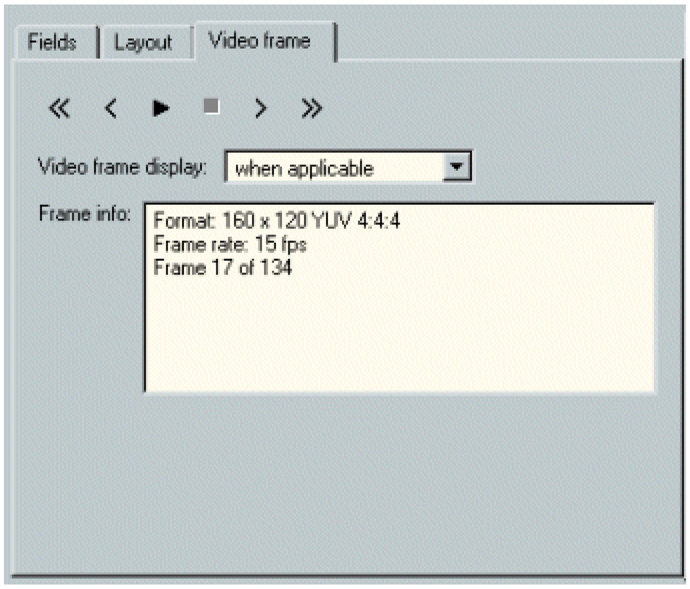
Offset 0xFFFFF0000220 Length 8 Num.Trans. 1

Fields | Layout

	Bandwidth
0	0x1333
0	0x983
0	0x1333

Video frame

This tab is displayed when an IIDC item is selected in the tree that contains video data:



when applicable

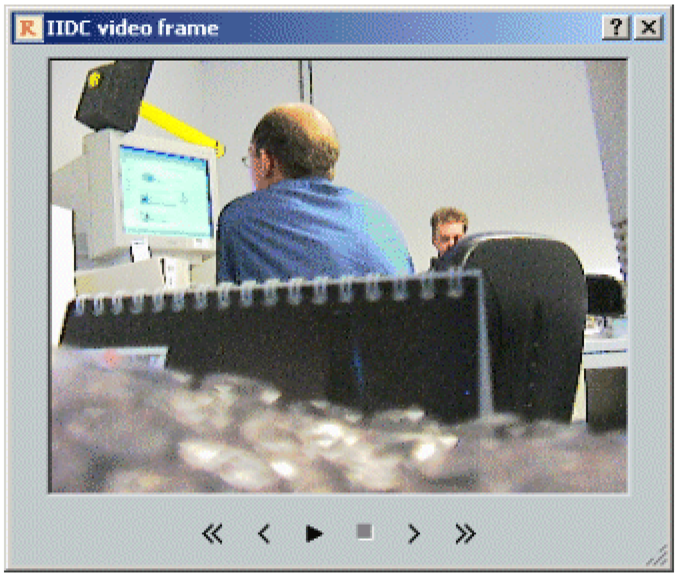
Frame info:

Format: 160 x 120 YUV 4:4:4
Frame rate: 15 fps
Frame 17 of 134

The tab displays information about the selected video frame and contains control buttons to select other video frames and to play back a sequence of video frames. The selected video frame is also displayed in a separate popup window shown below. The tab has a combo box that can be used to determine when the video window will be displayed. The possible choices are:

- when applicable: the video window is shown when a packet is selected that is part of a video frame and hidden when this is not the case. This is the default setting.
- hide: the video window is never displayed.
- always: the video window is always visible even when a packet is selected that is not part of a video frame.

Below, an example of the video window is shown.



The buttons on both the tab and the video window, control the same data and have the following meaning:

«

First frame
Clicking this button will select the first packet of all video frames sent by the unit.

‹

Previous frame
Clicking this button will select the first packet of the video frame before the selected frame.

Playback video
Clicking this button will play back the video frames starting at the selected frame. It tries to play the video back at the recorded rate, but depending on the format and speed the actual playback rate might be lower.

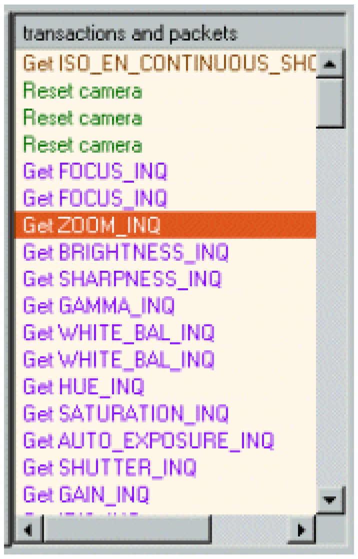
Stop playback
Clicking this button will stop the playback of video frames.

Next frame
Clicking this button will select the first packet of the video frame after the selected frame.

Last frame
Clicking this button will select the first packet of the last video frame sent by the unit.
Note: when the video window has been closed by clicking on the close button, it will remain hidden until it is reopened by selecting a different value for the 'video frame display' combo box in the video-frame tab.

14.2.4. Transactions and packets pane

The right-most part of the Protocol View shows the sequence of all found protocol packets and transactions (including those for IIDC). The order of items in this list corresponds to the order of recording. It is the same order as displayed in the Transaction View or Packets View of the Recorder. An example follows:



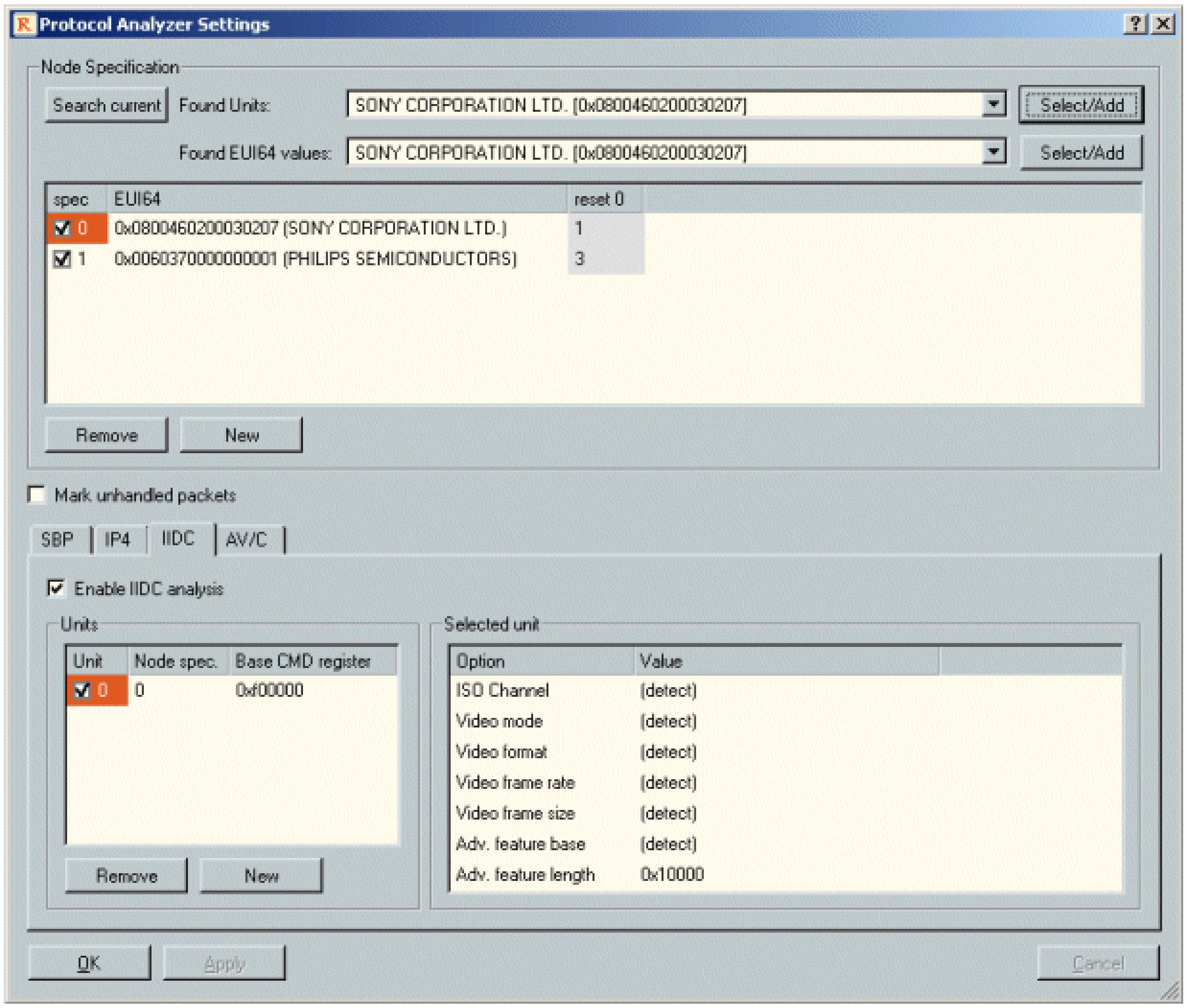
transactions and packets

Get ISO_EN_CONTINUOUS_SHO
Reset camera
Reset camera
Reset camera
Get FOCUS_INQ
Get FOCUS_INQ
Get ZOOM_INQ
Get BRIGHTNESS_INQ
Get SHARPNESS_INQ
Get GAMMA_INQ
Get WHITE_BAL_INQ
Get WHITE_BAL_INQ
Get HUE_INQ
Get SATURATION_INQ
Get AUTO_EXPOSURE_INQ
Get SHUTTER_INQ
Get GAIN_INQ

Note that the colors identify the type of access and are the same as used in the 'relations' pane (see above). All transactions and packets in this list that are not part of the protocol currently selected in the 'relation' pane are colored gray.
When an item is selected that consists of more than one packet or transaction, the items are highlighted in the 'transactions and packets' pane list with a gray bar (the selected item, which is also part of the item, will still be highlighted with the usual selection color).

14.2.5. IIDC Protocol Settings

When clicking the 'Protocol Settings' button in the toolbar of the Protocol View, the 'Protocol Settings' dialog will be displayed. This dialog shows information used by the protocol analyzers to find and correctly analyze the transactions. For each supported protocol, including IIDC, there is a tab page with protocol-specific settings. The dialog will be initialized with all information that can be found automatically. If this is not enough for a correct analysis (e.g. the IIDC unit's base command address was not recorded), you can add this information manually. An example of the dialog is shown below.



Found Units: SONY CORPORATION LTD. [0x0800460200030207]

Found EUI64 values: SONY CORPORATION LTD. [0x0800460200030207]

spec	EUI64	reset 0
☑ 0	0x0800460200030207 (SONY CORPORATION LTD.)	1
☑ 1	0x0060370000000001 (PHILIPS SEMICONDUCTORS)	3

☐ Mark unhandled packets

☑ Enable IIDC analysis

Unit	Node spec.	Base CMD register
☑ 0	0	0xf00000

Option	Value
ISO Channel	(detect)
Video mode	(detect)
Video format	(detect)
Video frame rate	(detect)
Video frame size	(detect)
Adv. feature base	(detect)
Adv. feature length	0x10000

In the dialog you see an upper part to specify one or more 'Node Specifications', and a lower part to specify protocol-specific information. In between these two parts, general options can be selected.
For a correct analysis the Unit information of the IIDC device to be analyzed must be present. To be able to specify a Unit, you will first need to specify the device (Node Specification) the Unit is part of.
For more information about Node Specifications and the protocol independent part of the Protocol Settings, see Protocol Settings. For the IIDC protocol specific settings, see below.

Node Specifications

If no IIDC Units were found automatically, you have to specify one or more Units manually. Before you can specify a Unit you have to specify the node the Unit is part of, using a Node Specification. When the IIDC device is still connected to the bus after recording, you can use the 'Search Current' button to automatically find IIDC Units. Adding such a Unit will create a Node Specification and Unit. The created Node Specification will be filled automatically when possible. You probably will have to select correct nodes for some reset segments. Make sure you have selected the IIDC node corrrectly for those reset segments for which transaction and packets are recorded that you want to analyze. For more information on Node Specifications, see Protocol Settings.

General options

Mark unhandled packets
For information about this checkbox, please see Protocol Settings.

Protocol-specific information

Enable IIDC analysis
Check this box to enable the IIDC analyzer. All protocol tabs in this dialog have such a checkbox, which can be useful if a recording contains data from several protocols, and you're not interested in some of these.

Units
In the Units table, one or more IIDC Units can be specified. The information in this table normally can be found in the Configuration ROM of the IIDC node. It consists of:

- *Unit:* Using the checkbox before each Unit in the table you can enable or disable each Unit individually for analysis. If the box is not checked, the IIDC-related transactions and packets for this Unit will be skipped.
- *Node Spec.*: A Unit is part of a node. As explained before, nodes are specified using Node Specifications. The number in this column corresponds to the Node Specification numbers of the table in the upper part of the dialog.
- *Base CMD register:* This is the base command register specified as an offset from the IIDC start address defined as Bus_ID, Node_ID , FFFF Fxxx xxxx. Normally, this is read from the Configuration ROM in which case the default value '(detect)' from the combo box can be used. If this information is missing the protocol analyzer can't do much, so it will need to be entered manually.

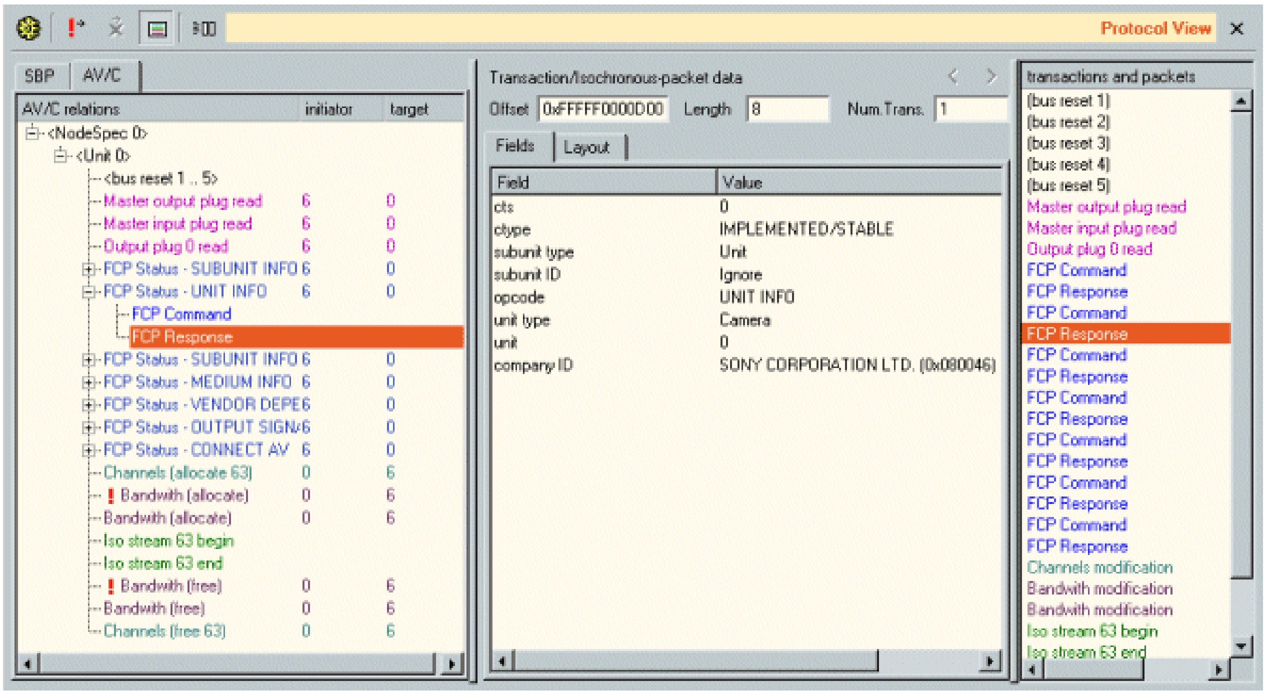
Extra IIDC settings
Most information the analyzer needs can be retrieved from the recorded packets. In some cases it might be necessary to manually specify certain values for an IIDC unit. The settings dialog shows the extra values for the selected unit in a group box that is labeled 'Selected unit'. The following values can be specified in that table:

- *ISO Channel:* Indicates the ISO Channel used by the unit to transmit video frame data. A value can be supplied here in case a recording has been made not containing the information (write of register 'ISO'). If a value is supplied the analyzer assumes video frame data is already transmitting on the indicated channel at the start of the recording. The analyzer will probably also need to know something about the mode and format of this data, so the next four options allow manual specifications of this as well.
- *Video mode:* Indicates the video mode used for the video frame-data transmitted. Selecting an option here determines which options are allowed and displayed for the next two values 'Video format' and 'Video frame rate'.
- *Video format:* Indicates the video format used for the video frame-data transmitted. The combo box will only show allowed values for the above selected 'Video mode'. Selecting an option here determines which options are allowed and displayed for the next value 'Video frame rate'.
- *Video frame rate:* Indicates the frame rate used for the video frame-data transmitted. The combo box will only show allowed values for the combination of the above selected value for 'Video mode' and 'Video format'. If all these 3 values have been entered, the next option ' Video frame size' will show the size of a single video frame packet.
- *Video frame size:* This displays the frame size (1394 bus packet size excluding header) for the video frame-data transmitted. It is fully determined by the choices made for the values of 'Video mode', 'Video format' and 'Video frame rate' described above. As an alternative to selecting values for these three options, it's sometimes easier to use this combo box to select the right value from all possible frame sizes. The combo box will show a list of all frame sizes including the mode and format that are possible taking into consideration the choices already made for 'Video mode', 'Video format' and 'Video frame rate'. So if all these three values are set to '(detect)', this combo box will show all possible packet sizes. If only a 'Video mode' has been selected, all possible frame sizes for any format and frame-rate combination for the selected video mode will be shown. If a value is chosen in this combo box, 'Video mode', 'Video format' and 'Video frame rate' will be updated to reflect that choice.
- *Adv. feature base:* The IIDC-protocol analyzer monitors reads of the register containing the base address of the advanced-feature registers. If a recording does not contain reads of that base address register, this option can be used to specify the base address of the advanced-feature registers, once known the analyzer can detect and display reads and writes of the advanced-feature registers.
- *Adv. feature length:* Given a base address offset of the advanced-feature registers, the IIDC-protocol analyzer needs to know the size of the area of these registers. Since this value is not specified in the standard nor can be found monitoring reads of an IIDC unit, it needs to be specified manually. This option can be used to change the range of registers monitored once a base address of the advanced-feature registers is known. A default value of 0x10000 will be used.

Chapter 15. Audio Video / Control protocol (AV/C)

15.1. Recorder Protocol View

One of the protocols supported by the Protocol View of the Recorder is the **Audio Video / Control (AV/C) protocol**. The AV/C protocol-analyzer software will scan through all transactions and packets to find AV/C-compliant Units and it will fill the 'relations' pane in the AV/C tab-page with AV/C-related items (including streams according to IEC 61883), as described in Protocol View and Protocol Settings. An example of the resulting Protocol View is displayed below.



Protocol View

SBP | AV/C

AV/C relations	initiator	target
<NodeSpec 0>		
<Unit 0>		
<bus reset 1 .. 5>		
Master output plug read	6	0
Master input plug read	6	0
Output plug 0 read	6	0
FCP Status - SUBUNIT INFO	6	0
FCP Status - UNIT INFO	6	0
FCP Command		
FCP Response		
FCP Status - SUBUNIT INFO	6	0
FCP Status - MEDIUM INFO	6	0
FCP Status - VENDOR DEPE	6	0
FCP Status - OUTPUT SIGN	6	0
FCP Status - CONNECT AV	6	0
Channels (allocate 63)	0	6
! Bandwith (allocate)	0	6
Bandwith (allocate)	0	6
Iso stream 63 begin		
Iso stream 63 end		
! Bandwith (free)	0	6
Bandwith (free)	0	6
Channels (free 63)	0	6

Transaction/Isochronous-packet data

Offset 0xFFFFF0000D00 Length 8 Num.Trans. 1

Fields | Layout

Field	Value
cts	0
ctype	IMPLEMENTED/STABLE
subunit type	Unit
subunit ID	Ignore
opcode	UNIT INFO
unit type	Camera
unit	0
company ID	SONY CORPORATION LTD. (0x080046)

transactions and packets

(bus reset 1)
(bus reset 2)
(bus reset 3)
(bus reset 4)
(bus reset 5)
Master output plug read
Master input plug read
Output plug 0 read
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
Channels modification
Bandwith modification
Bandwith modification
Iso stream 63 begin
Iso stream 63 end

It displays the result in three different parts. At the right (transactions and packets), all found packets and transactions are displayed in the order they were detected on the bus. The left part (relations) shows the relation between these items. The transactions and packets belonging together (e.g. all transactions that form a single FCP command) are grouped into tree items and these items are displayed hierarchically in the 'relations' pane. In this tree you will find for instance:

- FCP commands and responses
- Channel allocation and de-allocation
- Bandwidth allocation and de-allocation
- Plug register transactions
- Isochronous-stream start/stop events
- Bus resets

The details of the item selected in the relations tree, are displayed in the middle part of the Protocol View. For the AV/C protocol it is called the 'Transaction/Isochronous-packet data'. It displays the fields defined for the selected item and the values of those fields. The details of commands and responses are formatted according to the specifications defined for the corresponding unit if that unit is supported by the Analyzer. The following units are supported, unless information of descriptors is needed for the formatting of the commands or responses (descriptors not yet supported):

- AV/C Digital Interface Command Set General Specification Version 4.0
- AV/C Connection and Compatibility Management Specification 1.0
- AV/C Monitor Subunit Model and Command Set Version 1.0
- AV/C Audio Subunit Specification 1.0 AV/C Printer Subunit Specification 1.0
- AV/C Tape Recorder/Player Subunit Specification - version 2.1

- AV/C Tuner Subunit Model and Command Set version 2.0, including:
  - AV/C Tuner Broadcast System Specification - Analog Video - version 1.0
  - AV/C Tuner Broadcast System Specification - Analog Audio version 1.1
  - AV/C Tuner Broadcast System Specification - Digital Video Broadcast (DVB) - version 1.0
  - AV/C Tuner Broadcast System Specification - ATSC Digital Television System (DTV)
  - AV/C Tuner Broadcast System Specification - Rec. ITU-R BO. 1294 System B
- AV/C CA Subunit Specification - version 1.0

Optionally, a fourth part can be toggled on and off by clicking on the 'Toggle source-display on/off' button in the toolbar (see below).

15.1.1. How to use it

To display the AV/C results, select the AV/C tab page in the Protocol View of the Recorder. The Protocol View can be displayed by selecting it in the 'View' menu of the Recorder.

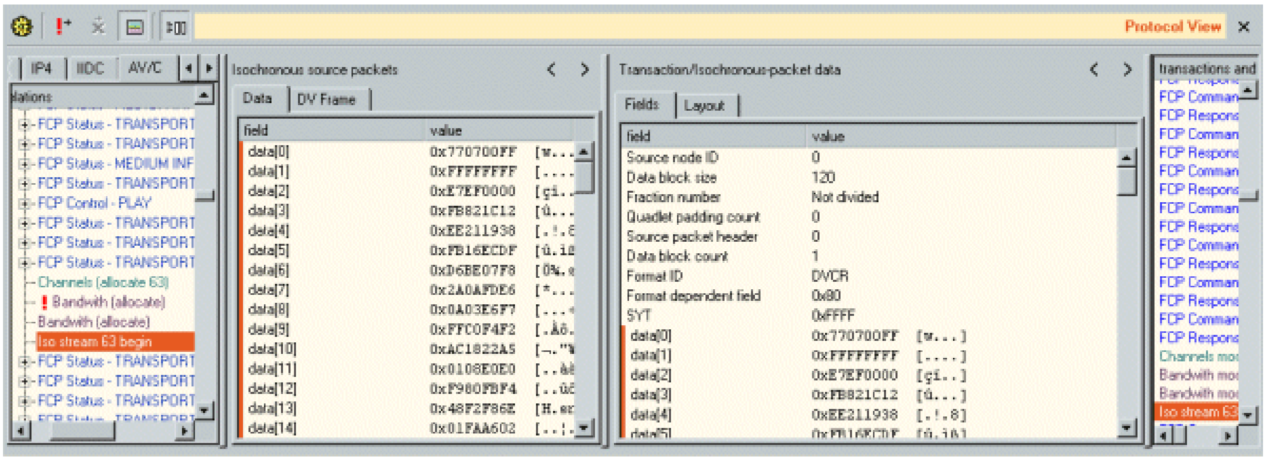
IMPORTANT:
You will need an AV/C-Protocol license key to be able to select this view. However, if you open a Recorder file that was made with a Analyzer with a valid AV/C license key installed, the Protocol View can be enabled, even if you do not have a valid AV/C license key yourself. For more information about license keys, click here.

The AV/C-protocol analyzer needs some information from the Configuration ROM of a AV/C device, to be able to analyze the AV/C transactions and packets corresponding to this device. There are two ways the analyzer can get this information: automatically or manually.

Automatically finding AV/C information
To make sure the AV/C analyzer finds the AV/C information automatically, you should make sure that the reading of Configuration ROM for the node id of the unit is recorded. The Configuration-ROM information is normally read after a bus reset. One way to do this is to (re)connect the AV/C device while the Recorder is recording data. You should also take care that this recorded information is not removed from the recorder buffer because of recorder buffer overflow (cyclic recorder buffer). One way to do this is to stop the recorder before the cyclic buffer (the buffer part before the trigger position) fills completely, or by generating a trigger before this part is filled completely (see the Recorder for more information).

Manually inputting AV/C information
If no Configuration-ROM reads are recorded, you will need to input this information manually. Also, if only part of this information can be found automatically, you will need to add information manually. You can do this in the 'Protocol Settings' dialog. It can be opened by clicking the 'Protocol Settings' button on the toolbar (see below). This dialog shows information used by the all-protocol analyzers. It initially shows all information that it found automatically and you can change or add information manually. See the description of the 'AV/C-Protocol Settings' below for more information.
Selecting a transaction or selecting a stream packet, will display the details of the transaction or stream packet in the details pane (the 'Transaction/Isochronous-packet data' pane). When selecting the 'Iso stream xx start' or 'Iso stream xx stop' event in the 'relations' pane (left) the first, respectively last isochronous packet of a continuous isochronous stream will be selected. The corresponding details of the selected isochronous packet will be displayed in the 'Transaction/Isochronous-packet data' pane.

When an isochronous packet is selected, you will be able to see the corresponding source packet as defined by IEC61883 in an optional 'Isochronous source packet' pane. This part can be toggled on and off with a button in the toolbar (see below). When this part is switched on, it is displayed next to the tree view.
In the example below this display is switched on:



lations
- FCP Status - TRANSPORT
- FCP Status - TRANSPORT
- FCP Status - MEDIUM INF
- FCP Status - TRANSPORT
- FCP Control - PLAY
- FCP Status - TRANSPORT
- FCP Status - TRANSPORT
- FCP Status - TRANSPORT
- Channels (allocate 63)
- Bandwith (allocate)
- Bandwith (allocate)
- Iso stream 63 begin
- FCP Status - TRANSPORT
- FCP Status - TRANSPORT
- FCP Status - TRANSPORT

Isochronous source packets

field	value
data[0]	0x770700FF [w...
data[1]	0xFFFFFFFF [....
data[2]	0xE7EF0000 [çí..
data[3]	0xFB821C12 [û...
data[4]	0xEE211938 [.!.8
data[5]	0xFB16ECDF [û.ì
data[6]	0xD6BE07F8 [Ö¾.ø
data[7]	0x2A0AFDE6 [*...
data[8]	0x0A03E6F7 [...÷
data[9]	0xFFC0F4F2 [.Àô.
data[10]	0xAC1822A5 [¬."¥
data[11]	0x0108E0E0 [..àà
data[12]	0xF980FBF4 [..ûô
data[13]	0x48F2F86E [H.ør
data[14]	0x01FAA602 [..¦.

Transaction/Isochronous-packet data

field	value
Source node ID	0
Data block size	120
Fraction number	Not divided
Quadlet padding count	0
Source packet header	0
Data block count	1
Format ID	DVCR
Format dependent field	0x80
SYT	0xFFFF
data[0]	0x770700FF [w...]
data[1]	0xFFFFFFFF [....]
data[2]	0xE7EF0000 [çí..]
data[3]	0xFB821C12 [û...]
data[4]	0xEE211938 [.!.8]
data[5]	0xFB16ECDF [û.ìß]

transactions and
- FCP Comman
- FCP Respons
- FCP Comman
- FCP Respons
- FCP Comman
- FCP Respons
- FCP Comman
- FCP Respons
- FCP Comman
- FCP Respons
- FCP Comman
- FCP Respons
- FCP Comman
- FCP Respons
- Channels mo
- Bandwith mo
- Bandwith mo
- Iso stream 63

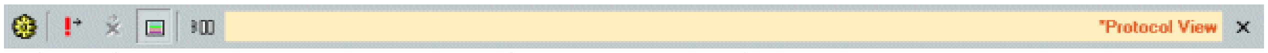
In the example above, an isochronous packet was selected and its details are displayed in the middle part as described above. The extra display, the 'Isochronous source packet' part, will display the isochronous source packet corresponding to the selected isochronous. As defined in the IEC61883 standard, isochronous streams are build from isochronous source packets. The data of one isochronous source packet can consist of data from a part of an isochronous packet, or from one or more complete isochronous packets.

15.1.2. Details

The AV/C-Protocol View consists of a toolbar on the top, with a maximum of four parts below it: the 'relations' pane, the 'Transaction/Isochronous-packet data' (details) pane, the 'transactions and packets' pane and optionally the 'Isochronous source packets' pane.

15.1.2.1. Toolbar

At the top of the Protocol View you will find a toolbar with a few buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol analysis settings have been modified.

Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See 'AV/C-Protocol Settings' below for more information.

Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane on the right side of the Protocol View.

Toggle source display on/off
With this button an optional 'Isochronous source packets' pane can be switched on or off. If switched on, it is added to the left of the 'Transaction/Isochronous-packet data' pane. See below for a description of this pane.

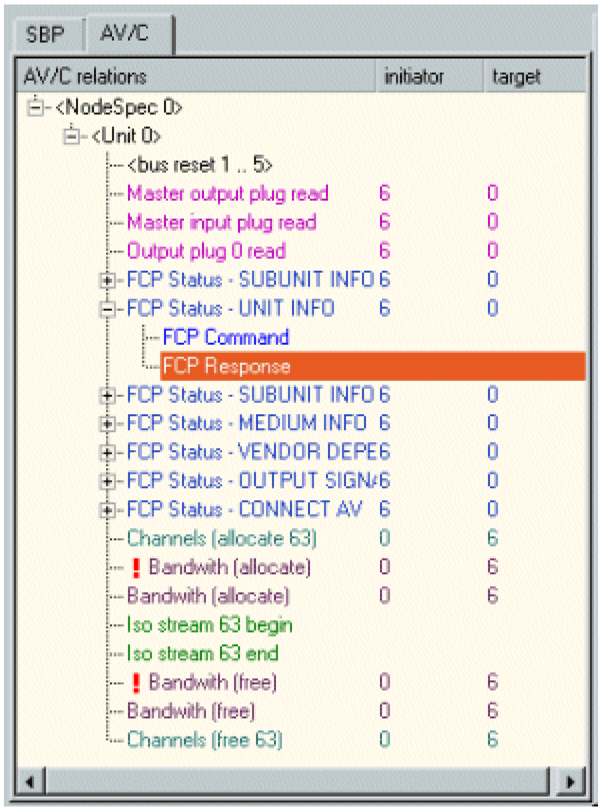
15.1.2.2. Relations Pane

This pane shows all transactions and packets found, grouped into items and displayed in a tree. The items are not necessarily recorded in the same order as displayed in this tree. However, all items inside the same item of the tree (at the same level) are listed in the same order as the first corresponding packet of each item was recorded. To find out the order of corresponding transactions and packets, see the 'transactions and packets' pane or take a look at one of the other views of the Recorder (e.g. the Transactions View).
For each found or manually specified Node Specification there will be a root item in the tree (NodeSpec 0 below). For information about NodeSpec's, see 'AV/C-Protocol Settings' below. For each found or manually specified AV/C Unit, there will be a Unit item inside the corresponding NodeSpec (Unit 0 in NodeSpec 0 below).

Inside the Unit items, the following items can displayed:

- FCP commands and responses
- Channel allocation and de-allocation
- Bandwidth allocation and de-allocation
- Plug register transactions
- Isochronous stream start/stop events
- Reset events

The colors used in the list above are the same as the colors used in the Protocol View.
FCP commands usually consist of a command transaction, followed by a response transaction. Both are grouped into the 'FCP command' items. See example below.


SBP | AV/C

```
AV/C relations                        initiator   target
⊟-<NodeSpec 0>
   ⊟-<Unit 0>
      ├-<bus reset 1 .. 5>
      ├-Master output plug read        6           0
      ├-Master input plug read         6           0
      ├-Output plug 0 read             6           0
      ⊞-FCP Status - SUBUNIT INFO      6           0
      ⊟-FCP Status - UNIT INFO         6           0
      │  ├-FCP Command
      │  └-FCP Response
      ⊞-FCP Status - SUBUNIT INFO      6           0
      ⊞-FCP Status - MEDIUM INFO       6           0
      ⊞-FCP Status - VENDOR DEPE       6           0
      ⊞-FCP Status - OUTPUT SIGN/      6           0
      ⊞-FCP Status - CONNECT AV        6           0
      ├-Channels (allocate 63)         0           6
      ├-! Bandwith (allocate)          0           6
      ├-Bandwith (allocate)            0           6
      ├-Iso stream 63 begin
      ├-Iso stream 63 end
      ├-! Bandwith (free)              0           6
      ├-Bandwith (free)                0           6
      └-Channels (free 63)             0           6
```

A black item indicates that it has no corresponding transactions or packets (e.g. NodeSpec and Unit above). It is just used for proper grouping of the items in the tree. All colored items however, correspond to one or more transactions or packets. When selecting such an item, the details of that item are displayed in the 'details' pane and the corresponding transactions and packets are highlighted in the 'transactions and packets' pane (see below). Note that bus resets like '' are also displayed in black, but have a corresponding item in the 'transactions and packets' pane.

15.1.2.3. Details Pane

When selecting a transaction or stream packet in the 'Transactions and packets' pane, the details of that transaction or stream packet are listed this pane. You can choose between a Field view or a Layout view.

We will describe the different parts below:

Offset
This box displays the address (offset) of the start of the selected item.

Length
The 'Length' box shows the size of the selected item in bytes.

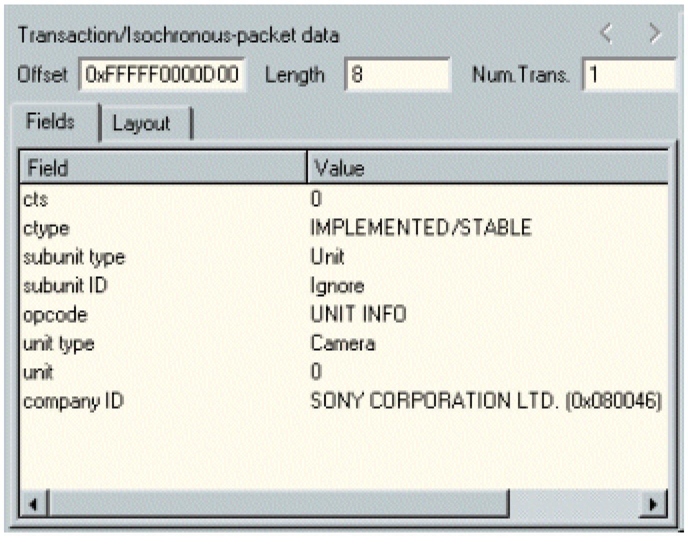
Num. Trans.
This box shows the number of transactions that corresponds to the selected item. For AV/C this will always be 1.

Fields
The 'Fields' table shows the fields that are defined for the selected item and the corresponding values of these fields.

Layout
The 'Layout' shows the layout of the selected item. Like the 'Fields' table it shows the fields that are defined for the selected item and the corresponding values of these fields. But now the fields are displayed graphically inside data item.

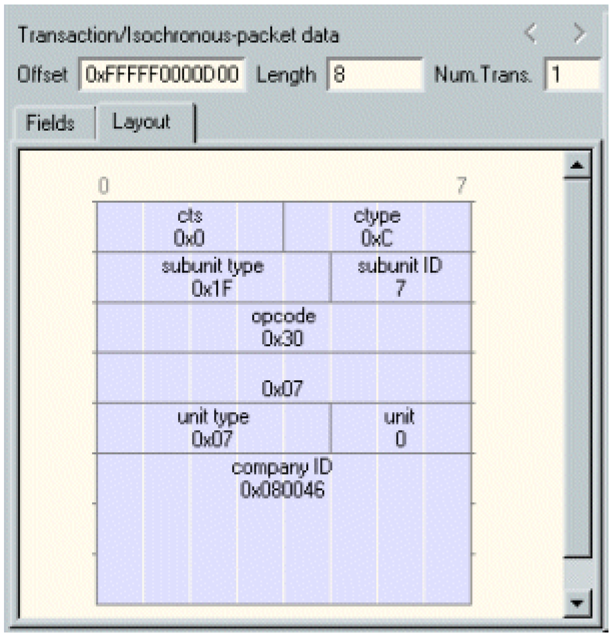
Below we show some field and layout examples of different types of items.

FCP command or response
If a FCP command transaction or FCP response transaction is selected, the fields and the values of these fields defined for the corresponding command or response are shown.
The formatting of the command and response depends on the subunit type, encoded in the command or response. If the Analyzer supports the defined subunit type, the command and response will be formatted corresponding the command set definition of that subunit type. The supported types are listed at the beginning of this manual.
In the two examples below the response of a UNIT INFO command is selected. The first example shows the Fields view, the second example shows the Layout view.



Offset 0xFFFFF0000D00 Length 8 Num.Trans. 1

Field	Value
cts	0
ctype	IMPLEMENTED/STABLE
subunit type	Unit
subunit ID	Ignore
opcode	UNIT INFO
unit type	Camera
unit	0
company ID	SONY CORPORATION LTD. (0x080046)



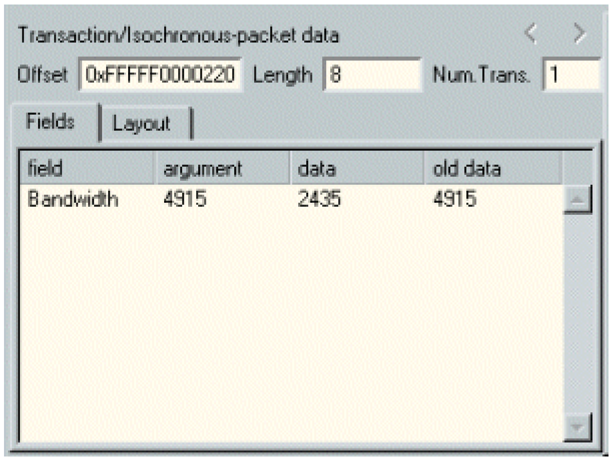
Transaction/Isochronous-packet data

Offset 0xFFFFF0000D00 Length 8 Num.Trans. 1

Fields | Layout

cts 0x0	ctype 0xC
subunit type 0x1F	subunit ID 7
opcode 0x30	
0x07	
unit type 0x07	unit 0
company ID 0x080046	

Bandwidth, Channel, Plug register
If a Bandwidth allocation/de-allocation transaction or Channel allocation/de-allocation transaction or Plug register transaction was selected in the 'details' pane, the 'details' pane will show the corresponding lock-transaction details. The displayed data consists of 3 parts: Argument, Data and Old data.
In the Fields view these 3 data parts are displayed in 3 columns. In the Layout view these 3 data parts are displayed below each other.
When the value(s) in the argument column and old-data column are the same, the lock transaction succeeded. Otherwise, a warning will be displayed for this transaction.
See the two examples below for a (successful) 'Bandwidth allocation' transaction.

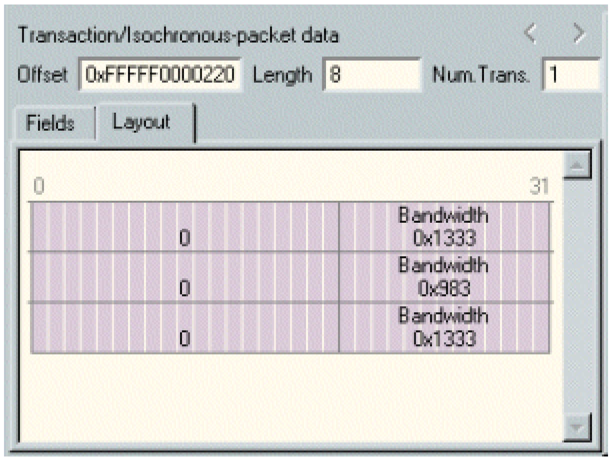


Transaction/Isochronous-packet data

Offset 0xFFFFF0000220 Length 8 Num.Trans. 1

Fields | Layout

field	argument	data	old data
Bandwidth	4915	2435	4915



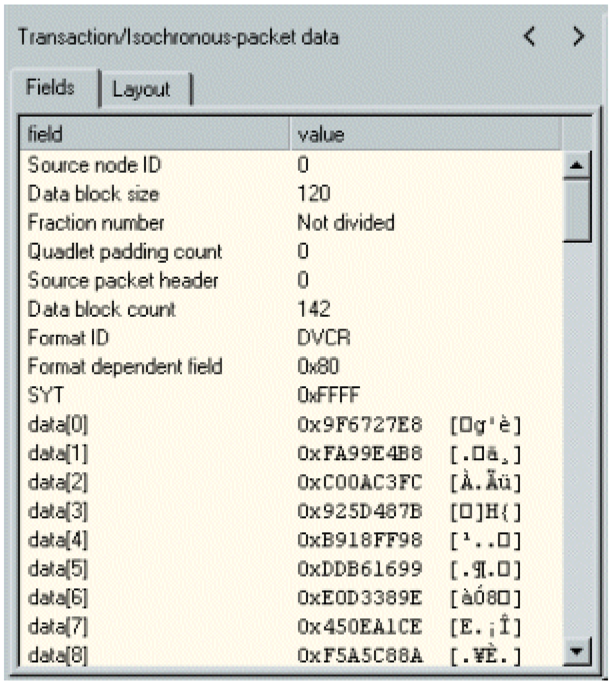
Transaction/Isochronous-packet data

Offset 0xFFFFF0000220 Length 8 Num.Trans. 1

Fields | Layout

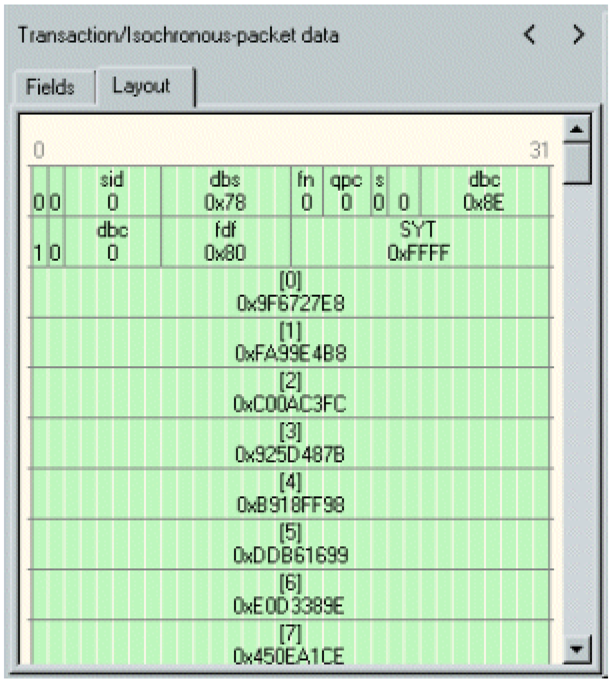
0	Bandwidth 0x1333
0	Bandwidth 0x983
0	Bandwidth 0x1333

Isochronous stream packet(br> If an isochronous stream packet is selected, the 'details' pane will show the details of this isochronous stream packet. Again, you can choose between Fields view and Layout view. See examples below.



Transaction/Isochronous-packet data

field	value
Source node ID	0
Data block size	120
Fraction number	Not divided
Quadlet padding count	0
Source packet header	0
Data block count	142
Format ID	DVCR
Format dependent field	0x80
SYT	0xFFFF
data[0]	0x9F6727E8 [□g'è]
data[1]	0xFA99E4B8 [.□ä¸]
data[2]	0xC00AC3FC [À.Ãü]
data[3]	0x925D487B [□]H{]
data[4]	0xB918FF98 [¹..□]
data[5]	0xDDB61699 [.¶.□]
data[6]	0xE0D3389E [àÓ8□]
data[7]	0x450EA1CE [E.¡Î]
data[8]	0xF5A5C88A [.¥È.]



Transaction/Isochronous-packet data

```
0                                                   31
0 0 | sid 0 | dbs 0x78 | fn 0 | qpc 0 | s 0 | 0 | dbc 0x8E
1 0 | dbc 0 | fdf 0x80 | SYT 0xFFFF
[0] 0x9F6727E8
[1] 0xFA99E4B8
[2] 0xC00AC3FC
[3] 0x925D487B
[4] 0xB918FF98
[5] 0xDDB61699
[6] 0xE0D3389E
[7] 0x450EA1CE
```

Additionally, the previous and/or next isochronous packet buttons (at the top-right corner) may be enabled. Using these buttons you can easily find the previous or next isochronous packet of a continuous isochronous stream.

<

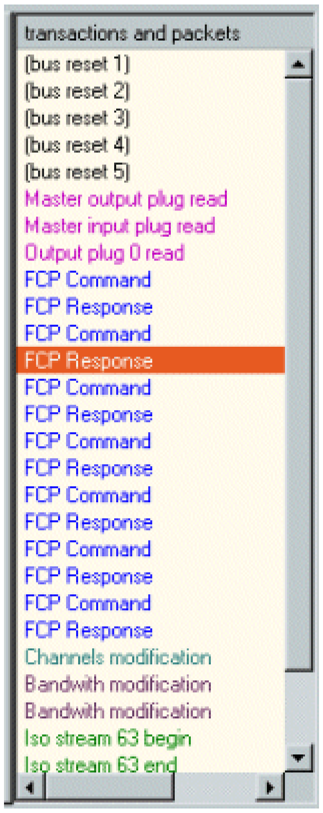
Previous isochronous packet same stream
Clicking this button will select the previous isochronous packet of the same isochronous stream. The button will be disabled as soon as the first packet of the stream is selected, which corresponds to the 'Iso stream xx start' event.

>

Next isochronous packet, same stream
Clicking this button will select the next isochronous packet of the same isochronous stream. The button will be disabled as soon as the last packet of the stream is selected, which corresponds to the 'Iso stream xx stop' event.

15.1.2.4. Transactions and packets pane

The right-most part of the Protocol View shows the sequence of all found protocol packets and transactions (including those for AV/C). The order of items in this list corresponds to the order of recording. It is the same order as displayed in the Transaction View or Packets View of the Recorder. An example follows:



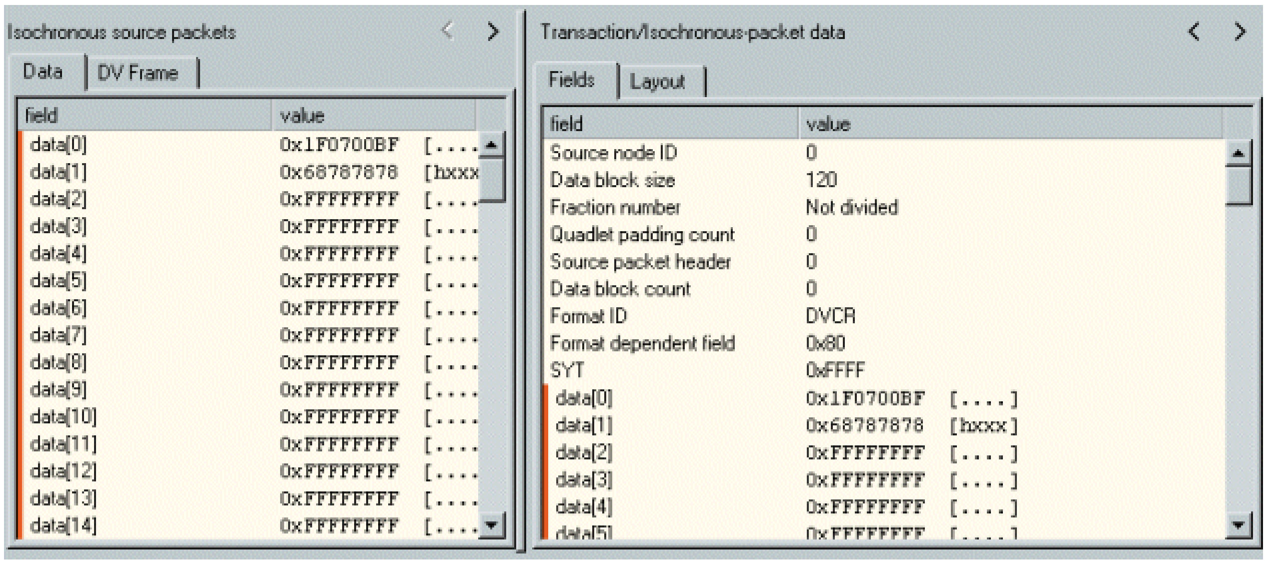
transactions and packets

[bus reset 1]
[bus reset 2]
[bus reset 3]
[bus reset 4]
[bus reset 5]
Master output plug read
Master input plug read
Output plug 0 read
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
FCP Command
FCP Response
Channels modification
Bandwith modification
Bandwith modification
Iso stream 63 begin
Iso stream 63 end

Note that the colors identify the type of access and are the same as used in the 'relations' pane (see above). All transactions and packets in this list that are not part of the protocol currently selected in the 'relation' pane are colored gray.
When an item is selected that consists of more than one packet or transaction, the items are highlighted in the 'transactions and packets' pane list with a gray bar (the selected item, which is also part of the item will still be highlighted with the usual selection color).

15.1.2.5. Isochronous source packets pane

This optional pane, displays the isochronous source packet (as defined in IEC61883) corresponding to the selected isochronous-stream packet(s). According to the definition of the source packets, one isochronous source packet can consist of data from a part of an isochronous packet, or from one or more complete isochronous packets. Below is an example of a combination of the 'Transaction/Isochronous-packet data' (details) pane and 'Isochronous source packets' pane.



Isochronous source packets
Data
DV Frame

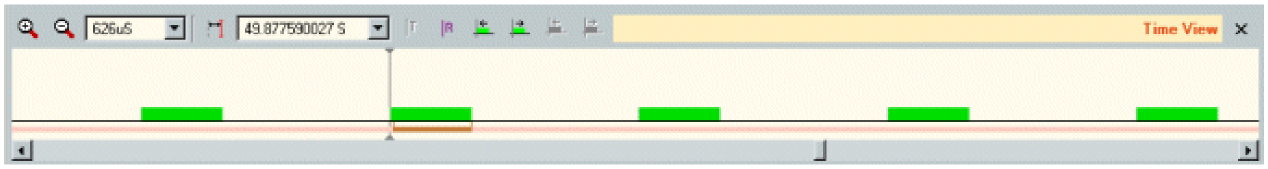
field	value	
data[0]	0x1F0700BF	[....
data[1]	0x68787878	[hxxx
data[2]	0xFFFFFFFF	[....
data[3]	0xFFFFFFFF	[....
data[4]	0xFFFFFFFF	[....
data[5]	0xFFFFFFFF	[....
data[6]	0xFFFFFFFF	[....
data[7]	0xFFFFFFFF	[....
data[8]	0xFFFFFFFF	[....
data[9]	0xFFFFFFFF	[....
data[10]	0xFFFFFFFF	[....
data[11]	0xFFFFFFFF	[....
data[12]	0xFFFFFFFF	[....
data[13]	0xFFFFFFFF	[....
data[14]	0xFFFFFFFF	[....

Transaction/Isochronous-packet data
Fields
Layout

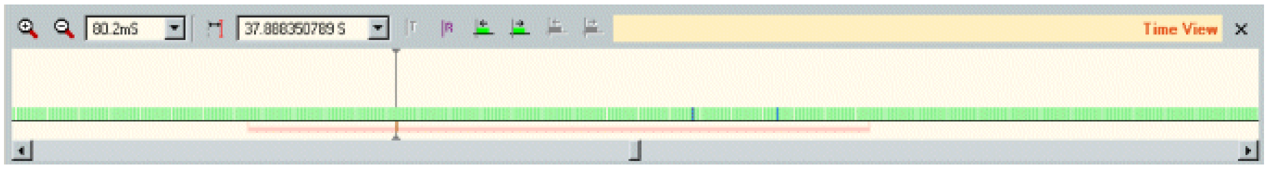
field	value	
Source node ID	0	
Data block size	120	
Fraction number	Not divided	
Quadlet padding count	0	
Source packet header	0	
Data block count	0	
Format ID	DVCR	
Format dependent field	0x80	
SYT	0xFFFF	
data[0]	0x1F0700BF	[....]
data[1]	0x68787878	[hxxx]
data[2]	0xFFFFFFFF	[....]
data[3]	0xFFFFFFFF	[....]
data[4]	0xFFFFFFFF	[....]
data[5]	0xFFFFFFFF	[....]

In this example, the data in one isochronous packet corresponds, again, exactly to one source packet (DV

video).
If the 'Isochronous source packets pane' is enabled, the Time View will indicate the source packet shown in the 'Isochronous source packets pane' by displaying a dark brown bar beneath those isochronous packets (or parts of isochronous packets) that are part of the source packet. In the example below, which is a DV video example, one source packet corresponds to one isochronous packet.



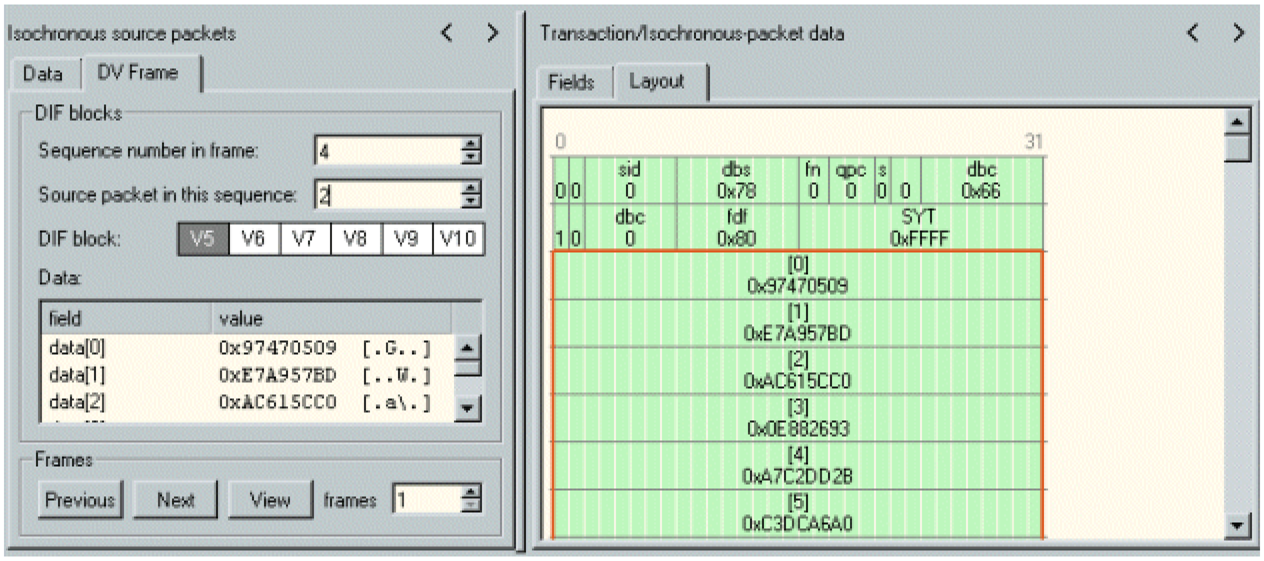
If the source packet is part of a DV video frame, the frame is indicated in the Time View with a light brown bar. Below, the same Time-View example is displayed as the example above, but now the view is zoomed out a few steps so that the complete frame becomes visible.



The 'Isochronous source packets' pane also has a previous and a next button:

Previous source packet
Clicking this button will select the previous source packet of the same isochronous stream. The button will be disabled as soon as the first source packet of the stream is selected.

Next source packet
Clicking this button will select the next source packet of the same isochronous stream. The button will be disabled as soon as the last source packet of the stream is selected.
It is also possible to view the data inside a video frame. The screenshot below shows what the DV Frame tab can be used for:


Isochronous source packets
Data
DV Frame
DIF blocks
Frames
Previous
Next
View
Transaction/Isochronous-packet data
Fields
Layout

Sequence number in frame: 4
Source packet in this sequence: 2
DIF block: V5 V6 V7 V8 V9 V10

Data:

field	value
data[0]	0x97470509 [.G..]
data[1]	0xE7A957BD [..W.]
data[2]	0xAC615CC0 [.a\.]

frames 1

0								31
0	0	sid 0	dbs 0x78	fn 0	qpc 0	s 0	0	dbc 0x66
1	0	dbc 0	fdf 0x80		SYT 0xFFFF			

[0] 0x97470509
[1] 0xE7A957BD
[2] 0xAC615CC0
[3] 0x0E882693
[4] 0xA7C2DD2B
[5] 0xC3DCA6A0

The DV-Frame tab can be used to walk through the logical data inside an isochronous video-frame packet. Each source packet consists of a series of DIF blocks. Using the edit control with spin box labeled 'Source packet in this sequence' all parts of the DIF block can be displayed. Clicking on a DIF block will display the content of that DIF block in the 'Data' table. Using the edit control with spin box labeled 'Sequence number in frame' all sequences in a video frame can be selected.
It's also possible to actually see the video from a series of frames. Clicking the 'View' button in the group 'Frames' at the bottom of the 'DV Frame' tab will save the selected number of video frames starting at the

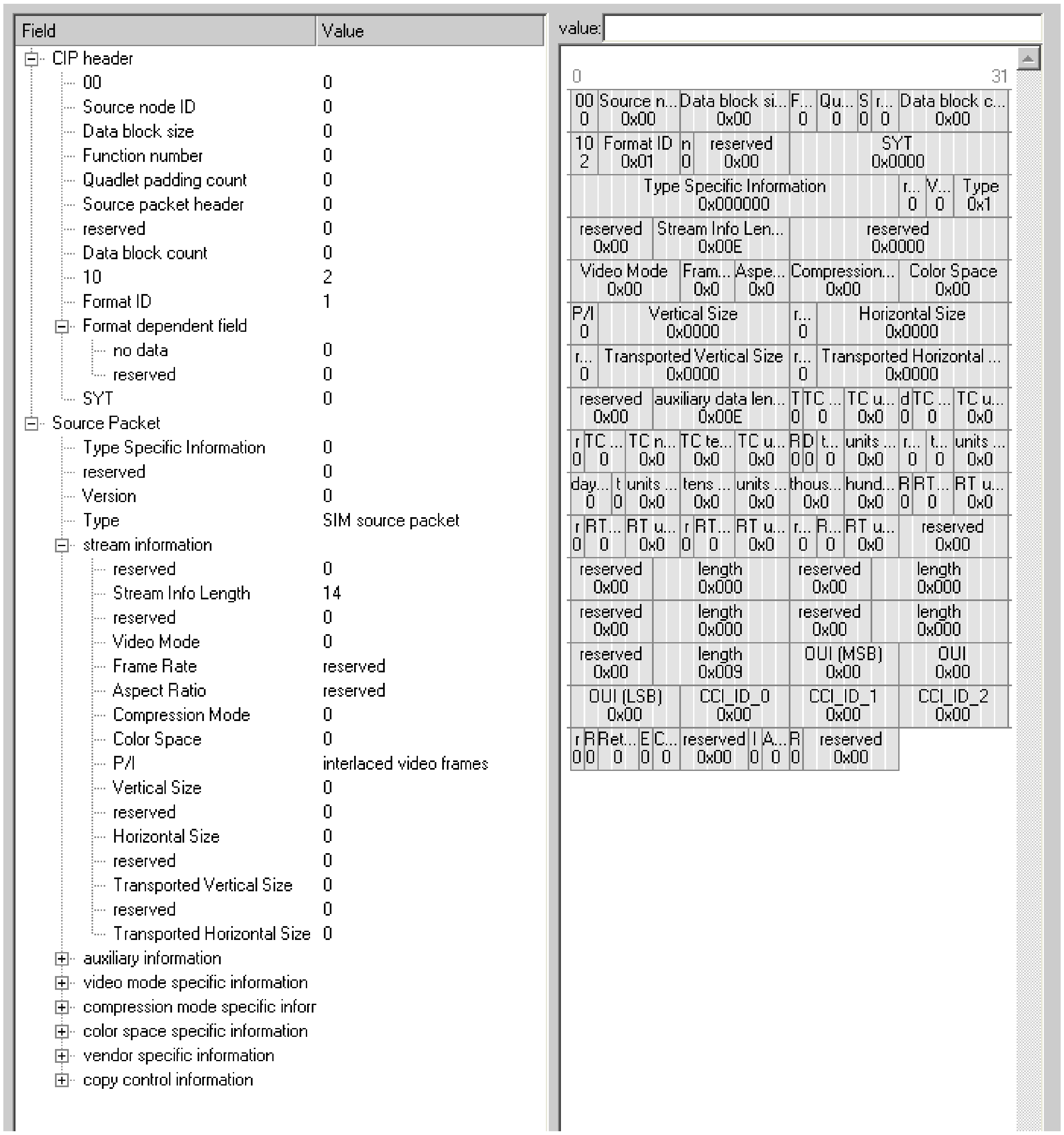
current frame to a *.dv file (a file with raw DV frames). This file will be automatically opened using the operating system dependent viewer for this type of file.

Note that a video player capable of playing a *.dv (raw DV frames) file need to be installed for this option to work properly.
The buttons 'Previous' and 'Next' can be used to select the previous or next video frame.

15.1.2.5.1 BT601 Support

As a part of the AV/C protocol support for BT601 is available.

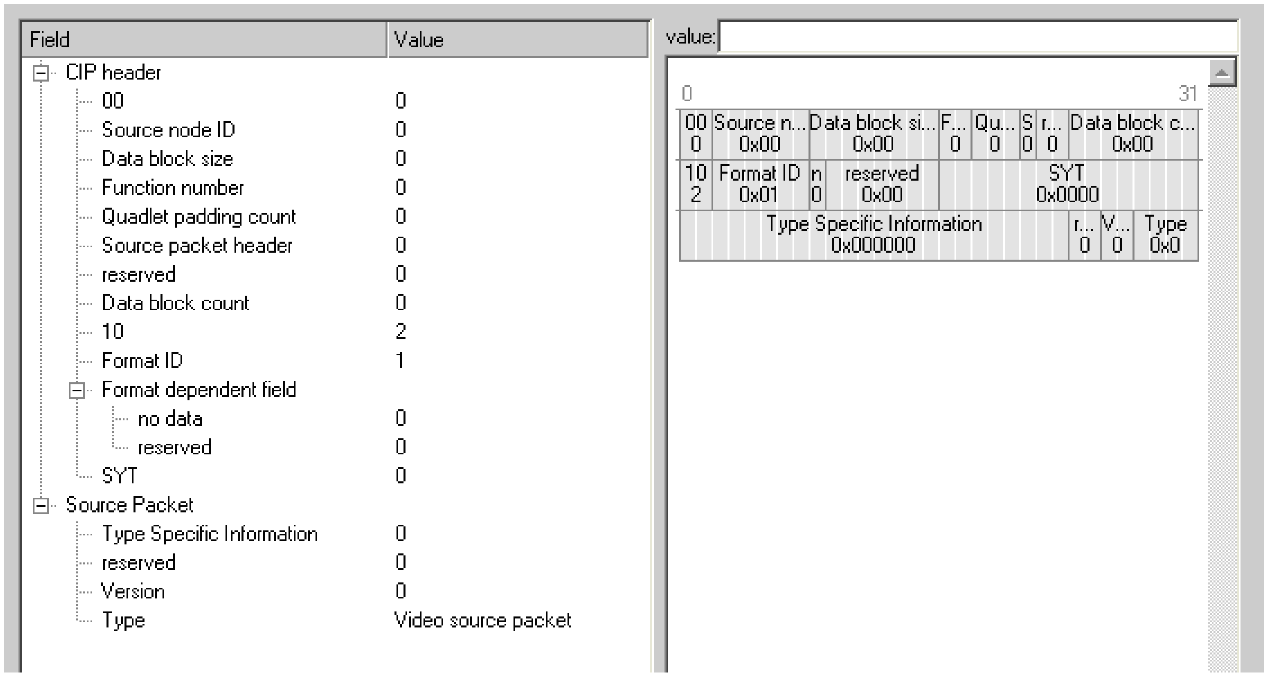
BT601 Message Set
Message set is defined in the specification “BT.601 Transport Over IEEE-1394” (document number 2006020). Specification defines two basic packet formats. One is related to the stream information and metadata – SIM packet, and another is used to transfer audio or video data. Figure below shows an example of the BT.601 SIM packet.



Field	Value
CIP header	
00	0
Source node ID	0
Data block size	0
Function number	0
Quadlet padding count	0
Source packet header	0
reserved	0
Data block count	0
10	2
Format ID	1
Format dependent field	
no data	0
reserved	0
SYT	0
Source Packet	
Type Specific Information	0
reserved	0
Version	0
Type	SIM source packet
stream information	
reserved	0
Stream Info Length	14
reserved	0
Video Mode	0
Frame Rate	reserved
Aspect Ratio	reserved
Compression Mode	0
Color Space	0
P/I	interlaced video frames
Vertical Size	0
reserved	0
Horizontal Size	0
reserved	0
Transported Vertical Size	0
reserved	0
Transported Horizontal Size	0
auxiliary information	
video mode specific information	
compression mode specific inforr	
color space specific information	
vendor specific information	
copy control information	

```
value:
0                                                                31
| 00 0 | Source n... 0x00 | Data block si... 0x00 | F... 0 | Qu... 0 | S 0 | r... 0 | Data block c... 0x00 |
| 10 2 | Format ID 0x01 | n 0 | reserved 0x00 | SYT 0x0000 |
| Type Specific Information 0x000000 | r... 0 | V... 0 | Type 0x1 |
| reserved 0x00 | Stream Info Len... 0x00E | reserved 0x0000 |
| Video Mode 0x00 | Fram... 0x0 | Aspe... 0x0 | Compression... 0x00 | Color Space 0x00 |
| P/I 0 | Vertical Size 0x0000 | r... 0 | Horizontal Size 0x0000 |
| r... 0 | Transported Vertical Size 0x0000 | r... 0 | Transported Horizontal ... 0x0000 |
| reserved 0x00 | auxiliary data len... 0x00E | T 0 | TC ... 0 | TC u... 0x0 | d 0 | TC ... 0 | TC u... 0x0 |
| r 0 | TC ... 0 | TC n... 0x0 | TC te... 0x0 | TC u... 0x0 | R 0 | D 0 | t... 0 | units ... 0x0 | r... 0 | t... 0 | units ... 0x0 |
| day... 0 | t 0 | units ... 0x0 | tens ... 0x0 | units ... 0x0 | thous... 0x0 | hund... 0x0 | R 0 | RT... 0 | RT u... 0x0 |
| r 0 | RT... 0 | RT u... 0x0 | r 0 | RT... 0 | RT u... 0x0 | r... 0 | R... 0 | RT u... 0x0 | reserved 0x00 |
| reserved 0x00 | length 0x000 | reserved 0x00 | length 0x000 |
| reserved 0x00 | length 0x000 | reserved 0x00 | length 0x000 |
| reserved 0x00 | length 0x009 | OUI (MSB) 0x00 | OUI 0x00 |
| OUI (LSB) 0x00 | CCI_ID_0 0x00 | CCI_ID_1 0x00 | CCI_ID_2 0x00 |
| r 0 | R 0 | Ret... 0 | E 0 | C... 0 | reserved 0x00 | I 0 | A... 0 | R 0 | reserved 0x00 |
```

Using BT601 protocol audio and video data can be transferred uncompressed or compressed. Depend on the data format packet has different size, but header remains the same. Figure below shows BT.601 video source header.



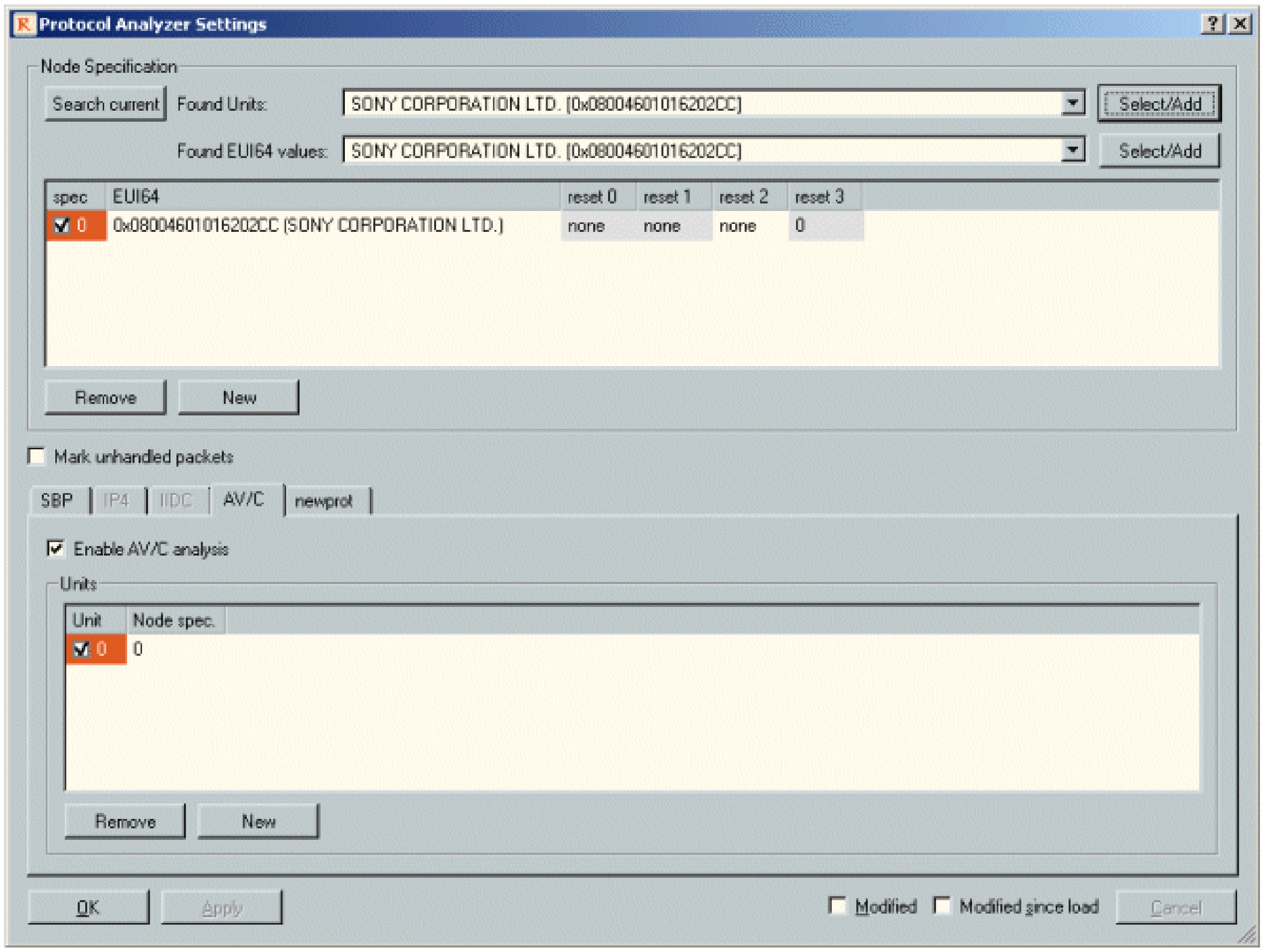
Field	Value
CIP header	
00	0
Source node ID	0
Data block size	0
Function number	0
Quadlet padding count	0
Source packet header	0
reserved	0
Data block count	0
10	2
Format ID	1
Format dependent field	
no data	0
reserved	0
SYT	0
Source Packet	
Type Specific Information	0
reserved	0
Version	0
Type	Video source packet

value:

```
0                                                           31
00 | Source n... | Data block si... | F... | Qu... | S | r... | Data block c...
0  | 0x00        | 0x00             | 0    | 0     | 0 | 0    | 0x00
10 | Format ID   | n | reserved     | SYT
2  | 0x01        | 0 | 0x00         | 0x0000
Type Specific Information           | r... | V... | Type
0x000000                            | 0    | 0    | 0x0
```

15.1.2.6. AV/C-Protocol Settings

When clicking the 'Protocol Settings' button in the toolbar of the Protocol View, the 'Protocol Settings' dialog will be displayed. This dialog shows information used by the protocol analyzers to find and correctly analyze the transactions. For each supported protocol, including AV/C, there is a tab page with protocol-specific settings. The dialog will be initialized with all information that can be found automatically. If this is not enough for a correct analysis, you can add this information manually. An example of the dialog is shown below.


Protocol Analyzer Settings

Node Specification

Found Units: SONY CORPORATION LTD. [0x08004601016202CC]

Found EUI64 values: SONY CORPORATION LTD. [0x08004601016202CC]

spec	EUI64	reset 0	reset 1	reset 2	reset 3
✓ 0	0x08004601016202CC (SONY CORPORATION LTD.)	none	none	none	0

Mark unhandled packets

Tabs: SBP, IP4, IIDC, AV/C, newprot

✓ Enable AV/C analysis

Units

Unit	Node spec.
✓ 0	0

Modified   Modified since load

In the dialog you see an upper part to specify one or more 'Node Specifications', and a lower part to specify protocol specific information. In between these two parts, general options can be selected.
For a correct analysis, the Unit information of the AV/C device to be analyzed must be present. To be able to specify a Unit, you will first need to specify the device (Node Specification) the Unit is part of.
For more information about Node Specifications and the protocol independent part of the Protocol Settings, see Protocol Settings. For the AV/C-protocol-specific settings, see below.

Node Specifications
If no AV/C Units were found automatically, you have to specify one or more Units manually. Before you can specify a Unit, you have to specify the node the Unit is part of using a Node Specification. When the AV/C device is still connected to the bus after recording, you can use the 'Search Current' button to automatically find AV/C Units. Adding such a Unit will create a Node Specification and Unit. The created Node Specification will be filled automatically when possible. You probably will have to select correct nodes for some reset segments. Make sure you have selected the AV/C node correctly for those reset segments for which transaction and packets are recorded that you want to analyze. For more information on Node Specifications, see Protocol Settings.

General options
Mark unhandled packets
For information about this checkbox, please see Protocol Settings.

Protocol specific information

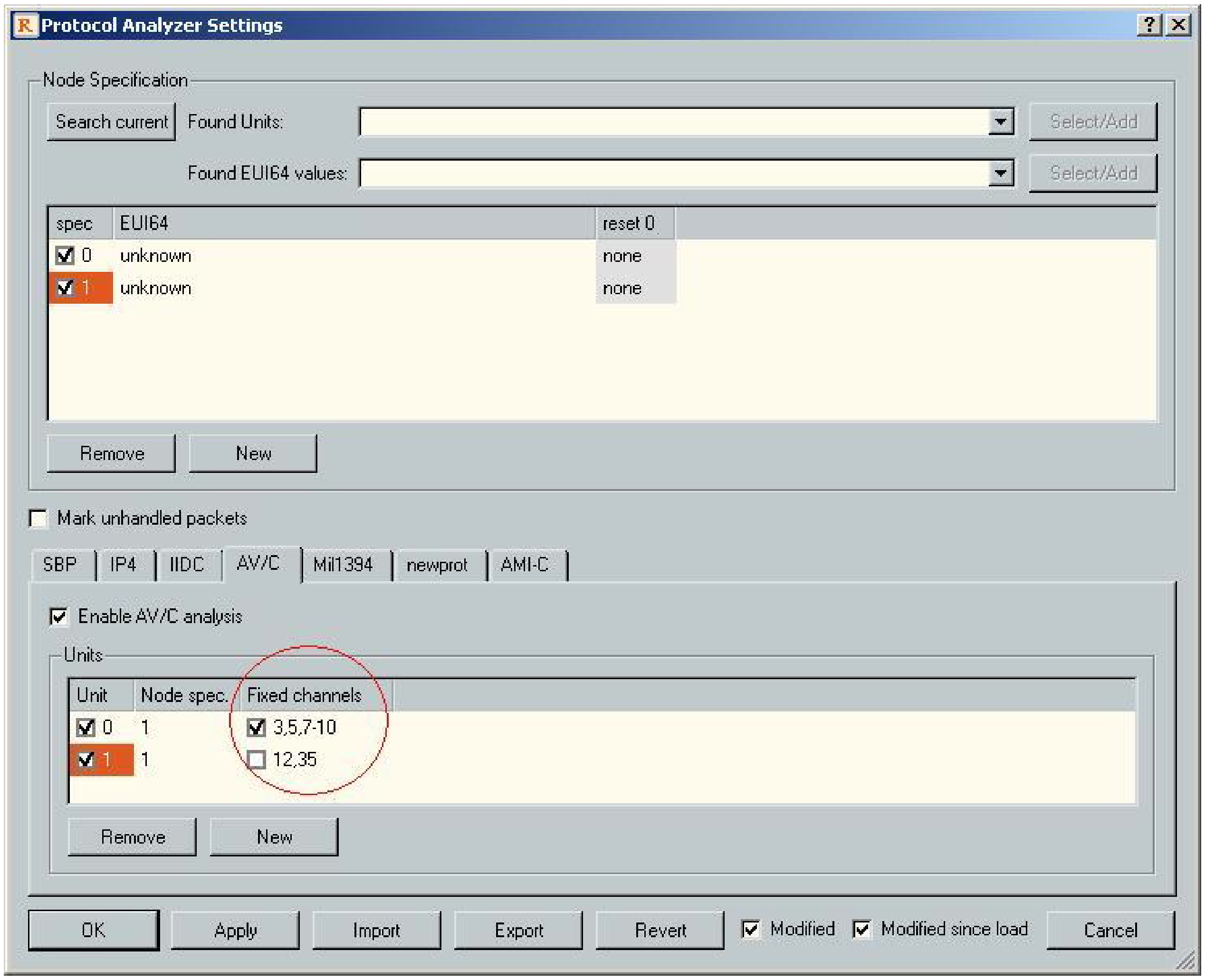
Enable AV/C analysis
Check this box to enable the AV/C analyzer. All protocol tabs in this dialog have such a checkbox, which can be useful if a recording contains data from several protocols, and you are not interested in some of these.

Units
In the Units table, one or more AV/C Units can be specified. The information in this table normally can be found in the Configuration ROM of the AV/C node. It consists of:

- *Unit:* Using the checkbox before each Unit in the table you can enable or disable each Unit individually for analysis. If the box is not checked, the AV/C-related transactions and packets for this Unit will be skipped.
- *Node Spec.:* A Unit is part of a node. As explained before, nodes are specified using Node Specifications. The number in this column corresponds to the Node-Specification numbers of the table in the upper part of the dialog.

Fixed channels
Some applications use 1394 channels for isochronous communication without previous resource allocation. Those channels are predefined, so the same has to be done in the FireWire Protocol Analyzer. In unit description, column “Fixed channel” can be used to assign per unit any channel between 0 and 63, and in case of more channels an array can be defined (e.g. 7-10). Assigned channels can be as a group enabled or disabled using check box.

Figure below shows User interface for Fixed channel settings.


Protocol Analyzer Settings

Node Specification

spec	EUI64	reset 0
✓ 0	unknown	none
✓ 1	unknown	none

☐ Mark unhandled packets

✓ Enable AV/C analysis

Units

Unit	Node spec.	Fixed channels
✓ 0	1	✓ 3,5,7-10
✓ 1	1	☐ 12,35

✓ Modified ✓ Modified since load

15.2. Editing AV/C Formats

The Analyzer software supports a number of standard AVC command sets for the AVC protocol. If, however, you need support for a command set not supported by the Analyzer AVC software, then you can define your own custom command set using the FormatEditor.

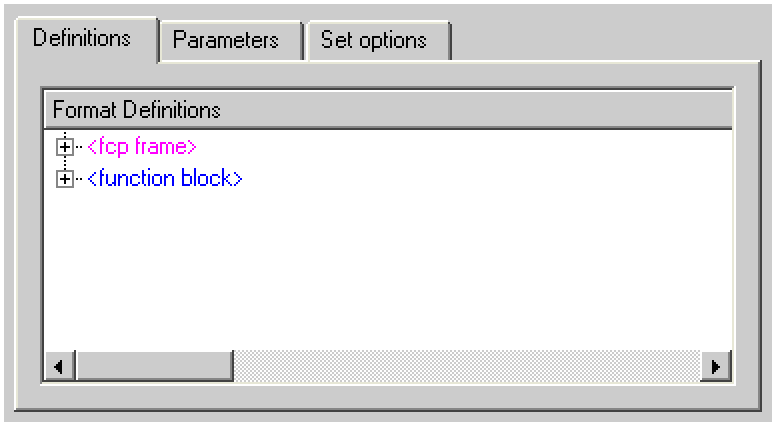
For each custom command set you have to create a format set file, with a format definition for the FCP frame. The defined command set can be used by the AVC analyzer to display the commands and associated responses in the correct format. It can also be used in the Filter Sets for the Filter/Trigger logic to define conditions on the fields in particular command or response.

You could create a new format set with the FormatEditor and start building your format defintions from scratch. But probably the easiest way is to open an existing format file for the AVC protocol and adapt it to your needs.

There is also a tempate file (avc_subunit_template.dff) which could be a good starting point.

15.2.1. Example

Below the file avc_Audio.dff has been loaded as an example of a AVC format set.

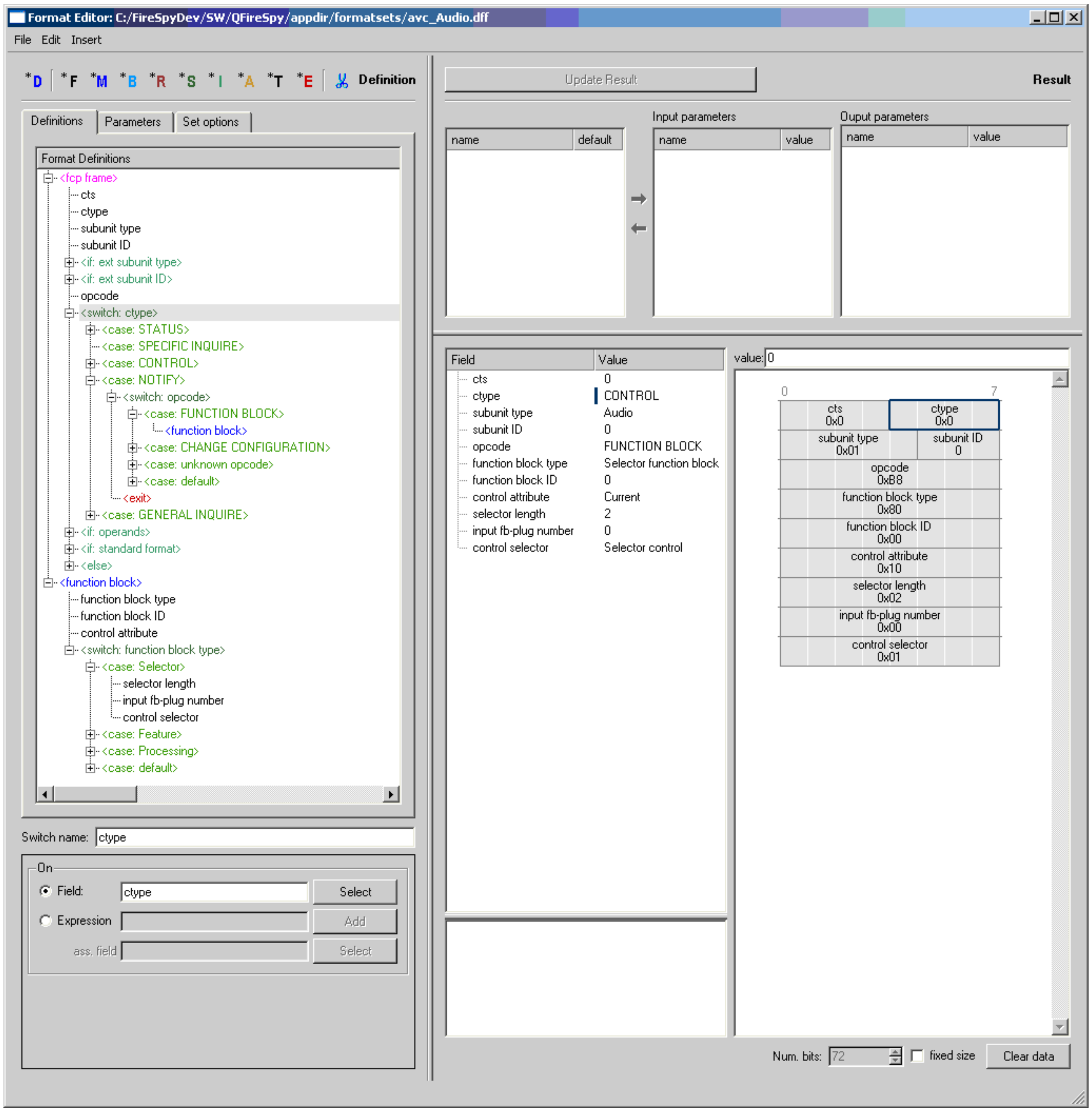


The first format definition, called 'fcp frame' (color magenta) defines the format for the AVC commands and responses. It always needs to be present in a format set for the AVC protocol.

The other definition (color blue) is a macro definition, used by the 'fcp frame' definition.

Fcp frame

The format definition 'fcp frame' defines the format of the commands and responses for AVC.
Below the 'fcp frame' definition is shown in more detail.



Format Definitions

```
<fcp frame>
    cts
    ctype
    subunit type
    subunit ID
    <if: ext subunit type>
    <if: ext subunit ID>
    opcode
    <switch: ctype>
        <case: STATUS>
        <case: SPECIFIC INQUIRE>
        <case: CONTROL>
        <case: NOTIFY>
            <switch: opcode>
                <case: FUNCTION BLOCK>
                    <function block>
                <case: CHANGE CONFIGURATION>
                <case: unknown opcode>
                <case: default>
            <exit>
        <case: GENERAL INQUIRE>
    <if: operands>
    <if: standard format>
    <else>
<function block>
    function block type
    function block ID
    control attribute
    <switch: function block type>
        <case: Selector>
            selector length
            input fb-plug number
            control selector
        <case: Feature>
        <case: Processing>
        <case: default>
```

Switch name: ctype

On
- (•) Field: ctype
- ( ) Expression:
- ass. field:

Field	Value
cts	0
ctype	CONTROL
subunit type	Audio
subunit ID	0
opcode	FUNCTION BLOCK
function block type	Selector function block
function block ID	0
control attribute	Current
selector length	2
input fb-plug number	0
control selector	Selector control

value: 0

0 … 7	
cts 0x0	ctype 0x0
subunit type 0x01	subunit ID 0
opcode 0xB8	
function block type 0x80	
function block ID 0x00	
control attribute 0x10	
selector length 0x02	
input fb-plug number 0x00	
control selector 0x01	

Num. bits: 72

As you can see above, the 'fcp frame' format definition starts with the 'cts', 'ctype', 'subunit type' and 'subunit ID' fields. Then there are some 'if' items for the optional extensions for the 'subunit type' and 'subunit ID' fields. Then the 'opcode' field follows. After this the format depends on the values of the 'ctype' and 'opcode' field. First we used a 'switch' item to select the correct format depending on the 'ctype' and inside each 'case' item of this switch we used a 'switch' item for the 'opcode' dependancy.

Note that this could of course also be done the other way around if you like.

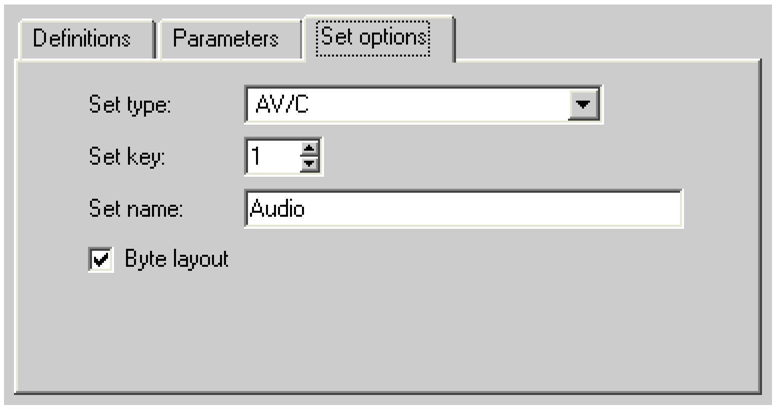
The 'case: NOTIFY' has been expanded in the example above. You can see the switch-on opcode and each case inside it. The 'case: FUNCTION BLOCK' has been expanded too. You can see that for the remaining fields for this format (opcode=FUNCTION BLOCK and ctype=NOTIFY) a macro has been used. The macro ('function block') is shown at the bottom of the definitions tree. It has been expanded too, so that you can see which field are added.
In the 'Result' pane at the right, the result for the 'fcp frame' is shown in a table and layout view. We filled in a ctype value of 'NOTIFY' and an opcode value of 'FUNCTION BLOCK' to check the correctness of this part of the definition tree.

input parameters
The following input parameters can be used by the 'fcp frame' format definition:

Input parameter name	value
prevCommandCType	This is the value of the ctype field of the corresponding AVC command. It can be usefull if the ctype for the command and response blocks are not the same.

Set options

To be able to use the format definitions in this format set for the AV/C protocol you need to set the correct 'Set options'. Below the 'Set options' of the example file are shown.



Set type: AV/C
Set key: 1
Set name: Audio
☑ Byte layout

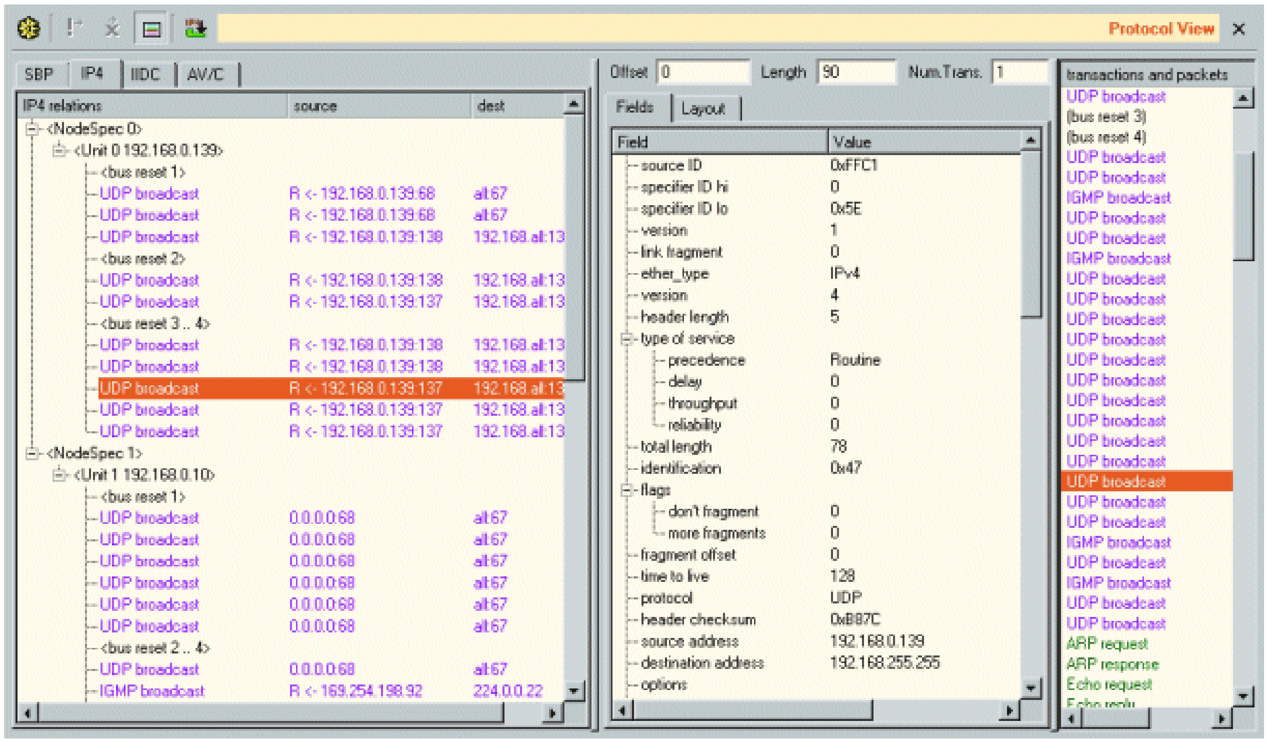
The 'Set type' should be 'AV/C'. The 'Set key' must be set to the subunit type corresponding to the command set you have implemented by this format set. The 'Set name' must be unique for all AV/C type files. So you should fill in some other suitable name here.
When the Analyzer application finds two AV/C format sets with the same 'Set name' then a warning will be displayed and the second one will be ignored. This is a valid way to overrule an existing format set.

Automatic set recognition

When the Analyzer needs to format an AV/C command or response block, it uses the value of the 'subunit type' field of that command or response block to select the correct format set and thus the correct formating for the block. It will search for a format set with 'Set type' AV/C and with a 'Set key' equal to the 'subunit type' field.

Chapter 16. Internet Protocol version 4 (IP4)

One of the protocols supported by the Protocol View of the Recorder is the **Internet Protocol version 4** (IP4) protocol. The IP4-protocol analyzer software will scan through all transactions and packets to find IP4-compliant Units and it will fill the 'relations' pane in the IP4 tab page with IP4-related items, as described in Protocol View and Protocol Settings. An example of the resulting Protocol View is displayed below.



Offset 0 Length 90 Num.Trans. 1

IP4 relations	source	dest
<NodeSpec 0>		
<Unit 0 192.168.0.139>		
<bus reset 1>		
UDP broadcast	R <- 192.168.0.139:68	all:67
UDP broadcast	R <- 192.168.0.139:68	all:67
UDP broadcast	R <- 192.168.0.139:138	192.168.all:13
<bus reset 2>		
UDP broadcast	R <- 192.168.0.139:138	192.168.all:13
UDP broadcast	R <- 192.168.0.139:137	192.168.all:13
<bus reset 3 .. 4>		
UDP broadcast	R <- 192.168.0.139:138	192.168.all:13
UDP broadcast	R <- 192.168.0.139:138	192.168.all:13
UDP broadcast	R <- 192.168.0.139:137	192.168.all:13
UDP broadcast	R <- 192.168.0.139:137	192.168.all:13
UDP broadcast	R <- 192.168.0.139:137	192.168.all:13
<NodeSpec 1>		
<Unit 1 192.168.0.10>		
<bus reset 1>		
UDP broadcast	0.0.0.0:68	all:67
UDP broadcast	0.0.0.0:68	all:67
UDP broadcast	0.0.0.0:68	all:67
UDP broadcast	0.0.0.0:68	all:67
UDP broadcast	0.0.0.0:68	all:67
UDP broadcast	0.0.0.0:68	all:67
<bus reset 2 .. 4>		
UDP broadcast	0.0.0.0:68	all:67
IGMP broadcast	R <- 169.254.198.92	224.0.0.22

Field	Value
source ID	0xFFC1
specifier ID hi	0
specifier ID lo	0x5E
version	1
link fragment	0
ether_type	IPv4
version	4
header length	5
type of service	
precedence	Routine
delay	0
throughput	0
reliability	0
total length	78
identification	0x47
flags	
don't fragment	0
more fragments	0
fragment offset	0
time to live	128
protocol	UDP
header checksum	0xB87C
source address	192.168.0.139
destination address	192.168.255.255
options	

transactions and packets
UDP broadcast
(bus reset 3)
(bus reset 4)
UDP broadcast
UDP broadcast
IGMP broadcast
UDP broadcast
UDP broadcast
IGMP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
IGMP broadcast
UDP broadcast
IGMP broadcast
UDP broadcast
UDP broadcast
ARP request
ARP response
Echo request
Echo reply

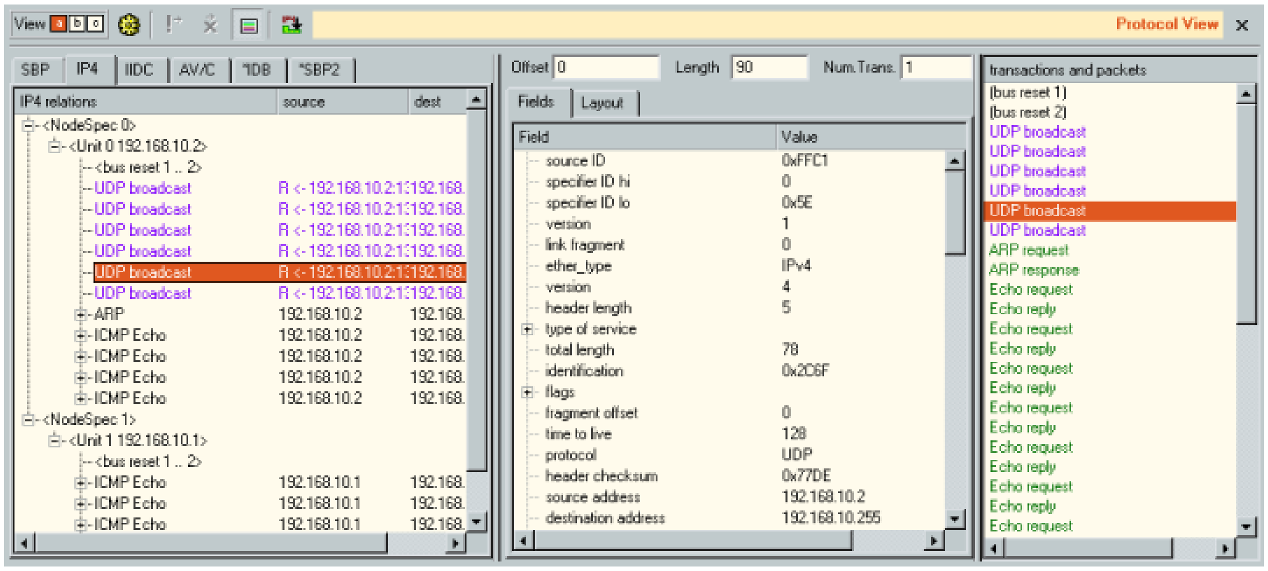
It displays the result in three different parts. At the right (transactions and packets), all found packets and transactions are displayed in the order they were detected on the bus. The left part (relations) shows the relation between these items. The transactions and packets belonging together (e.g. all transactions that form a single TCP connection) are grouped into tree items and these items are displayed hierarchically in the 'relations' pane. In this tree you will find for instance:

- ICMP traffic
- TCP connections
- UDP packets
- ARP packets
- Bus resets

The details of the item selected in the relations tree, are displayed in the middle part of the Protocol View. It displays the fields defined for the selected item and the values of those fields.

Triple FireSpy

For triple FireSpys the Protocol View will look as follows:



IP4 relations	source	dest
<NodeSpec 0>		
<Unit 0 192.168.10.2>		
<bus reset 1 .. 2>		
UDP broadcast	R <- 192.168.10.2:1	192.168.
UDP broadcast	R <- 192.168.10.2:1	192.168.
UDP broadcast	R <- 192.168.10.2:1	192.168.
UDP broadcast	R <- 192.168.10.2:1	192.168.
UDP broadcast	R <- 192.168.10.2:1	192.168.
UDP broadcast	R <- 192.168.10.2:1	192.168.
ARP	192.168.10.2	192.168.
ICMP Echo	192.168.10.2	192.168.
ICMP Echo	192.168.10.2	192.168.
ICMP Echo	192.168.10.2	192.168.
ICMP Echo	192.168.10.2	192.168.
<NodeSpec 1>		
<Unit 1 192.168.10.1>		
<bus reset 1 .. 2>		
ICMP Echo	192.168.10.1	192.168.
ICMP Echo	192.168.10.1	192.168.
ICMP Echo	192.168.10.1	192.168.

Offset 0 Length 90 Num. Trans. 1

Field	Value
source ID	0xFFC1
specifier ID hi	0
specifier ID lo	0x5E
version	1
link fragment	0
ether_type	IPv4
version	4
header length	5
type of service	
total length	78
identification	0x2C6F
flags	
fragment offset	0
time to live	128
protocol	UDP
header checksum	0x77DE
source address	192.168.10.2
destination address	192.168.10.255

```
transactions and packets
[bus reset 1]
[bus reset 2]
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
UDP broadcast
ARP request
ARP response
Echo request
Echo reply
Echo request
Echo reply
Echo request
Echo reply
Echo request
Echo reply
Echo request
Echo reply
Echo request
Echo reply
Echo request
```

16.1. Recorder Protocol View

16.1.1. How to use it

To display the IP4 results, select the IP4 tab page in the Protocol View of the Recorder. The Protocol View can be displayed by selecting it in the 'View' menu of the Recorder.

IMPORTANT:
You will need an IP4-Protocol license key to be able to select this view. However, if you open a Recorder file that was made with a Analyzer with a valid IP4 license key installed, the Protocol View can be enabled, even if you do not have a valid IP4 license key yourself. For more information about license keys, click here.

The IP4-protocol analyzer needs some information from the Configuration ROM of an IP4 device, to be able to analyze the IP4 transactions and packets corresponding to this device. It also needs to know the FIFO command address and IP address of the IP4 unit to be able to analyze IP4 data. There are two ways the analyzer can get this information: automatically or manually.

Automatically finding IP4 information
To make sure the IP4 analyzer finds the IP4 information automatically, you should make sure that a read of the Configuration ROM and an ARP request (which stands for address-resolution protocol) for the node id of the unit are recorded. The Configuration-ROM information is normally read after a bus reset. ARP packets are sent if an IP4 unit wants to communicate with another IP4 unit, but doesn't know the 1394 node id of that unit yet. One way to do this is to (re)connect the IP4 device while the Recorder is recording data. You should also take care that this recorded information is not removed from the recorder buffer because of recorder-buffer overflow (cyclic recorder buffer). One way to do this is to stop the recorder before the cyclic buffer (the buffer part before the trigger position) fills completely, or by generating a trigger before this part is filled completely (see the Recorder for more information).

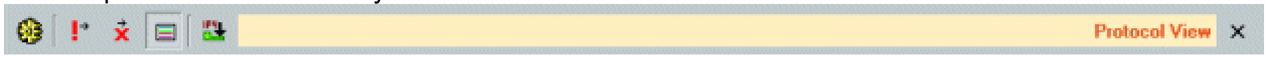
Manually inputting IP4 information
If no Configuration-ROM reads or ARP packets are recorded, you will need to input this information manually. Also if only part of this information can be found automatically, you will need to add information manually. You can do this in the 'Protocol Settings' dialog. It can be opened by clicking the 'Protocol Settings' button on the toolbar (see below). This dialog shows information used by the all-protocol analyzers. It initially shows all information that it found automatically and you can change or add information manually. See the description of the 'IP4-Protocol Settings' below for more information.

16.1.2. Details

The Protocol View consists of a toolbar on the top, with three parts below it: the 'relations' pane, the details pane and the 'transactions and packets' pane.

16.1.2.1. Toolbar

At the top of the Protocol View you will find a toolbar with a few buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol analysis settings have been modified.

Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See 'IP4-Protocol Settings' below for more information.

Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

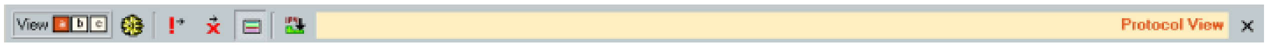
Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane at the right side of the Protocol View.

Export IP4 packets
Clicking this button will open an export to file dialog that can be used to export all packets found by the IP4 analyzer. Use this dialog to select the location and name of a file. After pressing OK the packet data will be written to this file using the LibPCap file format (UNIX standard Library Packet Capture format). Since there's no option currently in this file format for 1394 packets, the Ethernet format is abused: the source and destination mac address in the dump are used to hold the specification id's of the 1394 nodes. An Ethernet mac address is 6 bytes wide, so the first 2 bytes are set to all zeros. The timing information for each packet is added to a start date of 15 January 1970. Because the file format used is essentially for Ethernet packets, ARP requests using the 1394 format can't be stored in the file and are ignored.

An export file can be used to further analyze the data using an external application, for example Wireshark, a free application that can be used to record and view IP4 packets (www.wireshark.org). Within Wireshark, select File > Open, and select the exported data file.

Triple Analyzer
For triple Analyzers, the Toolbar looks as follows:

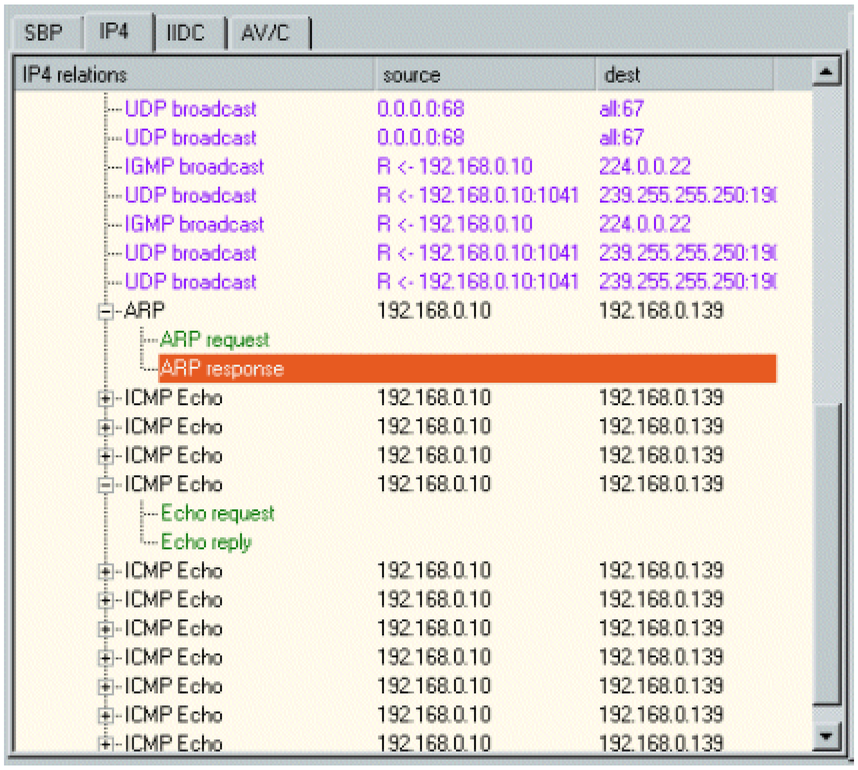


It contains the following additional control:

Analyzer node select control
With this control you are able to select the protocol analysis result of each supported Analyzer node.

16.1.2.2. Relations Pane

This pane shows all transactions and packets found, grouped into items and displayed in a tree. The tree structure indicates the relation between the items. The items are not necessarily recorded in the same order as displayed in this tree. However, all items inside the same item of the tree (at the same level) are listed in the same order as the first corresponding packet of each item was recorded. To find out the order of corresponding transactions and packets, see the 'transactions and packets' pane or take a look at one of the other views of the Recorder (e.g. the Transactions View).



IP4 relations	source	dest
UDP broadcast	0.0.0.0:68	all:67
UDP broadcast	0.0.0.0:68	all:67
IGMP broadcast	R <- 192.168.0.10	224.0.0.22
UDP broadcast	R <- 192.168.0.10:1041	239.255.255.250:19
IGMP broadcast	R <- 192.168.0.10	224.0.0.22
UDP broadcast	R <- 192.168.0.10:1041	239.255.255.250:19
UDP broadcast	R <- 192.168.0.10:1041	239.255.255.250:19
ARP	192.168.0.10	192.168.0.139
ARP request		
ARP response		
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
Echo request		
Echo reply		
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139
ICMP Echo	192.168.0.10	192.168.0.139

For each found or manually specified NodeSpec there will be a root item in the tree (NodeSpec 0 above). For information about NodeSpec's, see 'IP4-Protocol Settings' below. For each found or manually specified Unit, there will be a Unit item inside the corresponding NodeSpec (Unit 0 in NodeSpec 0 above).

A black item indicates that it has no corresponding transactions or packets (e.g. NodeSpec and Unit above). It just is used for proper grouping of the items in the tree. All colored items however, correspond to one or more transactions or packets. When selecting such an item, the details of that item are displayed in the 'details pane' and the corresponding transactions and packets are highlighted in the 'transactions and packets' pane (see below). Note that bus resets like '' are also displayed in black, but have a corresponding item in the 'transactions and packets' pane.

The type of items that can be displayed are:

- Node: displays a 1394 node with IP4 units inside that node below it. Each node gets a logical node id, which is fixed for the entire recording. This logical node id is called a NodeSpec and is displayed in the tree.
- Unit: displays a IP4 unit inside a 1394 node. The tree shows the unit number and IP address(es) of the unit as seen in packets sent by the unit.
- TCP connection: groups all TCP packets belonging to a single TCP connection.
- ICMP group: groups all ICMP packets belonging to a related series.
- ARP group: groups all ICMP packets belonging to a related series.
- UDP packet: a single UDP packet
- TCP packet: a single TCP packet
- Packet group: if a packet is fragmented the packet will be shown as a tree item with all the fragments listed as children. The children will have added ' - Data' to their name.
- Other IP protocols: any packet the analyzer doesn't handle specially is displayed on its own.

The following color scheme is used for these items:

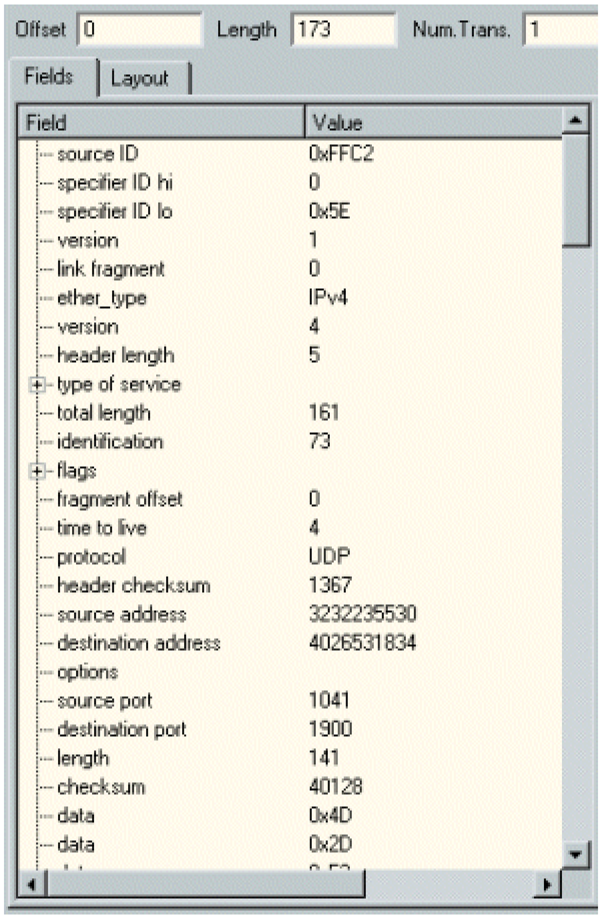
- item group
- item sent by the unit that is part of a group

- item received by the unit that is part of a group
- ungrouped ICMP packet
- un-handled IP packet
- broadcast packet

For each item in the tree two extra columns indicate the source and destination IP address. Where relevant, the port number is appended to the IP address (e.g. for UDP and TCP packets). Some parts of an IP address are in a special form: any part of an IP address ending on a series of .255 will be displayed as .all. For example, 192.168.0.255 will be displayed as 192.168.0.all and 192.168.255.255 will be shown as 192.168.all. Also the IP address 0.0.0.0 is a special case and is displayed as 'none'. In some cases the source and destination may have the symbols 'R<-' or 'R->' prepended to them. This indicates that a packet is sent to or from a unit using a different IP number than the IP number the unit has (which the analyzer assumes to be the IP number used in ARP requests of replies sent by the unit).

16.1.2.3. Details Pane

The middle part of the Protocol View shows the details of the item that is selected in the 'relations' pane. An example that shows the details of an 'UDP broadcast' item follows:



Field	Value
source ID	0xFFC2
specifier ID hi	0
specifier ID lo	0x5E
version	1
link fragment	0
ether_type	IPv4
version	4
header length	5
type of service	
total length	161
identification	73
flags	
fragment offset	0
time to live	4
protocol	UDP
header checksum	1367
source address	3232235530
destination address	4026531834
options	
source port	1041
destination port	1900
length	141
checksum	40128
data	0x4D
data	0x2D

We will describe the different parts below:

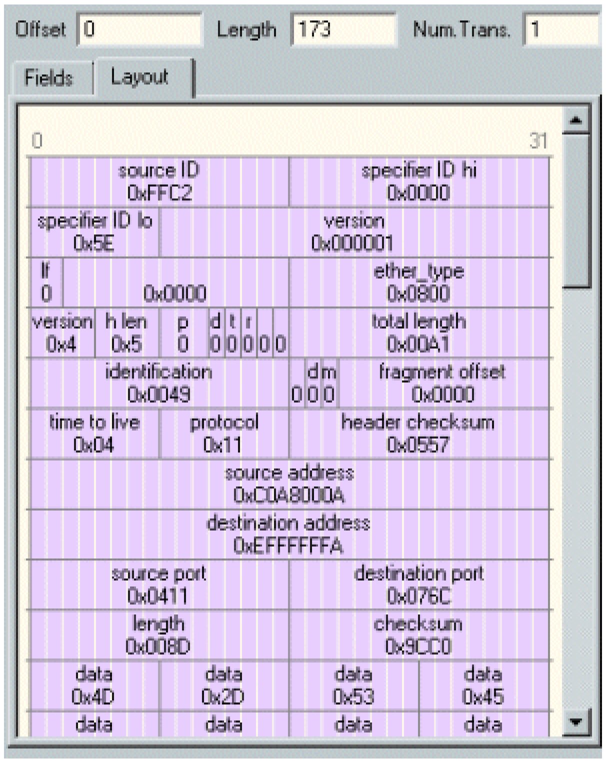
Offset
This box displays the address (offset) of the start of the selected item.

Length
The 'Length' box shows the size of the selected item in bytes.

Num. Trans.
This box shows the number of transactions that corresponds to the selected item. For IP4 this will always be 1.

Fields
The 'Fields' table shows the fields that are defined for the selected item and the corresponding values of these fields. See example above.

Layout
The 'Layout' shows the layout of the selected item. Like the 'Fields' table it shows the fields that are defined for the selected item and the corresponding values of these fields. But now the fields are displayed graphically inside data items. See below for an example of the layout view for the same item as shown in the example above.

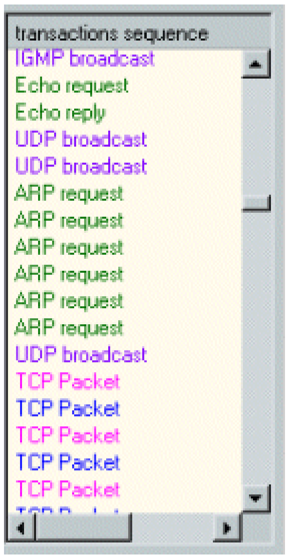


Offset 0 Length 173 Num.Trans. 1

source ID 0xFFC2 | specifier ID hi 0x0000
specifier ID lo 0x5E | version 0x000001
lf 0 | 0x0000 | ether_type 0x0800
version 0x4 | h len 0x5 | p 0 | d 0 | t 0 | r 0 | 0 | 0 | total length 0x00A1
identification 0x0049 | 0 | d 0 | m 0 | fragment offset 0x0000
time to live 0x04 | protocol 0x11 | header checksum 0x0557
source address 0xC0A8000A
destination address 0xEFFFFFFA
source port 0x0411 | destination port 0x076C
length 0x008D | checksum 0x9CC0
data 0x4D | data 0x2D | data 0x53 | data 0x45
data | data | data | data

16.1.2.4. transactions and packets pane

The right most part of the Protocol View shows the sequence of all protocol packets and transactions found (including those for IP4). The order of items in this list corresponds to the order of recording. It is the same order as displayed in the Transaction View or Packets View of the Recorder. An example follows:

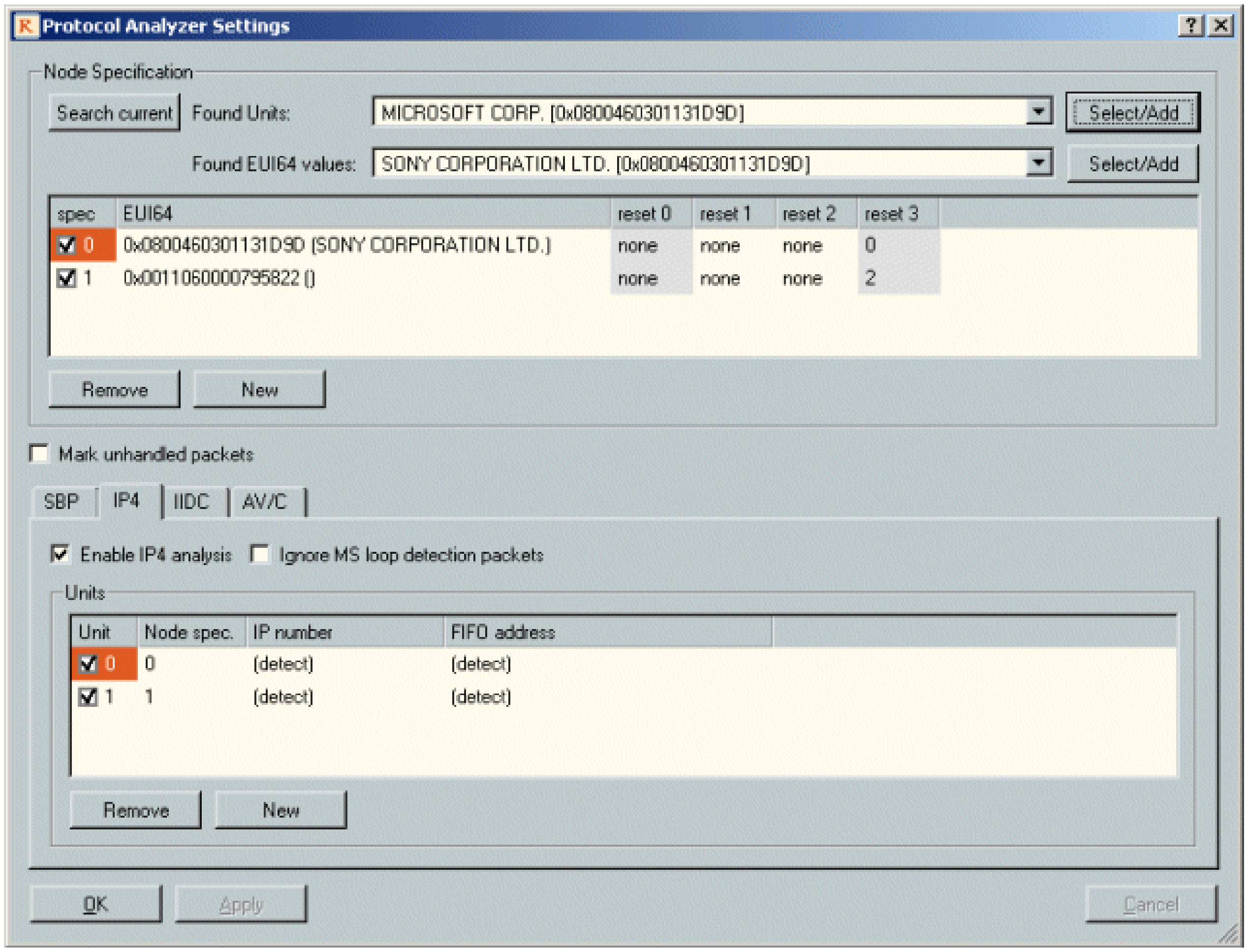


transactions sequence
IGMP broadcast
Echo request
Echo reply
UDP broadcast
UDP broadcast
ARP request
ARP request
ARP request
ARP request
ARP request
ARP request
UDP broadcast
TCP Packet
TCP Packet
TCP Packet
TCP Packet
TCP Packet

Note that the colors identify the type of access and are the same as used in the 'relations' pane (see above). All transactions and packets in this list that are not part of the protocol currently selected in the 'relation' pane are colored gray.
When a item is selected that consists of more than one packet or transaction, the items are highlighted with a gray bar in the list on the 'transactions and packets' pane. (the selected item, which is also part of the item, will still be highlighted with the usual selection color).

16.1.2.5. IP4 Protocol Settings

When clicking the 'Protocol Settings' button in the toolbar of the Protocol View, the 'Protocol Settings' dialog will be displayed. This dialog shows information used by the protocol analyzers to find and correctly analyze the transactions. For each supported protocol, including IP4, there is a tab page with protocol-specific settings. The dialog will be initialized with all information that can be found automatically. If this is not enough for a correct analysis (e.g. the IP4 units' FIFO address was not recorded), you can add this information manually. An example of the dialog is shown below.



Found Units: MICROSOFT CORP. [0x0800460301131D9D]

Found EUI64 values: SONY CORPORATION LTD. [0x0800460301131D9D]

spec	EUI64	reset 0	reset 1	reset 2	reset 3
✓ 0	0x0800460301131D9D (SONY CORPORATION LTD.)	none	none	none	0
✓ 1	0x0011060000795822 ()	none	none	none	2

☐ Mark unhandled packets

✓ Enable IP4 analysis ☐ Ignore MS loop detection packets

Unit	Node spec.	IP number	FIFO address
✓ 0	0	(detect)	(detect)
✓ 1	1	(detect)	(detect)

In the dialog you see an upper part to specify one or more 'Node Specifications', and a lower part to specify protocol-specific information. In between these two parts, general options can be selected.

For a correct analysis the Unit information of the IP4 device to be analyzed must be present. And if no FIFO address was recorded, you also need to specify this for the Units where it is missing. To be able to specify a Unit, you will first need to specify the device (Node Specification) the Unit is part of.
For more information about Node Specifications and the protocol-independent part of the Protocol Settings, see Protocol Settings. For the IP4 protocol-specific settings, see below.

Node Specifications
If no IP4 Units were found automatically, you have to specify one or more Units manually. Before you can specify a Unit you have to specify the node the Unit is part of, using a Node Specification. When the IP4 device is still connected to the bus after recording, you can use the 'Search Current' button to

automatically find IP4 Units. Adding such a Unit will create a Node Specification and Unit. The created Node Specification will be filled automatically when possible. You probably will have to select correct nodes for some reset segments. Make sure you have selected the IP4 node corrrectly for those reset segments for which transaction and packets are recorded that you want to analyze. For more information on Node Specifications, see Protocol Settings.

General options
Mark unhandled packets
For information about this checkbox, please see Protocol Settings.

Protocol-specific information
Enable IP4 analysis
Check this box to enable the IP4 analyzer. All protocol tabs in this dialog have such a checkbox, which can be useful if a recording contains data from several protocols, and you are not interested in some of these.

Ignore MS loop detection packets
While developing the IP4 analyzer at DAP Design, we sometimes made recordings containing IP packets with an 'ether_type' value of 0x0777, which is not in any standard (yet). After some time we were informed that this is a packet with a deliberate invalid signature that contains Microsoft specific data used to detect loops in routed networks. Since the format of these packets is unknown, it might be useful to hide their presence. Checking this box allows the analyzer to ignore such packets.

Units
In the Units table, one or more IP4 Units can be specified. The information in this table normally can be found in the Configuration ROM of the IP4 node. It consists of:

Unit
Using the checkbox before each Unit in the table you can enable or disable each Unit individually for analysis. If the box is not checked, the IP4-related transactions and packets for this Unit will be skipped.

Node Spec.
A Unit is part of a node. As explained before, nodes are specified using Node Specifications. The number in this column corresponds to the Node-Specification numbers of the table in the upper part of the dialog.

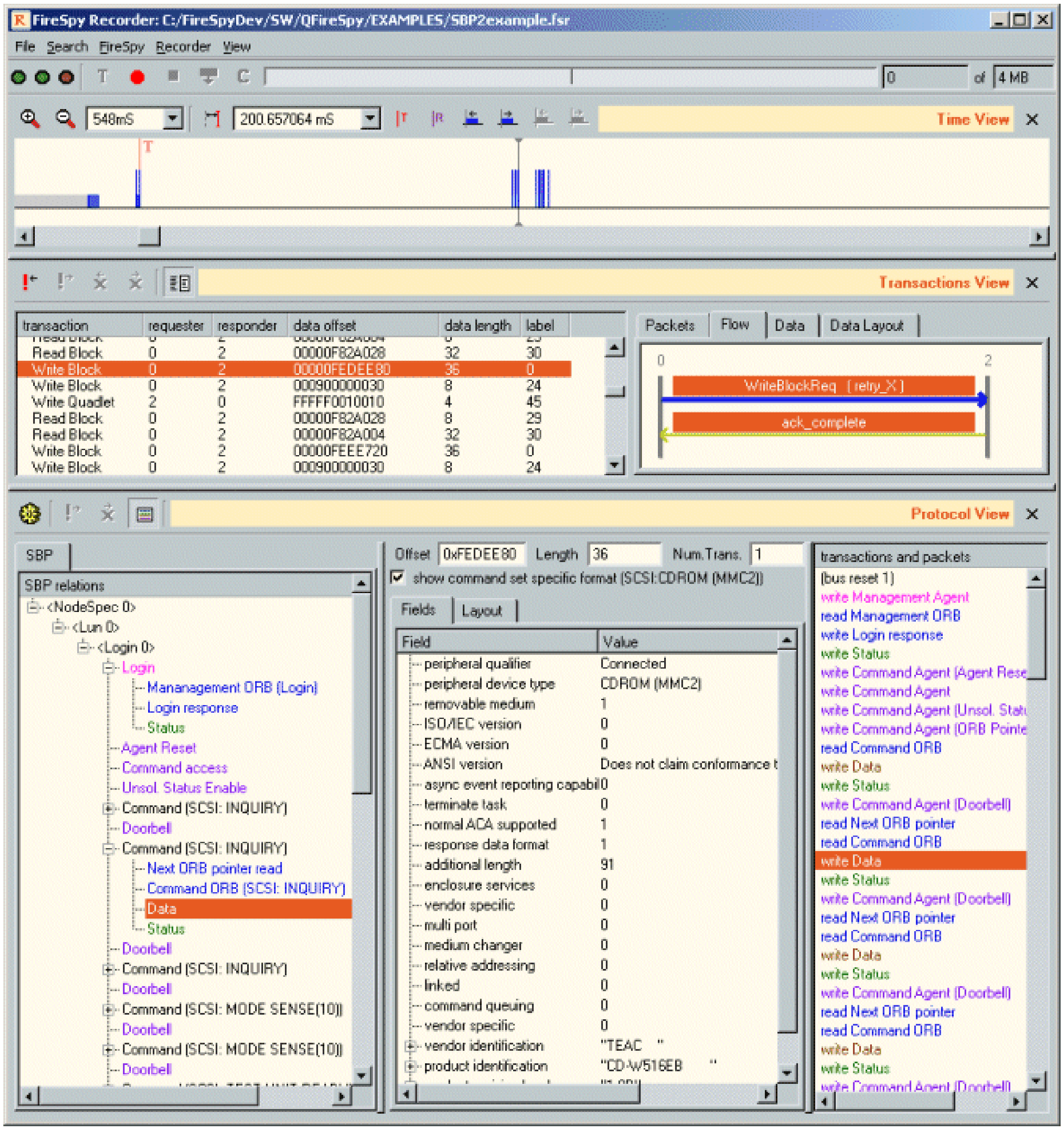
IP number
The IP number of the unit. Any packet sent to or from this node with another IP address is assumed to have been routed through this unit. Such routed packets will have a 'R->' prepended to the source or destination address in the tree view. For this field there are three options:
- Manually specify the IP number of the unit (use a format like w.x.y.z.)
- Let the protocol analyzer use this first IP sent to or from this unit (select '(first)' from the combo box)
- Let the protocol analyzer find the IP number by looking only at ARP packets (select '(detect)' from the combo box)

FIFO address
This is the address in the node where IP4 packets should be sent. This value can be automatically detected using ARP packets (use value 'detect') or manually entered. Note that any value entered is assumed to be in a hexadecimal notation.

Chapter 17. Serial Bus Protocol (SBP)

17.1. Recorder Protocol View

One of the protocols supported by the Protocol View of the Recorder is the Serial-Bus Protocol (SBP). The SBP-protocol analyzer software will scan through all transactions and packets to find SBP-compliant Units and it will fill the 'relations' pane in the SBP tab page with SBP-related items, as described in Protocol View and Protocol Settings. An example of the resulting Protocol View is displayed below.



transaction	requester	responder	data offset	data length	label
Read Block	0	2	00000F82A028	32	30
Write Block	0	2	00000FEDEE80	36	0
Write Block	0	2	000900000030	8	24
Write Quadlet	2	0	FFFFF0010010	4	45
Read Block	0	2	00000F82A028	8	29
Read Block	0	2	00000F82A004	32	30
Write Block	0	2	00000FEEE720	36	0
Write Block	0	2	000900000030	8	24

SBP relations

```
<NodeSpec 0>
  <Lun 0>
    <Login 0>
      Login
        Mananagement ORB (Login)
        Login response
        Status
      Agent Reset
      Command access
      Unsol. Status Enable
      Command (SCSI: INQUIRY)
      Doorbell
      Command (SCSI: INQUIRY)
        Next ORB pointer read
        Command ORB (SCSI: INQUIRY)
        Data
        Status
      Doorbell
      Command (SCSI: INQUIRY)
      Doorbell
      Command (SCSI: MODE SENSE(10))
      Doorbell
      Command (SCSI: MODE SENSE(10))
      Doorbell
```

Offset 0xFEDEE80 Length 36 Num.Trans. 1

☑ show command set specific format (SCSI:CDROM (MMC2))

Field	Value
peripheral qualifier	Connected
peripheral device type	CDROM (MMC2)
removable medium	1
ISO/IEC version	0
ECMA version	0
ANSI version	Does not claim conformance t
async event reporting capabil	0
terminate task	0
normal ACA supported	1
response data format	1
additional length	91
enclosure services	0
vendor specific	0
multi port	0
medium changer	0
relative addressing	0
linked	0
command queuing	0
vendor specific	0
vendor identification	"TEAC "
product identification	"CD-W516EB "

transactions and packets

```
(bus reset 1)
write Management Agent
read Management ORB
write Login response
write Status
write Command Agent (Agent Rese
write Command Agent
write Command Agent (Unsol. Stat
write Command Agent (ORB Pointe
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
```

It displays the result in three different parts. On the right (transactions and packets), all packets and transactions found are displayed in the order they were detected on the bus. The left part (relations) shows the relation between those items. The transactions and packets belonging together (e.g. all transactions that are involved in one command) are grouped into tree items and these items are displayed hierarchically in the 'relations' pane. In this tree you will find for instance:

- Management accesses
- Command-agent accesses
- Orb reads and next orb-pointer re-read
- Data transfers including data tables
- Orb status and unsolicited status
- Bus resets

The details of the item selected in the relations tree, are displayed in the middle part of the Protocol View. It displays the fields defined for the selected item and the values of those fields.

The SBP protocol is in fact a protocol to encapsulate other command-based protocols and supports the movement of large blocks of data. Examples of such encapsulated command sets are the SCSI command sets. The middle part of the SBP Protocol View (details) will show the contents of such commands if the corresponding command set is supported by the Analyzer software. The name of these commands are also shown in the left part (transactions relation).

Currently, the Analyzer supports **version 2** of the SBP protocol (**SBP2**) and the following SCSI command sets are supported:

- CDROM MMC2
- Direct-Access SBC
- Optical-Memory SBC
- Printer SSC
- Processor SPC
- Processor SPC2
- Sequential-Access SSC
- Simplified Direct-Access RBC
- SPC General
- SPC2 General
- Write-Once SBC

17.1.1. How to use it

To display the SBP results, select the SBP tab page in the Protocol View of the Recorder. The Protocol View can be displayed by selecting it in the 'View' menu of the Recorder.

IMPORTANT:
You will need an SBP-Protocol license key to be able to select this view. However, if you open a Recorder file that was made with a Analyzer with a valid SBP license key installed, the Protocol View can be enabled, even if you do not have a valid SBP license key yourself. For more information about license keys, see the section about the License Manager.

The SBP-protocol analyzer needs some information from the Configuration ROM of a SBP device, to be able to analyze the SBP transactions and packets corresponding to this device. It also needs the information of a login command and response to be able to anaylze SBP commands. There are two ways the analyzer can get this information: automatically or manually.

Automatically finding SBP information
To make sure the SBP analyzer finds the SBP information automatically, you should make sure that the reading of Configuration ROM and the login command for the SBP unit are recorded. The Configuration-ROM information is normally read after a bus reset. The login command normally is the first command for an SBP unit.
One way to do this is to (re)connect the SBP device while the Recorder is recording data. You should also take care that this recorded information is not removed from the recorder buffer because of recorder-buffer overflow (cyclic recorder buffer). One way to do this is to stop the recorder before the cyclic buffer (the buffer part before the trigger position) fills completely, or by generating a trigger before this part is filled completely (see the Recorder for more information).
However, a reconnect command may be issued (instead of a login command) when the SBP unit was disconnected for a short period (shorter than 20 seconds). In that case the analyzer misses some

information that was send during the login command and is not resend for a reconnect command. This missing information can be added manually, or you should make sure the SBP unit was disconnected for a period longer than 20 seconds, so that a login command will be recorded.

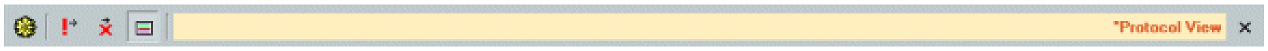
Manually inputting SBP information
If no Configuration-ROM reads login-command is recorded, you will need to input this information manually. Also if only part of this information can be found automatically, you will need to add information manually. You can do this in the 'Protocol Settings' dialog. It can be opened by clicking the 'Protocol Settings' button on the toolbar (see below). This dialog shows information used by the all-protocol analyzers. It initially shows all information that it found automatically and you can change or add information manually. See the description of the 'SBP-Protocol Settings' below for more information.

17.1.2. Details

The Protocol View consists of a toolbar on the top, with three parts below it: the 'relations' pane, the details pane and the 'transactions and packets' pane.

17.1.2.1. Toolbar

At the top of the Protocol View you will find a toolbar with a few buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol-analysis settings have been modified.

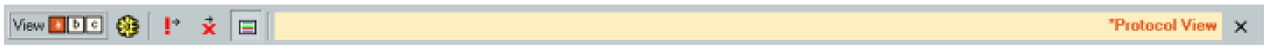
Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See 'SBP-Protocol Settings' below for more information.

Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane at the right side of the Protocol View.

Triple Analyzers
For triple Analyzers the Toolbar looks as follows:



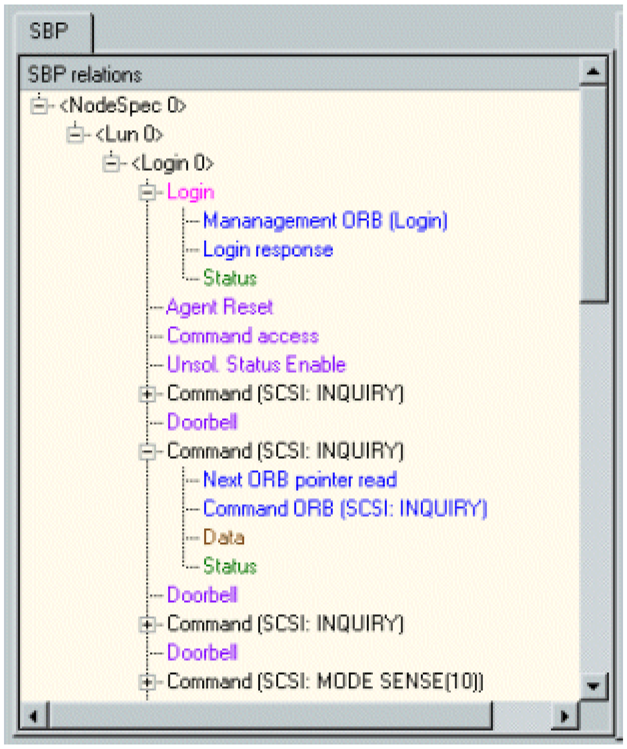
It contains the following additional control:

View a b c

Analyzer node select control
With this control you are able to select the protocol analysis result of each supported Analyzer node.

17.1.2.2. relations pane

This pane shows all transactions and packets found, grouped into items and displayed in a tree. The tree structure indicates the relation between the items. The items are not necessarily recorded in the same order as displayed in this tree. However, all items inside the same item of the tree (at the same level) are listed in the same order as the order of recording the first corresponding packet of each item. To find out the order of corresponding transactions and packets, see the 'transactions and packets' pane or take a look at one of the other views of the Recorder (e.g. the Transactions View).



```
<NodeSpec 0>
  <Lun 0>
    <Login 0>
      Login
        Mananagement ORB (Login)
        Login response
        Status
      Agent Reset
      Command access
      Unsol. Status Enable
      Command (SCSI: INQUIRY)
      Doorbell
      Command (SCSI: INQUIRY)
        Next ORB pointer read
        Command ORB (SCSI: INQUIRY)
        Data
        Status
      Doorbell
      Command (SCSI: INQUIRY)
      Doorbell
      Command (SCSI: MODE SENSE(10))
```

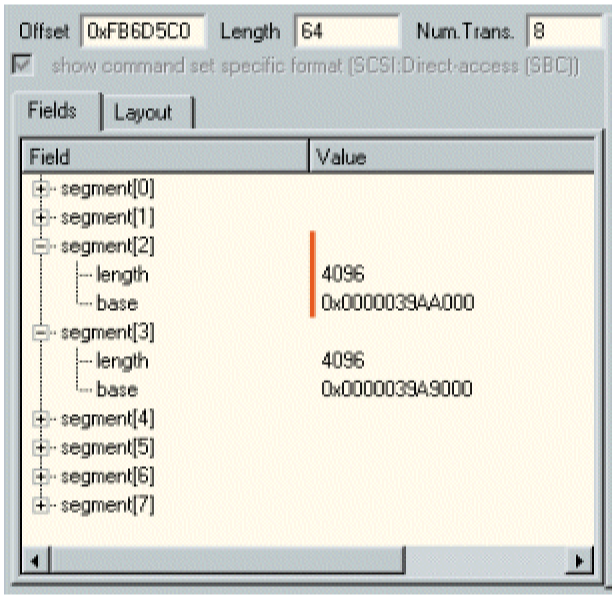
For each found or manually specified NodeSpec there will be a root item in the tree (NodeSpec 0 above). For information about NodeSpec's, see 'SBP-Protocol Settings' below. For each found or manually specified Unit, there will be a Unit item inside the corresponding NodeSpec (Unit 0 in NodeSpec 0 above). And for each found or manually specified login, there will be a Login item inside the corresponding Lun item (Login 0 in Lun 0 above). These NodeSpec, Lun and Login items are colored black in the tree.

A black item indicates that it has no corresponding transactions or packets (e.g. NodeSpec and Unit above). It just is used for proper grouping of the items in the tree. All colored items however correspond to one or more transactions or packets. When selecting such an item, the details of that item are displayed in the 'details pane' and the corresponding transactions and packets are highlighted in the 'transactions and packets' pane (see below). Note that the black 'Command' items (see example above) do not have corresponding transactions. It is also used to group the items belonging to the command. Note also that bus resets like '' are also displayed in black, but have a corresponding item in the 'transactions and packets' pane.
The type of items that can be displayed are (listed in the same color as displayed in the tree):

- item group
- Management accesses
- Command-agent accesses
- Orb reads and next-orb pointer re-read
- Data transfers including data tables
- Orb status and unsolicited status

17.1.2.3. details pane

The middle part of the Protocol View shows the details of the item that is selected in the 'relations' pane. An example that shows the details of a 'Login response' item follows:



Offset 0xFB6D5C0 Length 64 Num.Trans. 8

☑ show command set specific format [SCSI:Direct-access (SBC)]

Fields | Layout

```
Field                 Value
+ segment[0]
+ segment[1]
- segment[2]
    length            4096
    base              0x0000039AA000
- segment[3]
    length            4096
    base              0x0000039A9000
+ segment[4]
+ segment[5]
+ segment[6]
+ segment[7]
```

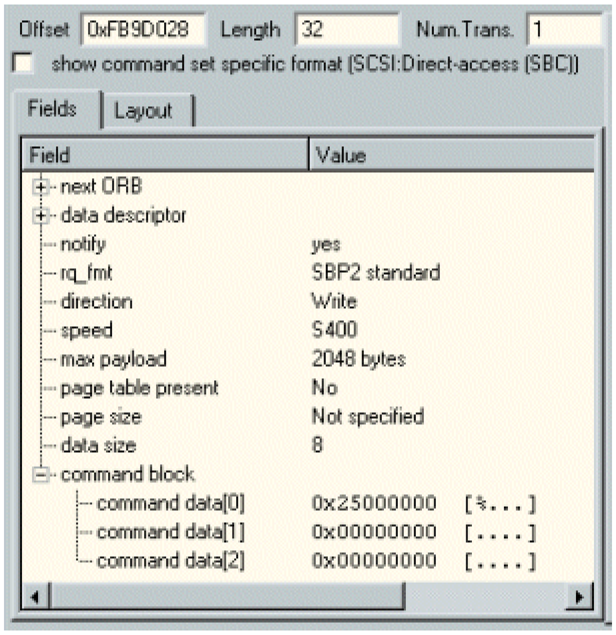
We will describe the different parts below:

Offset
This box displays the address (offset) of the start of the selected item.

Length
The 'Length' box shows the size of the selected item in bytes.

Num. Trans.
This box shows the number of transactions that corresponds to the selected item. In the two examples in the 'Fields' and 'Layout' descriptions below for instance, a data table is selected that consists of 8 transactions.

show command-set-specific format .
When a part of the item contains command-set-specific information, the user has the option to show the command-set-specific part of the item. When the item does not contain command-set-specific information, this option will be disabled. In the two example below, the same item is displayed without and with this option checked.



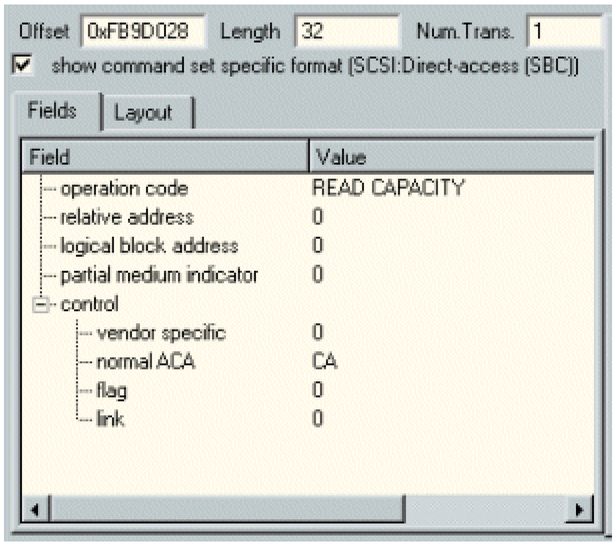
Offset 0xFB9D028 Length 32 Num.Trans. 1

☐ show command set specific format [SCSI:Direct-access (SBC)]

Fields | Layout

```
Field                   Value
+ next ORB
+ data descriptor
  notify                yes
  rq_fmt                SBP2 standard
  direction             Write
  speed                 S400
  max payload           2048 bytes
  page table present    No
  page size             Not specified
  data size             8
- command block
    command data[0]     0x25000000   [%...]
    command data[1]     0x00000000   [....]
    command data[2]     0x00000000   [....]
```

Without the option checked (example above), all the fields of the item are listed, including the command-set-specific part ('command block' in the example above). Note however, that the command-set-specific part is not formatted in this case. It is just the SBP definition of the ORB block.



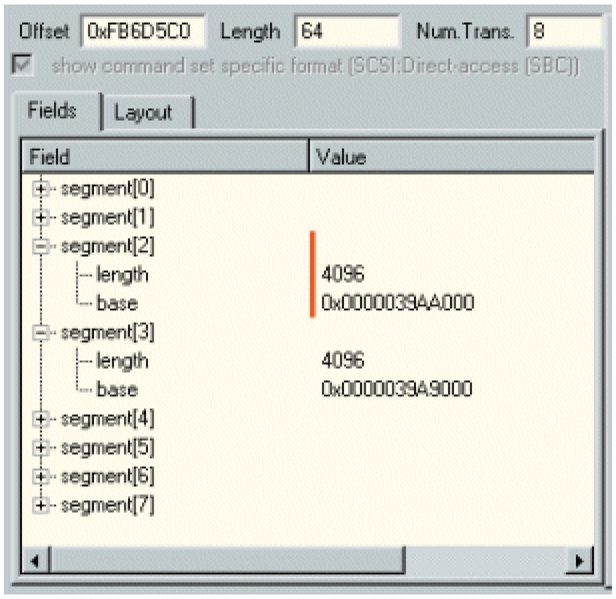
Offset 0xFB9D028 Length 32 Num.Trans. 1

[✓] show command set specific format (SCSI:Direct-access (SBC))

Field	Value
operation code	READ CAPACITY
relative address	0
logical block address	0
partial medium indicator	0
control	
vendor specific	0
normal ACA	CA
flag	0
link	0

When the 'show command-set-specific format' option is selected, only the command-set-specific part is displayed. In the example above, the same item is selected, but now only the 'command block' is shown. It is formatted according to the format set defined in the SBP-Protocol Settings (see below). The used command set is also shown after the check box. In the example above it is the SCSI-Direct-access (SBC) command set.

Fields
The 'Fields' table shows the fields that are defined for the selected item and the corresponding values of these fields.
When a part of the item contains command-set-specific information, the user has the option to show all the item fields (including the unformatted command data) or only the command data formatted corresponding to the defined command set. See 'show command-set-specific format .' above.
If the selected item consists of more then one transaction and one of these transactions is selected (in the 'transactions and packets' pane for instance), the fields corresponding to that transaction are highlighted. In the example below, the selected item consists of 8 transactions and the third transaction of the item was selected, which happens to correspond to the fields of the third table segment.



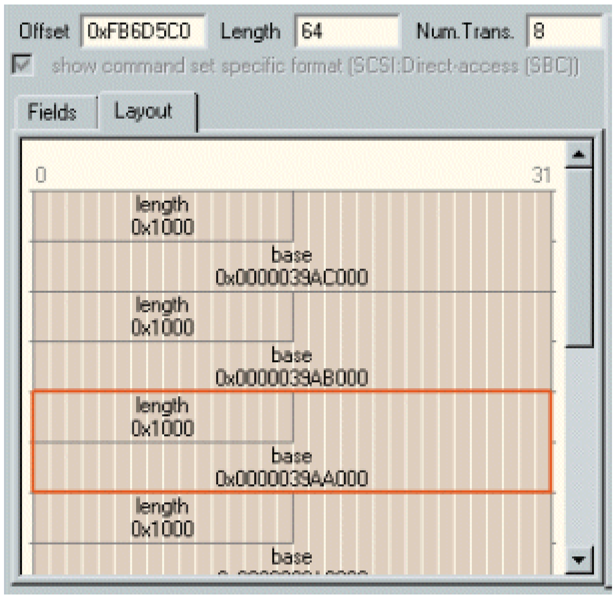
Offset 0xFB6D5C0 Length 64 Num.Trans. 8

[✓] show command set specific format (SCSI:Direct-access (SBC))

Field	Value
segment[0]	
segment[1]	
segment[2]	
length	4096
base	0x0000039AA000
segment[3]	
length	4096
base	0x0000039A9000
segment[4]	
segment[5]	
segment[6]	
segment[7]	

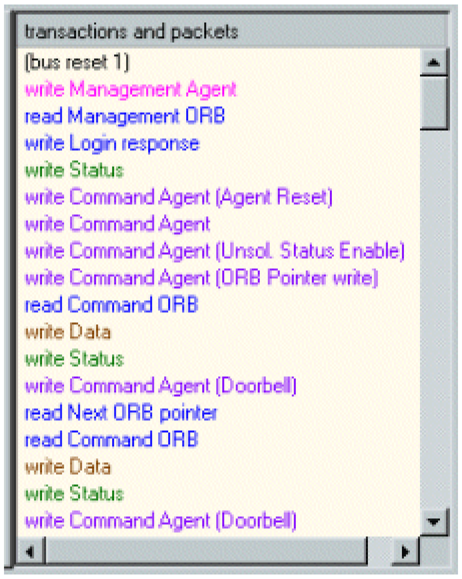
Layout
The 'Layout' shows the layout of the selected item. Like the 'Fields' table it shows the fields that are

defined for the selected item and the corresponding values of these fields. But now the fields are displayed graphically.
When a part of the item contains command-set-specific information, the user has the option to show all the item fields (including the unformatted command data) or only the command data formatted corresponding to the defined command set. See 'show command-set-specific format .' above.
If the selected item consists of more than one transaction and one of these transactions is selected (in the 'transactions and packets' pane for instance), the fields corresponding to that transaction are highlighted. In the example below, the item consists of 8 transactions and the third transaction of the item was selected, which happens to correspond to the fields of the third table segment. It is infact the same situation as the example above, but now the layout tab is selected.


Offset 0xFB6D5C0 Length 64 Num.Trans. 8
☑ show command set specific format (SCSI:Direct-access (SBC))
Fields Layout
0 31
length 0x1000
base 0x0000039AC000
length 0x1000
base 0x0000039AB000
length 0x1000
base 0x0000039AA000
length 0x1000
base

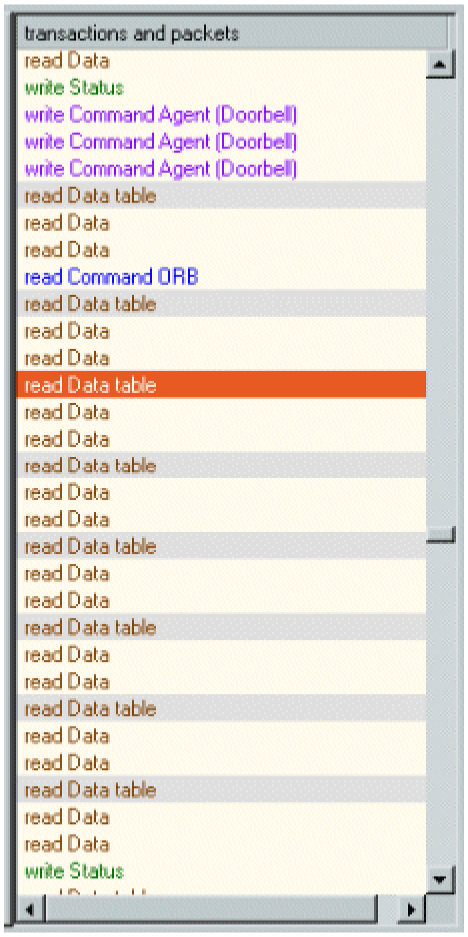
17.1.2.4. transactions and packets pane

The right-most part of the Protocol View shows the sequence of all protocol packets and transactions found (including those for SBP). The order of items in this list corresponds to the order of recording. It is the same order as displayed in the Transaction View or Packets View of the Recorder. An example follows:


transactions and packets
(bus reset 1)
write Management Agent
read Management ORB
write Login response
write Status
write Command Agent (Agent Reset)
write Command Agent
write Command Agent (Unsol. Status Enable)
write Command Agent (ORB Pointer write)
read Command ORB
write Data
write Status
write Command Agent (Doorbell)
read Next ORB pointer
read Command ORB
write Data
write Status
write Command Agent (Doorbell)

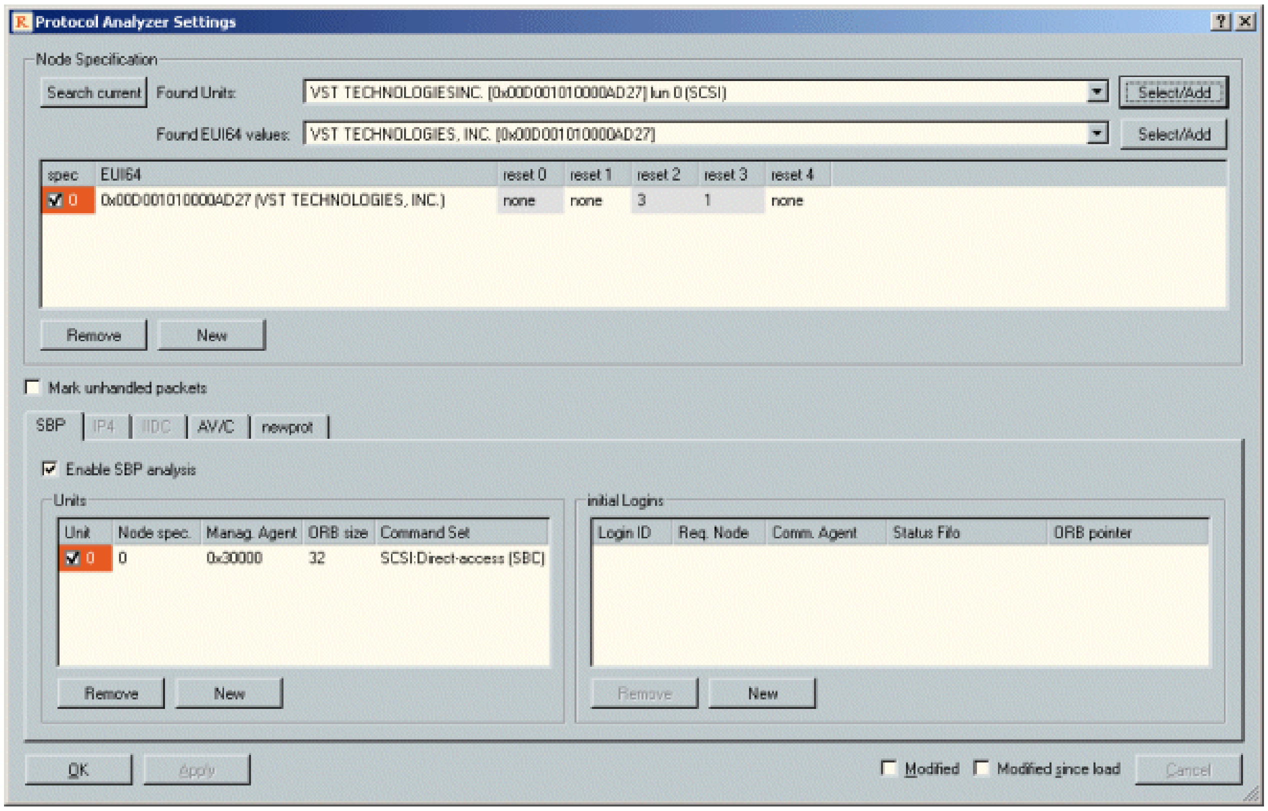
Note that the colors identify the type of access and are the same as used in the 'relations' pane (see above). All transactions and packets in this list that are not part of the protocol currently selected in the 'relation' pane are colored gray.
When an item is selected that consists of more than one packet or transaction, the items are highlighted

in the 'transactions and packets' pane list with a gray bar (the selected item, which is also part of the item will still be highlighted with the usual selection color).
The example below corresponds to the examples in the 'Fields' and 'Layout' descriptions above. Thus the selected item consists of 8 transactions and the third transaction is selected.


transactions and packets
read Data
write Status
write Command Agent (Doorbell)
write Command Agent (Doorbell)
write Command Agent (Doorbell)
read Data table
read Data
read Data
read Command ORB
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
read Data table
read Data
read Data
write Status

17.1.2.5. Protocol Settings

When clicking the 'Protocol Settings' button in the toolbar of the Protocol View, the 'Protocol Settings' dialog will be displayed. This dialog shows information used by the protocol analyzers to find and correctly analyze the transactions. For each supported protocol, including SBP, there is a tab page with protocol-specific settings. The dialog will be initialized with all information that can be found automatically. If this is not enough for a correct analysis (e.g. the correct login was not recorded), you can add this information manually. An example of the dialog is shown below.



Found Units: VST TECHNOLOGIESINC. [0x00D001010000AD27] lun 0 (SCSI)

Found EUI64 values: VST TECHNOLOGIES, INC. [0x00D001010000AD27]

spec	EUI64	reset 0	reset 1	reset 2	reset 3	reset 4
✓ 0	0x00D001010000AD27 (VST TECHNOLOGIES, INC.)	none	none	3	1	none

☐ Mark unhandled packets

✓ Enable SBP analysis

Unit	Node spec.	Manag. Agent	ORB size	Command Set
✓ 0	0	0x30000	32	SCSI:Direct-access (SBC)

Login ID	Req. Node	Comm. Agent	Status Fifo	ORB pointer

☐ Modified ☐ Modified since load

In the dialog you see an upper part to specify one or more 'Node Specifications', and a lower part to specify protocol-specific information. In between these two parts, general options can be selected.
In the protocol-specific part one or more 'Units' can be specified for each protocol. For a correct analysis the Unit information of the SBP device to be analyzed must be present. And if no login command was recorded, you also need to specify an initial login. To be able to specify a Unit, you will first need to specify the device (Node Specification) the Unit is part of.
For more information about Node Specifications and the protocol-independent part of the Protocol Settings, see Protocol Settings. For the SBP-protocol-specific settings, see below.

Node Specifications
If no SBP Units were found automatically, you have to specify one or more Units manually. Before you can specify a Unit you have to specify the node the Unit is part of, using a Node Specification. When the SBP device is still connected to the bus after recording, you can use the 'Search Current' button to automatically find SBP Units. Adding such a Unit will create a Node Specification and Unit. The created Node Specification will be filled automatically when possible. You probably will have to select correct nodes for some reset segments. Make sure you have selected the SBP node corrrectly for those reset segments for which transaction and packets are recorded that you want to analyze.

General Options
Mark unhandled packets
For information about this checkbox, please see Protocol Settings.

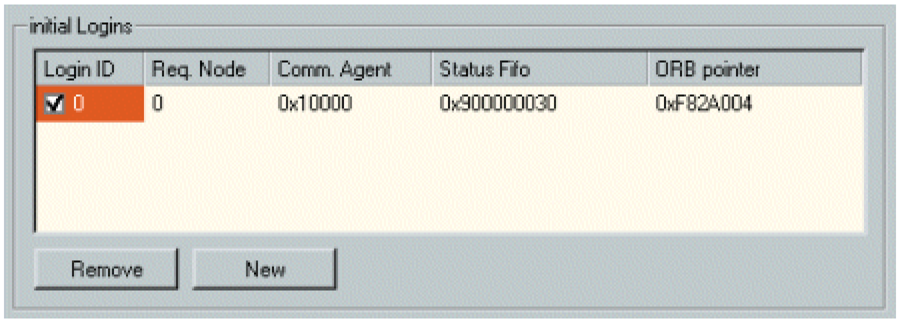
Protocol-specific information
Enable SBP analysis
Check this box to enable the SBP analyzer. All protocol tabs in this dialog have such a checkbox, which can be useful if a recording contains data from several protocols, and you're not interested in some of these.

Units
In the Units table, one or more SBP Units can be specified. The information in this table normally can be found in the Configuration ROM of the SBP node.
It consists of:

- *Unit*: Using the checkbox before each Unit in the table you can enable or disable each Unit individually for analysis. If the box is not checked, the SBP-related transactions and packets for this Unit will be skipped.

- *Node Spec.*: A Unit is part of a node. As explained before, nodes are specified using Node Specifications. The number in this column corresponds to the Node-Specification numbers of the table in the upper part of the dialog.
- *Management Agent*: This is the offset (in the CSR registers) of the Management-Agent registers. If this value can not be found automatically, you could try to figure out this value yourself by looking in the Transactions View of the Recorder and searching for one of the first 8-byte writes to the SBP device. This could be the write of the Management-agent register to start a login request. Just subtract 0xFFFFF0000000 from the offset written to and use that value as the Management-Agent offset value in the table.
- *ORB size*: This is the maximum size of the ORB block in bytes. Normally this is 32.
- *Command Set*: This defines the command set to be used for the command-set-specific formatting. If you have to fill it in manually, you can select one of the supported command sets. If you do not know which command set the SBP device supports, just try one and check for formatting errors after analyzing. If not all commands could be displayed in command-set-specific format, you probably have selected the wrong set.

initial Logins
If the login command (and response) are not recorded, you will have to fill in some login information that normally is obtained by the login command and response of the login command, to be able to analyze SBP commands. Note that if the login command and responses are recorded properly, no 'initial Logins' need to be specified. It is called 'initial' logins, because the analyzer uses this information and starts as if a login command and response were detected with the specified values, before it starts analyzing the transactions.



Login ID	Req. Node	Comm. Agent	Status Fifo	ORB pointer
✓ 0	0	0x10000	0x900000030	0xF82A004

The information consists of:

- *Login ID:* This is the login ID normally returned by the login response. Its value is only needed for some Management Agent commands like logout and reconnect. If you do not know the value, use 0. A wrong value results in not matching the logout or reconnect to the correct login. But after analyzing once, you will probably see additional logins in the 'relation' pane that lead you to the correct value.
- *Requesting Node:* This is the node ID of the node that requested the login. This value is needed to correctly identify accesses to the SBP device. If you do not know the value, try to find a Command-Agent write transaction (see below) and use the node ID of the requester of that transaction. Note that it should correspond to the node ID of the SBP device in the first reset segment. After the first reset, this value is not used anymore, because a new login or reconnect will result in a new value.
- *Command Agent:* To be able to detect Command-Agent access (doorbell write, ORB pointer write, etc) the Command-Agent offset is needed. Normally this value is returned by the login response. If a login is not recorded, you have to try to find this value yourself.
  To find this value, search in the Transaction View of the Recorder for write transactions to the SBP device, which write to one of the CSR registers of the SBP device. During the SBP protocol the requesting device probably does write a lot of times to the doorbel register (register offset 0x10 from start of Command Agent and 4 bytes size) or ORB pointer write register (register offset 0x08 form start of Command Agent and size of 8 bytes). From these transaction offsets you can calculate the Command-Agent offset (subtract 0xFFFFF0000000 and the little register offset mentioned above).
- *Status Fifo:* The address to which the SBP device should write status information is normally given in the login command. So to correctly analyze the commands that include status information, while a login command is not recorded, you should find out this value yourself.
  Search in the Transactions View for write transactions of 8 bytes (mostly) to maximal 32 bytes (seldom), with the SBP device as requester. If you see many of these transactions that write to the same address,

this could be the status fifo address. Fill it in, try it and if status formatting errors occur, you probably have the wrong one.
- *ORB pointer:* This field is needed if the first write to the 'ORB pointer write' register was not recorded. If your login was not recorded, this one is probably not recorded either. If you found the Command Agent above, it is easy to find this one. Just search for the first write transaction to the 'ORB pointer write' (offset 0x08 from Command Agent offset). And use the value of the data that is written (the 'ORB offset').

17.2. Editing SBP2 Formats

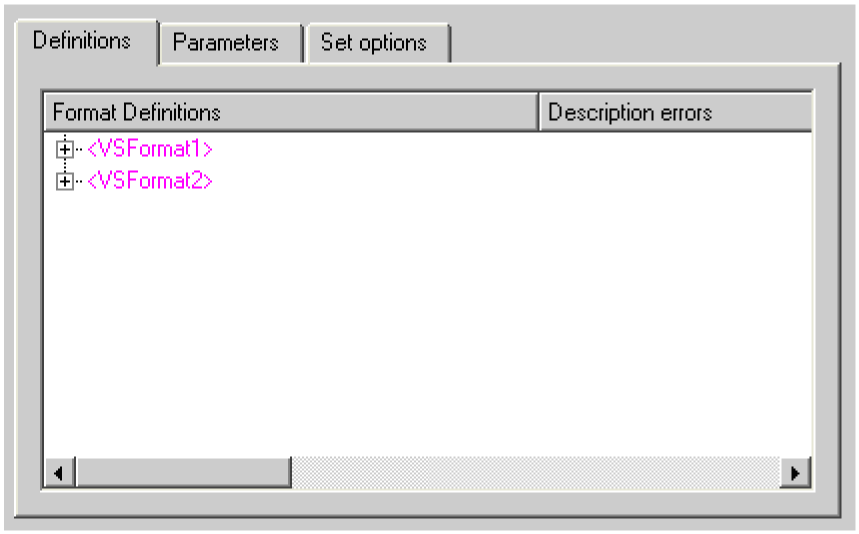
The Analyzer software supports a number of standard SCSI command sets for the SBP2 protocol. If you however need support for a command set not supported by the Analyzer SBP2 software, then you can define your own custom command set using the FormatEditor.

For each custom command set you have to create a file with a number of format defintions, called a format set. The defined command set can be used by the Recorder SBP2 analyzer to display the commands and associated data and status in the correct format. It also can be used in the Filter Sets for the Filter/Trigger logic to define conditions on the fields in particular command packets or data packets.

You can create a new format set with the FormatEditor and start building your format defintions from scratch. But probably the easiest way is to open an existing format file for the SBP2 protocol and adapt it to your needs.

17.2.1. Example

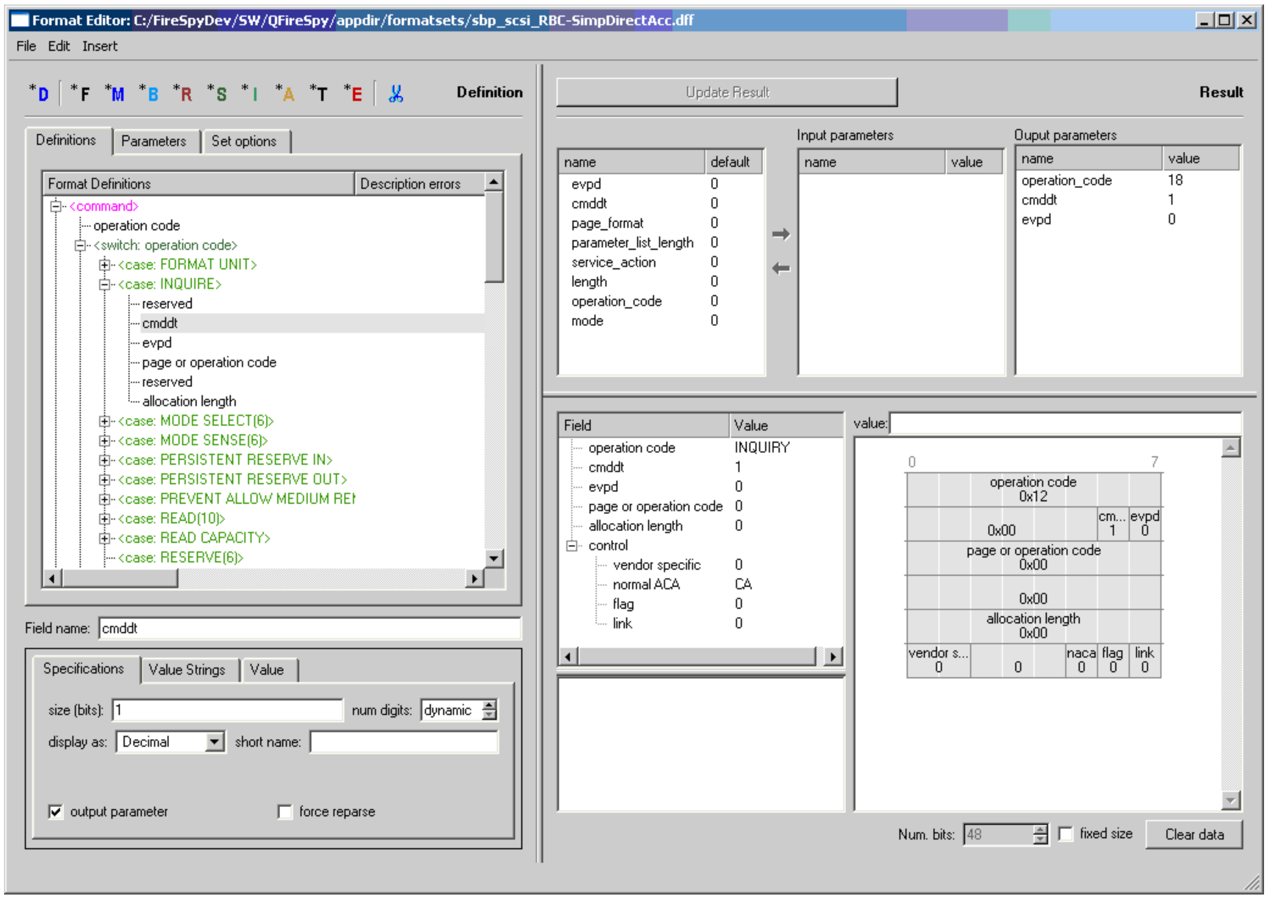
Below the file sbp_scsi_RBC-SimpDirectAcc.dff has been loaded as an example of a SBP2 format set.



The upper tree format defintions (color magenta) are definitions available for the Analyzer. The 'command' and 'data' format definitions must be present to implement a correct command set. The format definition 'status' is only needed when the status format differs from the standard SBP2 status format. The other defintions (color blue) are macro definitions, used by the upper tree definitions.

17.2.1.1. Command

The format definition 'command' defines the format of the command part of a SBP2 command ORB block.
Below the 'command' definition is shown in more detail.



Format Definitions	Description errors
<command>	
operation code	
<switch: operation code>	
<case: FORMAT UNIT>	
<case: INQUIRE>	
reserved	
cmddt	
evpd	
page or operation code	
reserved	
allocation length	
<case: MODE SELECT(6)>	
<case: MODE SENSE(6)>	
<case: PERSISTENT RESERVE IN>	
<case: PERSISTENT RESERVE OUT>	
<case: PREVENT ALLOW MEDIUM REI	
<case: READ(10)>	
<case: READ CAPACITY>	
<case: RESERVE(6)>	

Field name: cmddt

size (bits): 1 num digits: dynamic
display as: Decimal short name:
[✓] output parameter [ ] force reparse

name	default
evpd	0
cmddt	0
page_format	0
parameter_list_length	0
service_action	0
length	0
operation_code	0
mode	0

Input parameters

name	value

Ouput parameters

name	value
operation_code	18
cmddt	1
evpd	0

Field	Value
operation code	INQUIRY
cmddt	1
evpd	0
page or operation code	0
allocation length	0
control	
vendor specific	0
normal ACA	CA
flag	0
link	0

value:

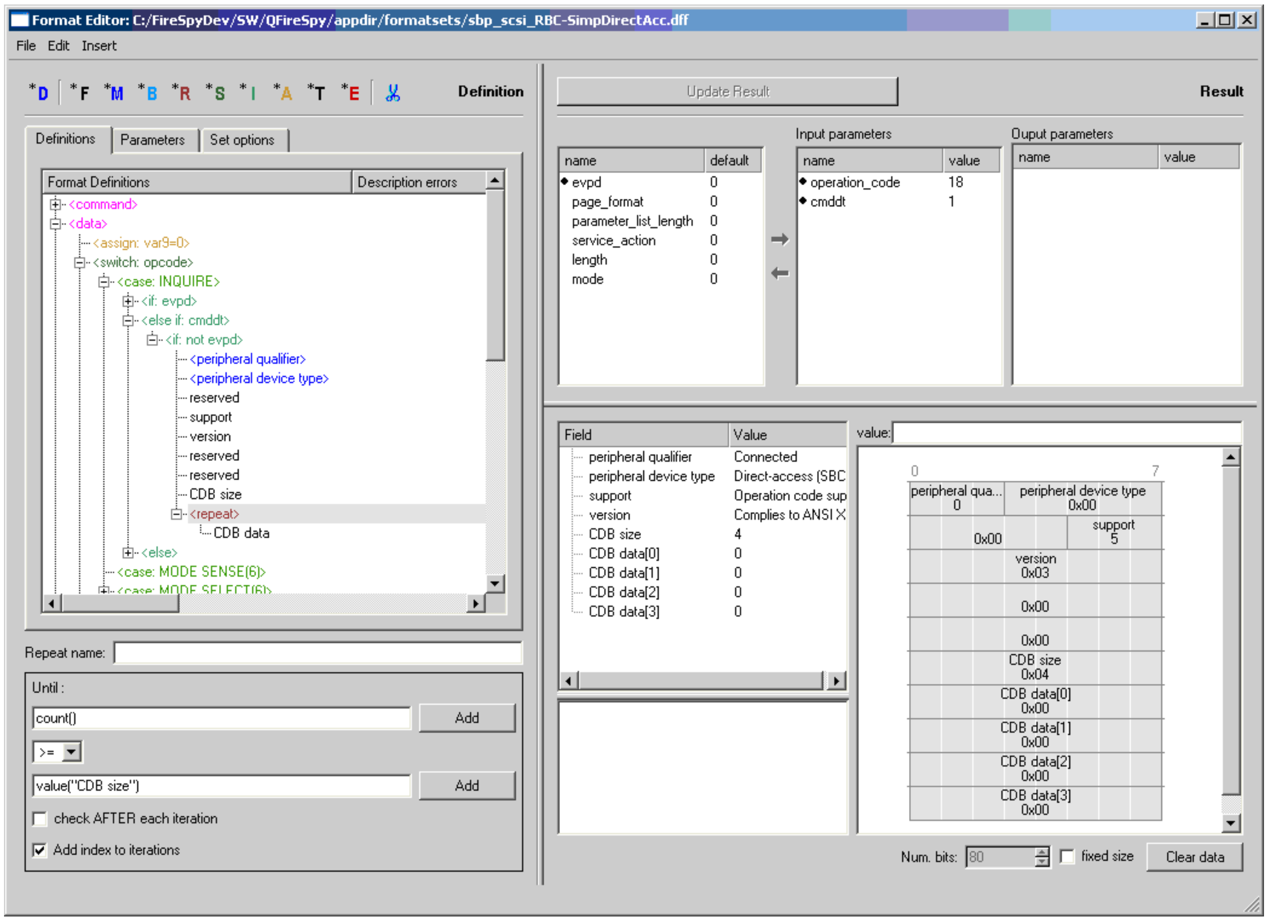
0 … 7					
operation code 0x12					
0x00				cm... 1	evpd 0
page or operation code 0x00					
0x00					
allocation length 0x00					
vendor s... 0	0		naca 0	flag 0	link 0

Num. bits: 48 [ ] fixed size

The command always starts with an operation code field. The format of the remaining bytes depend on the value of this operation code. A 'switch' item is used to implement the different commands. The implementation of the 'INQUIRE' command is shown ('case INQUIRE' item expanded) in more detail. One of its field is the 'cmddt' field. This field is selected and thus the properties of it are dispayed below the definitions tree. There you can see that it is a one bit field. Notice also that the 'output parameter' checkbox has been checked, meaning that the formatting of this command packet will result in an output parameter called 'cmddt' with the value of that bit. We will explain below why this is needed.
In the right pane ('Result' pane), the result of the selected definition ('command') is shown. We see the 'operation code' field at the start. For the 'operation code' field we filled in the value 18 to test the 'INQUIRE' command. It is followed by a reserved field (only visible in the layout view), followed by the one bit 'cmddt' field, etc. As you can see for this command tree output parameters are generated: 'operation_code', 'cmddt' and 'epvd'. These parameters are used to be able to format the data associated with the command (see below).

17.2.1.2. Data

The format definition 'data' defines the format of the data block associated with a particular command. The format of the data depends on the value of the operation code of the command and depending on the command type, it may depend on other fields from the command. Below the 'data' is shown in more detail.



Format Editor: C:/FireSpyDev/SW/QFireSpy/appdir/formatsets/sbp_scsi_RBC-SimpDirectAcc.dff

File Edit Insert

Definition | Update Result | Result

Definitions | Parameters | Set options

```
Format Definitions                      Description errors
⊞ <command>
⊟ <data>
   <assign: var9=0>
   ⊟ <switch: opcode>
      ⊟ <case: INQUIRE>
         ⊞ <if: evpd>
         ⊟ <else if: cmddt>
            ⊟ <if: not evpd>
               <peripheral qualifier>
               <peripheral device type>
               reserved
               support
               version
               reserved
               reserved
               CDB size
               ⊟ <repeat>
                  CDB data
         ⊞ <else>
      <case: MODE SENSE(6)>
      ⊞ <case: MODE SELECT(6)>
```

Repeat name:

Until :

count() [Add]

>=

value("CDB size") [Add]

☐ check AFTER each iteration

☑ Add index to iterations

name	default
evpd	0
page_format	0
parameter_list_length	0
service_action	0
length	0
mode	0

Input parameters

name	value
operation_code	18
cmddt	1

Ouput parameters

name	value

Field	Value
peripheral qualifier	Connected
peripheral device type	Direct-access (SBC
support	Operation code sup
version	Complies to ANSI X
CDB size	4
CDB data[0]	0
CDB data[1]	0
CDB data[2]	0
CDB data[3]	0

value:

```
0                                   7
peripheral qua... | peripheral device type
       0          |        0x00
          0x00              | support 5
          version 0x03
             0x00
             0x00
          CDB size 0x04
          CDB data[0] 0x00
          CDB data[1] 0x00
          CDB data[2] 0x00
          CDB data[3] 0x00
```

Num. bits: 80 ☐ fixed size [Clear data]

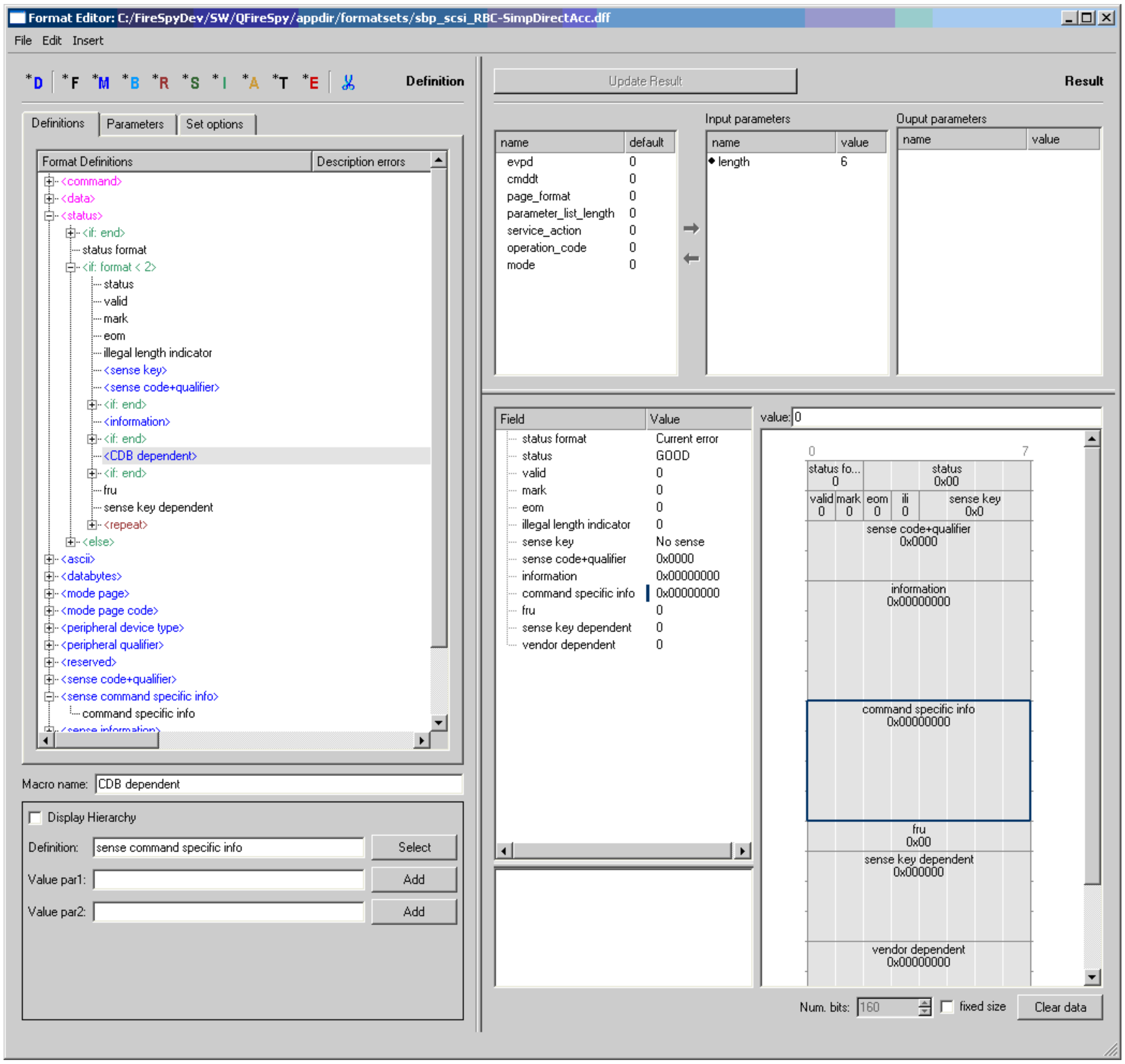
As for the 'command' format, the 'data' format depends on the 'operation code'. A 'switch' item is used to implement the formats for the different operation codes. The value of the operation code is however not included in the packet (as it was for the 'command' format). In fact the operation code value is the value of the operation code in the command packet. Therefore the 'operation_code' output parameter was generated by the 'command' format. For the 'data' format, the 'operation_code' is an input parameter. In fact, all output parameters generated by the corresponding 'command' format are used as input parameters for the 'data' format when the Analyzer formats the 'data' block.
The implementation of the 'INQUIRE' command is shown in more detail. The format for the data of this 'INQUIRE' command depends on some more field values from the associated command block. These fields are 'cmddt' and 'epvd'. Therefore these are input parameters for the 'data' defintion too, and their values will be set to the values of the corresponding output parameters that are generated when the corresponding 'command' block was formatted (in case the 'INQUIRE' command was formatted).

other input parameters
Beside the parameters generated by the corresponding 'command' format, there are also a few other input parameters which can be used by the 'data' format definition:

Input parameter name	**value**
dataSize	This is the value of the 'data size' field of the command ORB block of the corresponding command. It holds the size of the data block in number of bytes.
reading	This is the value of the 'direction' field of the command ORB block of the corresponding command. A value of 1 means that data is transfered 'from' the SBP2 device (note that this corresponds to write transactions from the SBP2 device point of view). A value of 0 means that data is transferws 'to' the SBP2 device (read transactions).

17.2.1.3. Status

The format definition 'status' defines the format of the command set dependent part of the status block. This is the part of the status block after the first two quadlets. It only need to be present when it differs from the standard format for a SBP2 status packet.
Below the 'status' definition is shown in more detail.



```
Format Definitions                Description errors
<command>
<data>
<status>
    <if: end>
    status format
    <if: format < 2>
        status
        valid
        mark
        eom
        illegal length indicator
        <sense key>
        <sense code+qualifier>
        <if: end>
        <information>
        <if: end>
        <CDB dependent>
        <if: end>
        fru
        sense key dependent
        <repeat>
    <else>
<ascii>
<databytes>
<mode page>
<mode page code>
<peripheral device type>
<peripheral qualifier>
<reserved>
<sense code+qualifier>
<sense command specific info>
    command specific info
<sense information>
```

Macro name: CDB dependent

Display Hierarchy

Definition: sense command specific info

Value par1:

Value par2:

name	default
evpd	0
cmddt	0
page_format	0
parameter_list_length	0
service_action	0
operation_code	0
mode	0

Input parameters

name	value
length	6

Ouput parameters

name	value

Field	Value
status format	Current error
status	GOOD
valid	0
mark	0
eom	0
illegal length indicator	0
sense key	No sense
sense code+qualifier	0x0000
information	0x00000000
command specific info	0x00000000
fru	0
sense key dependent	0
vendor dependent	0

value: 0

```
0                                   7
status fo... 0      status 0x00
valid 0  mark 0  eom 0  ili 0   sense key 0x0
sense code+qualifier 0x0000
information 0x00000000
command specific info 0x00000000
fru 0x00
sense key dependent 0x000000
vendor dependent 0x00000000
```

Num. bits: 160   fixed size

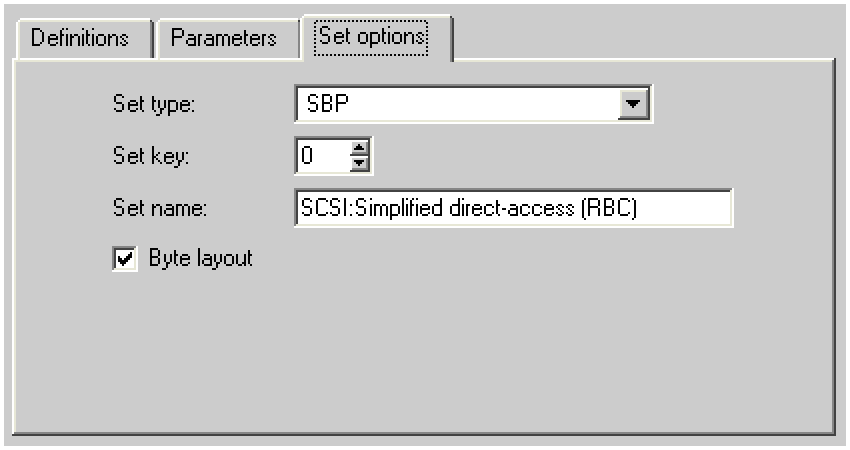
Here we can see an example of the use of a macro definition. The selected item in the definitions tree is a macro call. The properties below show that the macro with name 'sense command specific info' will be called. This macro is shown lower in the definitions tree. In this case it consists of only one field, the 'command specific info' field.

other input parameters
The following input parameters can be used by the 'status' format definition:

Input parameter name	value
length	This is the value of the 'length' field in the first quadlet of the status block. It holds the size of the total status block in number of quadlets minus 1.

17.2.2. Set Options

To be able to use the format definitions in this format set for the SBP2 protocol you need to set the correct 'Set options. Below the 'Set options' of the example file are shown.



Set type: SBP
Set key: 0
Set name: SCSI:Simplified direct-access (RBC)
[✓] Byte layout

The 'Set type' should be 'SBP'. The 'Set key' must be set to 0. The 'Set name' must be unique for all SBP2 type files. So you should fill in some other name here.
When the Analyzer application finds two SBP2 format sets with the same 'Set name' then a warning will be and the second one will be ignored. This is a valid way to overrule an existing format set.

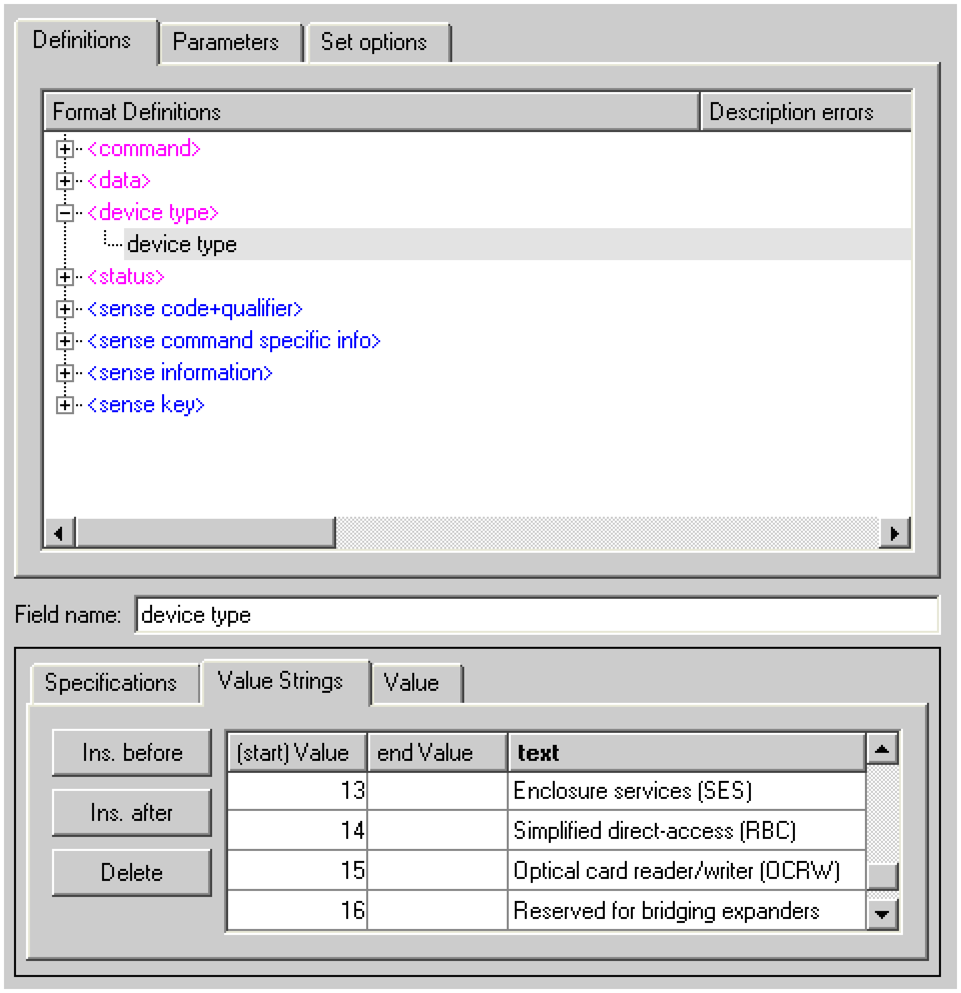
Automatic set recognition
SBP2 command sets can automatically be selected by special values in the CSR ROM. For all SBP2 devices, somewhere in this ROM there must be a device-type value. These device type numbers are associated with a device type name. If this name equals the part of the format set 'Set name' after the " SCSI:" text, then this format set will automatically be used for that SBP2 device.

Currently the following device-types and associated strings are defined:

Device type number	Device type name	Supported by Analyzer?
0	Direct-access (SBC)	Yes
1	Sequential-access (SSC)	Yes
2	Printer (SSC)	Yes
3	Processor (SPC2)	Yes
4	Write-once (SBC)	Yes
5	CDROM (MMC2)	Yes
6	Scanner (SCSI2)	No
7	Optical memory (SBC)	Yes
8	Medium changer (SMC)	No
9	Communication (SCSI2)	No
10	ASC IT8 device	No
11	ASC IT8 device	No
12	Storage array controller (SCC2)	No
13	Enclosure services (SES)	No
14	Simplified direct-access (RBC)	Yes
15	Optical card reader/writer (OCRW)	No
16	Reserved for bridging expanders	-
17	Object-based storage (OSD)	No

Thus the example command set described above will be used automatically when a device type value of 14 has been found in the CSR ROM.

If you want to implement a command set for a device type not listed above and you want that it will be recognized automatically, then you must expand the list above. This list is part of the format set with file name 'sbp_scsi_general.dff'. If you open this file, you will see a format definition named 'device type'. See below.



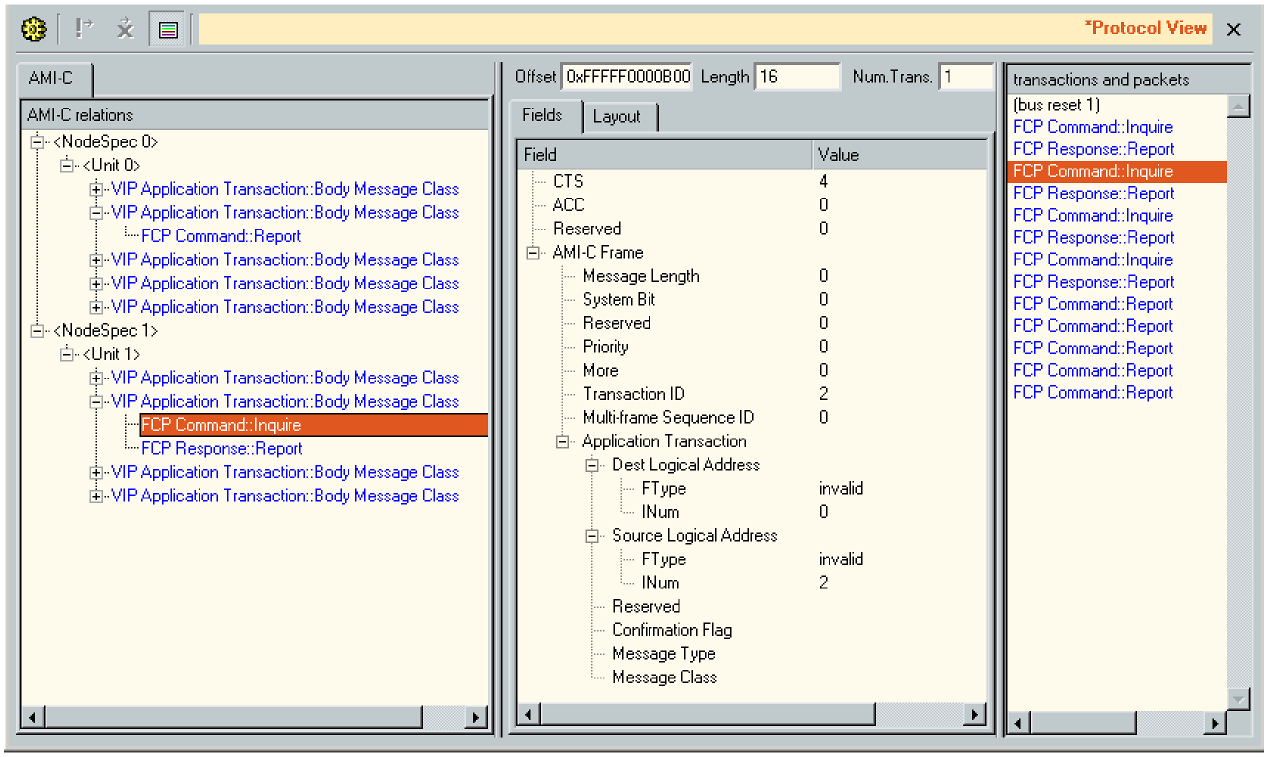
(start) Value	end Value	text
13		Enclosure services (SES)
14		Simplified direct-access (RBC)
15		Optical card reader/writer (OCRW)
16		Reserved for bridging expanders

The one and only field of this format definition has been selected and the properties of this field are shown below the tree. The 'Value Strings' tab of the properties has been selected. This page shows all the defined device type numbers and the associated strings. You may add your strings here. Keep its 'Set name' and 'Set type' the same and save the file at some other place. If you include the path name of the folder, where the file is stored, in the Custom Formats Directories for the Analyzer (see below), then this file will overrule the original and the new device types will be recognized by the Analyzer.

Chapter 18. Automotive Multimedia Interface - Collaboration protocol (AMI-C)

18.1. Recorder Protocol View

One of the protocols supported by the Protocol View of the Recorder is the **Automotive Multimedia Interface - Collaboration (AMI-C) protocol**. The AMI-C protocol-analyzer software will scan through all transactions and packets to find AMI-C-compliant Units and it will fill the 'relations' pane in the AMI-C tab-page with AMI-C-related items. An example of the resulting Protocol View is displayed below.



AMI-C relations
- <NodeSpec 0>
  - <Unit 0>
    - VIP Application Transaction::Body Message Class
    - VIP Application Transaction::Body Message Class
      - FCP Command::Report
    - VIP Application Transaction::Body Message Class
    - VIP Application Transaction::Body Message Class
    - VIP Application Transaction::Body Message Class
- <NodeSpec 1>
  - <Unit 1>
    - VIP Application Transaction::Body Message Class
    - VIP Application Transaction::Body Message Class
      - FCP Command::Inquire
      - FCP Response::Report
    - VIP Application Transaction::Body Message Class
    - VIP Application Transaction::Body Message Class

Offset 0xFFFFF0000B00 Length 16 Num.Trans. 1

Field	Value
CTS	4
ACC	0
Reserved	0
AMI-C Frame	
Message Length	0
System Bit	0
Reserved	0
Priority	0
More	0
Transaction ID	2
Multi-frame Sequence ID	0
Application Transaction	
Dest Logical Address	
FType	invalid
INum	0
Source Logical Address	
FType	invalid
INum	2
Reserved	
Confirmation Flag	
Message Type	
Message Class	

transactions and packets
- [bus reset 1]
- FCP Command::Inquire
- FCP Response::Report
- FCP Command::Inquire
- FCP Response::Report
- FCP Command::Inquire
- FCP Response::Report
- FCP Command::Inquire
- FCP Response::Report
- FCP Command::Report
- FCP Command::Report
- FCP Command::Report
- FCP Command::Report
- FCP Command::Report

It displays the result in three different parts. At the right (transactions and packets), all found packets and transactions are displayed in the order they were detected on the bus. The left part (relations) shows the relation between these items. The transactions and packets belonging together (e.g. all transactions that form a single FCP command) are grouped into tree items and these items are displayed hierarchically in the 'relations' pane. In this tree you will find for instance:

- FCP commands and responses
- Bus resets

The details of the item selected in the relations tree, are displayed in the middle part of the Protocol View. It displays the fields defined for the selected item and the values of those fields.

18.1.1. How to use it

To display the AMI-C results, select the AMI-C tab page in the Protocol View of the Recorder. The Protocol View can be displayed by selecting it in the 'View' menu of the Recorder.

IMPORTANT:
You will need an AMI-C-Protocol license key to be able to select this view. However, if you open a Recorder file that was made with a Analyzer with a valid AMI-C license key installed, the Protocol View can be enabled, even if you do not have a valid AMI-C license key yourself. For more information about license keys, click here.

The AMI-C-protocol analyzer needs some information from the Configuration ROM of a AMI-C device, to be able to analyze the AMI-C transactions and packets corresponding to this device. There are two ways the analyzer can get this information: automatically or manually.

Automatically finding AMI-C information
To make sure the AMI-C analyzer finds the AMI-C information automatically, you should make sure that the reading of Configuration ROM for the node id of the unit is recorded. The Configuration-ROM information is normally read after a bus reset. One way to do this is to (re)connect the AMI-C device while the Recorder is recording data. You should also take care that this recorded information is not removed from the recorder buffer because of recorder buffer overflow (cyclic recorder buffer). One way to do this is to stop the recorder before the cyclic buffer (the buffer part before the trigger position) fills completely, or by generating a trigger before this part is filled completely (see the Recorder for more information).

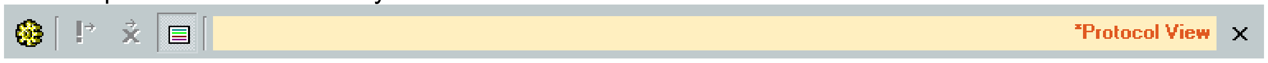
Manually inputting AMI-C information
If no Configuration-ROM reads are recorded, you will need to input this information manually. Also, if only part of this information can be found automatically, you will need to add information manually. You can do this in the 'Protocol Settings' dialog. It can be opened by clicking the 'Protocol Settings' button on the toolbar (see below). This dialog shows information used by the all-protocol analyzers. It initially shows all information that it found automatically and you can change or add information manually. See the description of the 'AMI-C-Protocol Settings' below for more information.

18.1.2. Details

The AMI-C-Protocol View consists of a toolbar on the top, with three parts below it: the 'relations' pane, the details pane and the 'transactions and packets' pane.

18.1.2.1. Toolbar

At the top of the Protocol View you will find a toolbar with a few buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol analysis settings have been modified.

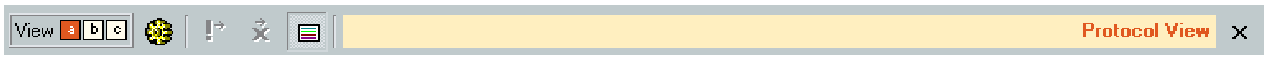
Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzers. It initially shows information that was found automatically and you will be able to change or add information. See ' AMI-C-Protocol Settings' below for more information.

Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane on the right side of the Protocol View.

Triple Analyzer
For triple Analyzers, the Toolbar looks as follows:



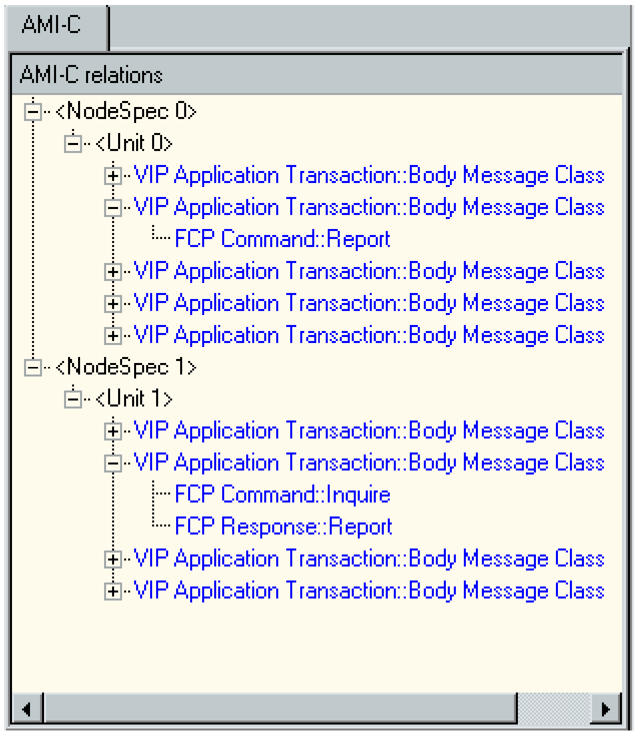
It contains the following additional control:

Analyzer node select control
With this control you are able to select the protocol analysis result of each supported Analyzer node.

18.1.2.2. Relations Pane

This pane shows all transactions and packets found, grouped into items and displayed in a tree. The tree structure indicates the relation between the items. The items are not necessarily recorded in the same order as displayed in this tree. However, all items inside the same item of the tree (at the same level) are listed in the same order as the first corresponding packet of each item was recorded. To find out the order of corresponding transactions and packets, see the 'transactions and packets' pane or take a look at one of the other views of the Recorder (e.g. the Transactions View).
Inside the Unit items, the following items can displayed:

FCP commands usually consist of a command transaction, followed by a response transaction. Both are grouped into the 'FCP command' items. See example below.


AMI-C

AMI-C relations

```
<NodeSpec 0>
    <Unit 0>
        VIP Application Transaction::Body Message Class
        VIP Application Transaction::Body Message Class
            FCP Command::Report
        VIP Application Transaction::Body Message Class
        VIP Application Transaction::Body Message Class
        VIP Application Transaction::Body Message Class
<NodeSpec 1>
    <Unit 1>
        VIP Application Transaction::Body Message Class
        VIP Application Transaction::Body Message Class
            FCP Command::Inquire
            FCP Response::Report
        VIP Application Transaction::Body Message Class
        VIP Application Transaction::Body Message Class
```

For each found or manually specified NodeSpec there will be a root item in the tree (NodeSpec 0 above). For information about NodeSpec's, see 'IP4-Protocol Settings' below. For each found or manually specified Unit, there will be a Unit item inside the corresponding NodeSpec (Unit 0 in NodeSpec 0 above).

18.1.2.3. Details Pane

The middle part of the Protocol View shows the details of the item that is selected in the 'relations' pane. An example that shows the details of an 'AMI-C message' item follows:

When selecting a transaction in the 'Transactions and packets' pane, the details of that transaction are listed this pane. You can choose between a Field view or a Layout view.

We will describe the different parts below:

Offset
This box displays the address (offset) of the start of the selected item.

Length
The 'Length' box shows the size of the selected item in bytes.

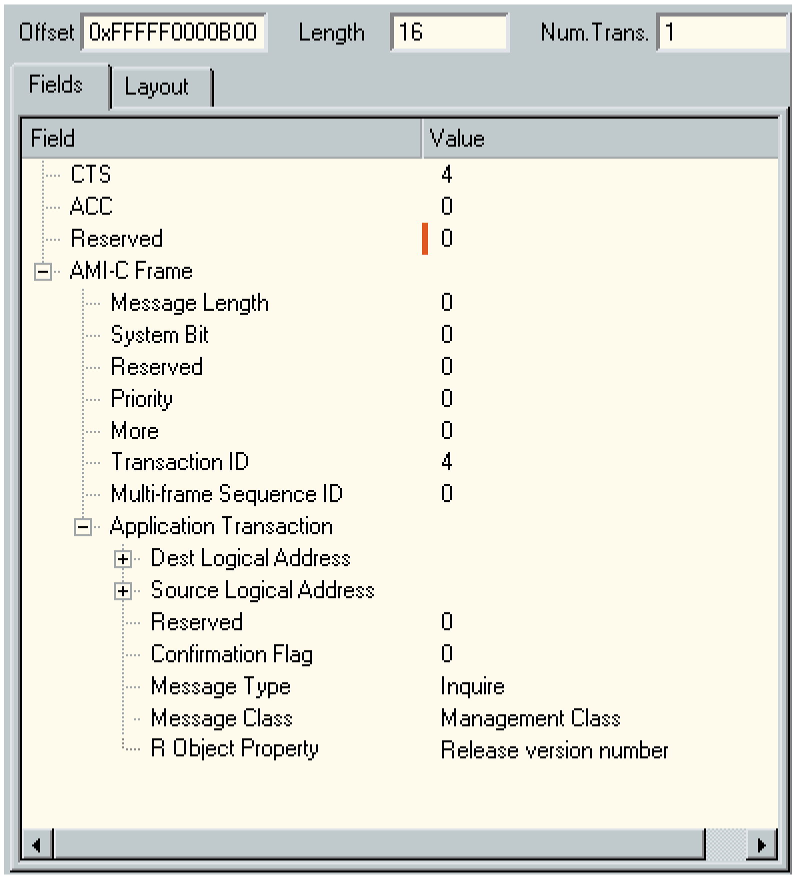
Num. Trans.
This box shows the number of transactions that corresponds to the selected item. For AMI-C this will always be 1.

Fields
The 'Fields' table shows the fields that are defined for the selected item and the corresponding values of these fields.

Layout
The 'Layout' shows the layout of the selected item. Like the 'Fields' table it shows the fields that are defined for the selected item and the corresponding values of these fields. But now the fields are displayed graphically inside data item.

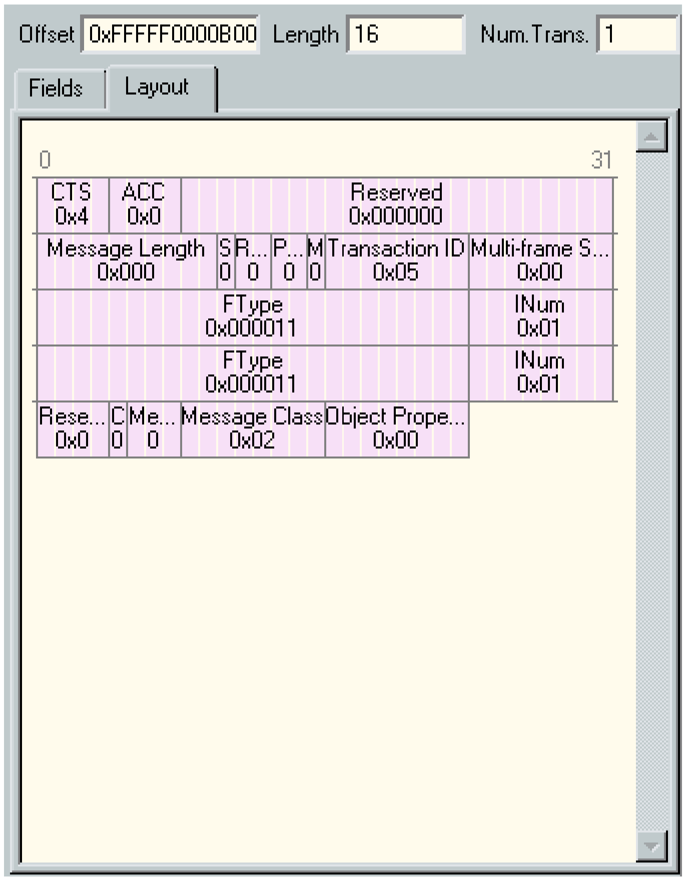
Below we show some field and layout examples of different types of items.

AMI-C command or response
If a AMI-C command transaction or AMI-C response transaction is selected, the fields and the values of these fields defined for the corresponding command or response are shown.
The formatting of the command and response depends on the subunit type, encoded in the command or response. If the Analyzer supports the defined subunit type, the command and response will be formatted corresponding the command set definition of that subunit type. The supported types are listed at the beginning of this manual.
In the two examples below the response of a INQUIRE command is selected. The first example shows the Fields view, the second example shows the Layout view.



Offset 0xFFFFF0000B00 Length 16 Num.Trans. 1

Field	Value
CTS	4
ACC	0
Reserved	0
AMI-C Frame	
Message Length	0
System Bit	0
Reserved	0
Priority	0
More	0
Transaction ID	4
Multi-frame Sequence ID	0
Application Transaction	
Dest Logical Address	
Source Logical Address	
Reserved	0
Confirmation Flag	0
Message Type	Inquire
Message Class	Management Class
R Object Property	Release version number

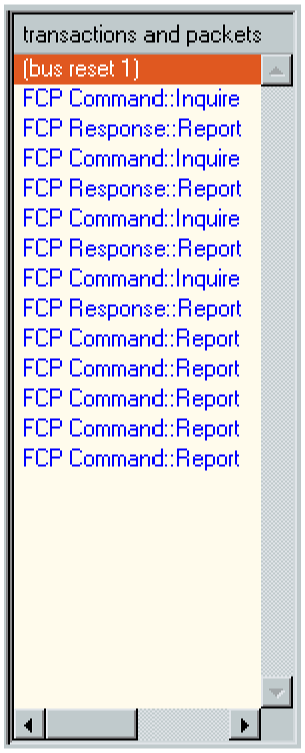


Offset 0xFFFFF0000B00 Length 16 Num.Trans. 1

CTS 0x4	ACC 0x0	Reserved 0x000000					
Message Length 0x000	S 0	R... 0	P... 0	M 0	Transaction ID 0x05	Multi-frame S... 0x00	
FType 0x000011						INum 0x01	
FType 0x000011						INum 0x01	
Rese... 0x0	C 0	Me... 0	Message Class 0x02	Object Prope... 0x00			

18.1.2.4. transactions and packets pane

The right-most part of the Protocol View shows the sequence of all found protocol packets and transactions (including those for AMI-C). The order of items in this list corresponds to the order of recording. It is the same order as displayed in the Transaction View or Packets View of the Recorder. An example follows:

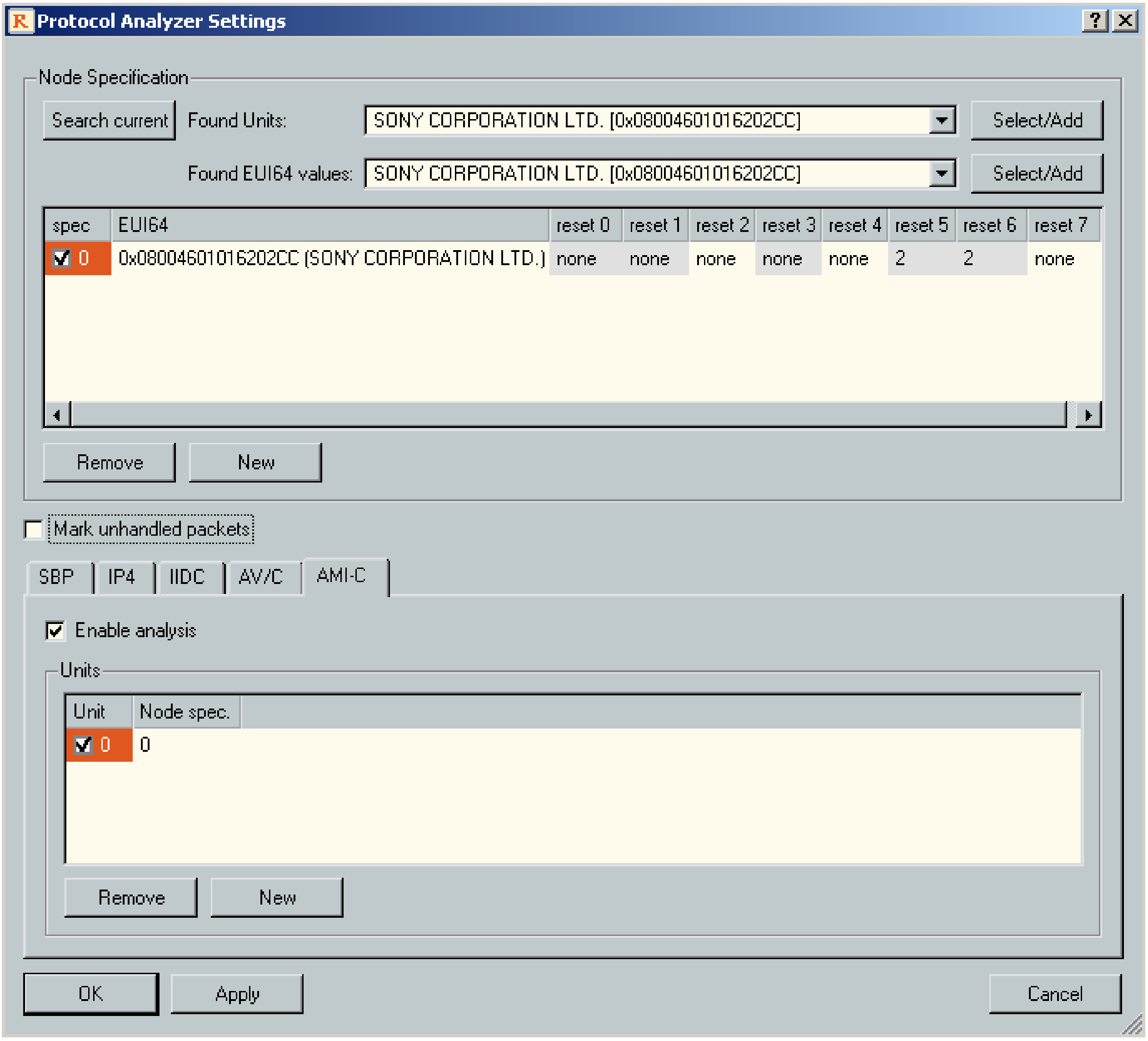


(bus reset 1)
FCP Command::Inquire
FCP Response::Report
FCP Command::Inquire
FCP Response::Report
FCP Command::Inquire
FCP Response::Report
FCP Command::Inquire
FCP Response::Report
FCP Command::Report
FCP Command::Report
FCP Command::Report
FCP Command::Report
FCP Command::Report

When an item is selected that consists of more than one packet or transaction, the items are highlighted in the 'transactions and packets' pane list with a gray bar (the selected item, which is also part of the item will still be highlighted with the usual selection color).

18.1.2.5. AMI-C-Protocol Settings

When clicking the 'Protocol Settings' button in the toolbar of the Protocol View, the 'Protocol Settings' dialog will be displayed. This dialog shows information used by the protocol analyzers to find and correctly analyze the transactions. For each supported protocol, including AMI-C, there is a tab page with protocol-specific settings. The dialog will be initialized with all information that can be found automatically. If this is not enough for a correct analysis, you can add this information manually. An example of the dialog is shown below.


Protocol Analyzer Settings
Node Specification
Search current
Found Units:
SONY CORPORATION LTD. [0x08004601016202CC]
Select/Add
Found EUI64 values:
SONY CORPORATION LTD. [0x08004601016202CC]
Select/Add

spec	EUI64	reset 0	reset 1	reset 2	reset 3	reset 4	reset 5	reset 6	reset 7
☑ 0	0x08004601016202CC (SONY CORPORATION LTD.)	none	none	none	none	none	2	2	none

Unit	Node spec.
☑ 0	0

In the dialog you see an upper part to specify one or more 'Node Specifications', and a lower part to specify protocol specific information. In between these two parts, general options can be selected.
For a correct analysis, the Unit information of the AMI-C device to be analyzed must be present. To be able to specify a Unit, you will first need to specify the device (Node Specification) the Unit is part of.
For more information about Node Specifications and the protocol independent part of the Protocol Settings, see Protocol Settings. For the AMI-C-protocol-specific settings, see below.

Node Specifications
If no AMI-C Units were found automatically, you have to specify one or more Units manually. Before you can specify a Unit, you have to specify the node the Unit is part of using a Node Specification. When the AMI-C device is still connected to the bus after recording, you can use the 'Search Current' button to automatically find AMI-C Units. Adding such a Unit will create a Node Specification and Unit. The created Node Specification will be filled automatically when possible. You probably will have to select correct nodes for some reset segments. Make sure you have selected the AMI-C node correctly for those reset segments for which transaction and packets are recorded that you want to analyze. For more information on Node Specifications, see Protocol Settings.

General options
Mark unhandled packets
For information about this checkbox, please see Protocol Settings.

Protocol specific information

Enable AMI-C analysis

Check this box to enable the AMI-C analyzer. All protocol tabs in this dialog have such a checkbox, which can be useful if a recording contains data from several protocols, and you are not interested in some of these.

Units
In the Units table, one or more AMI-C Units can be specified. The information in this table normally can be found in the Configuration ROM of the AMI-C node. It consists of:

- *Unit:* Using the checkbox before each Unit in the table you can enable or disable each Unit individually for analysis. If the box is not checked, the AMI-C-related transactions and packets for this Unit will be skipped.
- *Node Spec.:* A Unit is part of a node. As explained before, nodes are specified using Node Specifications. The number in this column corresponds to the Node-Specification numbers of the table in the upper part of the dialog.

18.2. Editing AMI-C Formats

The Analyzer software supports AMI-C command sets for the AMI-C protocol compliant with standard ISO 22902. If, however, you need support for a command set not supported by the Analyzer AMI-C software, then you can define your own custom command set using the FormatEditor.

For each custom command set you have to create a format set file, with a format definition for the VIP frame. The defined command set can be used by the AMI-C analyzer to display the commands and associated responses in the correct format. It can also be used in the Filter Sets for the Filter/Trigger logic to define conditions on the fields in particular command or response.

You could create a new format set with the FormatEditor and start building your format defintions from scratch. But probably the easiest way is to open an existing format file for the AMI-C protocol and adapt it to your needs.

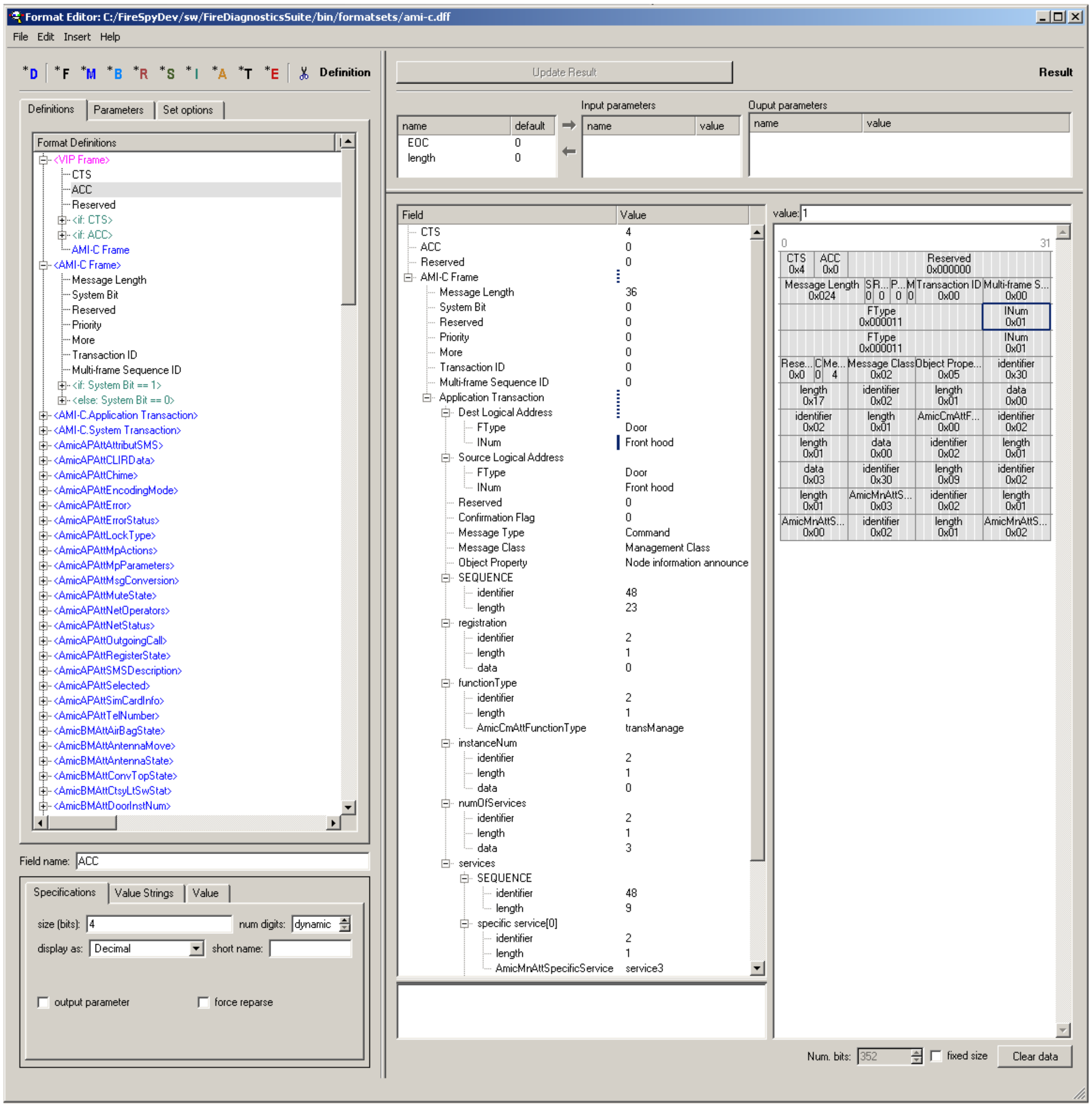
18.2.1. Example

The first format definition, called 'VIP Frame' (color magenta) defines the format for the AMI-C commands and responses. It always needs to be present in a format set for the AMI-C protocol.

The other definitions (color blue) are macro definitions.

VIP Frame

The format definition 'VIP Frame' defines the format of the commands and responses for AMI-C protocol. Below the 'VIP Frame' definition is shown in more detail.


Format Editor: C:/FireSpyDev/sw/FireDiagnosticsSuite/bin/formatsets/ami-c.dff

```
Format Definitions
<VIP Frame>
    CTS
    ACC
    Reserved
    <if: CTS>
    <if: ACC>
    AMI-C Frame
<AMI-C Frame>
    Message Length
    System Bit
    Reserved
    Priority
    More
    Transaction ID
    Multi-frame Sequence ID
    <if: System Bit == 1>
    <else: System Bit == 0>
<AMI-C Application Transaction>
<AMI-C System Transaction>
<AmicAPAttAttributSMS>
<AmicAPAttCLIRData>
<AmicAPAttChime>
<AmicAPAttEncodingMode>
<AmicAPAttError>
<AmicAPAttErrorStatus>
<AmicAPAttLockType>
<AmicAPAttMpActions>
<AmicAPAttMpParameters>
<AmicAPAttMsgConversion>
<AmicAPAttMuteState>
<AmicAPAttNetOperators>
<AmicAPAttNetStatus>
<AmicAPAttOutgoingCall>
<AmicAPAttRegisterState>
<AmicAPAttSMSDescription>
<AmicAPAttSelected>
<AmicAPAttSimCardInfo>
<AmicAPAttTelNumber>
<AmicBMAttAirBagState>
<AmicBMAttAntennaMove>
<AmicBMAttAntennaState>
<AmicBMAttConvTopState>
<AmicBMAttCtsyLtSwStat>
<AmicBMAttDoorInstNum>
```

Field name: ACC

size (bits): 4   num digits: dynamic
display as: Decimal   short name:
output parameter   force reparse

Input parameters

name	default
EOC	0
length	0

```
Field                              Value
CTS                                4
ACC                                0
Reserved                           0
AMI-C Frame
    Message Length                 36
    System Bit                     0
    Reserved                       0
    Priority                       0
    More                           0
    Transaction ID                 0
    Multi-frame Sequence ID        0
    Application Transaction
        Dest Logical Address
            FType                  Door
            INum                   Front hood
        Source Logical Address
            FType                  Door
            INum                   Front hood
        Reserved                   0
        Confirmation Flag          0
        Message Type               Command
        Message Class              Management Class
        Object Property            Node information announce
        SEQUENCE
            identifier             48
            length                 23
        registration
            identifier             2
            length                 1
            data                   0
        functionType
            identifier             2
            length                 1
            AmicCmAttFunctionType  transManage
        instanceNum
            identifier             2
            length                 1
            data                   0
        numOfServices
            identifier             2
            length                 1
            data                   3
        services
            SEQUENCE
                identifier         48
                length             9
            specific service[0]
                identifier         2
                length             1
                AmicMnAttSpecificService  service3
```

value: 1

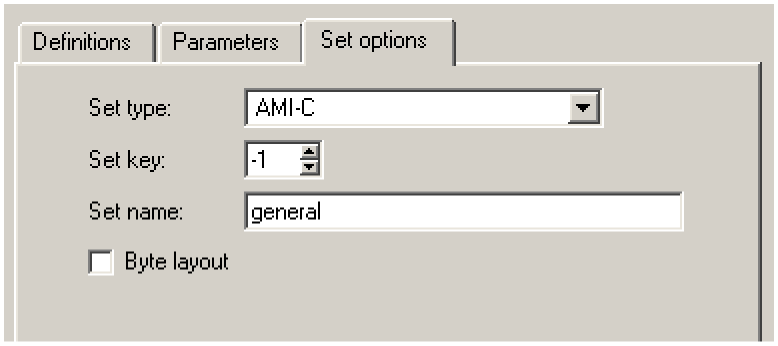
```
0                                                               31
| CTS 0x4 | ACC 0x0 |              Reserved 0x000000              |
| Message Length 0x024 | S 0 | R... 0 | P... 0 | M 0 | Transaction ID 0x00 | Multi-frame S... 0x00 |
|                FType 0x000011                  | INum 0x01 |
|                FType 0x000011                  | INum 0x01 |
| Rese... 0x0 | C 0 | Me... 4 | Message Class 0x02 | Object Prope... 0x05 | identifier 0x30 |
| length 0x17 | identifier 0x02 | length 0x01 | data 0x00 |
| identifier 0x02 | length 0x01 | AmicCmAttF... 0x00 | identifier 0x02 |
| length 0x01 | data 0x00 | identifier 0x02 | length 0x01 |
| data 0x03 | identifier 0x30 | length 0x09 | identifier 0x02 |
| length 0x01 | AmicMnAttS... 0x03 | identifier 0x02 | length 0x01 |
| AmicMnAttS... 0x00 | identifier 0x02 | length 0x01 | AmicMnAttS... 0x02 |
```

Num. bits: 352   fixed size

As you can see above, the 'VIP Frame' format definition starts with fields 'CTS' and 'ACC'. To identify 'VIP Frame' these values have to be 'CTS'=4 and 'ACC'=0. 'AMI-C Frame' is encapsulated in 'VIP Frame' and has two possible transactions:
- Application transaction, 'System Bit'=0,
- System transaction, 'System Bit'=1 and 'Priority'=3.

Set options

To be able to use the format definitions in this format set for the AMI-C protocol you need to set the correct 'Set options'. Below the 'Set options' of the AMI-C format set are shown.



Set type: AMI-C

Set key: -1

Set name: general

☐ Byte layout

The 'Set type' should be 'AMI-C'. The 'Set key' must be set to the subunit type corresponding to the command set you have implemented by this format set. The 'Set name' must be unique for all AMI-C type files. So you should fill in some other suitable name here.
When the Analyzer application finds two AMI-C format sets with the same 'Set name' then a warning will be displayed and the second one will be ignored. This is a valid way to overrule an existing format set.

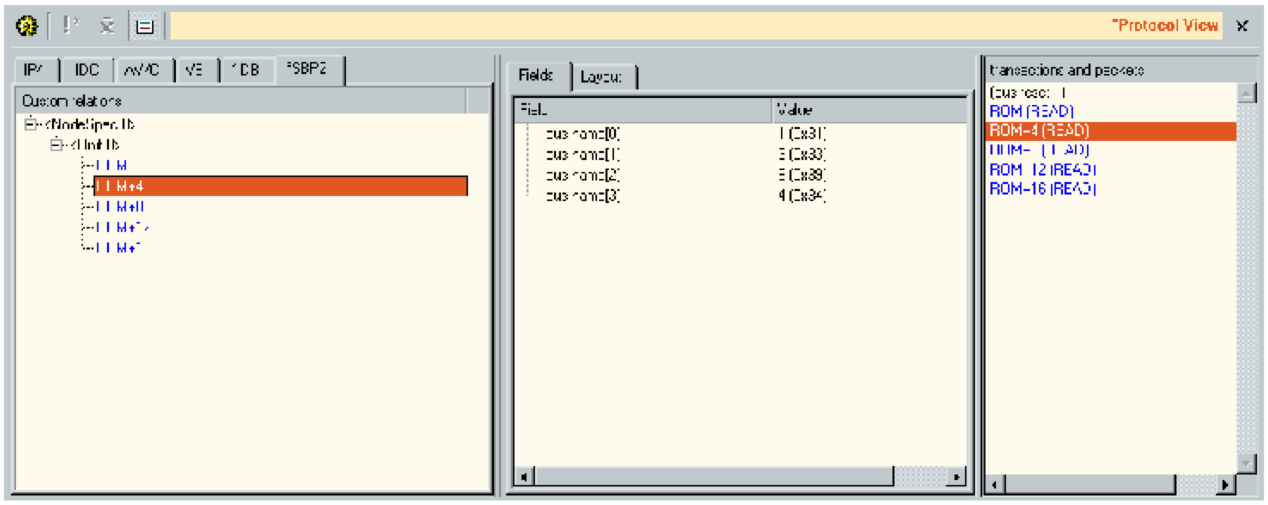
Automatic set recognition

When the Analyzer needs to format an AMI-C command or response block, it uses the value of the 'subunit type' field of that command or response block to select the correct format set and thus the correct formating for the block. It will search for a format set with 'Set type' AMI-C and with a 'Set key' equal to the 'subunit type' field.

Chapter 19. Custom Protocols

Custom-made protocols are supported by the Protocol View of the Recorder. The protocol analyzer takes into account the custom-made protocols for which protocol definitions are available. A protocol definition lay down the behavior of a custom-made protocol. Protocol definitions are created with the Protocol editor.

The protocol analyzer will scan through all transactions to find custom-made protocol compliant items and it will fill the tab page related to the custom-made protocol in the 'relation' pane with these items, as described in Protocol View and Protocol Settings. An example of the resulting Protocol View is displayed below.



19.1. What is it

The custom protocol view displays the result in three different parts. At the right (transactions and packets), all found resets and transactions are displayed in the order they were detected on the bus. The left part (relations) shows the relation between these items. The transactions belonging to tree items are displayed hierarchically in the 'relations' pane. At this moment only one to one relations will exists. The spin-off effect of a write or read transaction is not supported by the protocol definition yet.

The details of the item selected in the relations tree, are displayed in the middle part of the Protocol View. It displays the fields defined for the selected item and the values of those fields.

19.2. How to use it

To display the Custom-made protocol results, select the related tab page in the Protocol View of the Recorder. The Protocol View can be displayed by selecting it in the 'View' menu of the Recorder.

IMPORTANT:
In order to select this view the protocol analyzer must take into account the protocol definitions of the custom-made protocols. The location of the protocol definitions must therefore be specified in the Settings dialog tab page 'Protocols'. The Settings dialog is opened by selecting it in the 'Windows' menu.
The protocol analyzer needs some information from the Configuration ROM of a device, which support the custom-made protocol, to be able to analyze transactions corresponding to this device. There are two ways the analyzer can get this information: automatically or manually.

Automatically finding custom-made protocol information
To make sure the protocol analyzer finds the custom-made protocol information automatically, you should make sure that the Configuration-ROM is read. The Configuration-ROM information is normally read after a bus reset. One way to do this is to (re)connect the device while the Recorder is recording data. You should also take care that this recorded information is not removed from the recorder buffer because of recorder-buffer overflow (cyclic recorder buffer). One way to do this is to stop the recorder before the cyclic buffer (the buffer part before the trigger position) fills completely, or by generating a trigger before

this part is filled completely (see the Recorder for more information).

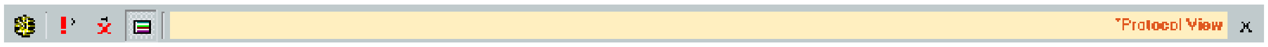
Manually inputting custom-made protocol information
If no Configuration-ROM reads are recorded, you will need to input this information manually. Also if only part of this information can be found automatically, you will need to add information manually. You can do this in the 'Protocol Settings' dialog. It can be opened by clicking the 'Protocol Settings' button on the toolbar (see below). This dialog shows information used by the protocol analyzer. It initially shows all information that it found automatically and you can change or add information manually. See the description of the 'Custom-made Protocol Settings' below for more information.

19.3. Details

The Protocol View consists of a toolbar on the top, with three parts below it: the 'relations' pane, the details pane and the 'transactions and packets' pane.

19.3.1. Toolbar

At the top of the Protocol View you will find a toolbar with a few buttons:



In front of the name 'Protocol View' a star '*' will be displayed if the protocol analysis settings have been modified.

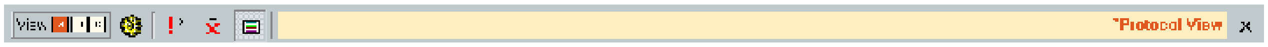
Protocol Settings
When clicking this button, the 'Protocol Settings' dialog will be displayed. It shows information used by the protocol analyzer. It initially shows information that was found automatically and you will be able to change or add information. See 'Custom-made Protocol Settings' below for more information.

Go to next item with warning(s)
When clicking this button, the next item that contains one or more warnings will be selected.

Go to next item with error(s)
When clicking this button, the next item that contains one or more errors will be selected.

Show Transactions and Packets
This button toggles the display of the right pane at the right side of the Protocol View.

Triple Analyzers
For triple Analyzers the toolbar looks as follows:

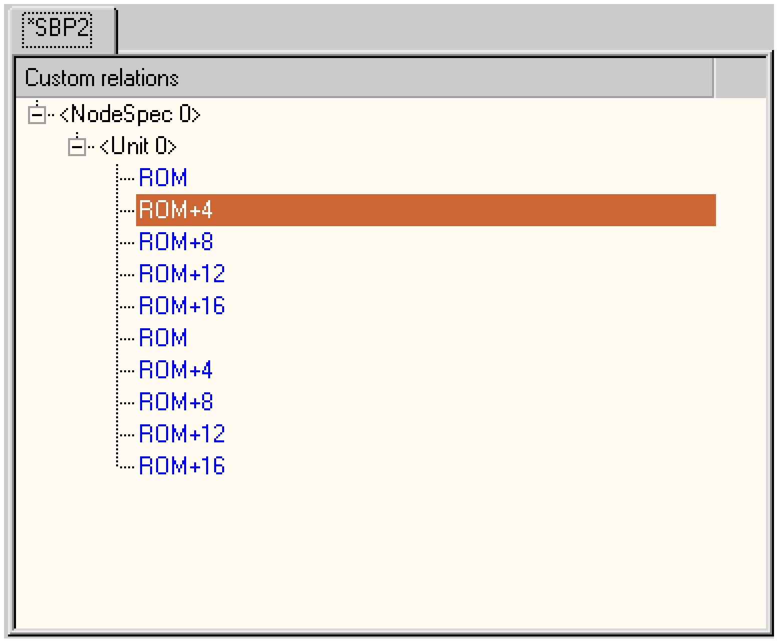


It contains the following additional control:

View a b c
Analyzer node select control
With this control you are able to select the protocol analysis result of each supported Analyzer node.

19.3.2. Relations Pane

This pane shows all custom-made protocol items found by the protocol analyzer and displayed in a tree. The tree structure indicates the relation between the items. All items of the tree are listed in the same order as the corresponding transaction of each item was recorded.



*SBP2

Custom relations

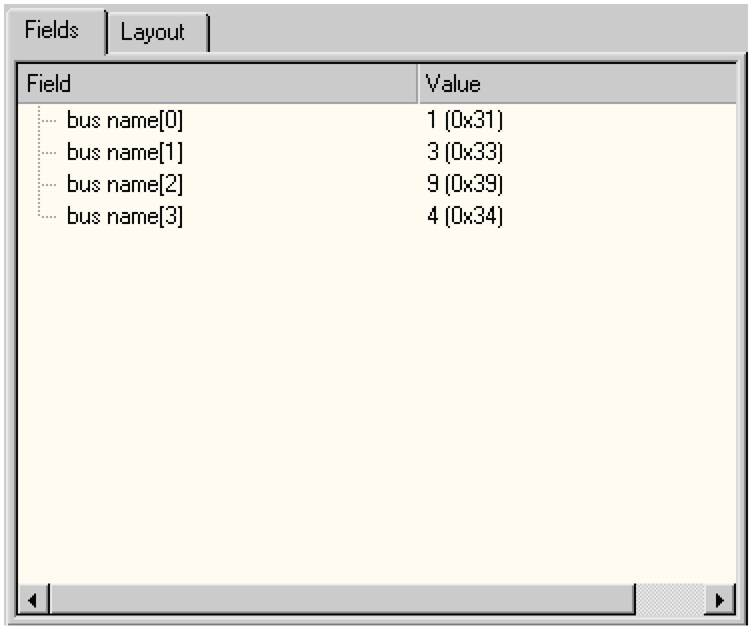
```
<NodeSpec 0>
  <Unit 0>
    ROM
    ROM+4
    ROM+8
    ROM+12
    ROM+16
    ROM
    ROM+4
    ROM+8
    ROM+12
    ROM+16
```

For each found or manually specified NodeSpec there will be a root item in the tree (NodeSpec 0 above). For information about NodeSpec's, see 'Custom-made Protocol Settings' below. For each found or manually specified Unit, there will be a Unit item inside the corresponding NodeSpec (Unit 0 in NodeSpec 0 above).

A black item indicates that it has no corresponding transactions or packets (e.g. NodeSpec and Unit above). It just is used for proper grouping of the items in the tree. All colored items however correspond to one transaction. When selecting such an item, the details of that item are displayed in the 'details pane' and the corresponding transaction is highlighted in the 'transactions and packets' pane (see below).

19.3.3. Details Pane

The middle part of the Protocol View shows the details of the item that is selected in the 'relations' pane. An example that shows the details of a ROM+4 item follows:



Fields | Layout

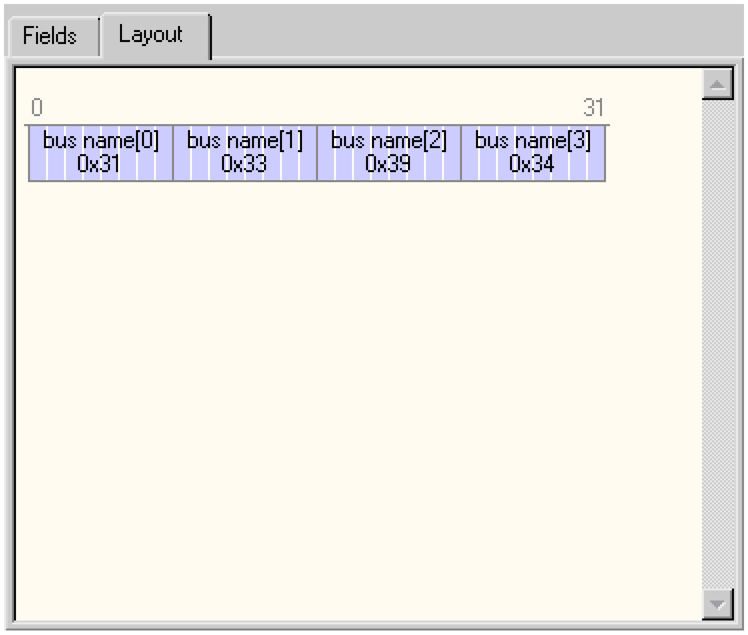
Field	Value
bus name[0]	1 (0x31)
bus name[1]	3 (0x33)
bus name[2]	9 (0x39)
bus name[3]	4 (0x34)

We will describe the different parts below:

Fields

The 'Fields' table shows the fields that are defined for the selected item and the corresponding values of these fields. See example above.

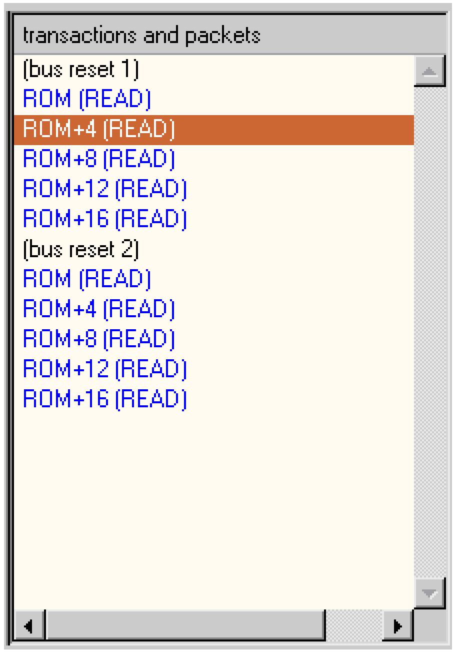
Layout
The 'Layout' shows the layout of the selected item. Like the 'Fields' table it shows the fields that are defined for the selected item and the corresponding values of these fields. But now the fields are displayed graphically inside data items. See below for an example of the layout view for the same item as shown in the example above.



bus name[0]	bus name[1]	bus name[2]	bus name[3]
0x31	0x33	0x39	0x34

19.3.4. Transaction and packets pane

The right most part of the Protocol View shows the sequence of all protocol packets and transactions found (including those for custom-made protocol). The order of items in this list corresponds to the order of recording. It is the same order as displayed in the Transaction View or Packets View of the Recorder. An example follows:



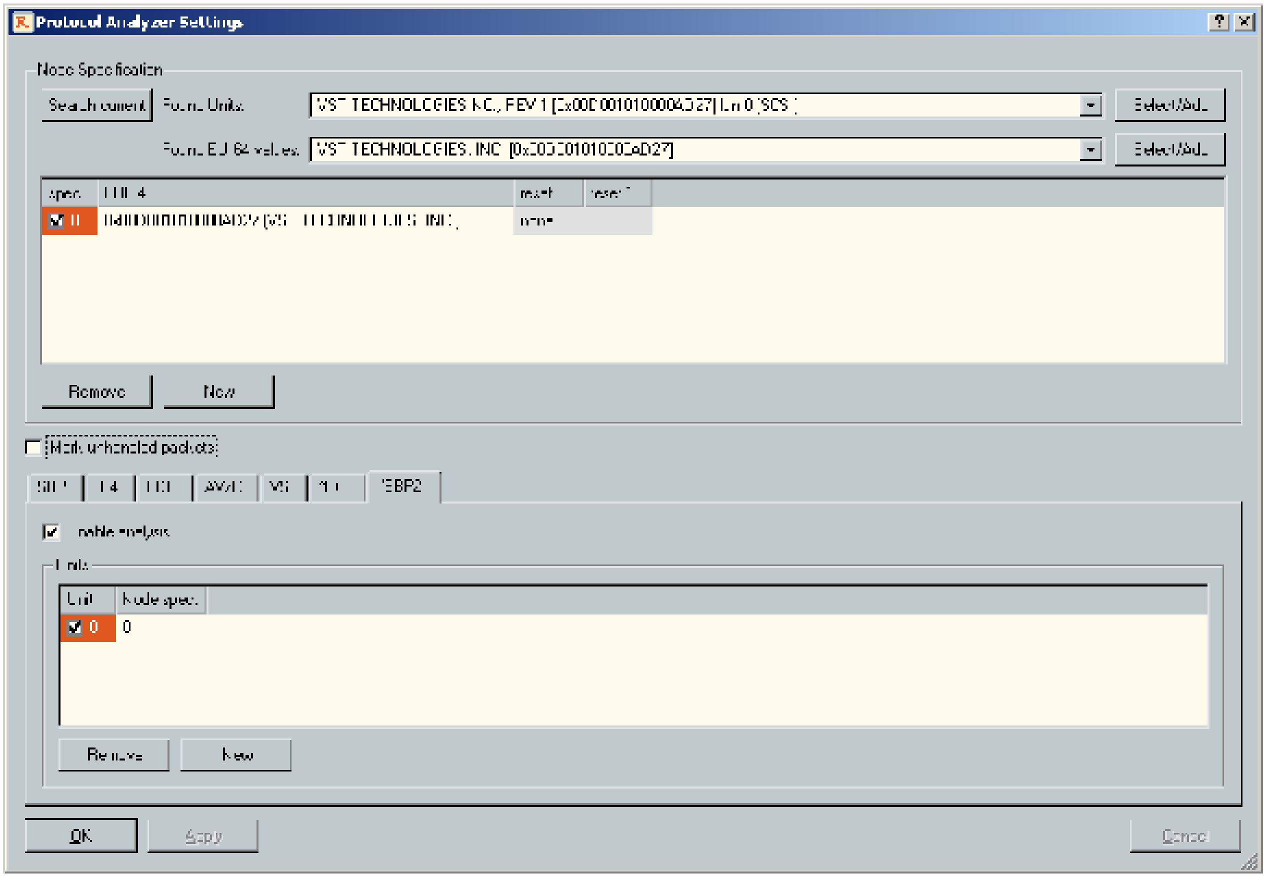
(bus reset 1)
ROM (READ)
ROM+4 (READ)
ROM+8 (READ)
ROM+12 (READ)
ROM+16 (READ)
(bus reset 2)
ROM (READ)
ROM+4 (READ)
ROM+8 (READ)
ROM+12 (READ)
ROM+16 (READ)

Note that the colors identify the type of access and are the same as used in the 'relations' pane (see above). All transactions and packets in this list that are not part of the protocol currently selected in the 'relation' pane are colored gray.

19.3.5. Custom Protocol Settings

When clicking the 'Protocol Settings' button in the toolbar of the Protocol View, the 'Protocol Settings' dialog will be displayed. This dialog shows information used by the protocol analyzer to find and correctly analyze the transactions. For each supported protocol, including custom-made protocols, there is a tab page with protocol-specific settings. The dialog will be initialized with all information that can be found automatically. If this is not enough for a correct analysis you can add this information manually.

An example of the dialog is shown below.



Found Units: VST TECHNOLOGIES INC., REV 1 [0x00D0010100004D27] Lun 0 (SCS)

Found EUI 64 values: VST TECHNOLOGIES, INC [0x00D0010100004D27]

spec.	EUI 64	reset	reset
✓ 0	[illegible] (VST TECHNOLOGIES, INC.)	none	

☐ Mark unhandled packets

SBP2

✓ Enable analysis

Units

Unit	Node spec.
✓ 0	0

In the dialog you see an upper part to specify one or more 'Node Specifications', and a lower part to specify protocol-specific information. In between these two parts, general options can be selected.

For a correct analysis the Unit information of the device which support the custom-made protocol to be analyzed must be present To be able to specify a Unit, you will first need to specify the device (Node Specification) the Unit is a part of.
For more information about Node Specifications and the protocol-independent part of the Protocol Settings, see Protocol Settings. For the custom-made protocol-specific settings, see below.

Node Specifications

If no custom-made protocol Units were found automatically, you have to specify one or more Units manually. Before you can specify a Unit you have to specify the node the Unit is part of, using a Node Specification. When the device which support the custom-made protocol is still connected to the bus after recording, you can use the 'Search Current' button to automatically find these Units. Adding such a Unit will create a Node Specification and Unit. The created Node Specification will be filled automatically when possible. You probably will have to select correct nodes for some reset segments. Make sure you have selected the correct node for those reset segments for which transaction and packets are recorded that you want to analyze. For more information on Node Specifications, see Protocol Settings.

General options

Mark unhandled packets
For information about this checkbox, please see Protocol Settings.

Protocol-specific information
Enable analysis
Check this box to enable the analysis of this custom-made protocol. All protocol tabs in this dialog have such a checkbox, which can be useful if a recording contains data from several protocols, and you are not interested in some of these.

Units
In the Units table, one or more custom-made protocol Units can be specified. The information in this table normally can be found in the Configuration ROM of the node of which Unit is part of. It consists of:

Unit
Using the checkbox before each Unit in the table you can enable or disable each Unit individually for analysis. If the box is not checked, the custom-made protocol related transactions for this Unit will be skipped.

Node Spec.
A Unit is part of a node. As explained before, nodes are specified using Node Specifications. The number in this column corresponds to the Node-Specification numbers of the table in the upper part of the dialog.

19.3.6. SBP-2 Example

For the custom protocol SBP-2 example it’s prerequisite to add directory ‘example’ to 2 tab-pages of the settings dialog. These are tab-pages ‘Protocols’ and ‘Directories’. For a detail description of these tab-pages, see sections Protocol Settings and Directory Settings.

The directory addition to tab-page ‘Protocols’ will include directory ‘examples’ to the custom protocols search paths. The custom protocol SBP-2 is defined in file ‘sbp2.pfs’.

The directory addition to tab-page ‘Directories’ will include directory ‘examples’ to the custom formats search paths. The custom format of the custom protocol SBP-2 is defined in file ‘ROMConfig.dff’.

The custom protocol SBP-2 is recognized by the Protocol View of the Recorder after restart of the Analyzer application.

Chapter 20. Settings

With the settings dialog you can control all kinds of Analyzer settings. There are general settings like which windows should open when the application is started, and there are settings for individual windows. The settings are persistent. So, when starting the Analyzer application the next time, the last settings will be applied automatically.Since the redesign of the settingsdialog in version 4.1, device dependent settings are saved per analyzer.

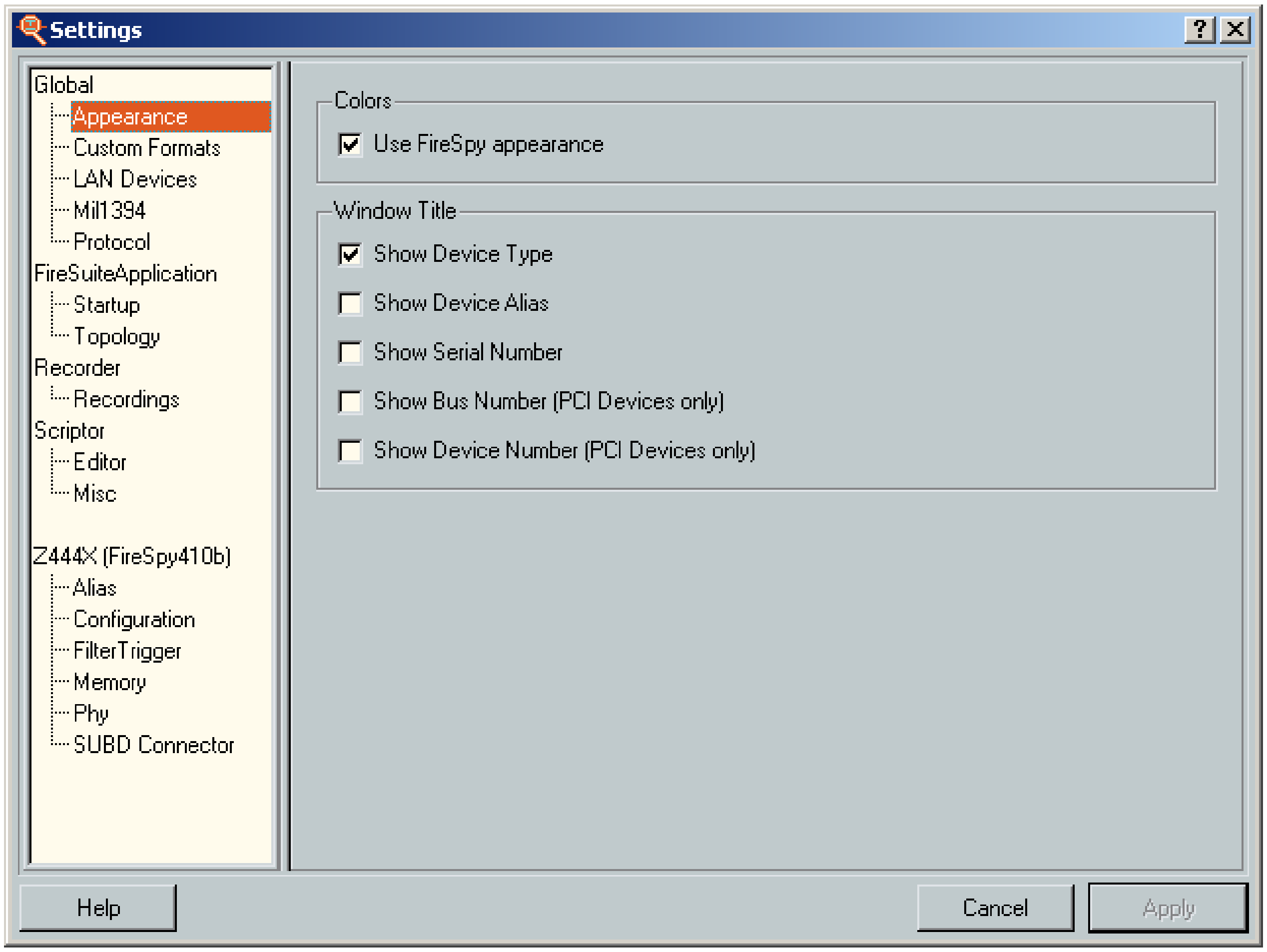
20.1. How to use it

You can open the Settings window by selecting the 'Settings' command from the Tools menu at the top of the main window or one of the other open windows. An example of the Settings dialog is shown below. All pages of the settings dialog will be described in the following sections.

20.2. Details

20.2.1. Global Settings

20.2.1.1. Appearance



Colors

[✓] Use FireSpy appearance

Window Title

[✓] Show Device Type

[ ] Show Device Alias

[ ] Show Serial Number

[ ] Show Bus Number (PCI Devices only)

[ ] Show Device Number (PCI Devices only)

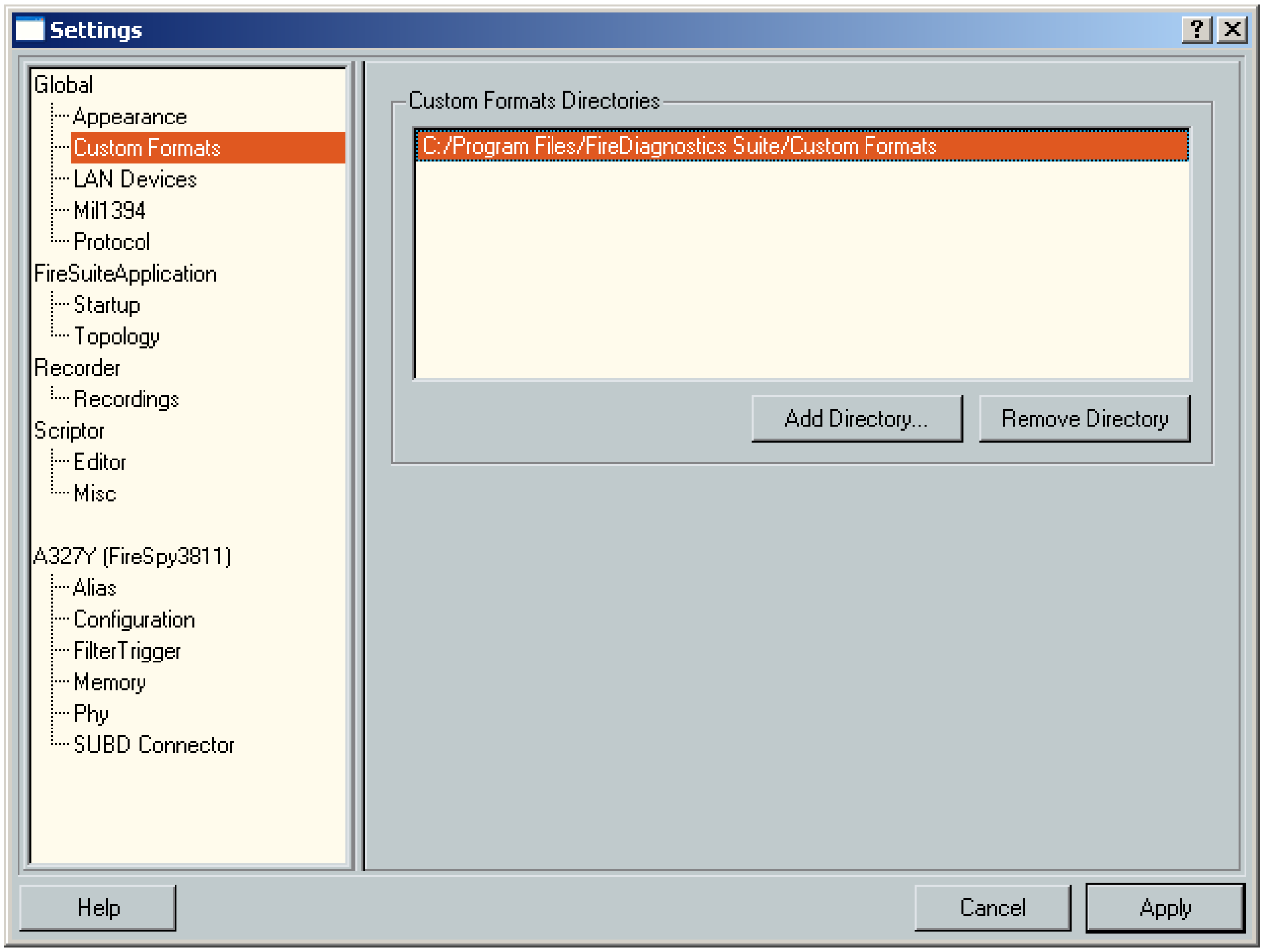
The example above shows the Appearance settings page in the Global section. It contains the following setting groups:

Colors

If the 'Use Analyzer appearance' box is checked, the Analyzer-type colors will be used for all windows. If this box is not checked, the standard system colors will be used. All examples in this manual use the Analyzer-type colors.
After changing the checkbox, you can use the 'Apply' or 'OK' button to activate it. Note that the little main window always uses the Analyzer-type colors.

Window Title
Every window of the application can show a number of items:
-Device Type. FireSpy810, FireStealth,..
-Device Alias. Can be chosen by the user in the Alias section of each device
-Serial Number.
-Bus Number. The PCI bus enumerates devices by their Bus en Device number.

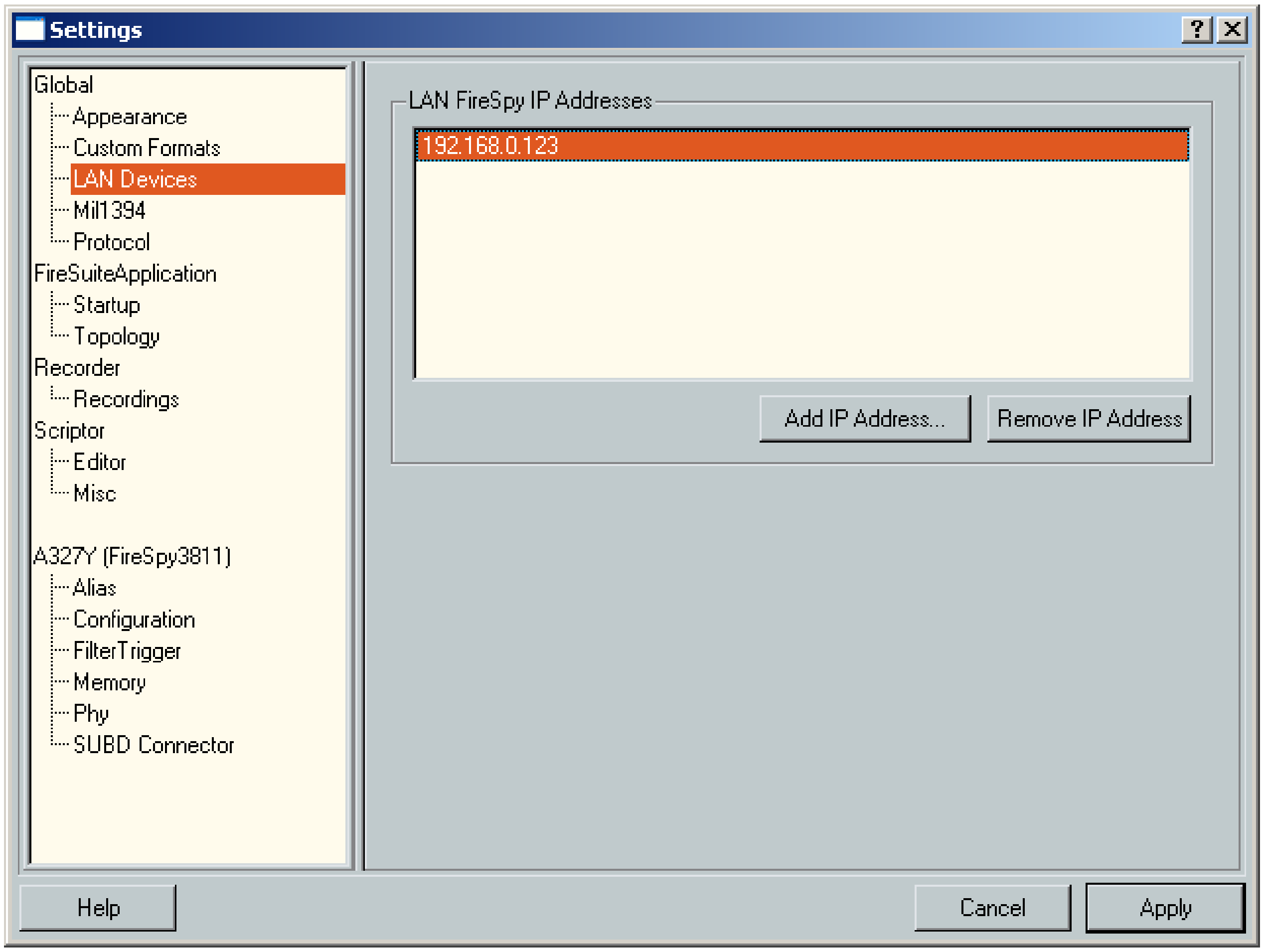
20.2.1.2. Custom Formats



C:/Program Files/FireDiagnostics Suite/Custom Formats

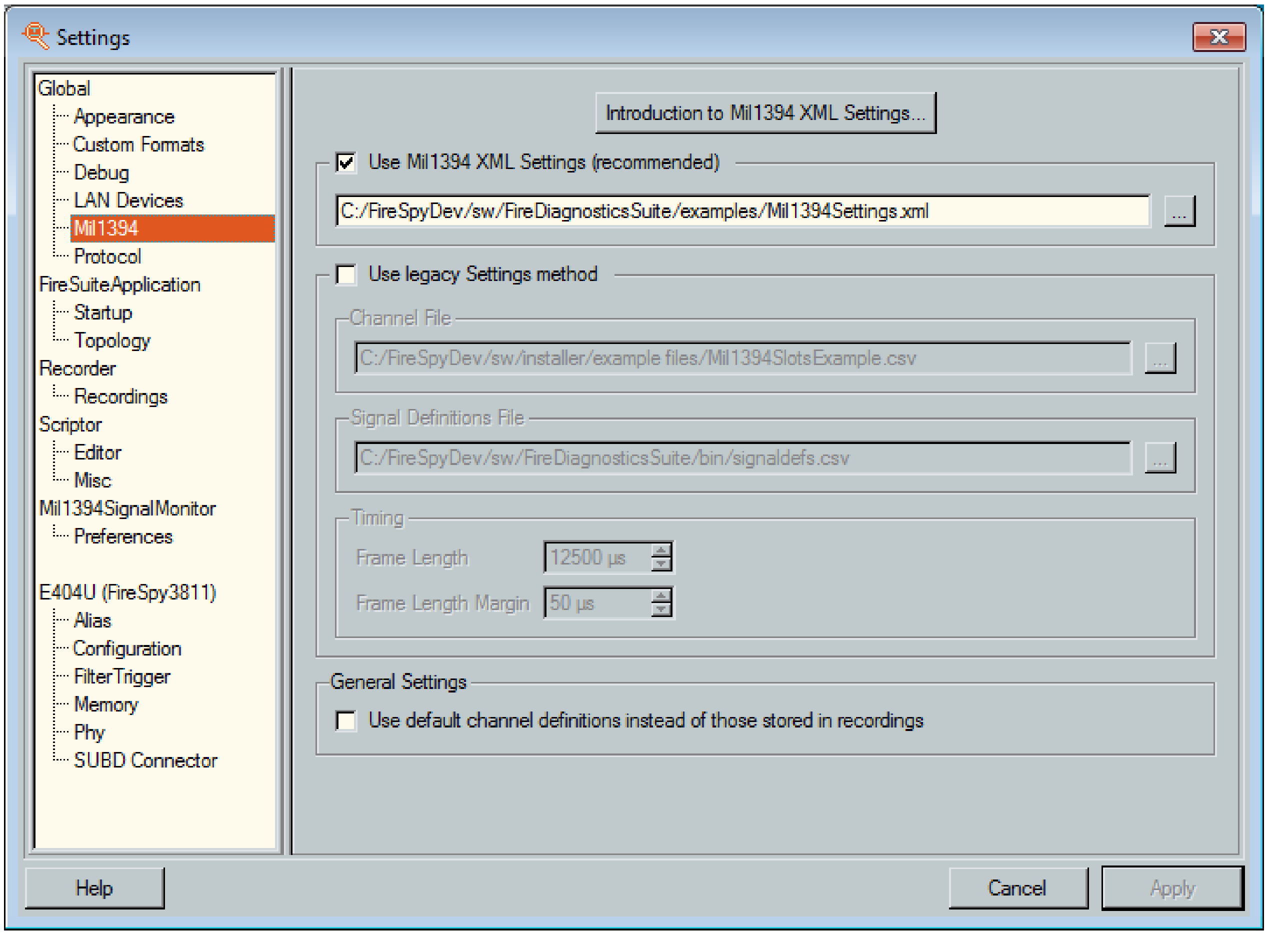
Custom Formats Directories
This list box can be used to define all locations the software should search for custom formats. For more information about custom formats, please read the Format Editor section.

20.2.1.3. LAN Devices



The FireDiagnosticsSuite can use Analyzers connected via ethernet after they are configured with the EthernetConfig tool. To have the application look for an ethernet device, add its IP Address in the box above.

20.2.1.4. Mil1394



[✓] Use Mil1394 XML Settings (recommended)

C:/FireSpyDev/sw/FireDiagnosticsSuite/examples/Mil1394Settings.xml

[ ] Use legacy Settings method

Channel File: C:/FireSpyDev/sw/installer/example files/Mil1394SlotsExample.csv

Signal Definitions File: C:/FireSpyDev/sw/FireDiagnosticsSuite/bin/signaldefs.csv

Timing

Frame Length: 12500 µs

Frame Length Margin: 50 µs

General Settings

[ ] Use default channel definitions instead of those stored in recordings

Mil1394 Protocol
The picture above shows the Mil1394 protocol settings page.

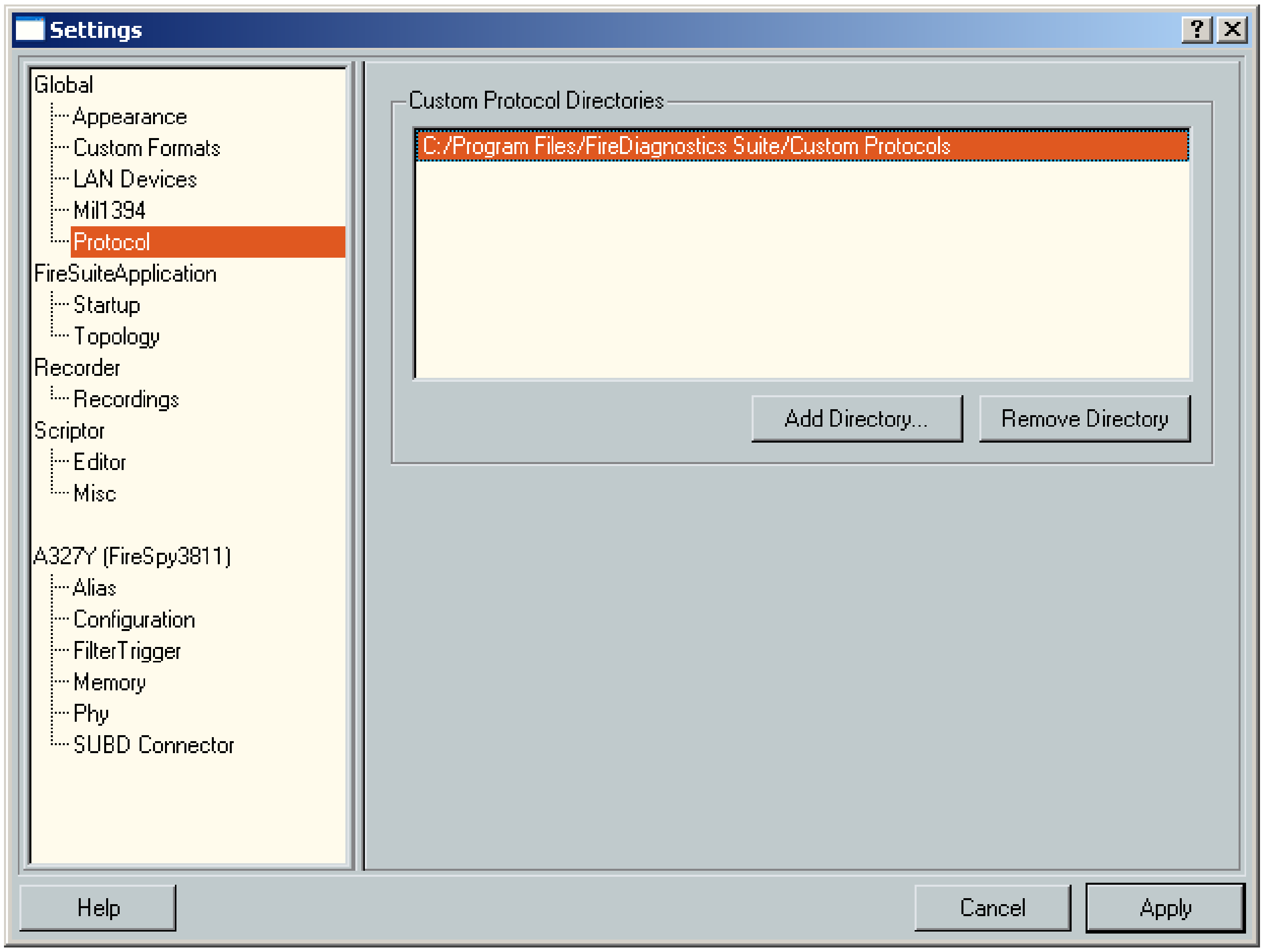
As of software version 5 it is recommended to use an XML file to define all important aspects of the AS5643 protocol according to the specifications of the program. An example file named "Mil1394Settings.xml" is installed with the application and can be changed as needed. Please contact support@daptechnology.com if you are wondering an XML file already exists for the specific program you are working on, for example JSF.

For more information about the XML file format, see the XML settings file format section.

For more information about slot definitions, please read the Mil1394 Protocol Settings section. Signal definitions are explained in the Signal Monitor and the Signal Definitions file format sections.

General Settings
When the checkbox is checked and a FireSpy recording is opened, the channel definitions in the file will be overruled by the channel definitions configured through this dialog.

20.2.1.5. Protocol



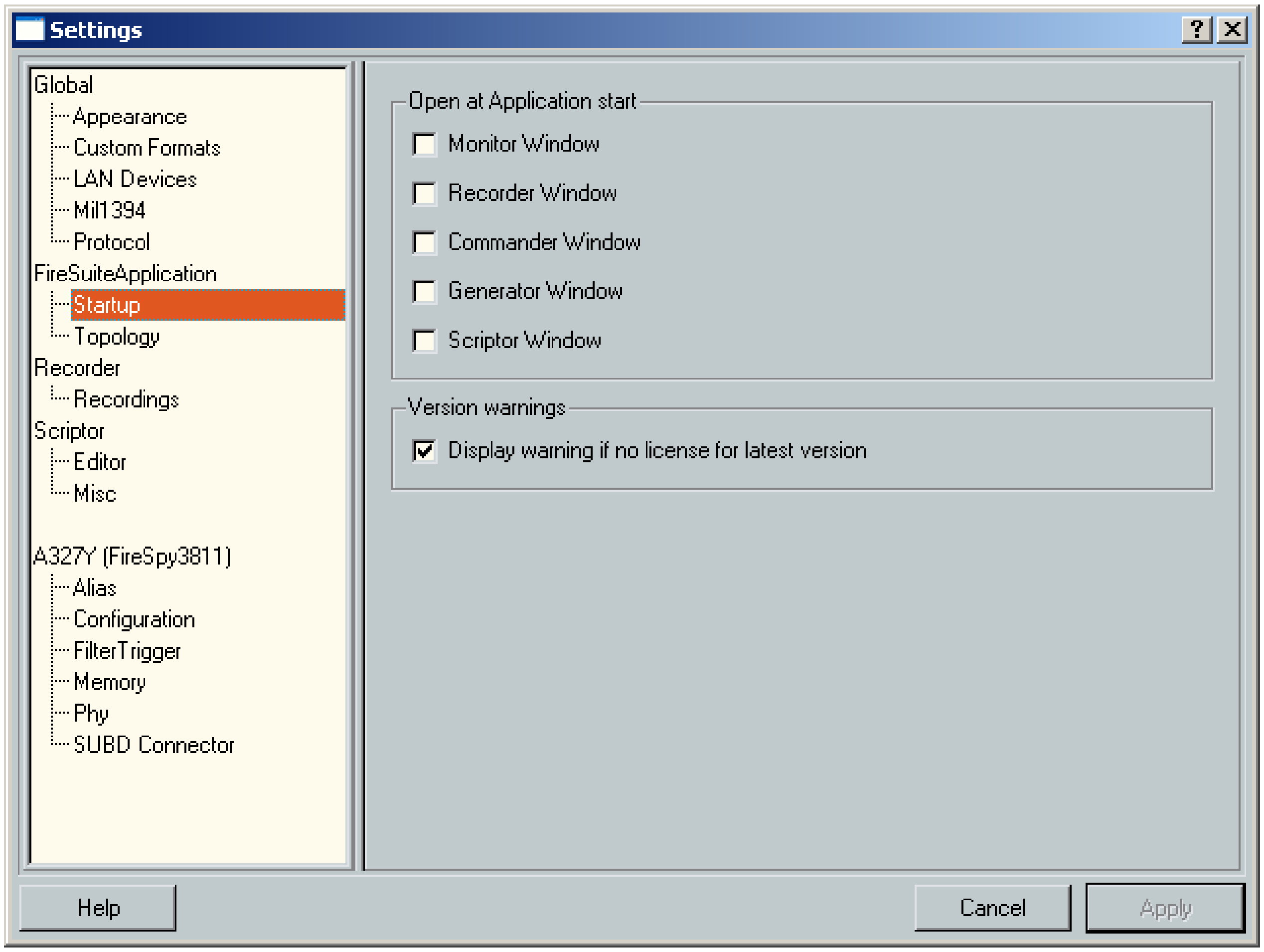
Custom Protocol Directories

C:/Program Files/FireDiagnostics Suite/Custom Protocols

Custom Protocol Directories
The picture above shows the custom protocols. It can be used to define all locations the application should search for custom protocol definitions. For more information about custom protocols, please read the Protocol Editor section.
Depending on your license, this setting might be disabled.

20.2.2. FireSuiteApplication

20.2.2.1. Startup



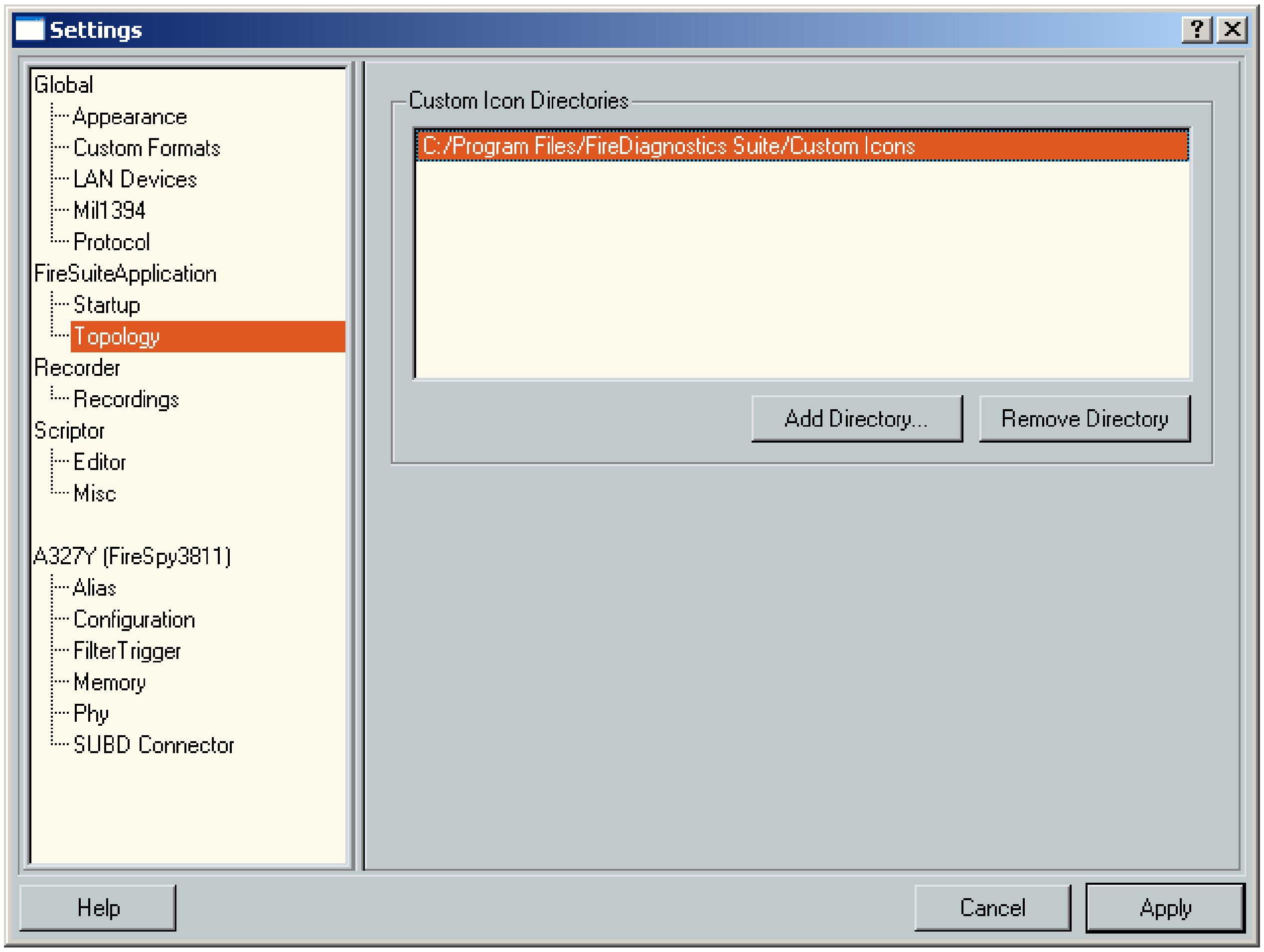
Open at Application start
- [ ] Monitor Window
- [ ] Recorder Window
- [ ] Commander Window
- [ ] Generator Window
- [ ] Scriptor Window

Version warnings
- [✓] Display warning if no license for latest version

Open at Application start
In this box you can specify which windows should open at application start. In the example above, none of the windows will open. The little main window will always open at startup.
If you want to change these settings, you need to apply the new settings. The next time the application is started, the new settings will be loaded and the specified windows will open.

Version warnings
If the checkbox with label 'Display warning if no license for latest version' is checked, the application will display a warning if no license key is installed for the latest version the software supports. If this latest license key is not installed, the latest features will be disabled.
You can use the License Manager to add new license keys. If you want new keys or want to try a temporary key (for evaluation) please contact support@daptechnology.com.

20.2.2.2. Topology



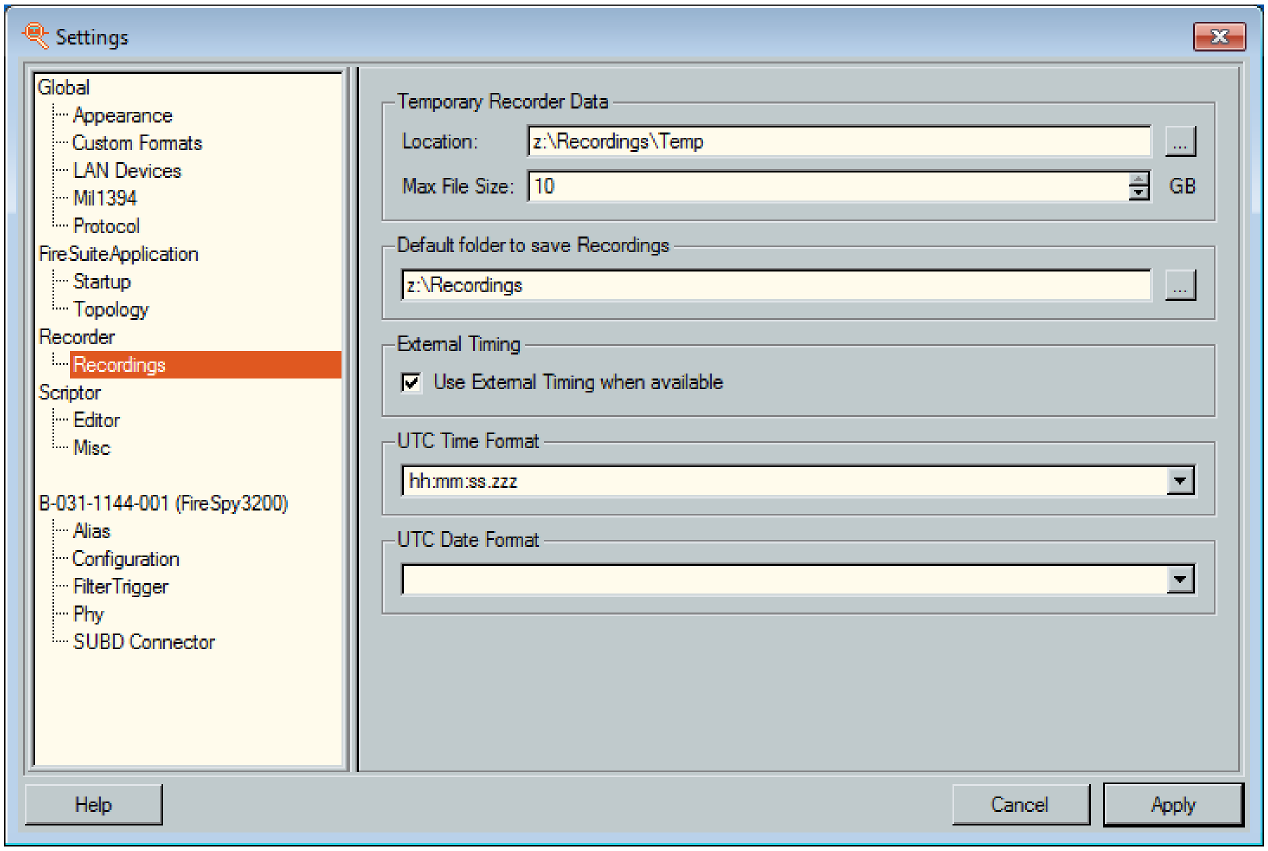
Custom Icon Directories

C:/Program Files/FireDiagnostics Suite/Custom Icons

Custom Icons Directories
This list box can be used to define all locations the software should search for custom icons used by the Topology view of the Commander. For more details, please read the Commander Topology section.

20.2.3. Recorder

20.2.3.1. Recordings



Temporary Recorder Data
Location: z:\Recordings\Temp
Max File Size: 10 GB

Default folder to save Recordings
z:\Recordings

External Timing
☑ Use External Timing when available

UTC Time Format
hh:mm:ss.zzz

UTC Date Format

Temporary Recorder Data
This setting only applies to Third Generation Analyzers. In this box you can choose a folder location where to store temporary recording files that are created during recording and the maximum file size. Please choose a folder on a fast hard drive or SSD to keep up with the data that is received on the bus. If the host can not keep up writing the data to disk while making the Recording, the Recording will be stopped as soon as the FireSpy on-board memory is filled.

Initial location of recordings
In this box you can specify the folder that is initially used for opening and saving Recorder files. The first time you open or save a Recorder file after changing this setting, the file dialog to open or save a file will display the specified folder.

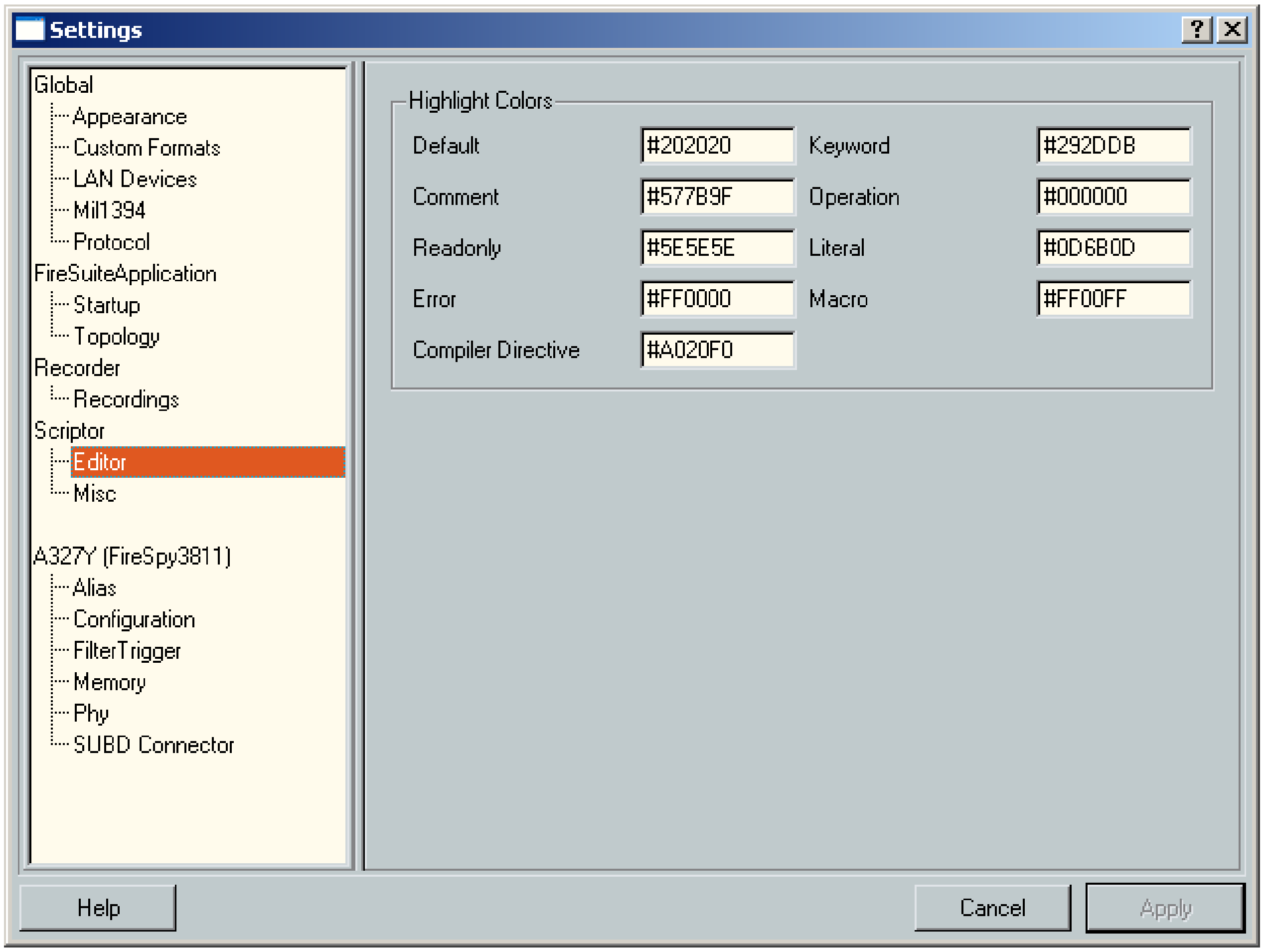
External Timing
Some devices can be configured to use external timing, for example from an IRIG source. If you want to have the recorder use this external timing, you should enable this option. The recorder then can show an extra field which represents the external timing.

UTC Time Format
This combo box can be used to specify a time format to be used in the Recorder packet view to show the UTC packet time stamp

UTC Data Format
This combo box can be used to specify a time format to be used in the Recorder packet view to show the UTC packet time stamp date

20.2.4. Scriptor

20.2.4.1. Editor



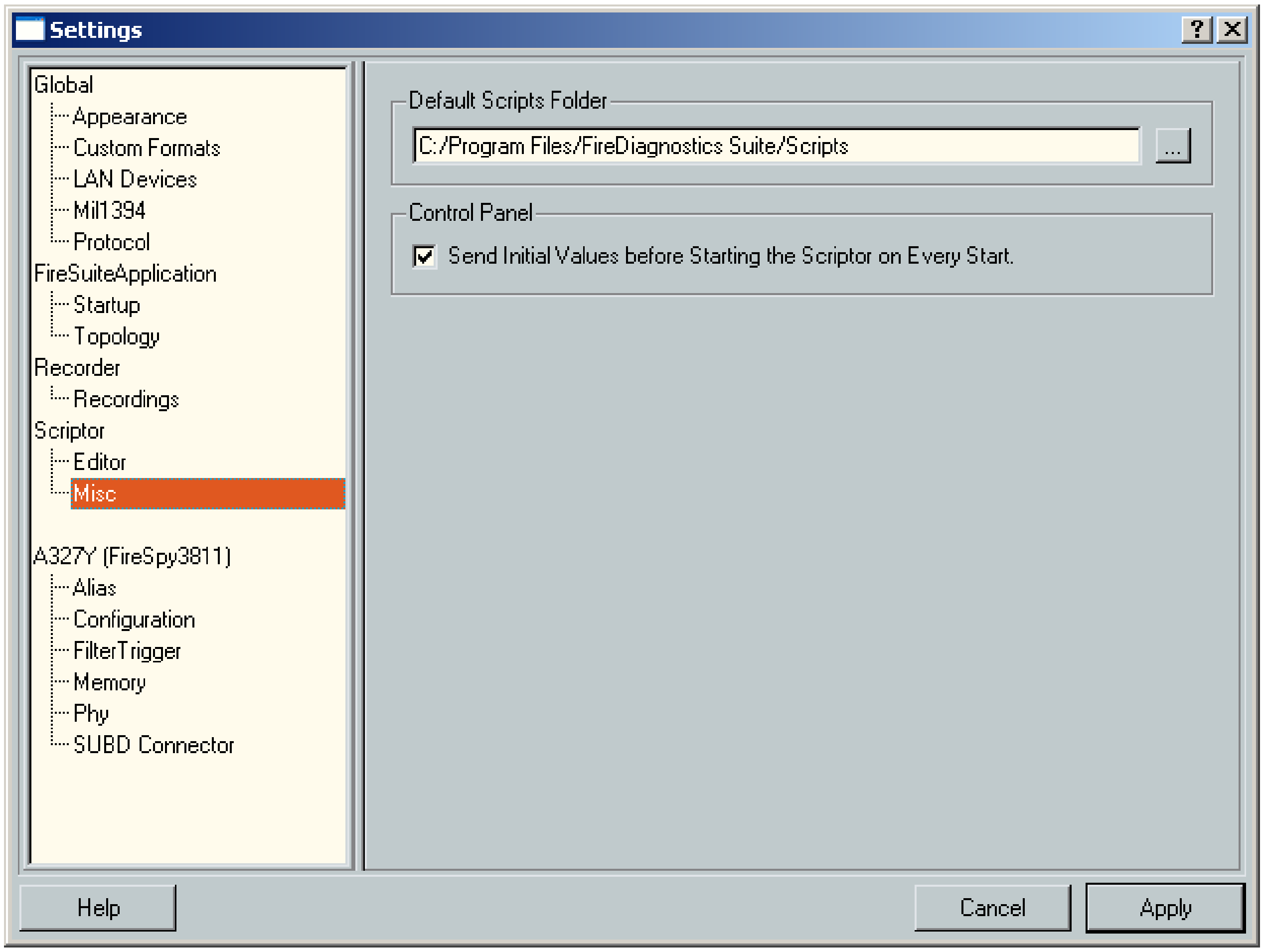
Highlight Colors

Default	#202020	Keyword	#292DDB
Comment	#577B9F	Operation	#000000
Readonly	#5E5E5E	Literal	#0D6B0D
Error	#FF0000	Macro	#FF00FF
Compiler Directive	#A020F0		

Highlight colors
This section contains several color settings that can be changed to let the Scriptor tree editor use different colors for syntax highlighting. It contains the following items:

- Default: All text that is not covered by one of the other highlight items
- Comment: Source code that is commented out by using the "//" characters
- Readonly: All source code that is read-only. For example: The built-in function declarations and the sub nodes of a Macro
- Error: All source code lines that contain an error are displayed in red if the cursor is not at that line
- Compiler Directive: All keywords that start with a "#" character
- Keyword: All script keywords. For example: while, for, if and all data types
- Operation: All script operation symbols. For example: "+", "-" or "=="
- Literal: All literal values. For example "1.123", "0xFFF" or "true"
- Macro: All macro names

20.2.4.2. Misc



Default Scripts Folder: C:/Program Files/FireDiagnostics Suite/Scripts

Control Panel

[✓] Send Initial Values before Starting the Scriptor on Every Start.

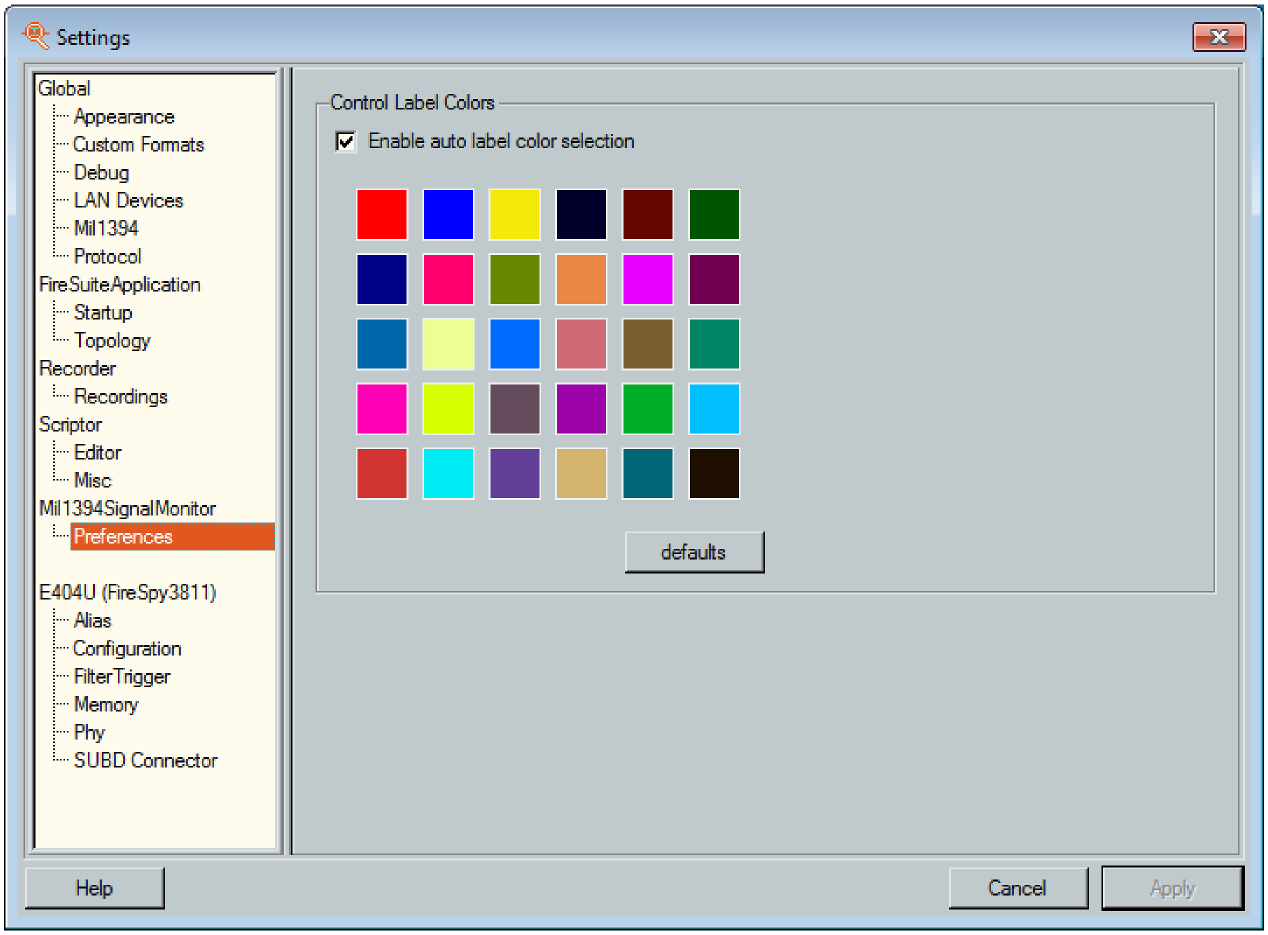
Default Scripts Folder
Specify a folder here that will be used as the default folder for scripts. This folder will be the starting point of a file open or file save dialog is displayed.

Send initial values before starting the Scriptor on every start.
When input control are used on the Control Panel, they only send their values to the scriptor when they are changed. This option will cause the ControlPanel to send its values before the script is started.

20.2.5. Mil1394 Signal Monitor

Enter topic text here.

20.2.5.1. Preferences



Control Label Colors

[✓] Enable auto label color selection

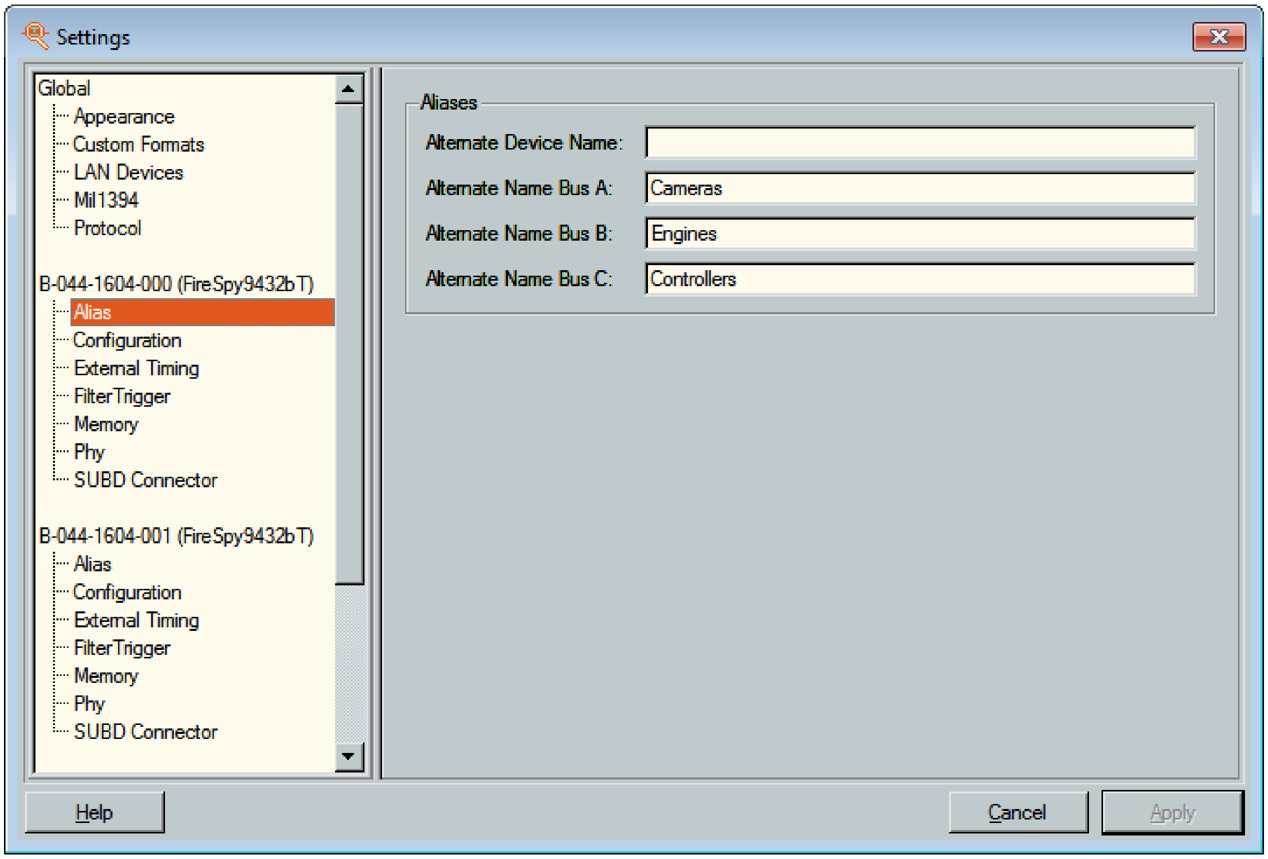
Control Label Colors
When Controls are added to the Mil1394 Signal Monitor Controls View the label color can automatically be chosen from a predefined set of colors. To enable this functionality the check box "Enable auto label color selection" needs to be checked. When unchecked, labels will be set to black.

A selection of 30 different colors can be predefined by the user. Simply click on one of the color boxes and select a color from the dialog that will be shown. By pressing the "defaults" button, all colors will be reset to the default value.

The software does not look at the order in which the colors are defined. It simply picks the color that it believes is ideal based on colors already used (can be one of the colors picked from this dialog or colors that the user manually chose for their control label) and the background color set for the Control Panel.

20.2.6. Device

20.2.6.1. Alias

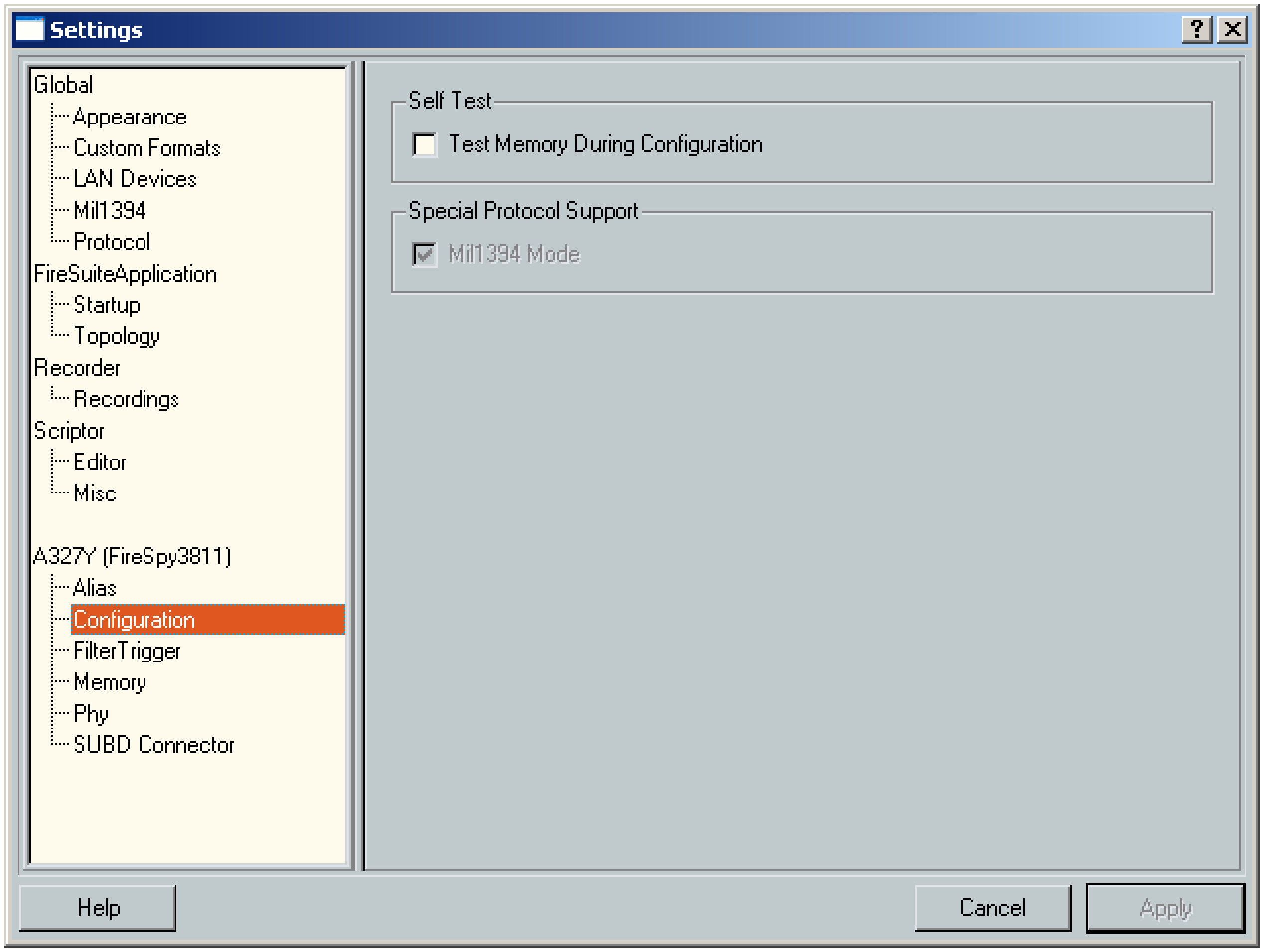


Aliases

Alternate Device Name:	
Alternate Name Bus A:	Cameras
Alternate Name Bus B:	Engines
Alternate Name Bus C:	Controllers

Aliases
The Aliases setting allows an user to enter a nickname to a device and/or its busses. The device name can be used to change the window name. The bus names are currently only used by the monitor.

20.2.6.2. Configuration



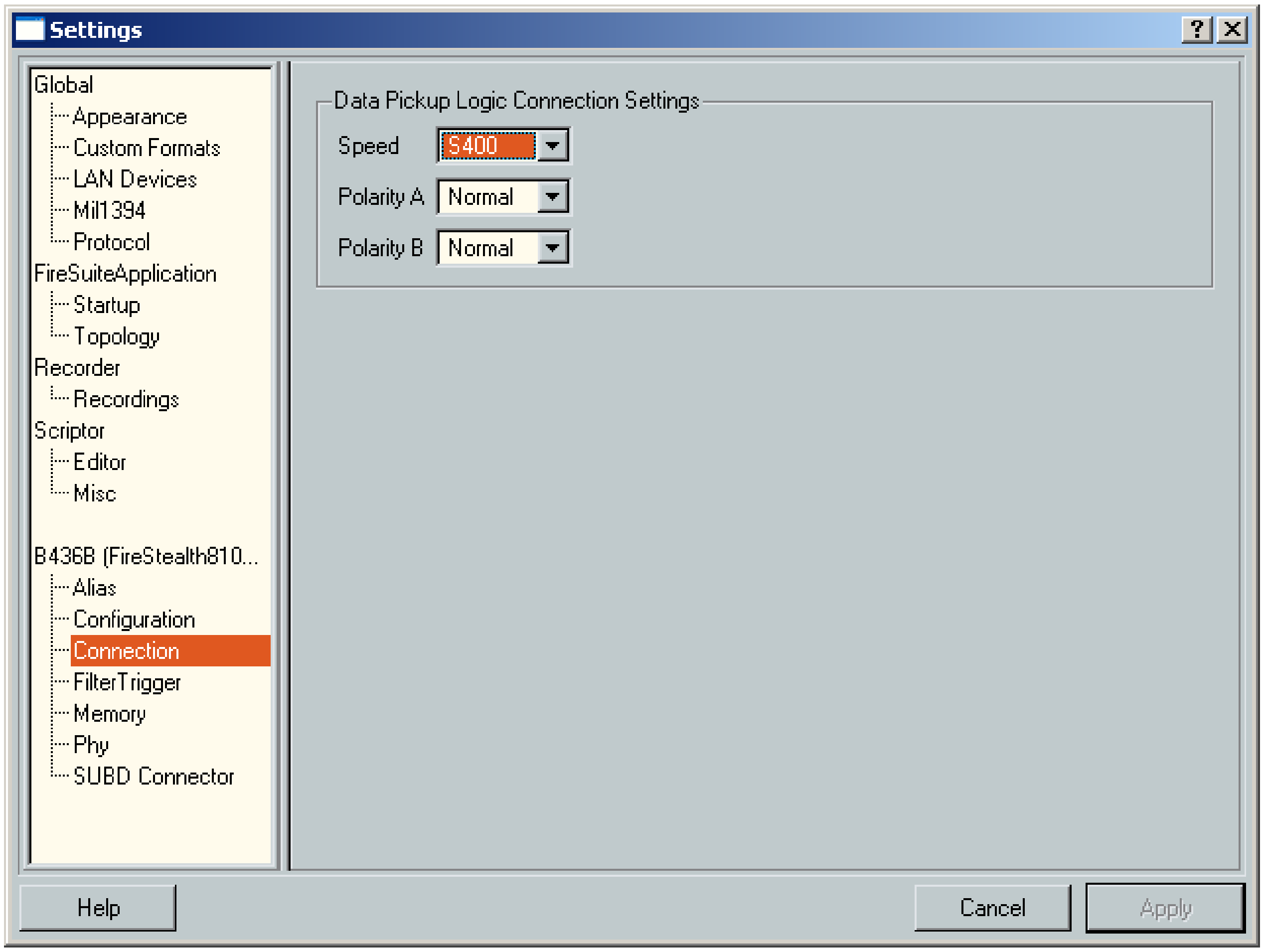
Self Test
- [ ] Test Memory During Configuration

Special Protocol Support
- [✓] Mil1394 Mode

Self Test
When the "Test Memory During Configuration" option is enabled, the analyzer will tests its own RAM during the configuration phase of the device. It will be visible to the user by showing a progress dialog.

Mil1394 Mode
If enabled, the generator can be used for generating Mil1394 Streams instead of standard 1394 isochronous streams. Furthermore, the Scriptor will also contain some additional Mil1394 Functions for Frame timing.

20.2.6.3. Connection



Data Pickup Logic Connection Settings

Speed: S400

Polarity A: Normal

Polarity B: Normal

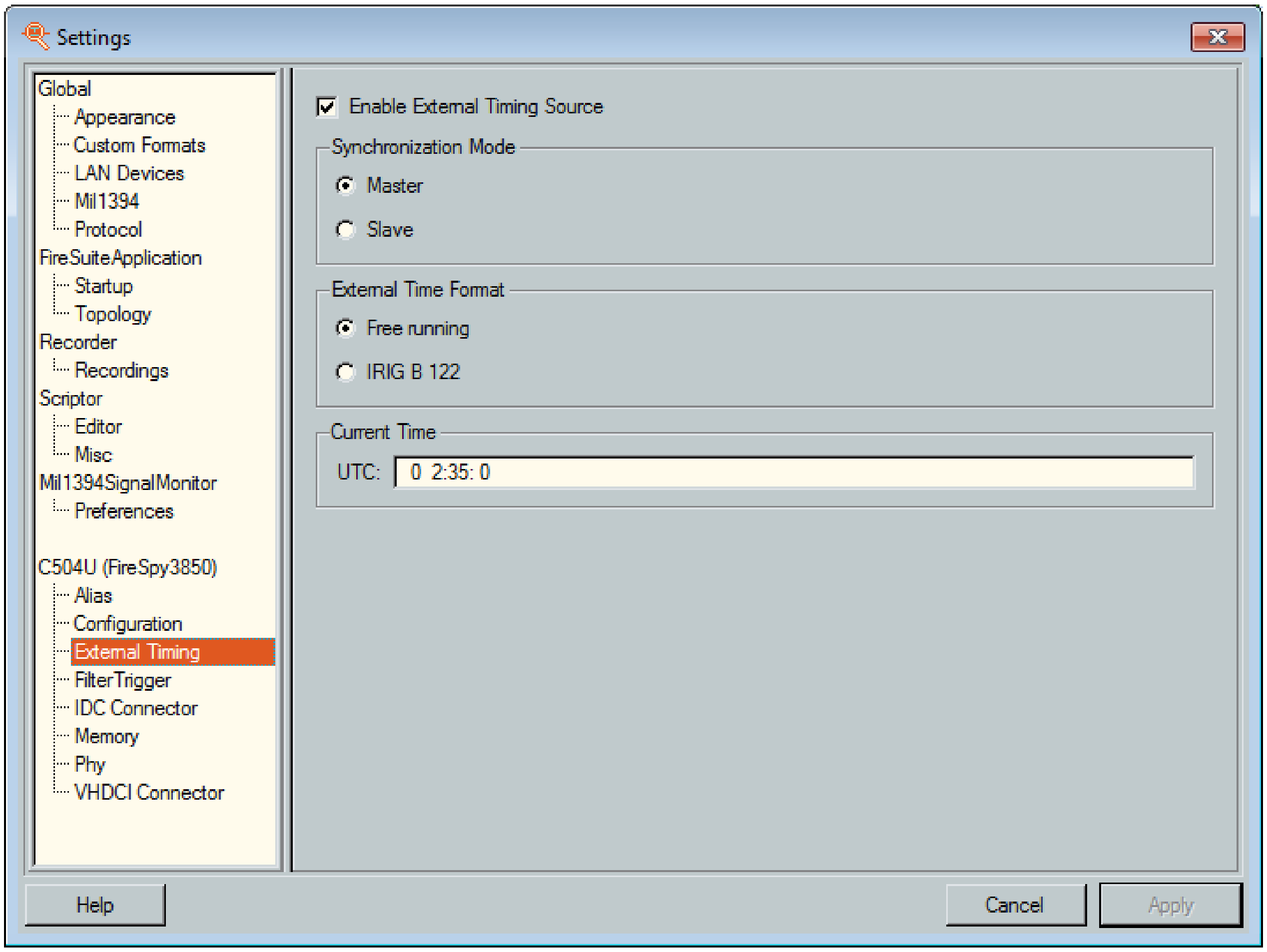
Data-Pickup Logic Connection-Settings

As the FireStealth does not have auto detection of the bus speed, the user must supply it by means of these configuration settings.

Setting the line speed can be done by making a selection in the Speed box between S800, S400, S200 and S100.

Setting the cable polarity of the cable connected to port A of the FireStealth can be done by making a selection in the PolarityA box between Normal and Inverted.
Setting the cable polarity of the cable connected to port B of the FireStealth can be done by making a selection in the PolarityB box between Normal and Inverted.
Note that any change in the Data Pickup Logic Connection Settings will result in a resynchronization attempt of the FireStealth to the IEEE 1394b bus.

20.2.6.4. External Timing



[✓] Enable External Timing Source

Synchronization Mode
[✓ Master] Slave

External Time Format
[✓ Free running] IRIG B 122

Current Time
UTC: 0 2:35: 0

The FireSpy385X is capable of synchronization between multiple analyzers by setting one as a time master and the others as time slave. Time on the master can be configured either using an internal clock running in free running mode or using an IRIG timing source (in combination with a DapTechnology IRIG Demodulator) to synchronize the internal clock. Currently the Recorder is the only module capable of using the external timing.

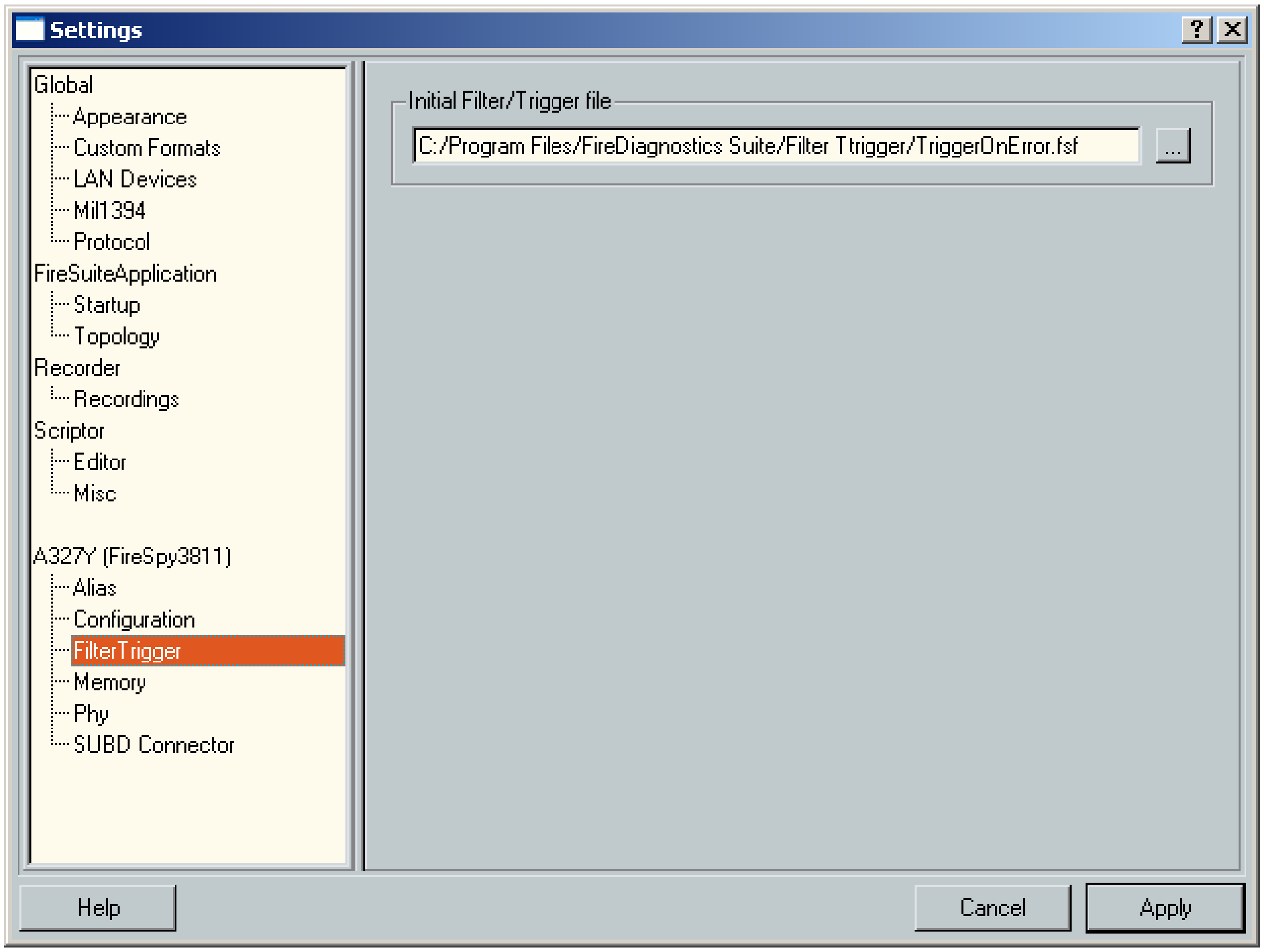
Synchronization Mode
When using External synchronization, one analyzer must be set to Master and other devices must be set to slave.

External Time Format
The following time formats are supported:

- Free Running
- IRIG B 122
  B: BCD Time-Of-Year in 1 second steps, Straight Binary Seconds-Of-Day and 27 Control Function bits
  1: Sine Wave, amplitude modulated
  2: 1kHz
  2: BCD Time-Of-Year

Current Time
This field will show the time from the IRIG source when connected correctly. If this field shows "Could not detect a clock", Recorder will not be able to use external timing.

20.2.6.5. FilterTrigger

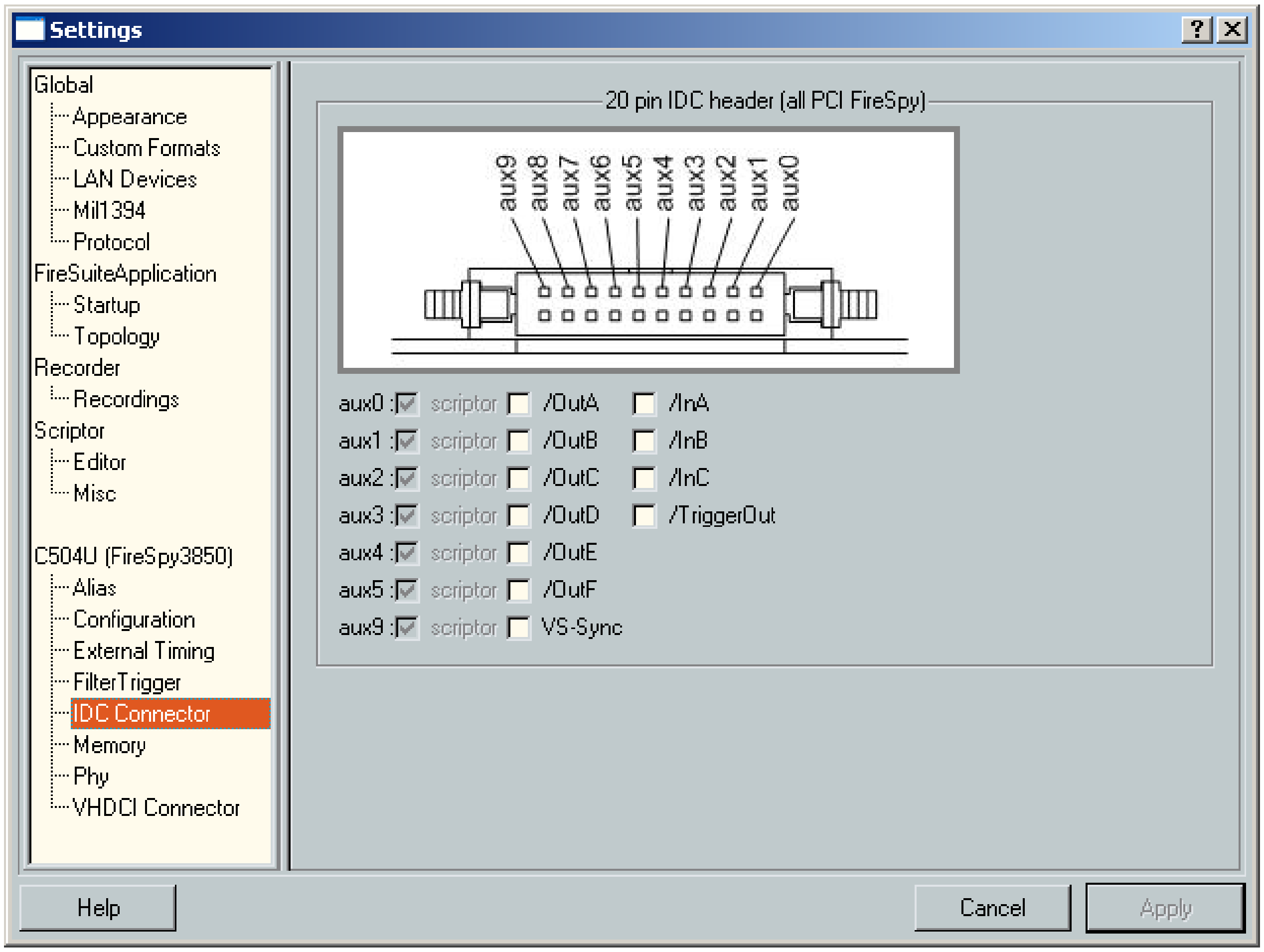


Initial Filter/Trigger file

C:/Program Files/FireDiagnostics Suite/Filter Ttrigger/TriggerOnError.fsf

Initial Filter/Trigger file
This box specifies the file that should be opened in the Filter/Trigger Settings dialog and that should be uploaded to the Analyzer (if connected) at application start. You can change the file by using the 'Browse' button, which will start a file dialog to choose a file. Filter/Trigger files have a default extension of .fsf. If you want no initial Filter/Trigger file please click the 'Clear' button. The next time the application is started, the specified Filter/Trigger setting will be read from this file.

20.2.6.6. IDC Connector



aux0 : [✓] scriptor ☐ /OutA ☐ /InA
aux1 : [✓] scriptor ☐ /OutB ☐ /InB
aux2 : [✓] scriptor ☐ /OutC ☐ /InC
aux3 : [✓] scriptor ☐ /OutD ☐ /TriggerOut
aux4 : [✓] scriptor ☐ /OutE
aux5 : [✓] scriptor ☐ /OutF
aux9 : [✓] scriptor ☐ VS-Sync

/OutA, B, C, D, E, F and /InA, B ,C
These ports can be used as a part of the Trigger Sequencer and the General Trigger. The ports can also be controlled by Port I/O in the Scriptor.

/TriggerOut
This port can be associated with a trigger event.

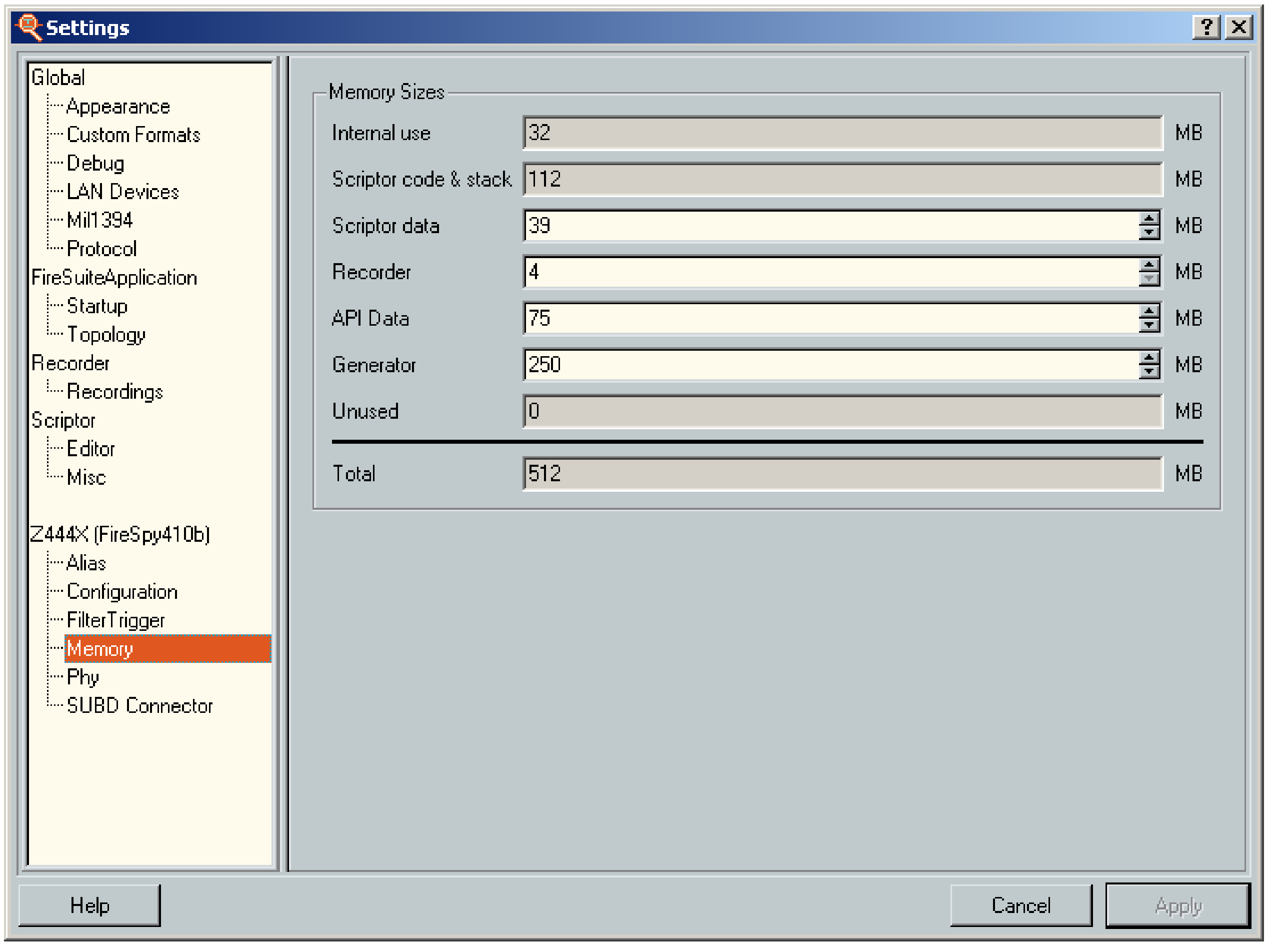
Mil1394-Sync
This port can be programmed as an external frame synchronization signal for the Scriptor function.

Electric characteristics

Input:
VIL: Vmin -0.2 Vmax 0.8
VIH: Vmin 2.0 Vmax 3.45

Output:
VOL: Vmax 0.4
IOL: 24 mA

20.2.6.7. Memory



Memory Sizes

Internal use	32	MB
Scriptor code & stack	112	MB
Scriptor data	39	MB
Recorder	4	MB
API Data	75	MB
Generator	250	MB
Unused	0	MB
Total	512	MB

The total memory is shown as the last item in the list. Part of the internal memory is allocated by the Analyzer itself. This allocation is fixed and cannot be changed. The following fixed memory allocations exist:

Internal Use
This part of the memory is used by the Analyzer itself for internal housekeeping.

Scriptor code & stack
This part of the memory is always allocated to the Scriptor for storing the code of a script and the stack it uses.

The following parts of internal memory can be allocated by the user. Note that the Recorder, Generator and Scriptor need to be cleared before their memory allocation can be changed.

Scriptor Data
This part of the memory is used for the heap of the scriptor and the parts of the code and stack that do not fit in the fixed allocation. The heap is used to store large data objects like packets, file stream buffers and the control output buffer.

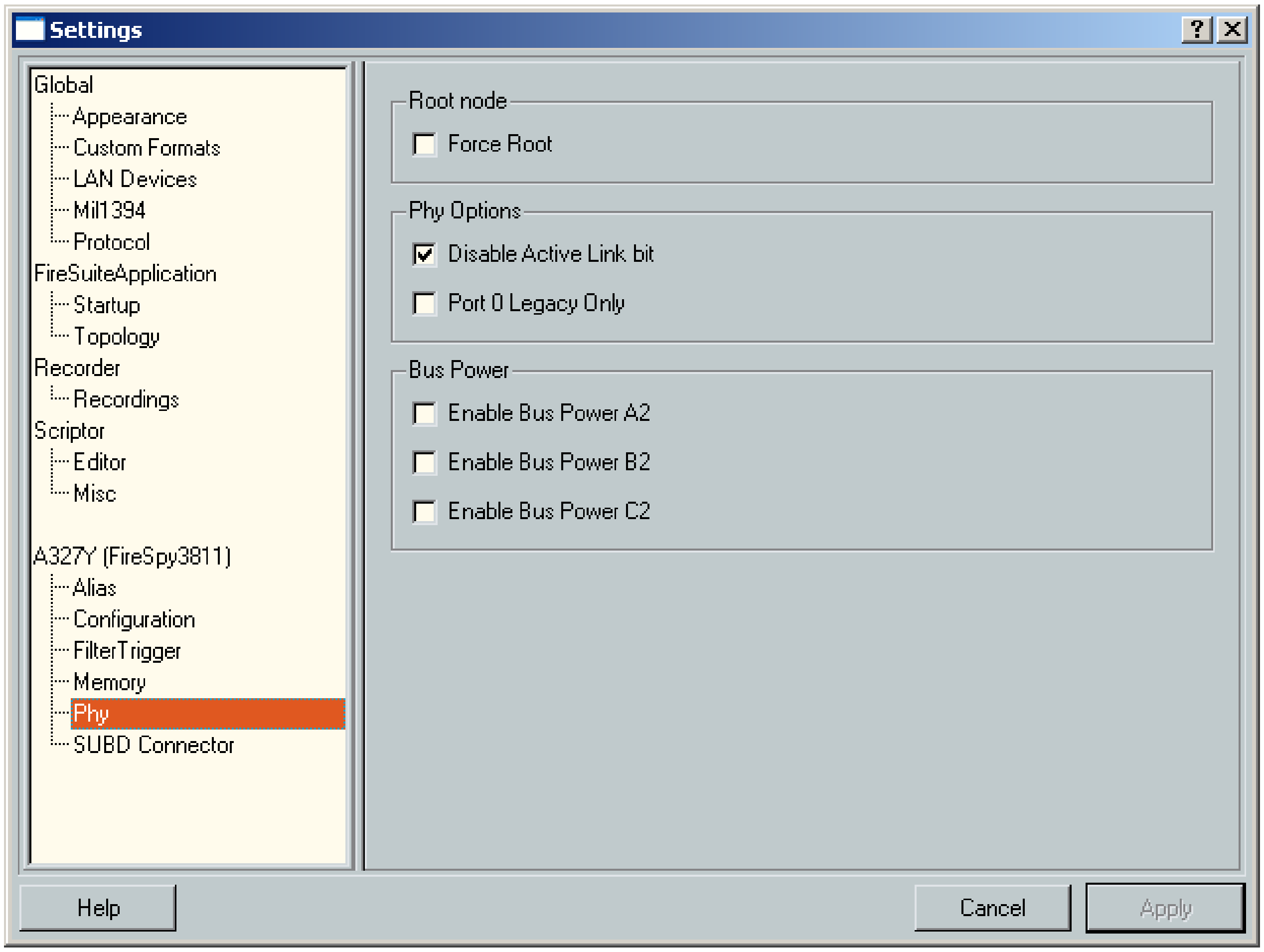
Recorder
This part of the memory is used to store all packets and bus events recorded by the recorder.

API Data
This part of the memory is used by the API.

Generator
This part of the memory is used by the generator data for sending Isochronous packets.

Please note that all available memory has to be assigned before the settings can be applied; there can be no unused memory left.

20.2.6.8. Phy



Root node
- [ ] Force Root

Phy Options
- [✓] Disable Active Link bit
- [ ] Port 0 Legacy Only

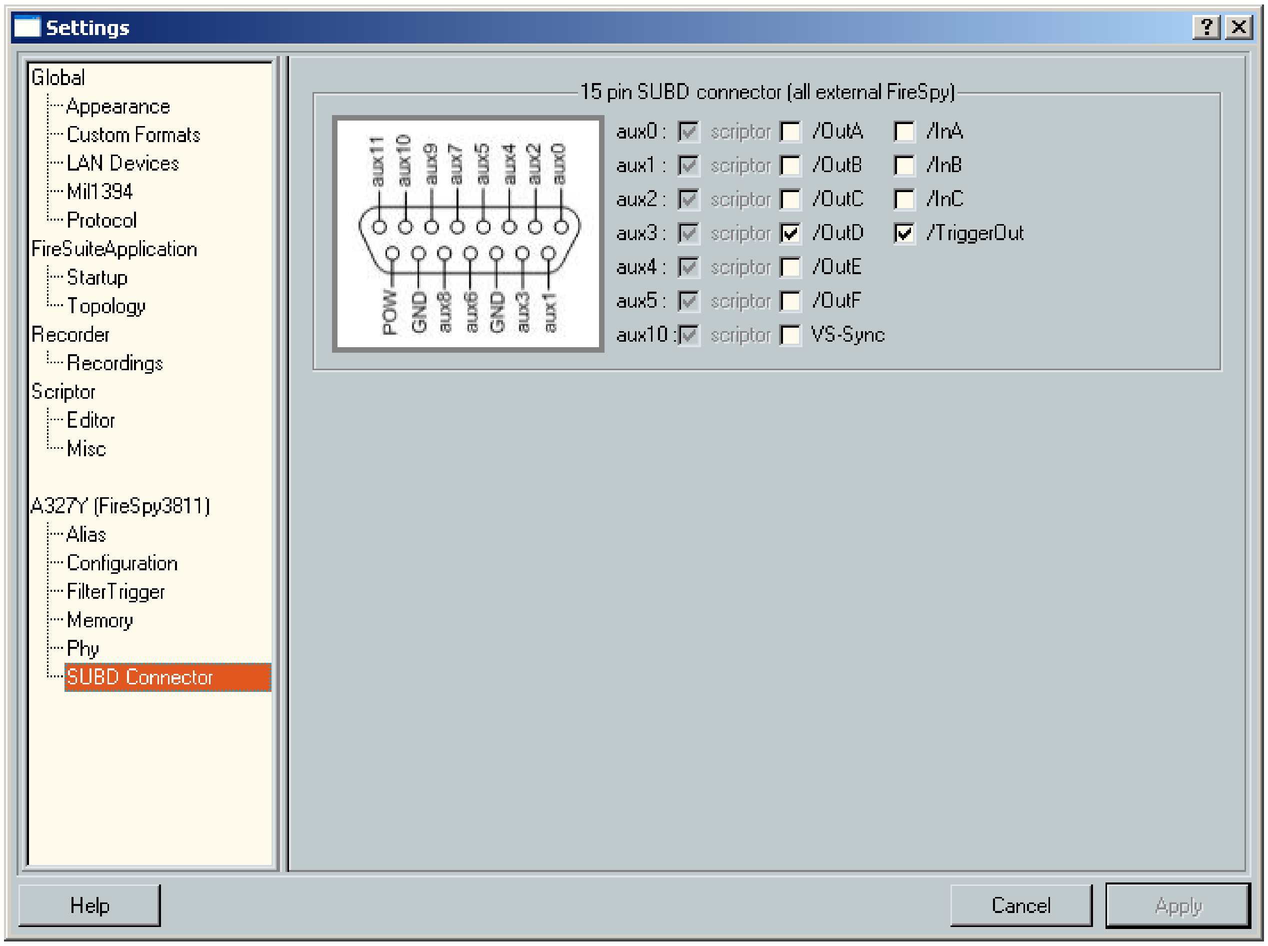
Bus Power
- [ ] Enable Bus Power A2
- [ ] Enable Bus Power B2
- [ ] Enable Bus Power C2

Root Node
When checked, the Analyzer will always try to become the root node after a bus reset.

Phy Options
Disable Active Link bit at startup: When set the Analyzer will set no active link layer in SelfID packets. This way the Analyzer looks like a "dumb" node on the bus.

Bus Power
This section is only available for the advanced series of analyzers.It contains a check box for each Analyzer node. By enabling the check box for a specific node, this node will provide power to the 1394 bus.
Analyzers with only a single node will only show "Enable Bus Power".
The FireSpy3810 and the FireSpy3850 contain a VHDCI connector, the VHDCI power is not on the ports itself but on the AUX cable.

20.2.6.9. SUBD Connector



15 pin SUBD connector (all external FireSpy)

aux0 :	[✓ scriptor]	[ ] /OutA	[ ] /InA
aux1 :	[✓ scriptor]	[ ] /OutB	[ ] /InB
aux2 :	[✓ scriptor]	[ ] /OutC	[ ] /InC
aux3 :	[✓ scriptor]	[✓ /OutD]	[✓ /TriggerOut]
aux4 :	[✓ scriptor]	[ ] /OutE	
aux5 :	[✓ scriptor]	[ ] /OutF	
aux10 :	[✓ scriptor]	[ ] VS-Sync	

/OutA, B, C, D, E, F and /InA, B ,C
These ports can be used as a part of the Trigger Sequencer and the General Trigger. The ports can also be controlled by Port I/O in the Scriptor.

/TriggerOut
This port can be associated with a trigger event.

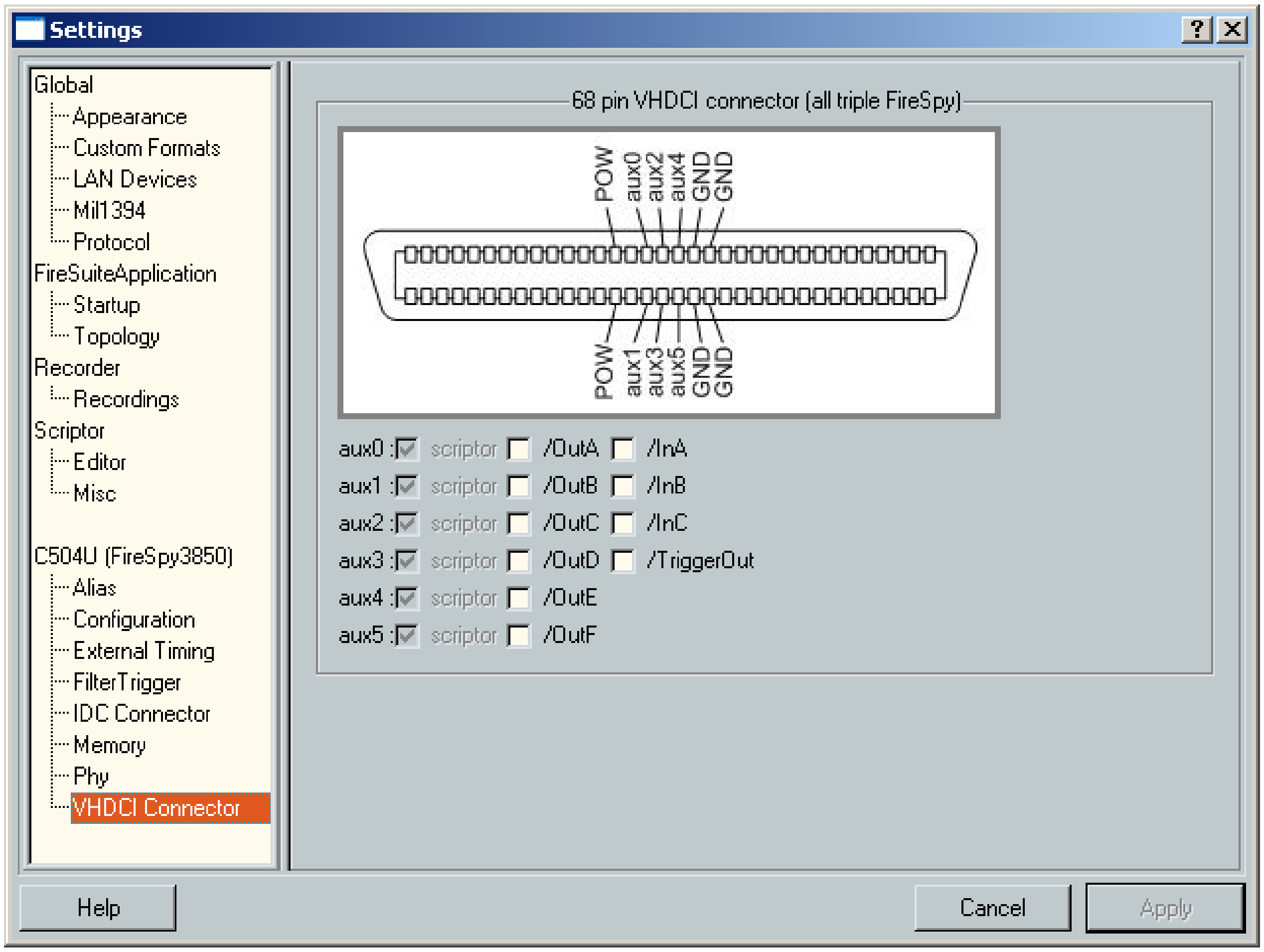
Mil1394-Sync
This port can be programmed as an external frame synchronization signal for the Scriptor function.

Electric characteristics

Input:
VIL: Vmin -0.2 Vmax 0.8
VIH: Vmin 2.0 Vmax 3.45

Output:
VOL: Vmax 0.4
IOL: 24 mA

20.2.6.10. VHDCI Connector



aux0 : [✓ scriptor]	[ ] /OutA	[ ] /InA
aux1 : [✓ scriptor]	[ ] /OutB	[ ] /InB
aux2 : [✓ scriptor]	[ ] /OutC	[ ] /InC
aux3 : [✓ scriptor]	[ ] /OutD	[ ] /TriggerOut
aux4 : [✓ scriptor]	[ ] /OutE	
aux5 : [✓ scriptor]	[ ] /OutF	

/OutA, B, C, D, E, F and /InA, B ,C
These ports can be used as a part of the Trigger Sequencer and the General Trigger. The ports can also be controlled by Port I/O in the Scriptor.

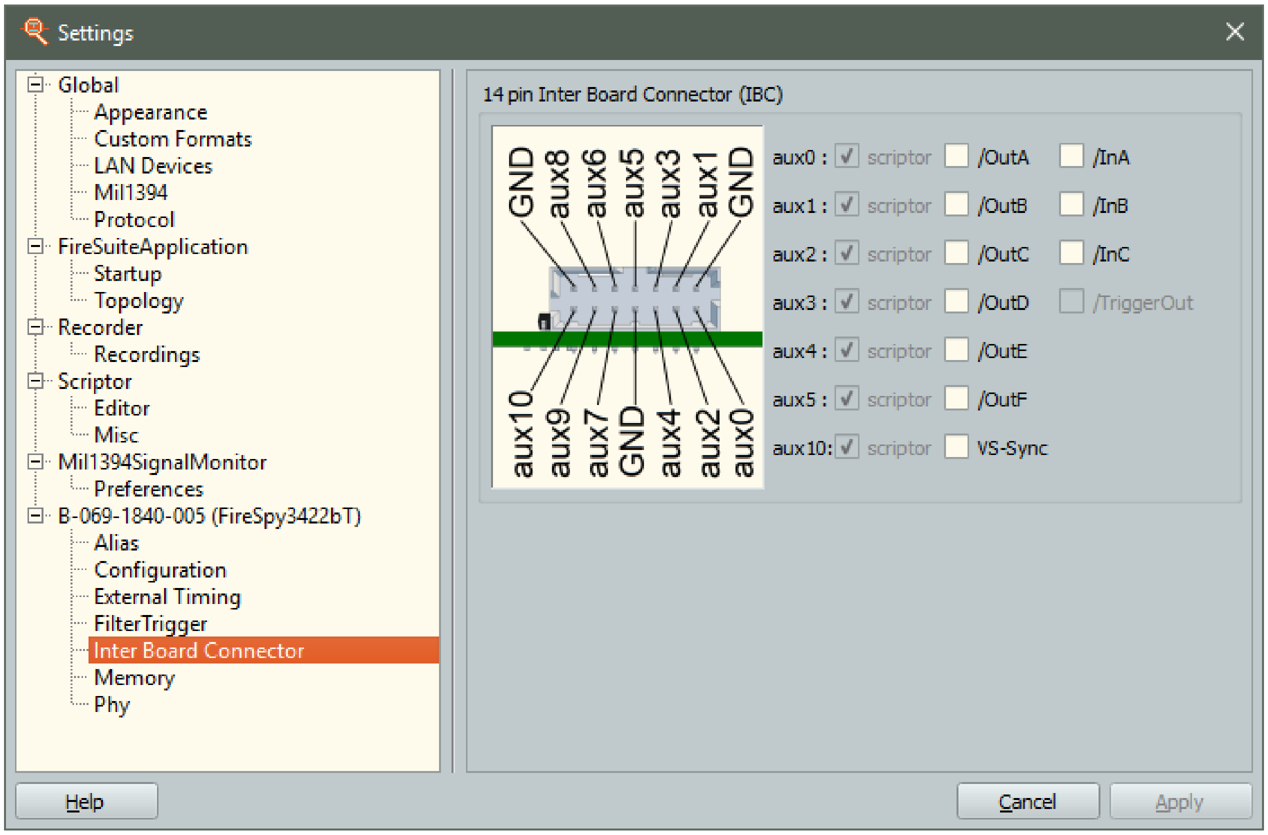
/TriggerOut
This port can be associated with a trigger event.

Electric characteristics

Input:
VIL: Vmin -0.2 Vmax 0.8
VIH: Vmin 2.0 Vmax 3.45

Output:
VOL: Vmax 0.4
IOL: 24 mA

20.2.6.11. Inter Board Connector



- Global
  - Appearance
  - Custom Formats
  - LAN Devices
  - Mil1394
  - Protocol
- FireSuiteApplication
  - Startup
  - Topology
- Recorder
  - Recordings
- Scriptor
  - Editor
  - Misc
- Mil1394SignalMonitor
  - Preferences
- B-069-1840-005 (FireSpy3422bT)
  - Alias
  - Configuration
  - External Timing
  - FilterTrigger
  - Inter Board Connector
  - Memory
  - Phy

14 pin Inter Board Connector (IBC)

aux0 :	✓ scriptor	☐ /OutA	☐ /InA
aux1 :	✓ scriptor	☐ /OutB	☐ /InB
aux2 :	✓ scriptor	☐ /OutC	☐ /InC
aux3 :	✓ scriptor	☐ /OutD	☐ /TriggerOut
aux4 :	✓ scriptor	☐ /OutE	
aux5 :	✓ scriptor	☐ /OutF	
aux10:	✓ scriptor	☐ VS-Sync	

/OutA, B, C, D, E, F and /InA, B ,C
These ports can be used as a part of the Trigger Sequencer and the General Trigger. The ports can also be controlled by Port I/O in the Scriptor.

/TriggerOut
This port can be associated with a trigger event.

Electric characteristics

Input:
VIL: Vmin -0.2 Vmax 0.8
VIH: Vmin 2.0 Vmax 3.45

Output:
VOL: Vmax 0.4
IOL: 24 mA

Chapter 21. Format Editor

Using the FormatEditor you can define format sets. A format set is stored in a file with .dff extension. Each format set consists of one or more format definitions which the Analyzer application can use to display formatted data in a table form or layout form.

For instance for the SBP2 protocol support of the Analyzer, a number of format sets have been defined. One format set for each supported SBP2 device type.
In a format set for the SBP2 protocol you will find format definitions that define the format for the 'command' or 'data' blocks. See 'FormatEditor and the SBP2 protocol' document for more details on this.

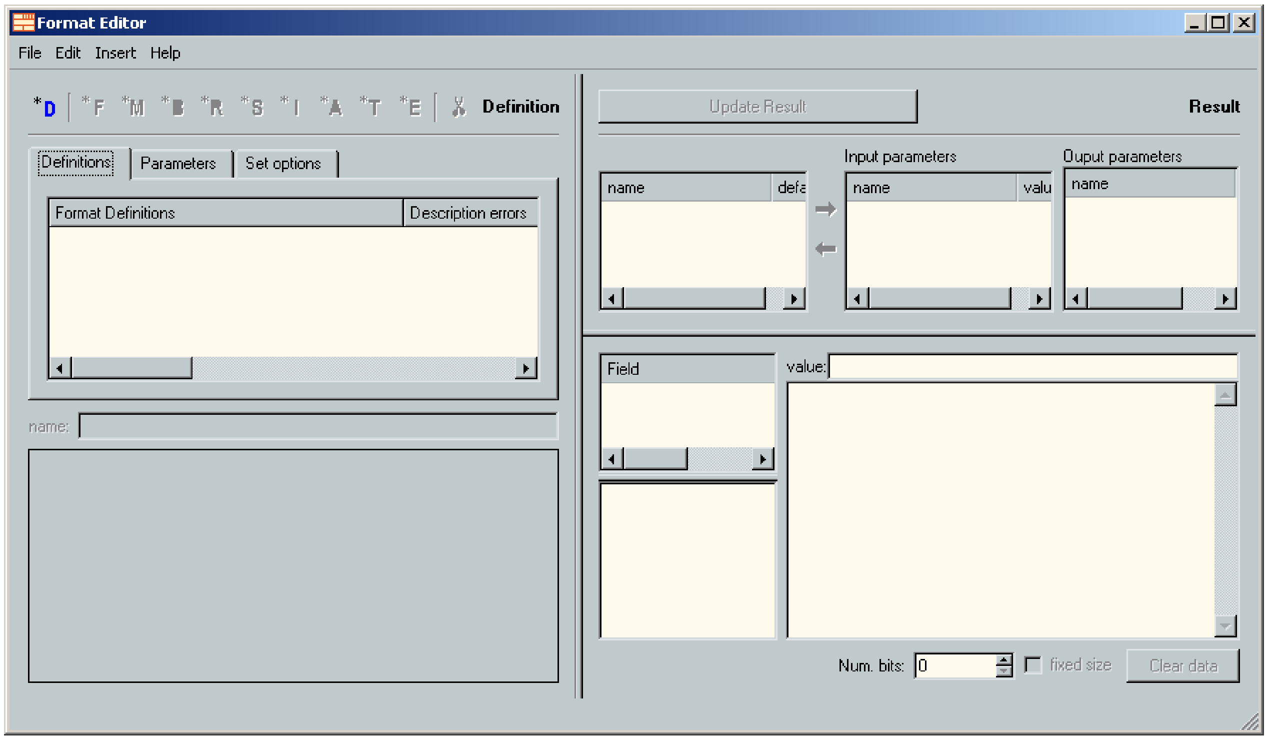
A format definition defines how the data block is build up with fields. A field is associated with one or more bits of the data block. It has a field name, a field type and some other properties.

The format definition defines sequentially all the fields of data. It can use a 'repeat' construction (like a while in C-language), it can use the 'if' and 'else' construction (similar to the C-language), it can use the 'switch' and 'case' construction (similar to the C-language) and it can use 'macro' calls (similar to function calls in the C-language).

A format definition can be defined using a simple to use interface, where you can add the 'field', 'repeat', 'if', 'switch' (etc) items in a tree and edit their properties.
After creating a definition, you can check the result. The result will be displayed in a table view and a layout view. You can change field values to check conditional behaviour of the format definition.

21.1. How to use it

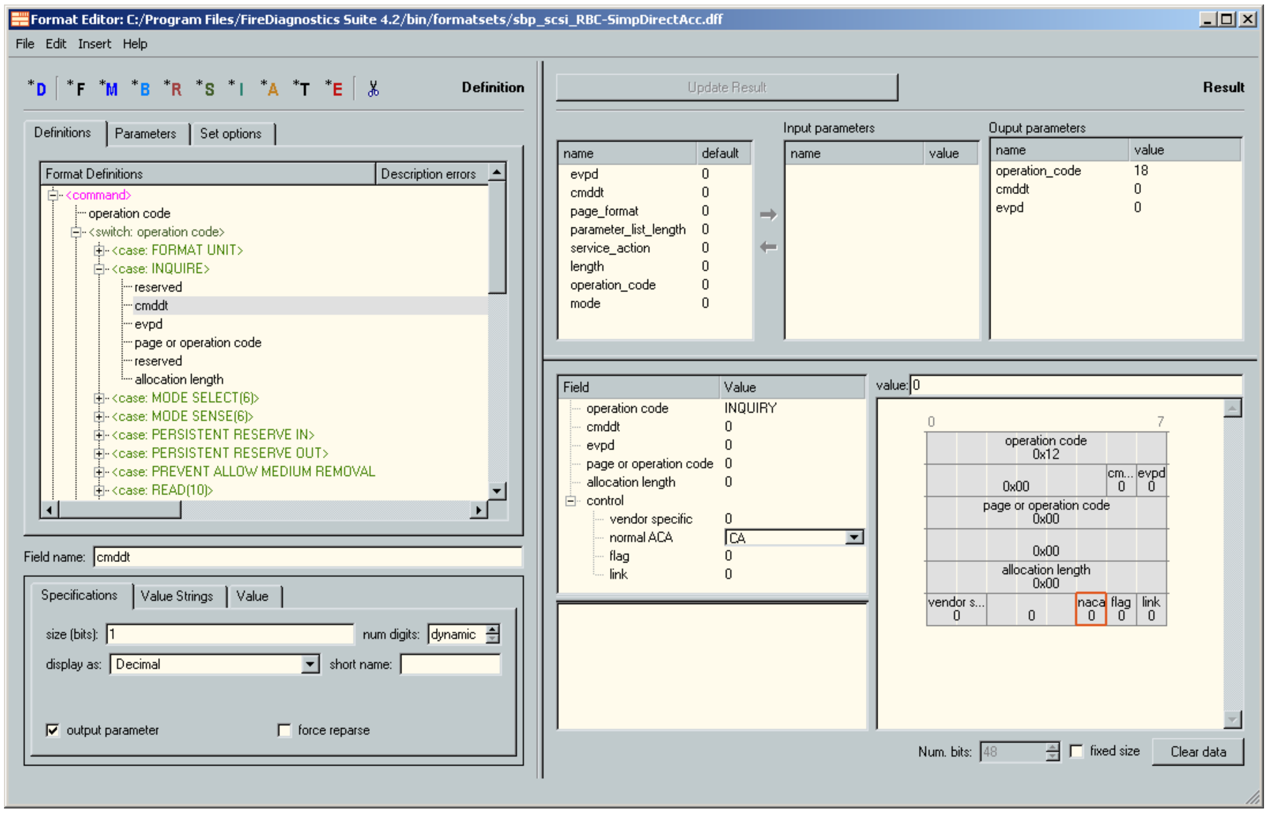
The Format Editor can be found in the same 'Start' menu as the Analyzer application. After starting it, the following window will show up:



In the left pane of the window (Definition pane), you can create the definitions tree. The properties of a selected item is shown below the tree and can be changed there.

In the right pane of the window (Result pane), you can check the resulting format of a selected definition. When the 'Update Result' button is clicked, the resulting format will be shown as a table and a layout view. You can change field values to verify correct behaviour.

In the following image, the format definition file of a SBP2 format set has been opened.



Format Editor: C:/Program Files/FireDiagnostics Suite 4.2/bin/formatsets/sbp_scsi_RBC-SimpDirectAcc.dff

Format Definitions	Description errors
<command>	
operation code	
<switch: operation code>	
<case: FORMAT UNIT>	
<case: INQUIRE>	
reserved	
cmddt	
evpd	
page or operation code	
reserved	
allocation length	
<case: MODE SELECT(6)>	
<case: MODE SENSE(6)>	
<case: PERSISTENT RESERVE IN>	
<case: PERSISTENT RESERVE OUT>	
<case: PREVENT ALLOW MEDIUM REMOVAL	
<case: READ(10)>	

Field name: cmddt

size (bits): 1 num digits: dynamic
display as: Decimal short name:
[✓] output parameter [ ] force reparse

name	default
evpd	0
cmddt	0
page_format	0
parameter_list_length	0
service_action	0
length	0
operation_code	0
mode	0

Input parameters

name	value

Ouput parameters

name	value
operation_code	18
cmddt	0
evpd	0

Field	Value
operation code	INQUIRY
cmddt	0
evpd	0
page or operation code	0
allocation length	0
control	
vendor specific	0
normal ACA	CA
flag	0
link	0

value: 0

0				7
operation code 0x12				
0x00			cm... 0	evpd 0
page or operation code 0x00				
0x00				
allocation length 0x00				
vendor s... 0	0	naca 0	flag 0	link 0

Num. bits: 48 [ ] fixed size

In the 'Definition' pane you see the format definitions tree. The name of the first definition is 'command'. It starts with a field with name 'operation code'. This item is selected, so the properties of this 'Field' item are displayed below. It is a 8 bit field of type 'Hexadecimal'.

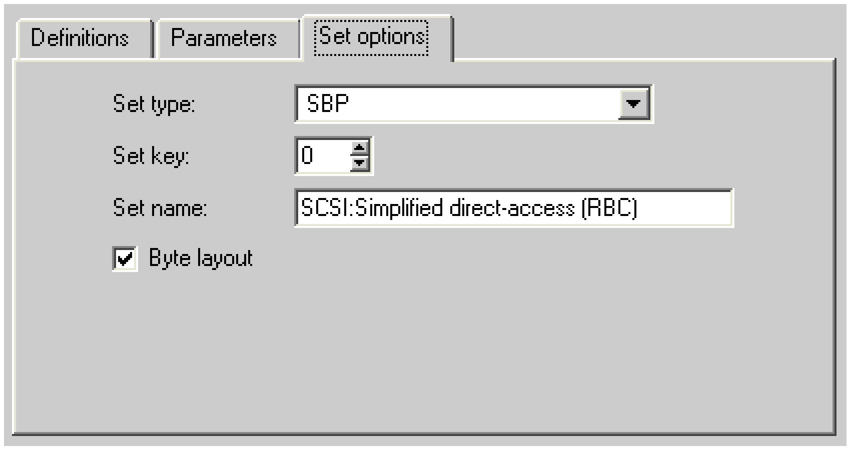
The fields following the operation code depend on the value of the operation code. Therefore a 'switch' item and its 'case' items are used to conditionally add additional fields. In the case of a 'MODE SENSE(6)' operation code, a reserved field is added followed by a 'disable block descriptors' field, another reserved field, a 'page control' field, etc.

In the 'Result' pane you see the result of the 'command' definition. For the 'operation code' we filled in the value 'MODE SENSE(6)' and as a result we got the fields as defined in the definition tree under the 'case' item for the 'MODE SENSE(6) case. Note that the reserved fields do not show up in the table view and are displayed in the layout view without a field name.

You can create your own definition starting with a new (empty) format set or load an existing one, change it to your needs and save it as a new format set. To be able to use the new format set you probably have to change some set options and you should tell the Analyzer application where it can find the new format set. This is described in one of the following sections.

21.1.1. Changing set options

To change the format set options, you have to select the 'Set options' tab (inside the 'Definition' pane. Below an example of the 'Set options' page is shown.


Definitions | Parameters | Set options

Set type: SBP

Set key: 0

Set name: SCSI:Simplified direct-access (RBC)

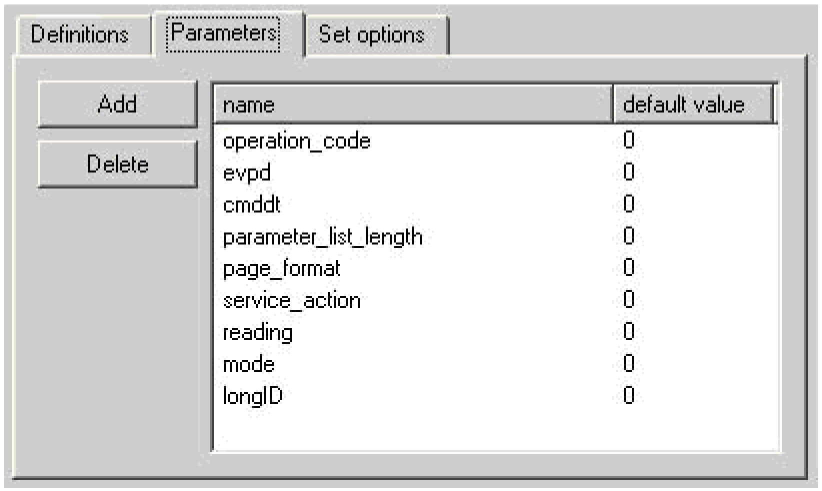
[✓] Byte layout

It defines an SBP (SBP2) Set type for a Direct-access (SBC) device. The Set type and Set name and sometimes the Set key are used by the Analyzer to find the correct format definitions. How these fields need to be filled in depends on the protocol the format set is used for. For details on this, see the corresponding document file (e.g. the 'Editing SBP2 formats' section for more details on these Set options for the SBP2 protocol).

21.1.2. Input and Output parameters

Input parameters
Each definition can use input parameters to control the format. Each parameter you want to use needs to be defined in the 'Parameters' tab in the 'Definitions' pane. When a parameter has been defined, you can use it in any expression.

In the following image, the 'Parameters' page of a SBP2 format set is shown.



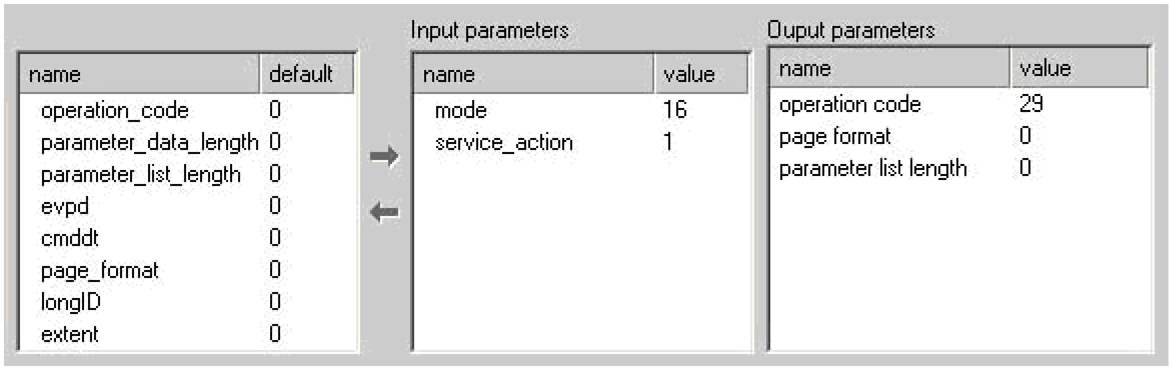
name	default value
operation_code	0
evpd	0
cmddt	0
parameter_list_length	0
page_format	0
service_action	0
reading	0
mode	0
longID	0

When one of the definitions of the format set is executed, zero or more of these parameters will be filled in (for the others, the default value will be used).
For an example of the use of input parameters, see Editing SBP2 Formats.

Output parameters
When a definition is executed, it can also generate output parameters. The 'command' definition shown above for instance, will generate the 'operation_code' parameter. The value will be the value of the corresponding field. For an example of the use of output parameters, see Editing SBP2 Formats.

Checking Input and Output parameters
In the 'Result' pane you can define values for all input parameters by moving them to the 'Input parameters' box. All generated Output parameters will be shown in the 'Output parameters' box.

The following images shows an example where a few input parameters are defined and where the selected format definition results in some output parameters (note that in this example the input parameters are not really needed for the selected format definition).



name	default
operation_code	0
parameter_data_length	0
parameter_list_length	0
evpd	0
cmddt	0
page_format	0
longID	0
extent	0

Input parameters

name	value
mode	16
service_action	1

Ouput parameters

name	value
operation code	29
page format	0
parameter list length	0

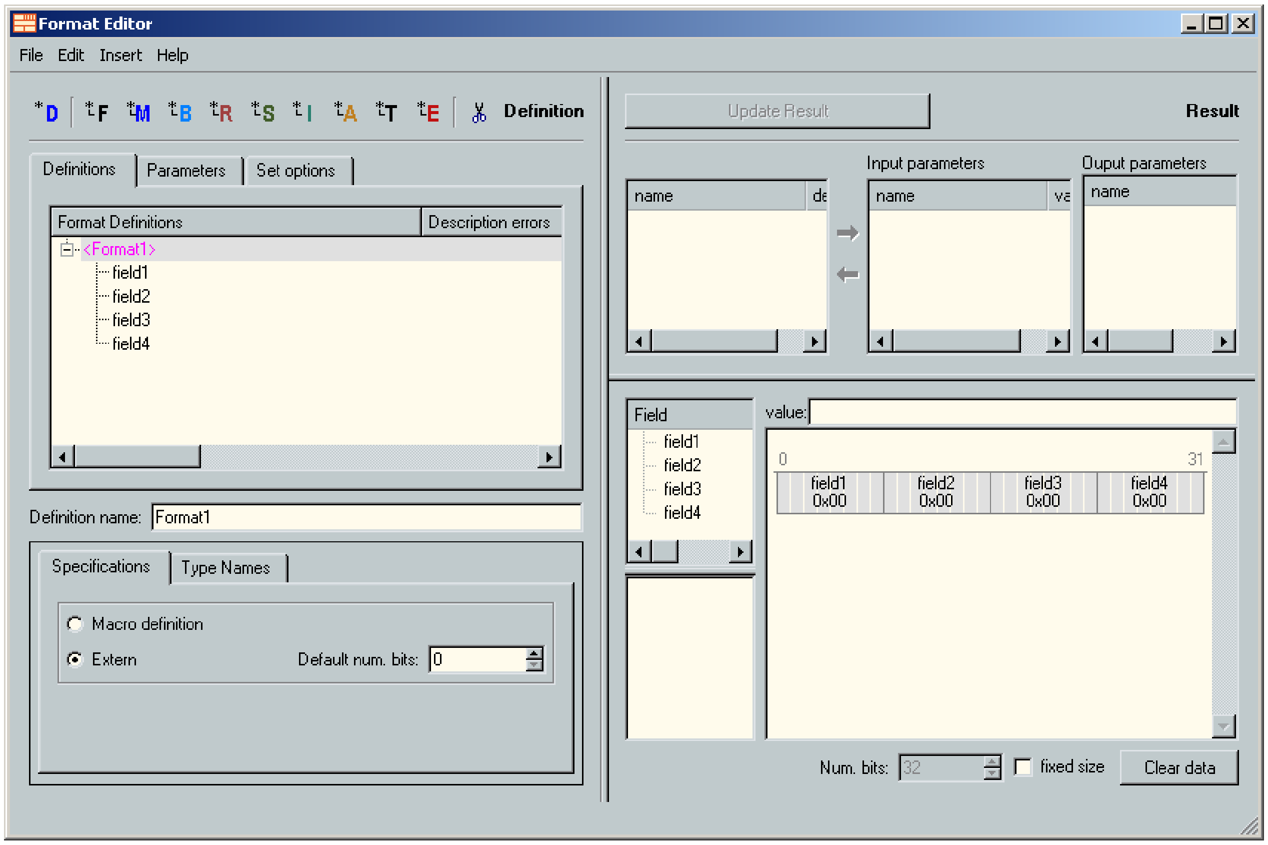
21.1.3. Built-in variables and functions

The following built-in variables and functions can be used in expressions:

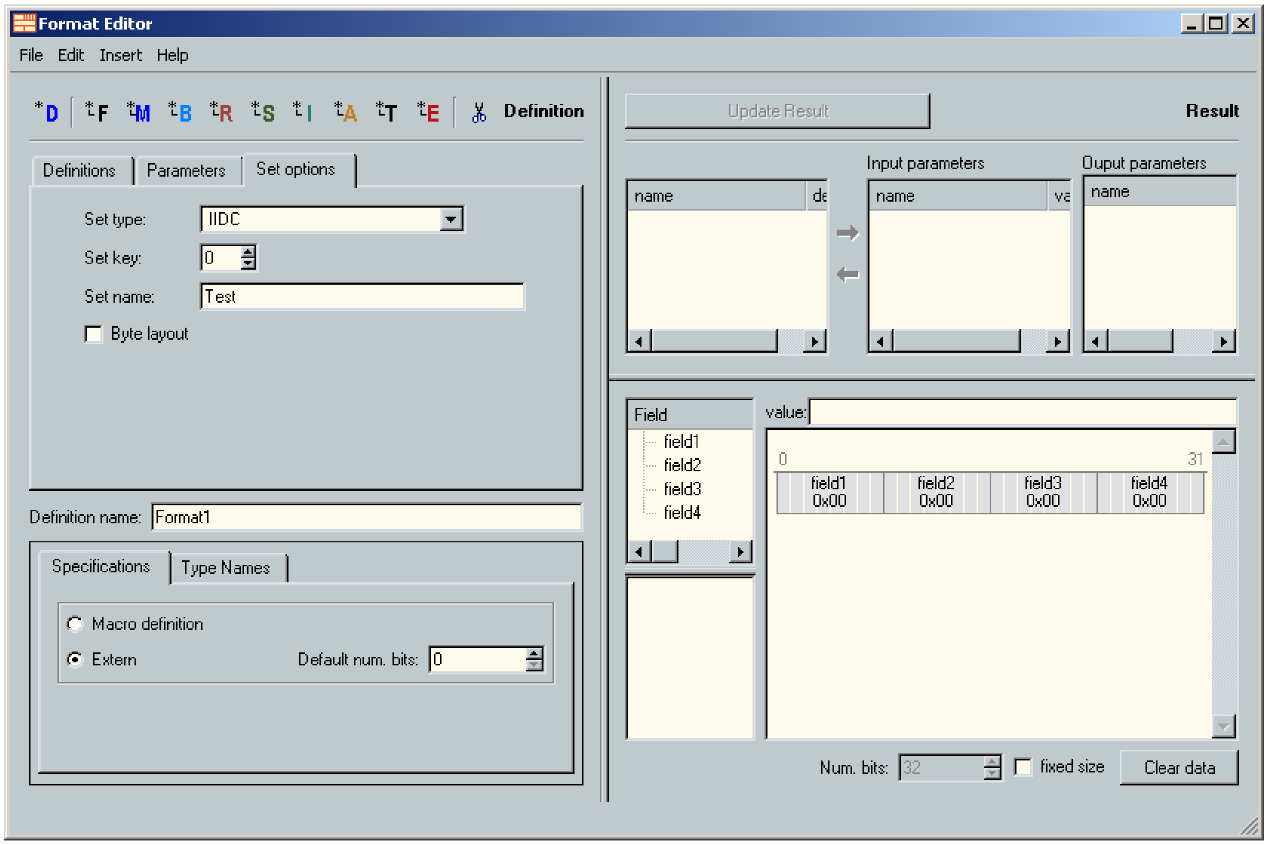
Name	Description
databits	This variable reflects the number of bits in the data block to be formatted.
formatbits	This vaiable reflect the number of bits already formatted.
remainingbits	This is databits-formatbits.
par0, par1	These are parameters for the macros
var0, var1, var2,var3, var4, var5, var6, var7, var8, var9	These are global variables that can be used anywhere for temporary holding values.
count()	This function returns the number of times the last repeat loop has been executed.
size("field")	This function returns the size of the indicated field in bits. Parameter field should be the name of a field that can be found by searching upwards in the definition tree.
value("field")	This function returns the value of the indicated field. Parameter field should be the name of a field that can be found by searching upwards in the definition tree.
valuenextbits(n)	This function returns the value of the n data bits starting at the current position.
valuebits(s,n)	This function returns the value of the n data bits starting at bit number s.

21.1.4. Using Format Sets in the FireSpy Application

Suppose we have a format definition as in the example below. This format set "exports" one format as is indicated by the purple color of the tree node "Format 1". Also, the "Specifications" tab for this tree node indicates that it is "Extern" and therefore, will be shown in the Recorder or Scriptor when this format set is loaded.


Format Editor
File Edit Insert Help
D F M B R S I A T E Definition
Update Result
Result
Definitions Parameters Set options
Format Definitions Description errors
<Format1>
field1
field2
field3
field4
Definition name: Format1
Specifications Type Names
Macro definition
Extern Default num. bits: 0
Input parameters Ouput parameters
name de name va name
Field value:
field1 field2 field3 field4
0 31
field1 0x00 field2 0x00 field3 0x00 field4 0x00
Num. bits: 32 fixed size Clear data

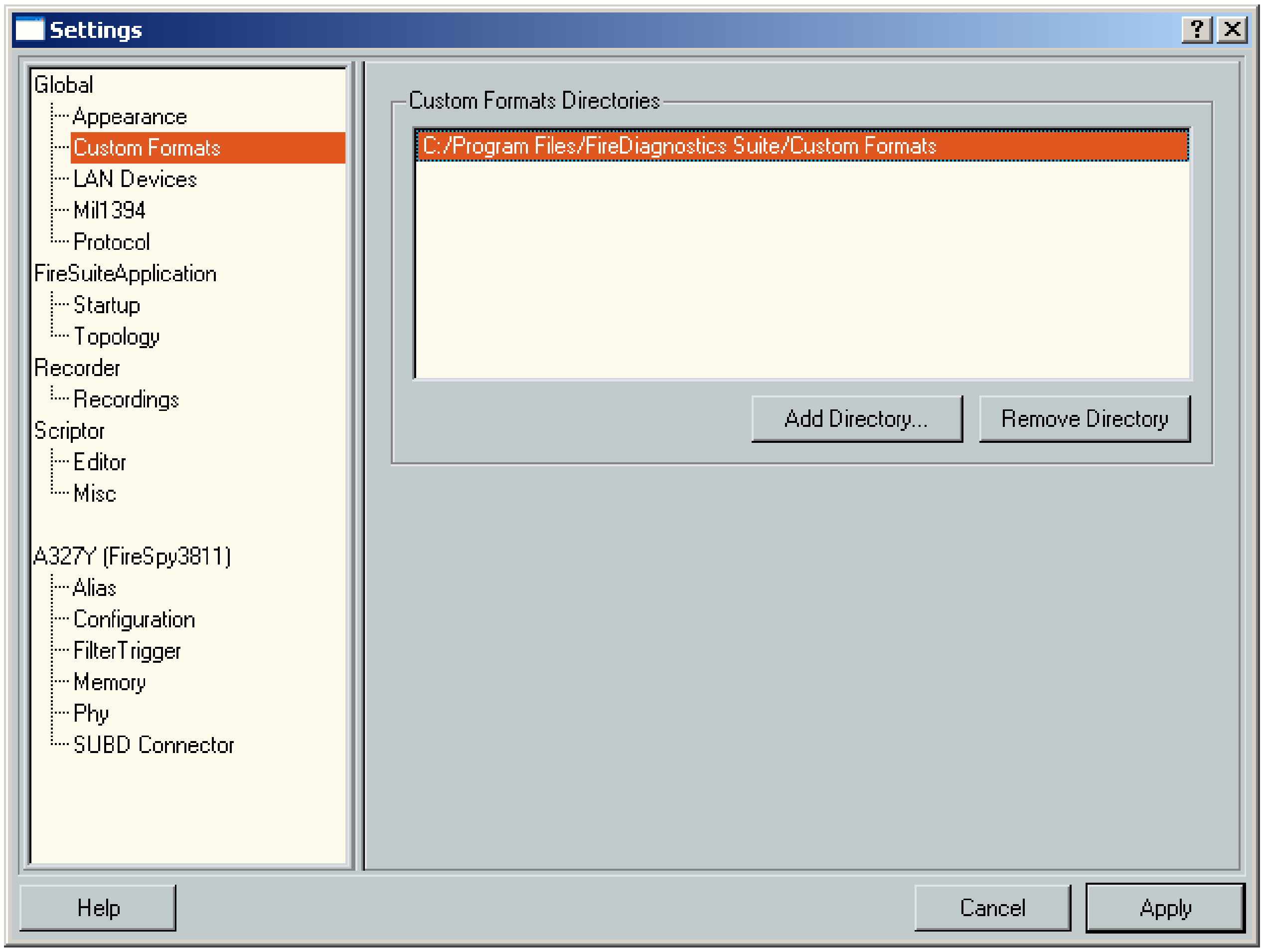
The "Set Options" tab for this example looks as follows:


Format Editor
File Edit Insert Help
D F M B R S I A T E Definition
Update Result
Result
Definitions Parameters Set options
Byte layout
Definition name: Format1
Specifications Type Names
Macro definition
Extern Default num. bits: 0
Input parameters Ouput parameters
name de name va name
Field value:
field1 field2 field3 field4
0 31
field1 0x00 field2 0x00 field3 0x00 field4 0x00
Num. bits: 32 fixed size Clear data

Set type: IIDC
Set key: 0
Set name: Test

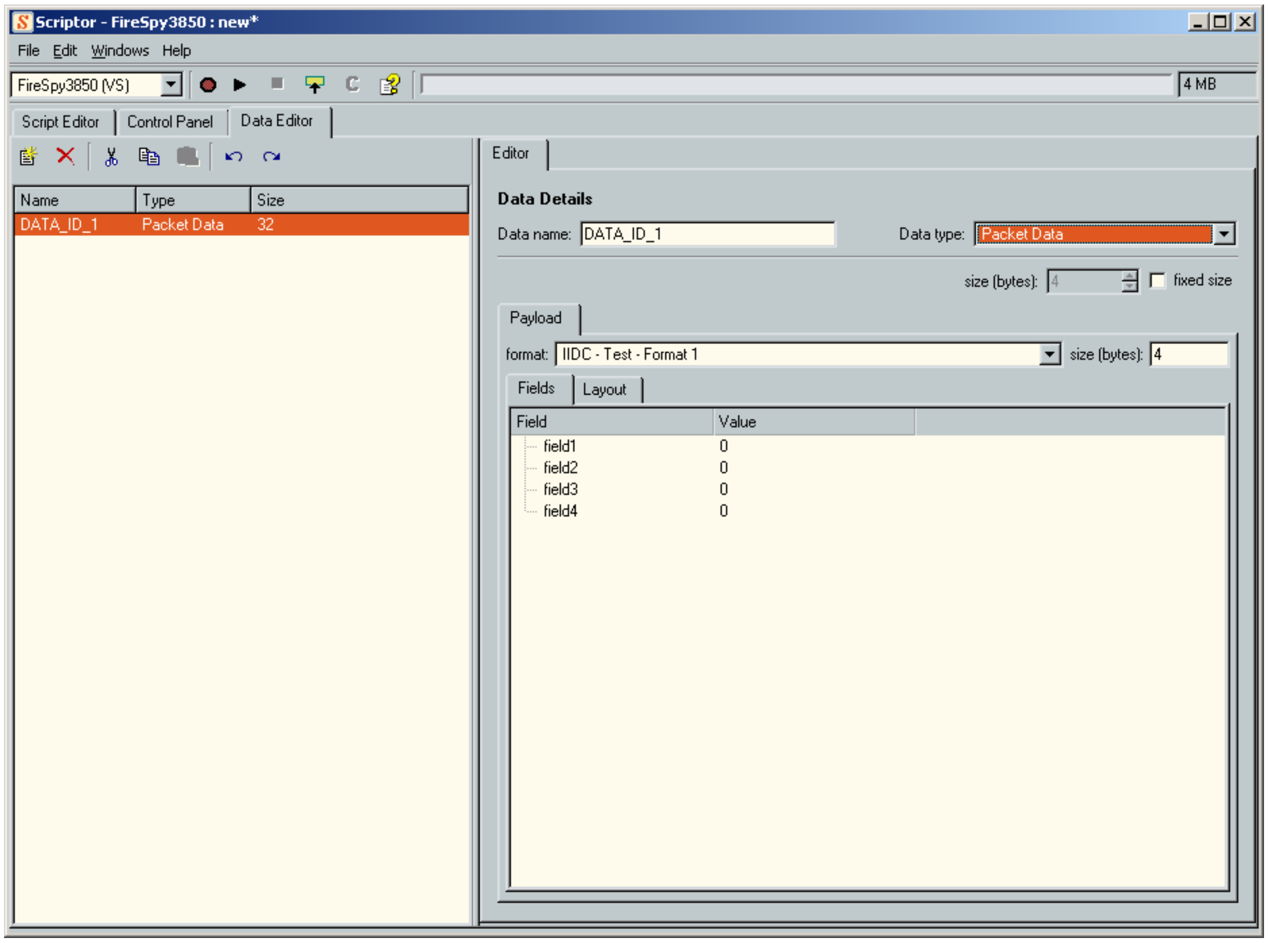
A set should always be of one of the known "Set Types". This enables automatic loading by the Analyzer application. Furthermore, the field "Set name" should not be set to "general" as this name is used by all built-in format sets. Please use something else.

To use your own format sets, you have to tell the Analyzer application where to find them. This can be done in the Settings dialog of the Analyzer application. The picture below shows the relevant section of the Settings dialog for the above example.



C:/Program Files/FireDiagnostics Suite/Custom Formats

After restarting the Analyzer application, the new format should be available. The following image shows how to locate it in the Packet Data Editor of the Scriptor.



Name	Type	Size
DATA_ID_1	Packet Data	32

Data Details

Data name: DATA_ID_1

Data type: Packet Data

size (bytes): 4 ☐ fixed size

format: IIDC - Test - Format 1

size (bytes): 4

Field	Value
field1	0
field2	0
field3	0
field4	0

Chapter 22. Protocol Editor

The Protocol editor is a tool to create a protocol definition for a specific high-level protocol. This high-level protocol could be one of the standard high-level protocols for example: AV/C, SBP-2, IIDC, etc. or your own custom-made high-level protocol. The protocol definition lays down the behaviour of the high-level protocol. At this moment only read and write actions of a high-level protocol can be lie down. In the near future it will also be possible to lay down the spin-off effects of a read or write action.

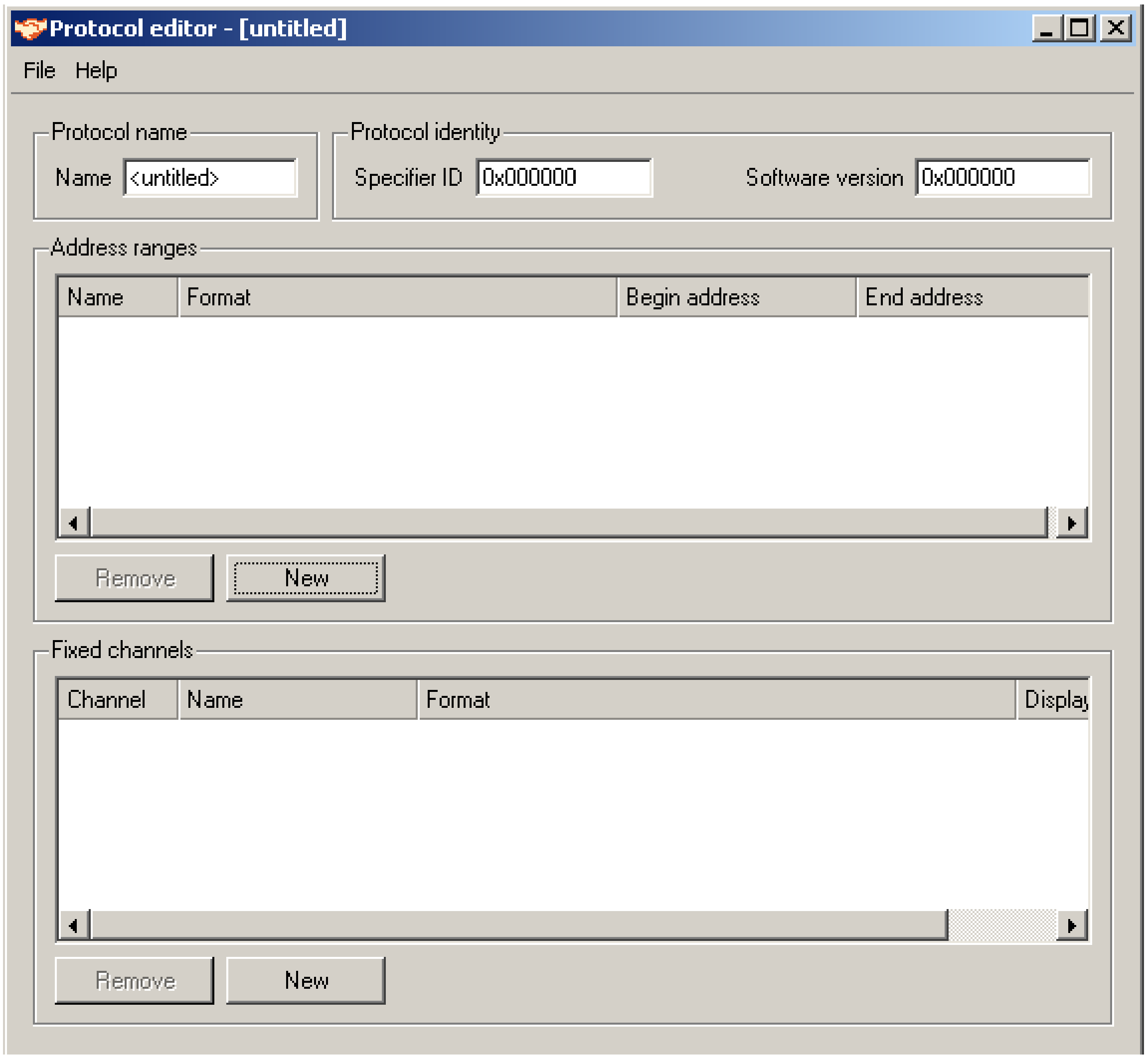
With help of the created protocol definitions the protocol analyzer of the Analyzer Recorder is capable to take into account the high-level protocols the protocol definitions lay down. Its result will be displayed in the Protocol View.

See section Custom Protocols for a description about how to use custom protocols in the Recorder.

22.1. How to use it

You can open the Protocol editor by double clicking the file ProtocolEditor.exe. This file is located in the directory: AnalyzerB x_x_x\ bin (the x stands for the version number of the release).

After double clicking this file the Protocol editor will be displayed on the desktop. It is shown in the figure below.



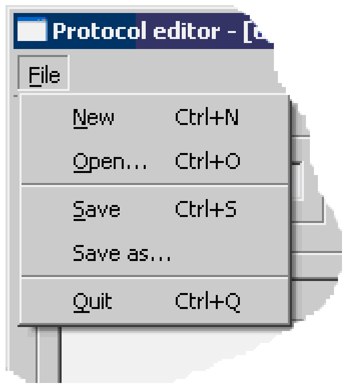
22.2. Details

22.2.1. Caption Bar

In the caption bar you see the filename between brackets of the displayed protocol definition. If a new created protocol definition is displayed as shown above and it hasn't been saved to file yet 'untitled' is displayed between the brackets. If you edit the displayed protocol definition a '*' is shown behind the filename. It indicates the protocol definition is changed and it's not in sync with the one saved to file.

22.2.2. File Menu

At the top of the Protocol editor you see the File menu. The menu items of the File menu are shown in the figure below.



Menu item New creates a new protocol definition and will display it. If the current displayed protocol definition has changes you will be asked to save these changes to file.

Menu item Open… opens a File open dialog. In this dialog you can specify the filename and location of the protocol definition file (extension .psf) to open. If the current displayed protocol definition has changes you will be asked to save these changes to file.

Menu item Save saves the displayed protocol definition to file. If this is the first time the protocol definition is saved to file the Save as dialog will be opened. With this dialog you can specify the filename and location for the protocol definition to save to file.

Menu item Save as… opens the Save as dialog. In this dialog you can specify a filename and location for the displayed protocol definition to save to file. This filename may be different than the filename you used to open the protocol definition. In this case the protocol definition will be saved to another file.

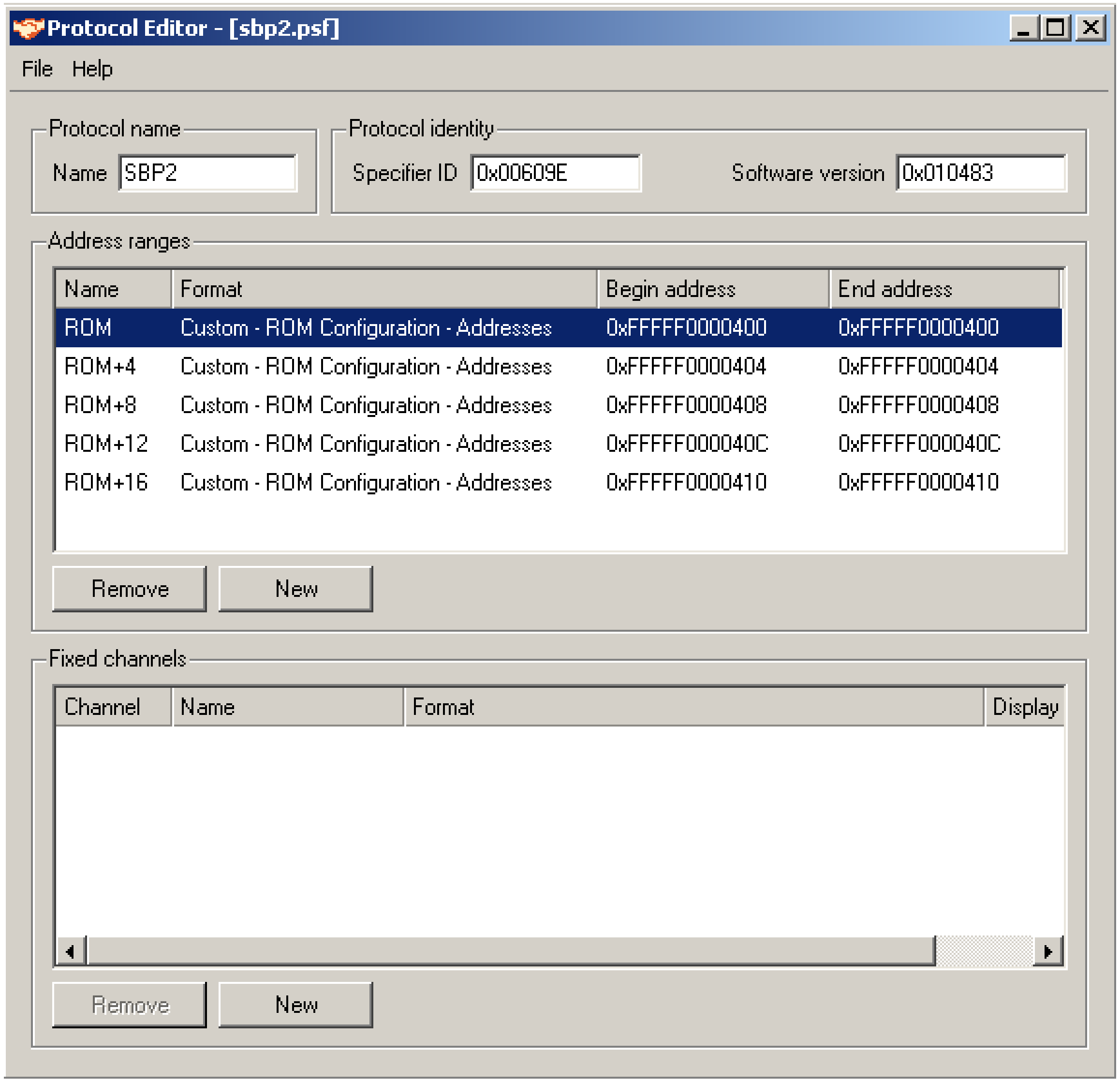
Menu item Quit quits the Protocol editor. If the displayed protocol definition has changes you will be asked to save these changes to file.

22.2.3. Protocol name & identity

Below the File menu you see the text fields for the Protocol name and Protocol identity.

With the Name text field you can specify the name of the protocol. The Protocol View will use this name for a tab page. This page will show the result of the protocol after a protocol analyze of the recorded data.

With the Specifier ID and Software version text fields you can specify the specifier ID and software version of the protocol. These 2 fields should uniquely identify the protocol. The specifier ID value and software version value for the SBP-2 protocol is shown in the figure below. In this figure file ‘sbp2.psf’ in directory ‘examples’ is opened.



Protocol name

Name: SBP2

Protocol identity

Specifier ID: 0x00609E

Software version: 0x010483

Address ranges

Name	Format	Begin address	End address
ROM	Custom - ROM Configuration - Addresses	0xFFFFF0000400	0xFFFFF0000400
ROM+4	Custom - ROM Configuration - Addresses	0xFFFFF0000404	0xFFFFF0000404
ROM+8	Custom - ROM Configuration - Addresses	0xFFFFF0000408	0xFFFFF0000408
ROM+12	Custom - ROM Configuration - Addresses	0xFFFFF000040C	0xFFFFF000040C
ROM+16	Custom - ROM Configuration - Addresses	0xFFFFF0000410	0xFFFFF0000410

Fixed channels

Channel	Name	Format	Display

22.2.4. Address ranges

Below the text fields Protocol name and Protocol identity you see the Address ranges table and its buttons Remove and New.

Table Address ranges displays the created address ranges. Each row of the table displays an address range. An address range comprises a name, a format, a begin address and an end address.

The address range name is displayed in the column Name. You can edit it by clicking on it. The name of an address range identifies an address range displayed in the Protocol View.

The address range format is displayed in column Format. You can select a format from a combo-box by clicking on it. The combo-box displays the formats of type Custom. You must create these formats with the Format editor. As an example, please open format ROMConfig.dff in the Format editor.

The address range begin address is displayed in column Begin address. You can edit it by clicking on it.

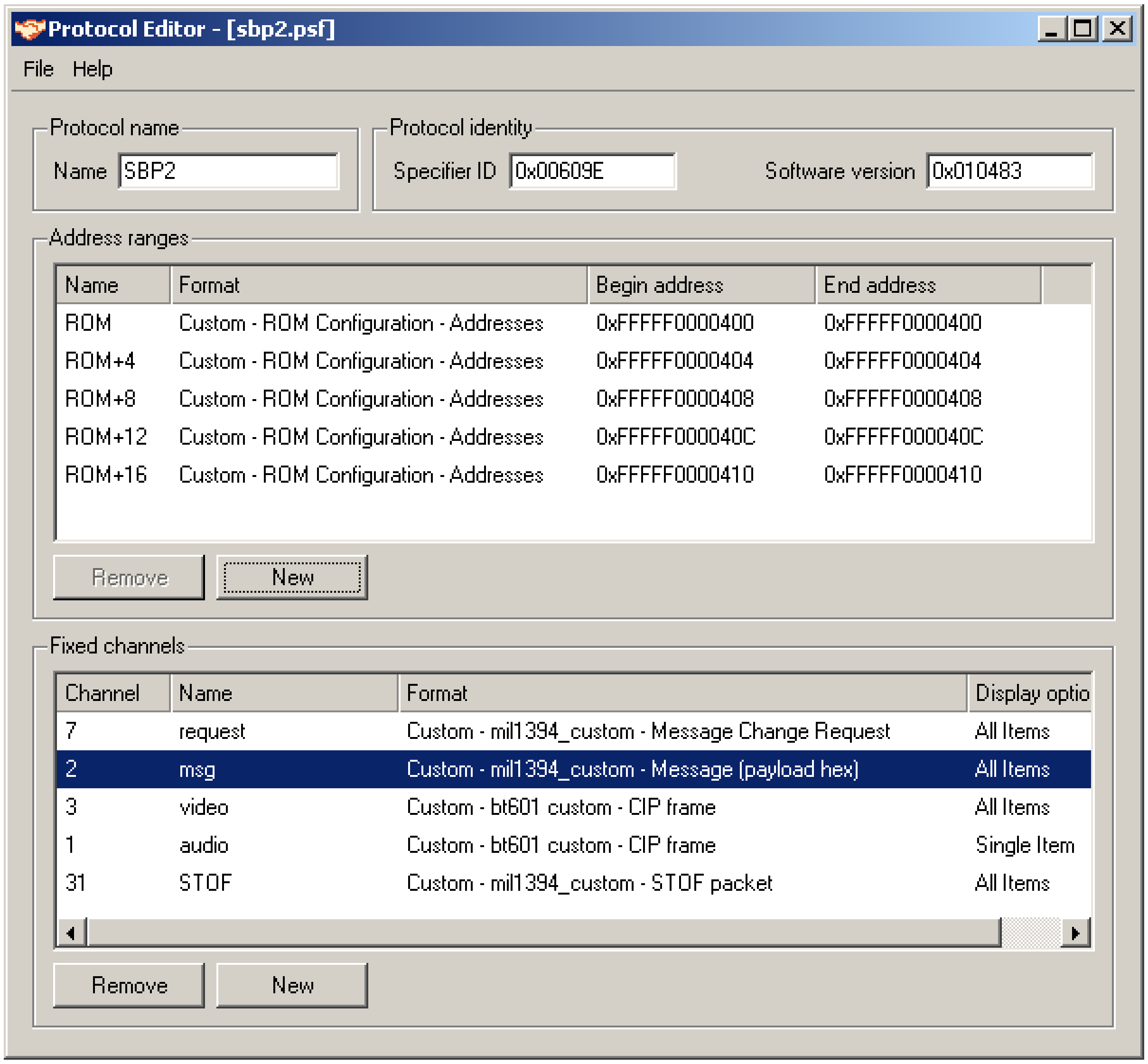
The address range end address is displayed in column End address. You can edit it by clicking on it.

Button Remove is enabled if a row in table Address ranges is highlighted. Clicking this button will remove the highlighted row from table Address ranges.

By clicking button New you can create a new address range. The new created address range is displayed at the last row of the table Address ranges.

22.2.5. Fixed channels

Already defined custom protocol you can extend with fixed channels. It means that you can assign custom isochronous protocol to one channel of your choice.
Image below illustrates five fixed channels within SBP-2 protocol.



Protocol name
Name: SBP2

Protocol identity
Specifier ID: 0x00609E
Software version: 0x010483

Address ranges

Name	Format	Begin address	End address
ROM	Custom - ROM Configuration - Addresses	0xFFFFF0000400	0xFFFFF0000400
ROM+4	Custom - ROM Configuration - Addresses	0xFFFFF0000404	0xFFFFF0000404
ROM+8	Custom - ROM Configuration - Addresses	0xFFFFF0000408	0xFFFFF0000408
ROM+12	Custom - ROM Configuration - Addresses	0xFFFFF000040C	0xFFFFF000040C
ROM+16	Custom - ROM Configuration - Addresses	0xFFFFF0000410	0xFFFFF0000410

Fixed channels

Channel	Name	Format	Display optio
7	request	Custom - mil1394_custom - Message Change Request	All Items
2	msg	Custom - mil1394_custom - Message (payload hex)	All Items
3	video	Custom - bt601 custom - CIP frame	All Items
1	audio	Custom - bt601 custom - CIP frame	Single Item
31	STOF	Custom - mil1394_custom - STOF packet	All Items

Chapter 23. File Formats

This chapter describes all file formats supported by the Analyzer software that were intended for use by the customer.

23.1. Hex Data file

Hex data files have the extension .hex and contain one or more data blocks in hexadecimal format. The file is an ascii file and the new lines may be indicated with a cariage-return or line-feed, or both. The hex characters may use lower case or upper case characters.

The data is stored as a sequence of hexadecimal bytes. Between bytes a space or new line may be present. The following example defines 16 bytes of data:

```
FF FF 00 A3 33 8a ee ee
12 34 cd ef 00 00 00 00
```

The same 16 bytes could also be written as:

```
FFFF 00A3 338a eeee 1234 cdef 0000 0000
```

Or even as:

```
FFFF00A3338aeeee1234cdef00000000
```

If more than one blocks of data are present, the blocks are separated by one or more lines containing no data. The example below for instance contains three data blocks. All three are 20 bytes long. All three have a different way of byte spacing. These blocks could for instance be packets.

```
FF FF 00 8F FF C5 FF FF F0 00 02 00 BA B2 40 24
48 5C EE AA

FFFF008FFFC5FFFFF0000200BAB25024580D75B9

FFFF008F
FFC5FFFF
F0000200
BAB26024
68FFD88C
```

23.2. Quadlets Data file

Quadlets data files have the extension .qdl and contain one or more data blocks consisting of quadlets, stored as hexadecimal, decimal or floating-point numbers. The file is an ascii file and the new lines may be indicated with a cariage-return or line-feed, or both. The hex characters may use lower-case or upper-case characters.

The data is stored as a sequence of quadlets. Between the qaudlets a space or new line may be present. Each qaudlet can be written as:

- *Hexadecimal*
  Hexadecimal quadlets are followed by a 'h' character.
  For example: `FFFF0008h` or `0h`.
- *Decimal*
  Decimal quadlets start with a '-' or digit and only contain digits.
  For example: -65152 or 1450744508 or 0.
- *Floating point*
  Floating-point quadlets start with a '-' or '.' or digit and contain minimal a '.', 'e' or 'E'.
  For example: 3.14159 or -6.666e-8 or .11 or 0.0 or 1E6.
  Note: When quadlets are imported from a quadlet file, then these floating-point values will be converted to standard 32 bits IEEE754 values. Thus the quadlet 1000 (decimal) will not be the same as 1e3 (floating point)!.If quadlets are reported to a quadlets file in floating-point format, then the quadlets are supposed to be IEEE754 values.

The following example defines 5 quadlets of data:

```
FFFF008Fh
FFC5FFFFh
F0000200h
BAB27024h
78AE439Fh
```

The same 5 quadlets could also be written as:

```
4294901903 4291166207 4026532352
3132256292 2024686495
```

If more than one blocks of data are present, the blocks are separated by one or more lines containing no data. The example below for instance contains three data blocks. All three are 20 bytes long.

```
FFFF008Fh FFC5FFFFh F0000200h BAB27024h 78AE439Fh

FFFF008Fh
FFC5FFFFh
F0000200h
BAB27024h
78AE439Fh

4294901903 4291166207 4026532352
3132256292 2024686495
```

23.3. FireSpy packet file

Analyzer Packet files have the extension .fsp and contain one or more packets in binary form.

The file consists of an array of 32-bit values, and thus its size will be a multiple of 4 bytes long.
These 32-bit values are stored with least-significant byte first (little endian).

One 32-bits value:

least-significant byte	byte	byte	most-significant byte

Using these 32-bits values, the file starts with a header and contains one or more blocks.

File:

header
block 1
....
block N

The header consists of two 32-bit values, one file identification and one version number.

Header:

file id
version

The file id has a fixed value of `0x705346AE`.

Version has the value `0xLLMMHH00`, with LL = lower version number, MM = middle version number and HH = high version number. The current software version is 1.0.0 and thus the value is `0x00000100`.

Each block consists of the block header, followed by the block data.

block header
block data

The block header is one 32-bit value.
The upper 12 bits define a block id and the lower 20 bits define the length of the block data in bytes.
If the length is not a multiple of 4 bytes, the block data is padded with zero-value bytes.

The only block type defined at this moment is a packet block it has a block id of 0, the specified length in the header equals the number of packet bytes and the block data consists of the packet quadlets.

example

The following (32-bit) values define a packet file with two isochronous packets, the first is 8 bytes long and the second is 20 bytes long.

```
value----------remarks----------------------------------

0x705346AE file id
0x00000100 version
0x00000008 block header (packet block with length 8)
0x000001A0 packet bytes 1-4
0x1BCDB556 packet bytes 5-8
0x00000014 block header (packet block with length 20)
0x000800A0 packet bytes 1-4
0xC71110B2 packet bytes 5-8
0x0178006A packet bytes 9-12
0x8080FFFF packet bytes 13-16
0x6719CF8D packet bytes 17-20
```

Note that the file is a binary file. The quadlets of the example above are stored binary. The example file will be 44 bytes long (11 quadlets)

23.4. Recorder Regeneration file

Recorder Regeneration File Definition

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
File ID																															
Reserved for future use																					TF			File Version							
Fields																															
Items																															

File ID: Fixed value of 0xAE52476E (32 bits)
TF: Time Format (3 bits)
 0: No time field
 1: Absolute time field (64 bits)
 2: Frame/Cycle Offset time field - Selected if "Time Offset from STOF" is checked. (32 bits)
File Version: Currently 1 (8 bits)
Fields: List of Fields, end is indicated with 0
Items: List of Items, end is indicated with 0

Field Definition

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	Size															Field Type ID															
Number (indicated by Size) of data Quadlets																															

C: Custom Field Indicator (1 bit)
 0: Dap Field Definition
 1: User-defined Custom Field Definition
Size: Number of Data Quadlets
Filed Type ID: 0 - Empty field (last one in list)

Filed Contents - Type 1 - Time Resolution Frequency

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x01															0x01															
Time Resolution Frequency																															

Time Resolution Frequency: 32 bits

Item Definition

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Item Type Specific																								Item Type ID							
Fields																															
Item Type Specific																															

Item Type ID: (8 bits)
- 0: Empty Item (last one in list)
- 1: Start of frame / cycle
- 2: Unformatted packet
- 3: Stream packet

Fields: List of Fields, end is indicated with 0

Item Contents - Type 1 - Start of Frame / Cycle

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FS Node				Reserved																				0x01							
Fields																															
Previous Frame Length																															

Inserted before every STOF packet if "Time Offset from STOF" is checked.

FS Node: Analyzer node - for Triple models (4 bits)
- 0: Node A
- 1: Node B
- 2: Node C

Fields: List of Fields, end is indicated with 0

Previous Frame Length: Delta time in microseconds between this item and the previous Start of Frame / Cycle item.

Item Contents - Type 2 - Unformatted Packet

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FS Node				Speed Code				Reserved																0x02							
Fields																															
Size in Quadlets																															
Raw Packet																															

FS Node: Analyzer node - for Triple models (4 bits)
- 0: Node A
- 1: Node B
- 2: Node C

Speed Code: Transmission speed (4 bits)
- 0: Unknown
- 1: 100 Mbps
- 2: 200 Mbps
- 3: 400 Mbps
- 4: 800 Mbps

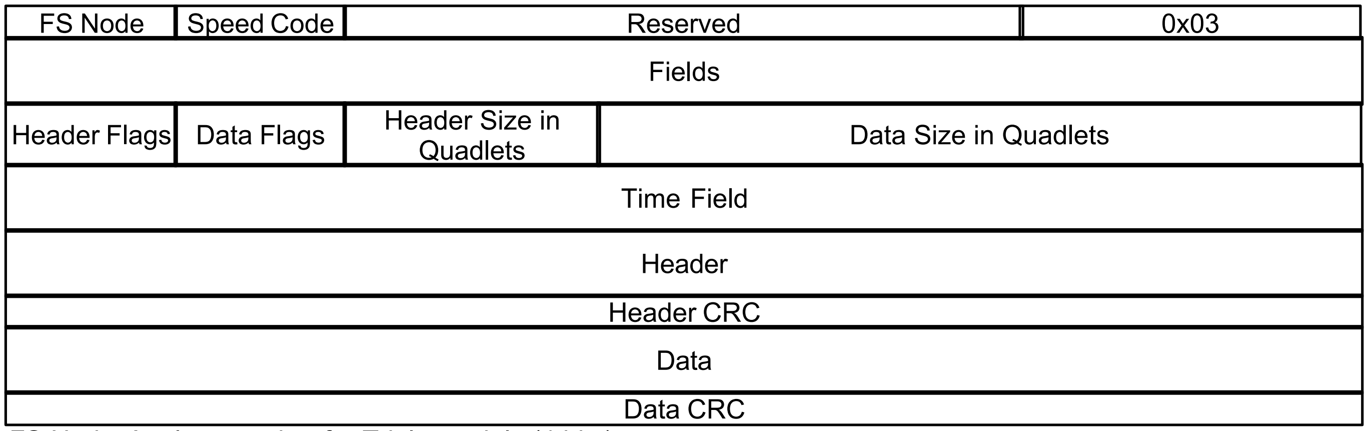
Fields: List of Fields, end is indicated with 0

Size in Quadlets: Complete packet size in number of Quadlets (including CRCs) (32 bits)

Raw Packet: Packet Contents (Quadlet aligned)

Item Contents - Type 3 - Stream Packet

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0



FS Node: Analyzer node - for Triple models (4 bits)
- 0: Node A
- 1: Node B
- 2: Node C

Speed Code: Transmission speed (4 bits)
- 0: Unknown
- 1: 100 Mbps
- 2: 200 Mbps
- 3: 400 Mbps
- 4: 800 Mbps

Fields: List of Fields, end is indicated with 0
Header Flags: 0x1 - CRC Error (4 bits)
Data Flags: 0x1 - CRC Error (4 bits)
Header Size in Quadlets: Number of Header Quadlets (6 bits)
Data Size in Quadlets: Number of Data Quadlets (including CRCs) (18 bits)
Time Field: 0, 32 or 64 bits depending on Time Format
Header: 1394 Header Contents (Quadlet aligned)
Header CRC: 1394 Header CRC (Quadlet aligned)
Data: 1394 Data Contents
Data CRC: 1394 Data CRC

23.5. Signal Definitions file

The signal definitions file format is a comma separated file (CSV), used by the Signal Monitor.

These columns have been defined:

Number	What	Type	Description	Example
1	Name	ascii text	Name of signal	engine1_heartbeat
2	Channel	unsigned integer	Channel number of this signal	10
3	MessageID	unsigned integer	The ID of the message to monitor	0
4	QuadletOffset	unsigned integer	Offset in the packet of the quadlet that contains the signal.	5
5	FirstBits	unsigned integer	Offset of the signal in bits inside a quadlet	16
6	NumBits	unsigned integer	Number of bits that a signal occupies	16
7	SignExtend	boolean	Whether or not to sign extend a value	1 or 0
8	RangeLow	double	Lower boundary, used by Control Panel	-1000
9	RangeHi	double	Upper boundary, used by Control Panel	123.45
10	Resolution	double	The raw value in the signal is multiplied by this constant to create the interpreted	0.000230708

			value.	
11	Units	ascii text	Used to decorate the Control Panel	psi, V/m
12	DataType	ascii text	The data type of the signal. See below for the special data type 'enum'.	`uint32, int32, float32, float64, enum`

Note : if a signal is of data type 'enum', additional columns with label, value pairs must follow. The number of pairs is user-defined. In the example below, sensors_heartbeat is of type 'enum' with defined values on (1) and off(0).

Example
A valid example of a signal definition file would be:

```
engine1_heartbeat,10,0,5,0,32,0,,,,,int32,,,,
engine1_pressure,10,0,6,0,32,0,0,200,0.000230708,,int32,,,,
engine1_temperature,10,0,7,16,16,1,-50,150,0.5,,int32,,,,
sensors_heartbeat,12,0,5,0,32,0,,,,,enum,on,1,off,0
sensors_sensor1,12,0,7,0,32,0,-1000,6000,1,,int32,,,,
```

23.6. Mil1394 Settings XML file

The application can load Mil1394 settings from the `examples/Mil1394Settings.xml` file. This has to be enabled first, as explained in the Mil1394 settings section. The location of the file can be changed in the application's settings dialog as well. This setting is used across different versions, so using a relative path might be a good idea.

A formal definition of this format is given by the XML schema installed as `examples/Mil1394Settings.xsd` in the application directory. Please be sure to validate the XML first, as described below.

The settings file contains a `Mil1394Settings` node, which may contain a list of channel definitions, a collection of ASM message definitions, the definitions of the STOF packet and ASM packet header and trailer, enumeration definitions and time settings. The semantics behind the XML tags are outlined below.

Validation
Be sure to validate the document first. By default, the format is formally defined by linking it to the XML schema file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<Mil1394Settings
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
        xs:noNamespaceSchemaLocation="Mil1394Settings.xsd">

        ...

</Mil1394Settings>
```

An example script to validate the document is provided in `examples/Mil1394SettingsValidation.html`. You can also use a special XML editor that supports validation, such as Microsoft's free XML Notepad. More information about the schema definition format itself can be found at http://www.w3.org/XML/Schema.

Please note that you are not supposed to alter the XSD file.

Time Settings
The `FrameLength`, `STOFAccuracy` and `ASMPacketAccuracy` nodes contain the length of the frame in microseconds, and the margins it may have for STOF and ASM packets.

```
<!-- Global Settings -->
<Properties
        Name="SAE Example"

        FrameLength="12500"
        STOFAccuracy="250"
        ASMPacketAccuracy="25"
/>
```

Channel Definitions

The `ChannelList` node contains a list of `Channel` nodes, each of which define a channel's properties. The `ID`, `Heartbeat` and `DeviceName` can be defined, as well as the `Offset` and `Length` properties of the `Transmit`, `Receive` and `Datapump` frames (within the respective childnodes). These nodes can be omitted or their attributes set to zero to invalidate those kinds of messages for the respective channel.

```
<!-- Pre-Assigned Channels -->
<ChannelList>
        <Channel ID="4" Heartbeat="80" DeviceName="Remote I/O 1">
                <Transmit Offset="2000" Length="100"/>
                <Receive  Offset="2200" Length="100"/>
                <DataPump Offset="2400" Length="100"/>
        </Channel>
        <Channel ID="5" Heartbeat="80" DeviceName="Remote I/O 2">
                <Transmit Offset="2000" Length="100"/>
                <Receive  Offset="2200" Length="100"/>
                <DataPump Offset="2400" Length="100"/>
        </Channel>
        <Channel ID="22" Heartbeat="80" DeviceName="Display Computer">
                <Transmit Offset="8000" Length="100"/>
                <Receive  Offset="8500" Length="100"/>
                <DataPump Offset="0"    Length="0"/>
        </Channel>
</ChannelList>
```

Enumerations

String representations for certain values may be defined in the `Enums` node, which consists of a list of `Enum` nodes with Item and `Range` childnodes. Both childnodes contain a `Label` attribute with which the respective string may be defined. An Item is used for a single value (defined in the `Value` attribute), whereas a `Range` represents a range of value from the `Start` to the `End` attribute's value.

```
<!-- String Representations -->
<Enums>
        <Enum Name="switch">
                <Item Value="0" Label="off"/>
                <Item Value="1" Label="on"/>
                <Range Start="2" End="255" Label="unspecified"/>
        </Enum>
</Enums>
```

Structures

Fields or signals that are used multiple times can be defined once in the Structures section and then referenced later on (using the `Struct` tag and the corresponding `Name` attribute). This makes it also easier to modify sections or just single fields that are used multiple times. Structures may be defined in the `Structs` node, which consists of a list of `Struct` nodes with `Signal` or `Field`-based childnodes, as explained in the Packet Formats section.

```
<!-- String Representations -->
<Structs>
        <Struct Name="switch">
                <Field Name="Reserved" FirstBit="0" NumBits="2" DataType="hex"/>
                <Speed                 FirstBit="2" NumBits="3" DataType="udec" EnumName="SPEED"/>
                <BetaMode              FirstBit="5" NumBits="1" DataType="udec" EnumName="bool"/>
                <ReceiveOK             FirstBit="6" NumBits="1" DataType="udec" EnumName="bool"/>
                <Connected             FirstBit="7" NumBits="1" DataType="udec" EnumName="bool"/>
        </Struct>
</Structs>
```

Packet Formats

A packet format node is made up of `Field` nodes, which in turn contain the following properties:

`QuadOffset` *(required)*
The number of quadlets from the start of the block containing this field, at which the bits of this field start. It must be a positive integer value. It may also be omitted or set to -1 to indicate it should be the same or the directly following quadlet from the previous field (this depends on the context).

`FirstBit` *(default 0)*
The number of bits from the start of the selected quadlet at which the bits of this field start. Bit 0 is the most significant bit, usually transmitted first, and 31 is the least significant bit (higher numbers are not allowed). When omitted or set it to -1 it will be placed after the previous field within the same quadlet, or at the start. If QuadOffset is also omitted or set to -1, the field will be placed directly after the last

specified field. For the sake of clarity it is recommended to explicitly define the offsets.

Please note that the fields still have to be in bitwise order.

`NumBits` *(default 32)*
The number of bits that mark the value of the field. This may span accross quadlet alignments.

`Name` *(optional)*
The name of the field.

`ShortName` *(optional)*
The name to display when only a small amount of space is reserved for it on the screen.

`DataType` *(default "hex")*
This can be one of the types `udec`, `dec`, `hex`, or `float`, to indicate whether the value is signed, real (32 or 64 bit IEEE floating point) and shown in decimal or hexadecimal base.

`EnumName` *(optional)*
The name of the enumeration defined in the `Enum` section, which should be used in case `Type` is "enum".

`Factor` *(optional)*
The factor with which the value derived from the bits should be multiplied. This value can be a real number, such as "2.5", but cannot be used for fields that are actually floating point.

`Offset` *(optional)*
The offset that should be added to the value derived from the bits. This value can also be a real number, but cannot be used for fields that are actually floating point.

`Units` *(optional)*
A string containing the units that should be shown after the value. This attribute cannot be used for floating point values.

`MinValue/MaxValue` *(optional)*
The minimal and maximal value expected.

When fields do not directly follow each other up, the empty space will be filled with "Unspecified" fields.

A field can be split up in multiple fields simply by inserting the inner fields as childnodes. This will be rendered as a collapsed tree when displayed in the packet format table.

A field value may also be split in parts of decimal digits instead. It will then for instance be shown like: "00/123/0000", when automatic formatting is enabled. This can be done using a `DecimalParts` childnode, which in turn may contain `Name` and `NumDigits` attributes. The latter signifies the number of leftmost digits to use for the value of the subfield, following the digits from the previous part.

An ASM message packet format is defined as follows. Please consult the Mil1394 specification for their exact meaning.

Note that the specified fields have their own tag, so they can be better validated.

Please note that the fields have to be defined in the order of the quad/bit offset, starting with the lower bits, even if `QuadOffset` and `FirstBit` are explicitly defined.

```
<!-- Packet Formats -->
<ASMPacketFormat>
        <ASMHeader>
                <MessageID                QuadOffset="0" FirstBit="0" NumBits="32"
                                          DataType="udec">
                        <DecimalParts>
                                <Branch      NumDigits="2"/>
                                <Channel     NumDigits="2"/>
                                <Message     NumDigits="6"/>
                        </DecimalParts>
                </MessageID>
                <Reserved                 QuadOffset="1"  FirstBit="0"  NumBits="32"
                                          DataType="hex"/>
                <NodeID                   QuadOffset="2"  FirstBit="0"  NumBits="32"
                                          DataType="udec"/>
                <MessagePayloadDataLength QuadOffset="3"  FirstBit="0"  NumBits="24"
                                          DataType="udec"/>
                <Priority                 QuadOffset="3"  FirstBit="24" NumBits="8"
                                          DataType="udec"/>
        </ASMHeader>

        <PayloadData>
                <HealthStatus             QuadOffset="0"  FirstBit="0"  NumBits="32"
                                          DataType="hex">
                        <Port Name="Port 2">
                                <Struct Name="Port Struct"/>
                        </Port>

                        <Port Name="Port 1">
                                <Struct Name="Port Struct"/>
                        </Port>

                        <Port Name="Port 0">
                                <Struct Name="Port Struct"/>
                        </Port>

                        <NodeError                        FirstBit="29" NumBits="1"
                                          DataType="udec" EnumName="bool"/>
                        <SubsystemError                   FirstBit="30" NumBits="1"
                                          DataType="udec" EnumName="bool"/>
                        <PacketError                      FirstBit="31" NumBits="1"
                                          DataType="udec" EnumName="bool"/>
                </HealthStatus>

                <Heartbeat                QuadOffset="1" FirstBit="0"  NumBits="32"
                                          DataType="udec"/>
                <MessageData              QuadOffset="2"/>
        </PayloadData>

        <PacketTrailer>
                <STOFTransmitOffset       QuadOffset="0" FirstBit="0"  NumBits="32"
                                          DataType="udec"/>
                <STOFReceiveOffset        QuadOffset="1" FirstBit="0"  NumBits="32"
                                          DataType="udec"/>
                <STOFDatapumpOffset       QuadOffset="2" FirstBit="0"  NumBits="32"
                                          DataType="udec"/>
                <VerticalParityCheck      QuadOffset="3" FirstBit="0"  NumBits="32"
                                          DataType="hex"/>
        </PacketTrailer>
</ASMPacketFormat>
```

The STOF packet format is similarly defined; see the XML file for its specification.

The `MessageData` tag denotes the location of the signal value payload within the packet; this definition is explained below.

Message/Signal Definitions

The `MessageDefinitions` section lists the `MessageDefinition` nodes, which define how the signals should be extracted from the ASM message packets, per channel and message ID.

The `Signal` nodes use the exact same attributes as the `Field` nodes (except that they cannot contain inner fields or signals), and may be grouped in `SignalGroup` nodes. A group can be grouped recursively, and mixed with `Signal` nodes.

The `PacketSize` attribute can be used to set the size of the payload, expressed in a positive number of bits. When omitted or set to -1, the size is derived from the location and size of the last signal within the respective definition.

```
<!-- Message Definitions -->
<MessageDefinitions>
        <MessageDefinition Channel="4" MessageID="6000004" PacketSize="320">
                <SignalGroup>
                        <Signal Name="engine1_heartbeat" DataType="udec"  QuadOffset="0"/>
                        <Signal Name="engine1_pressure"  DataType="float" QuadOffset="1"
                                Factor="2.5" Offset="-1.5" MinValue="-50" MaxValue="99.5"
                                Units="bar"/>
                </SignalGroup>
                <Signal Name="engine1_temperature"         DataType="dec"   QuadOffset="2"
```

```
                                FirstBit="16" NumBits="16"/>
                </MessageDefinition>

                <MessageDefinition Channel="5" MessageID="0">
                        <SignalGroup>
                                <Signal Name="sensors_heartbeat" DataType="udec"  QuadOffset="0"/>
                                <Signal Name="sensors_sensor1"   DataType="dec"   QuadOffset="2"
                                        EnumName="switch"/>
                        </SignalGroup>
                </MessageDefinition>
        </MessageDefinitions>
```

Chapter 24. Hardware

24.1. Gen4 Analyzer Series

The Gen4 Series of analyzers embodies a complete redesign of the well-known Advanced and Triple series analyzers. This brand-new line offers improved performance due to a more powerful embedded processor, the very latest FPGA technology and modern interfaces like USB3, enhanced Scriptor capabilities and more data capture memory.

The Gen4 Series uses a more powerful architectural design with two 667Mhz on-board ARM processors. It was designed with performance and flexibility in mind. Modularity not only results in several configuration options with respect to 1394 speeds, transformer coupling and host connectivity options but also results in the possibility of combining up-to 3 Triple FireSpy Mainboards (Units) into a full 9-bus IEEE-1394 analyzer. The FS9432bT/FS9832bT is the first Gen4 Series model released and several different configurations followed its initial release.

A standard off-the-shelf Physical Layer chip connects to a highly optimized FPGA which not only acts as a Link layer but also incorporates the data recording functionality, Trigger and Filter engines as well as the Data Monitor. Recorded data is directly captured into dedicated onboard data memory of 4GB per unit.

24.1.1. FireSpy 9432bT, 9832bT

The FireSpy9x32bT bus analyzer is world’s only nine-channel IEEE-1394 bus analyzer. Based on the 4th generation FireSpy analyzer architecture the FireSpy9x32bT is the most advanced 1394 test equipment in the market. The FireSpy9x32bT in fact combines nine FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host.

The FireSpy9x32bT has nine 1394 nodes connected to nine synchronized analysis engines. They are controlled by three dual core ARM processors running at 667MHz. Each node is connected to two 1394 ports. Both ports of each node are connected to a separate LEMO connector which was chosen over standard 1394 connectors for improved ruggedness.

The FireSpy9x32bT is equipped with 15 GB internal memory and extensive hardware filtering and trigger possibilities. The analyzer can be connected to a host computer using the USB3 interface. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way; or you can use the API to program your own control software.

The seamless integration of the SAE AS5643 protocol makes the FireSpy9x32bT the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the AS5643 protocol in all major functional areas of the FireSpy9x32bT and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.1.1.1. Main Feature Summary

GENERAL

- IEEE 1394-2008 Beta compliant
- Supports S100B-S800B transfer rates depending on exact model

- Connects to host using USB3.0 interface
- 15GByte memory for embedded OS and packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
- Software runs on Windows^{TM} Operating Systems

MONITOR

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

RECORDER

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

GENERATOR

- Simultaneous generation of up to 63 iso streams on 9 buses
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous
- packet generation
- Special AS5643 stream generator package (optional)

SCRIPTOR

- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor

- Control Panel
    - Graphical display elements for data value representation
    - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

COMMANDER

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.1.1.2. Specifications

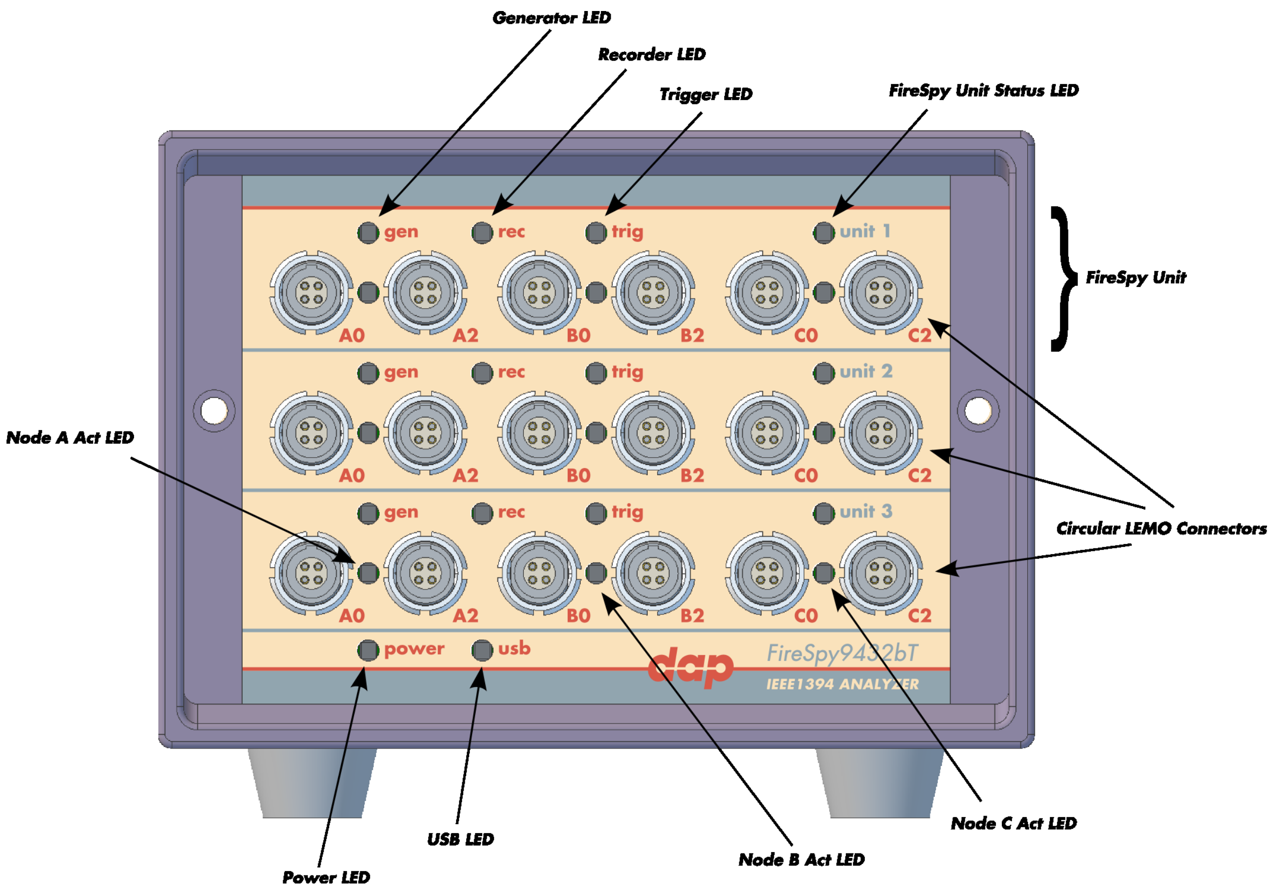
Dimensions:	125 mm x 96 mm x 301 mm
Weight:	2260 g
Operating Range:	0 – 45 C
Power Requirements:	12 V, 40Watt max.
Compliance:	FCC Class A
Connections:	• USB 3.0 connector for host-computer • 18x LEMO connectors (EYG.0B.304) for 1394 Beta connections • Auxiliary connector • Power connector
Indicators:	• Multi-colored LEDs for: Unit status, Recorder, Generator, Trigger, Active (per node) • Red LEDs for: USB, Power
Switches:	Toggle switch for Power On/Off
Package Content:	FireSpy9x32bT Power Adapter (12V, 5A) USB 3.0 Cable
Product warranty:	36 months limited warranty
Part Number:	FS9432bT or FS9432bTAS5643 analyzer with AS5643 SW protocol package FS9832bT or FS9832bTAS5643 analyzer with AS5643 SW protocol package
Optional Configuration:	N/A
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package AMI-C protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.1.1.3. FireSpy Front



Generator LED
This LED will light up green while the FireSpy Generator is active. See Generator.

Recorder LED
This LED will light up red while recording is in progress. After the recording stops it will be orange during the post-processing phase. As soon as data is ready for download to the host, the LED will turn green. See Recorder.

Trigger LED
This LED will light up green while the Trigger/Sequencer is active but not yet triggered. When the Trigger occurs, the LED turns red. See Filter/Trigger.

FireSpy Unit
This FireSpy model consists of three triple FireSpy Analyzer Units to form a total of 9 analyzer nodes. Each Unit has its own section on the front panel with connectors and status LEDs. Units are numbered top-down from 1 to 3. The FireSpy application can be used to control one unit at a time or multiple units simultaneously.

FireSpy Unit Status LED
This led will light up orange when the FireSpy Unit is switched on but not yet configured. The bootloader will configure the device with the firmware version last used. When configuration is successful the LED will turn green. A red LED indicates an error.

Node A,B and C Act LEDs
(Not yet implemented) The function of these LEDs can be configured through the settings dialog of the FireSpy Application. These LEDs can also be controlled by the FireSpy Scriptor.

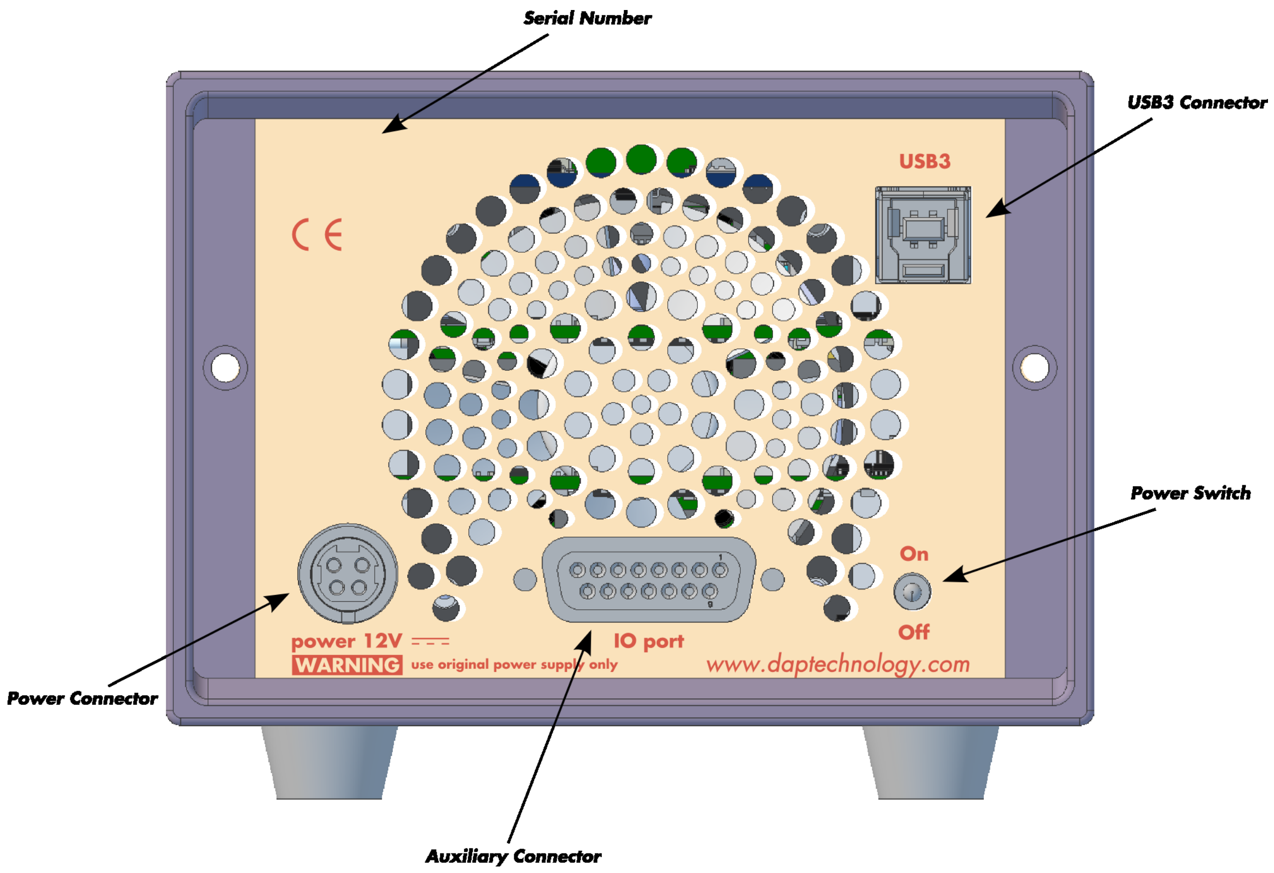
Power LED

The Power LED will light up red when power is supplied to the power connector on the back. Please be aware that this LED also lights up when the Power Switch on the back is in the Off position.

USB LED
The USB LED will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

LEMO Connectors
With these connectors the FireSpy can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. The IEEE1394b ports are (from left to right) A0, A2, B0, B2, C0, C2.
Each connector is transformer coupled.

24.1.1.4. FireSpy Rear



Power Switch
Using this switch the FireSpy can be switched on and off. When switched on, the Unit LEDs on the front panel will light up orange to indicate they are starting up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

Power Connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used. Whenever power is supplied to the analyzer, the Power LED on the front panel will light up red. (Even when the Power Switch is in the Off position).

Auxiliary Connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

USB3 Connector
The FireSpy must be connected to the computer using this connector. A USB3 cable, which is part of the FireSpy-package, is connected between this connector and the USB3 port of the computer.

Serial Number

Each FireSpy has an 11-character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. The software will only work when a valid license certificate is installed for the serial number of the currently connected FireSpy. See License Manager for more information on license certificates.

24.1.2. FireSpy 6432bT, 6832bT

The FireSpy6x32bT bus analyzer is world's only six-node IEEE-1394 bus analyzer. Based on the 4th generation FireSpy analyzer architecture the FireSpy6x32bT is the most advanced 1394 test equipment in the market. The FireSpy6x32bT in fact combines six FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host.

The FireSpy6x32bT has six 1394 nodes connected to six synchronized analysis engines. They are controlled by two dual core ARM processors running at 667MHz. Each node is connected to two 1394 ports. Both ports of each node are connected to a separate LEMO connector which was chosen over standard 1394 connectors for improved ruggedness.

The FireSpy6x32bT is equipped with 10 GB internal memory and extensive hardware filtering and trigger possibilities. The analyzer can be connected to a host computer using the USB3 interface. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way; or you can use the API to program your own control software.

The seamless integration of the SAE AS5643 protocol makes the FireSpy6x32bT the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the AS5643 protocol in all major functional areas of the FireSpy6x32bT and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.1.2.1. Main Feature Summary

GENERAL

- IEEE 1394-2008 Beta compliant
- Supports S100B-S800B transfer rates depending on exact model
- Connects to host using USB3.0 interface
- 10GByte memory for embedded OS and packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
- Software runs on Windows^{TM} Operating Systems

MONITOR

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition

- Counts number of bus resets

RECORDER
- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

GENERATOR
- Simultaneous generation of up to 63 iso streams on 6 buses
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous
- packet generation
- Special AS5643 stream generator package (optional)

SCRIPTOR
- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

COMMANDER
- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.1.2.2. Specifications

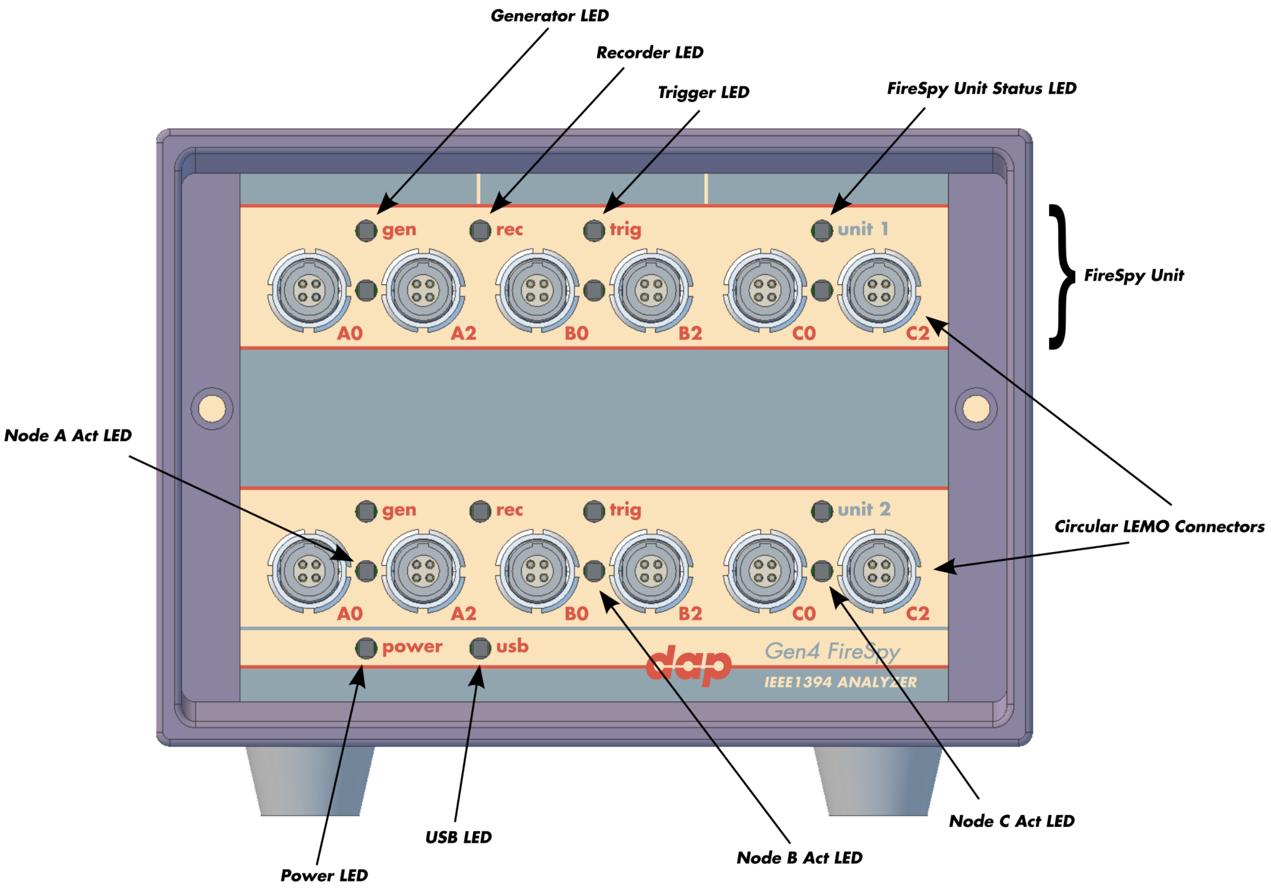
Dimensions:	125 mm x 96 mm x 301 mm
Weight:	2090 g
Operating Range:	0 – 45 C
Power Requirements:	12 V, 40Watt max.
Compliance:	FCC Class A
Connections:	• USB 3.0 connector for host-computer • 12x LEMO connectors (EYG.0B.304) for 1394 Beta connections • Auxiliary connector • Power connector
Indicators:	• Multi-colored LEDs for: Unit status, Recorder, Generator, Trigger, Active (per node) • Red LEDs for: USB, Power
Switches:	Toggle switch for Power On/Off
Package Content:	FireSpy6x32bT Power Adapter (12V, 5A) USB 3.0 Cable
Product warranty:	36 months limited warranty
Part Number:	FS6432bT or FS6432bTAS5643 analyzer with AS5643 SW protocol package FS6832bT or FS6832bTAS5643 analyzer with AS5643 SW protocol package
Optional Configuration:	N/A
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package AMI-C protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.1.2.3. FireSpy Front



Generator LED
This LED will light up green while the FireSpy Generator is active. See Generator.

Recorder LED
This LED will light up red while recording is in progress. After the recording stops it will be orange during the post-processing phase. As soon as data is ready for download to the host, the LED will turn green. See Recorder.

Trigger LED
This LED will light up green while the Trigger/Sequencer is active but not yet triggered. When the Trigger occurs, the LED turns red. See Filter/Trigger.

FireSpy Unit
This FireSpy model consists of two triple FireSpy Analyzer Units to form a total of 6 analyzer nodes. Each Unit has its own section on the front panel with connectors and status LEDs. Units are numbered top-down from 1 to 2. The FireSpy application can be used to control one unit at a time or multiple units simultaneously.

FireSpy Unit Status LED
This led will light up orange when the FireSpy Unit is switched on but not yet configured. The bootloader will configure the device with the firmware version last used. When configuration is successful the LED will turn green. A red LED indicates an error.

Node A,B and C Act LEDs
(Not yet implemented) The function of these LEDs can be configured through the settings dialog of the FireSpy Application. These LEDs can also be controlled by the FireSpy Scriptor.

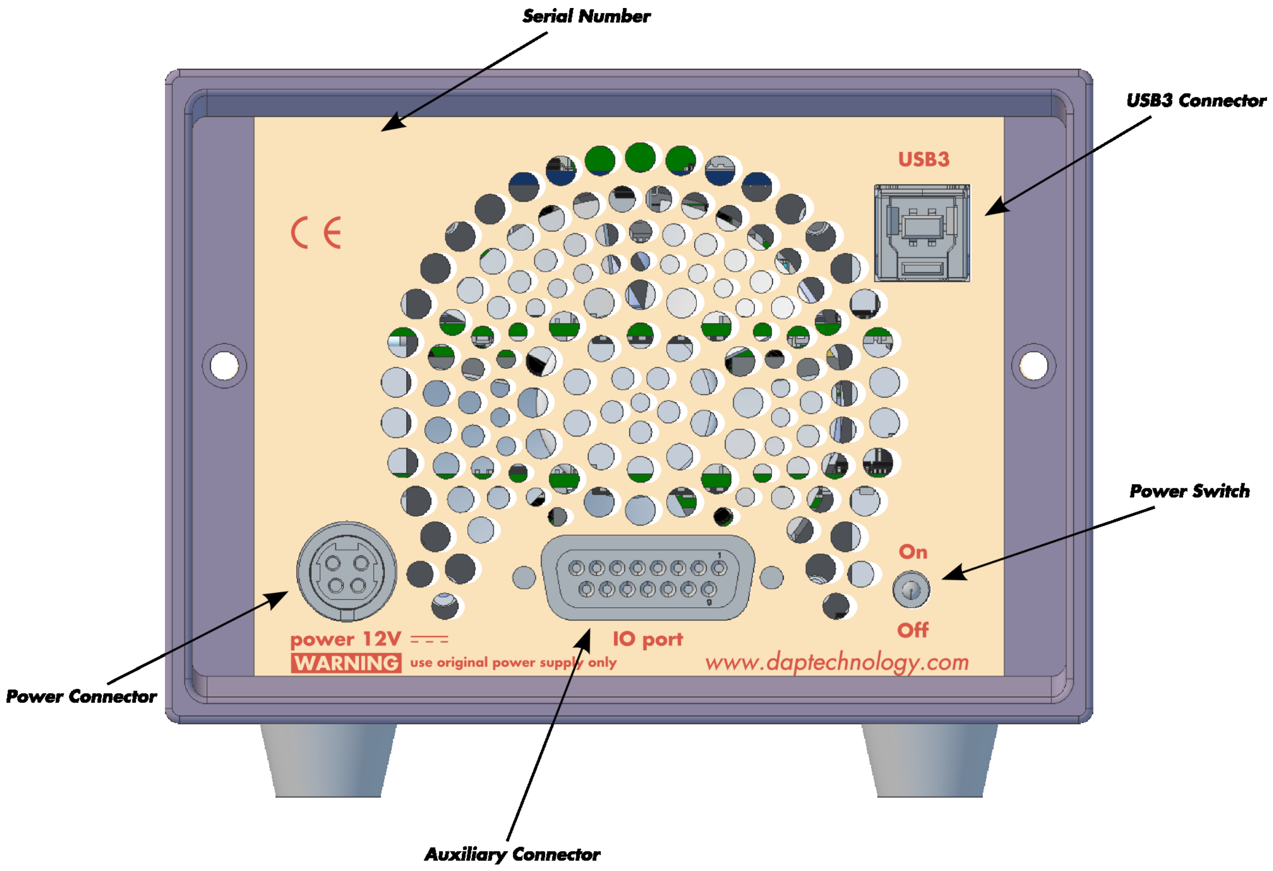
Power LED

The Power LED will light up red when power is supplied to the power connector on the back. Please be aware that this LED also lights up when the Power Switch on the back is in the Off position.

USB LED
The USB LED will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

LEMO Connectors
With these connectors the FireSpy can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. The IEEE1394b ports are (from left to right) A0, A2, B0, B2, C0, C2.
Each connector is transformer coupled.

24.1.2.4. FireSpy Rear



Power Switch
Using this switch the FireSpy can be switched on and off. When switched on, the Unit LEDs on the front panel will light up orange to indicate they are starting up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

Power Connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used. Whenever power is supplied to the analyzer, the Power LED on the front panel will light up red. (Even when the Power Switch is in the Off position).

Auxiliary Connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

USB3 Connector
The FireSpy must be connected to the computer using this connector. A USB3 cable, which is part of the FireSpy-package, is connected between this connector and the USB3 port of the computer.

Serial Number

Each FireSpy has an 11-character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. The software will only work when a valid license certificate is installed for the serial number of the currently connected FireSpy. See License Manager for more information on license certificates.

24.1.3. FireSpy 3430b, 3430bT, 3830, 3830bT

The FireSpy3x30xx bus analyzer is based on the 4th generation FireSpy analyzer architecture and is the most advanced 1394 test equipment in the market. The FireSpy3x30xx in fact combines three FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host.

The FireSpy3x30xx has three 1394 nodes connected to three synchronized analysis engines. They are controlled by a dual core ARM processor running at 667MHz. Each node is connected to three 1394 ports. All ports of each node are connected to a separate IEEE-1394 connector with screw mount holes which was chosen over standard 1394 connectors for improved ruggedness.

The FireSpy3x30xx is equipped with 5 GB internal memory and extensive hardware filtering and trigger possibilities. The analyzer can be connected to a host computer using the USB3 interface. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way; or you can use the API to program your own control software.

The seamless integration of the SAE AS5643 protocol makes the FireSpy3x30xx the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the AS5643 protocol in all major functional areas of the FireSpy3x30xx and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.1.3.1. Main Feature Summary

GENERAL

- IEEE 1394-2008 Beta compliant
- Supports S100B-S800B transfer rates depending on exact model
- Connects to host using USB3.0 interface
- 5GByte memory for embedded OS and packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
- Software runs on Windows^{TM} Operating Systems

MONITOR

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition

- Counts number of bus resets

RECORDER
- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

GENERATOR
- Simultaneous generation of up to 63 iso streams on 3 buses
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous
- packet generation
- Special AS5643 stream generator package (optional)

SCRIPTOR
- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

COMMANDER
- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.1.3.2. Specifications

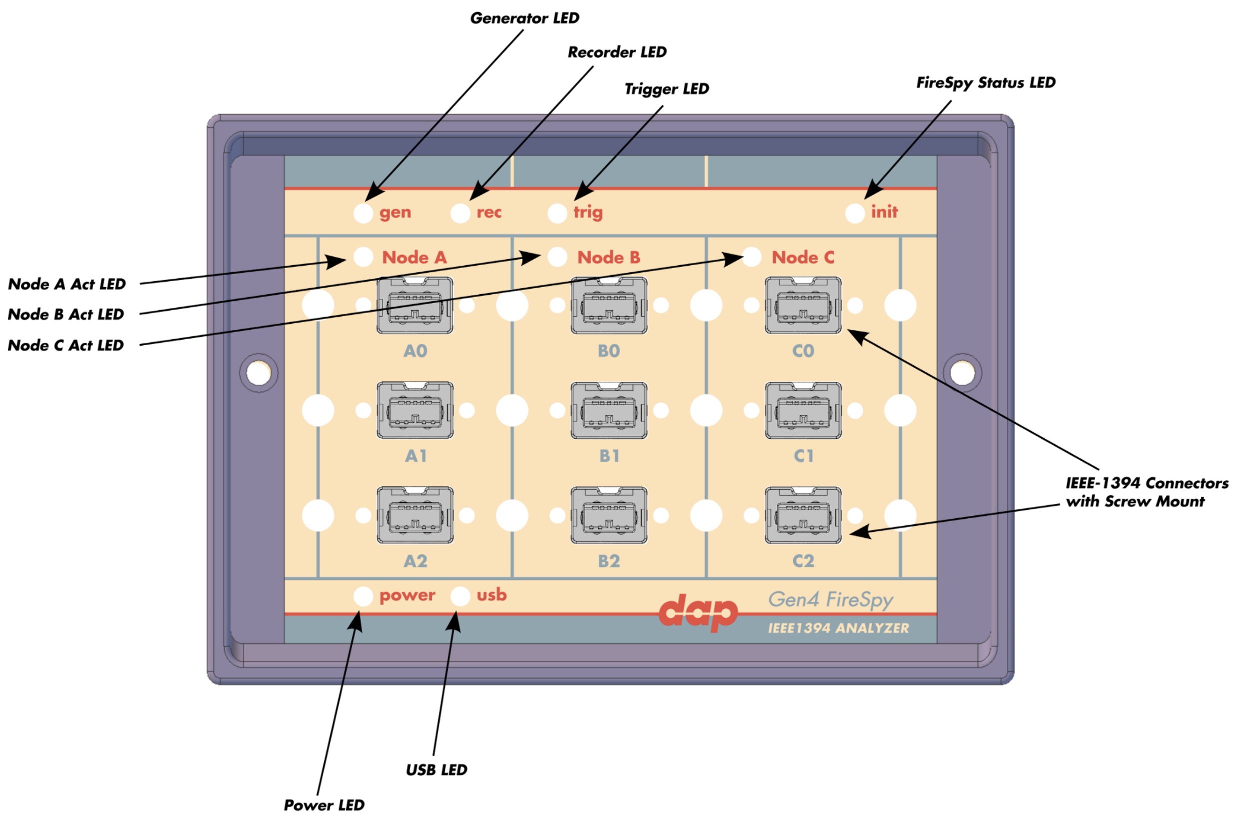
Dimensions:	125 mm x 96 mm x 301 mm
Weight:	1790 g
Operating Range:	0 – 45 C
Power Requirements:	12 V, 40Watt max.
Compliance:	FCC Class A
Connections:	• USB 3.0 connector for host-computer • 9x IEEE-1394 connectors with screw holes • Auxiliary connector • Power connector
Indicators:	• Multi-colored LEDs for: Unit status, Recorder, Generator, Trigger, Active (per node) • Red LEDs for: USB, Power
Switches:	Toggle switch for Power On/Off
Package Content:	FireSpy3x30xx Power Adapter (12V, 5A) USB 3.0 Cable
Product warranty:	36 months limited warranty
Part Number:	FireSpy3430b or FireSpy3430bAS5643 analyzer with AS5643 SW protocol package FireSpy3430bT or FireSpy3430bTAS5643 analyzer with AS5643 SW protocol package FireSpy3830 or FireSpy3830AS5643 analyzer with AS5643 SW protocol package FireSpy3830bT or FireSpy3830bTAS5643 analyzer with AS5643 SW protocol package
Optional Configuration:	N/A
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package AMI-C protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.1.3.3. FireSpy Front



Generator LED
This LED will light up green while the FireSpy Generator is active. See Generator.

Recorder LED
This LED will light up red while recording is in progress. After the recording stops it will be orange during the post-processing phase. As soon as data is ready for download to the host, the LED will turn green. See Recorder.

Trigger LED
This LED will light up green while the Trigger/Sequencer is active but not yet triggered. When the Trigger occurs, the LED turns red. See Filter/Trigger.

FireSpy Status LED
This led will light up orange when the FireSpy Unit is switched on but not yet configured. The bootloader will configure the device with the firmware version last used. When configuration is successful the LED will turn green. A red LED indicates an error.

Node A,B and C Act LEDs
(Not yet implemented) The function of these LEDs can be configured through the settings dialog of the FireSpy Application. These LEDs can also be controlled by the FireSpy Scriptor.

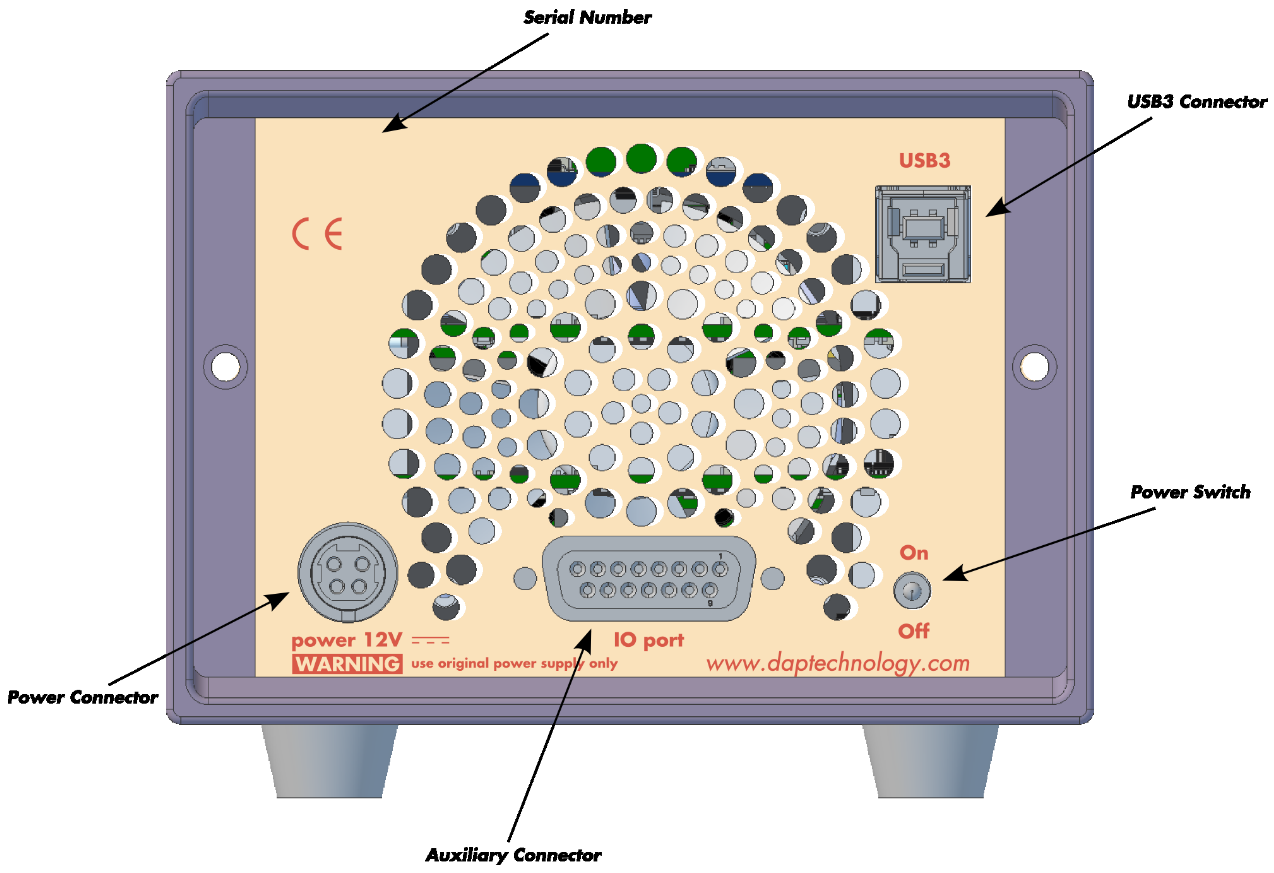
Power LED
The Power LED will light up red when power is supplied to the power connector on the back. Please be aware that this LED also lights up when the Power Switch on the back is in the Off position.

USB LED
The USB LED will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

IEEE-1394 Connectors

With these connectors the FireSpy can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. The IEEE1394b ports are (top-down, left-right) A0, A1, A2, B0, B1, B2, C0, C1, C2.
Depending on the exact model, each connector is transformer coupled.

24.1.3.4. FireSpy Rear



Power Switch
Using this switch the FireSpy can be switched on and off. When switched on, the Unit LEDs on the front panel will light up orange to indicate they are starting up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

Power Connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used. Whenever power is supplied to the analyzer, the Power LED on the front panel will light up red. (Even when the Power Switch is in the Off position).

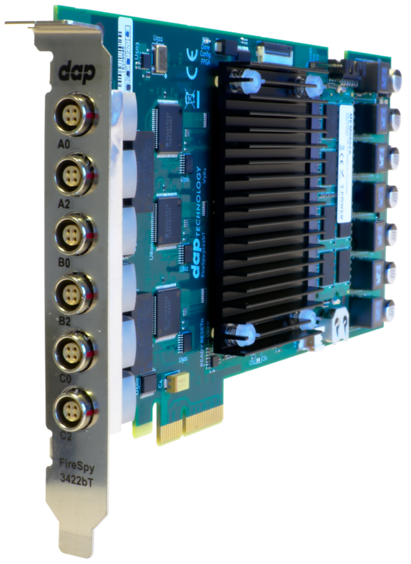
Auxiliary Connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

USB3 Connector
The FireSpy must be connected to the computer using this connector. A USB3 cable, which is part of the FireSpy-package, is connected between this connector and the USB3 port of the computer.

Serial Number
Each FireSpy has an 11-character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. The software will only work when a valid license certificate is installed for the serial number of the currently connected FireSpy. See License Manager for more information on license certificates.

24.1.4. FireSpy 3422bT, 3822bT

The FireSpy3x22bT bus analyzer series is world's only PCIe-based multi-channel FireWire bus analyzer product line. The different models differ only in their support for different speeds, architecturally they based on the latest Gen4 concept. Each analyzer card in fact combines three FireSpy analyzers in one single instrument.



The PCIe's format takes flexibility to a higher level and addresses the industries growing support for this data interface. This solution enables users to install the FireSpy hardware in any modern PC that has at least one 4-lane PCIe slot available. This allows for installation of the FireSpy in developers existing PCs or installing the analyzer in a rugged field service laptop or lunchbox computer.

The FireSpy3x22bT have three 1394 nodes connected to three synchronized analysis engines. They are controlled by a dual core ARM processor running at 667MHz. Each node is connected to two 1394 ports. Two ports of each 1394 node are connected to a LEMO connector.

The FireSpy3x22bT are equipped with 5 GB internal memory and extensive hardware filtering and trigger possibilities. The analyzer can be connected into a host computer using its PCIe interface. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way; or you can use the API to program your own control software.

24.1.4.1. Main Feature Summary

GENERAL

- IEEE 1394-2008 Beta compliant
- Supports S100-S800 transfer rates depending on exact model
- Connects to host using PCIe interface
- 5GByte memory for embedded OS and packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
  - Time synchronization
- GUI and API for Windows^{TM} Operating Systems

MONITOR

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

- Measurement of bus power voltages (bilingual ports)

RECORDER
- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

GENERATOR
- Simultaneous generation of up to 63 iso streams on 9 buses
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous
- packet generation
- Special AS5643 stream generator package (optional)

SCRIPTOR
- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

COMMANDER
- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.1.4.2. Specifications

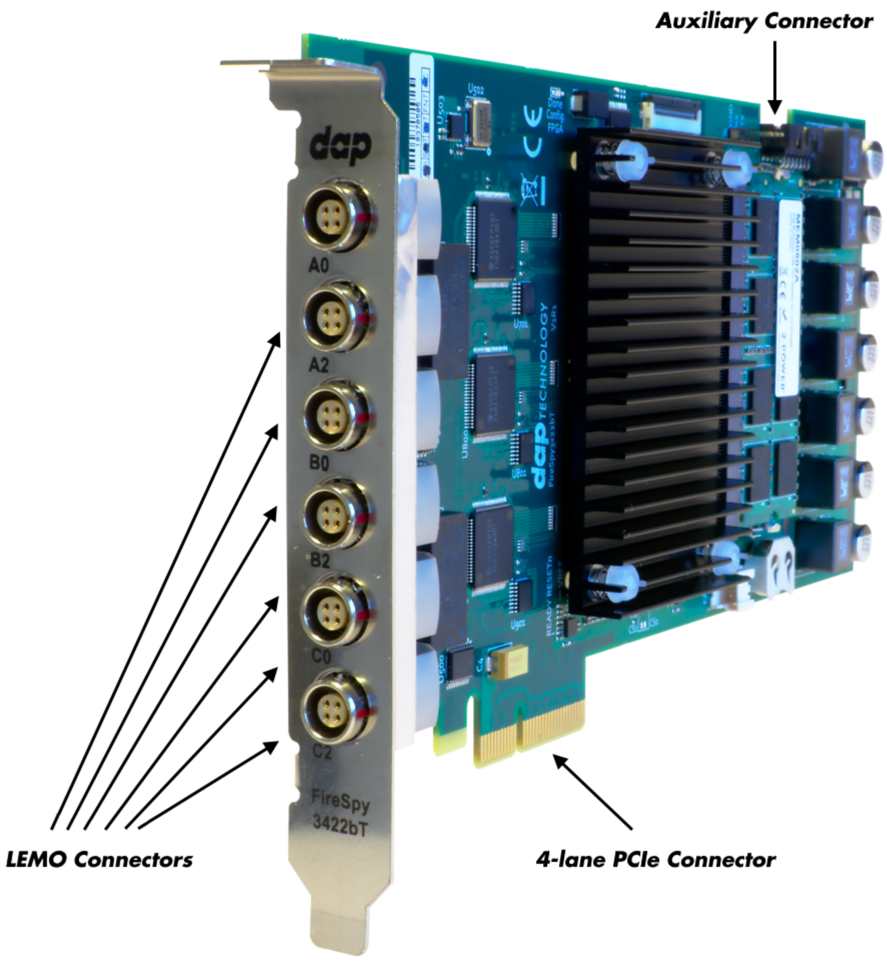
Dimensions:	4-lane PCIe: card length approx. 205 mm
Weight:	TBD
Operating Range:	0 – 70 C
Power Requirements:	25Watt max.
Compliance:	FCC Class A
Connections:	• PCIe connector • 6x LEMO connectors • Auxiliary connector
Indicators:	-
Switches:	-
Package Content:	FireSpy3x22bT 3x LEMO to Bilingual cables
Product warranty:	36 months limited warranty
Part Number:	FS3822bT or FS3822bTAS5643 analyzer with AS5643 SW protocol package FS3422bT or FS3422bTAS5643 analyzer with AS5643 SW protocol package
Optional Configuration:	N/A
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package AMI-C protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.1.4.3. FireSpy PCI Board

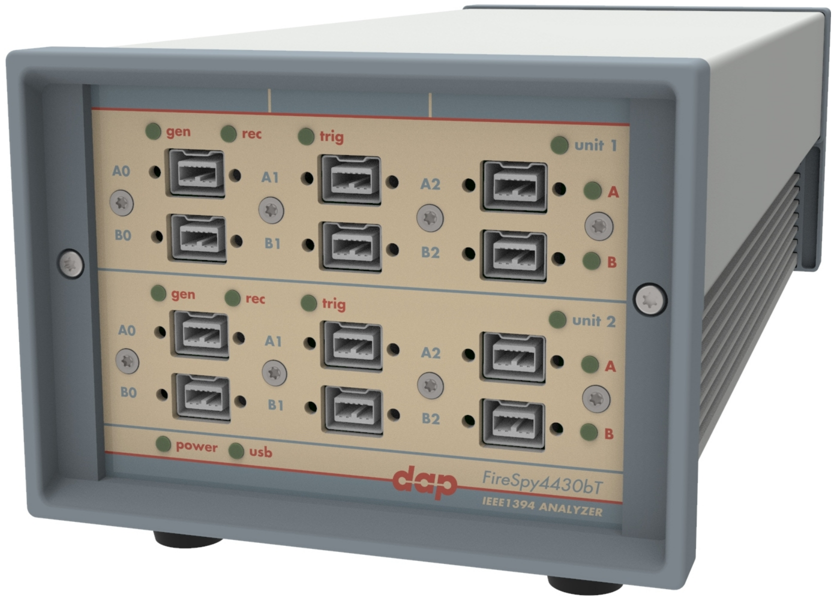


LEMO Connectors
With these connectors the FireSpy can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. The IEEE1394b ports are (from left to right) A0, A2, B0, B2, C0, C2.
Each connector is transformer coupled.

Auxiliary Connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

24.1.5. FireSpy 4430b, 4430bT, 4830, 4830bT

The FireSpy4x30xx bus analyzer is world’s only four-node IEEE-1394 bus analyzer. Based on the 4th generation FireSpy analyzer architecture the FireSpy4x30xx is the most advanced 1394 test equipment in the market. The FireSpy4x30xx in fact combines four FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host.



The FireSpy4x30xx has four 1394 nodes connected to four synchronized analysis engines. They are controlled by two dual core ARM processors running at 667MHz. Each node is connected to three 1394 ports. All ports of each node are connected to a separate IEEE-1394 connector with screw mount holes which was chosen over standard 1394 connectors for improved ruggedness.

The FireSpy4x30xx is equipped with 10 GB internal memory and extensive hardware filtering and trigger possibilities. The analyzer can be connected to a host computer using the USB3 interface. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way; or you can use the API to program your own control software.

The seamless integration of the SAE AS5643 protocol makes the FireSpy4x30xx the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the AS5643 protocol in all major functional areas of the FireSpy4x30xx and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.1.5.1. Main Feature Summary

GENERAL

- IEEE 1394-2008 Beta compliant
- Supports S100B-S800B transfer rates depending on exact model
- Connects to host using USB3.0 interface
- 10GByte memory for embedded OS and packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
- Software runs on Windows^{TM} Operating Systems

MONITOR

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

RECORDER

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets

- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

GENERATOR

- Simultaneous generation of up to 63 iso streams on 4 buses
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous
- packet generation
- Special AS5643 stream generator package (optional)

SCRIPTOR

- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

COMMANDER

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.1.5.2. Specifications

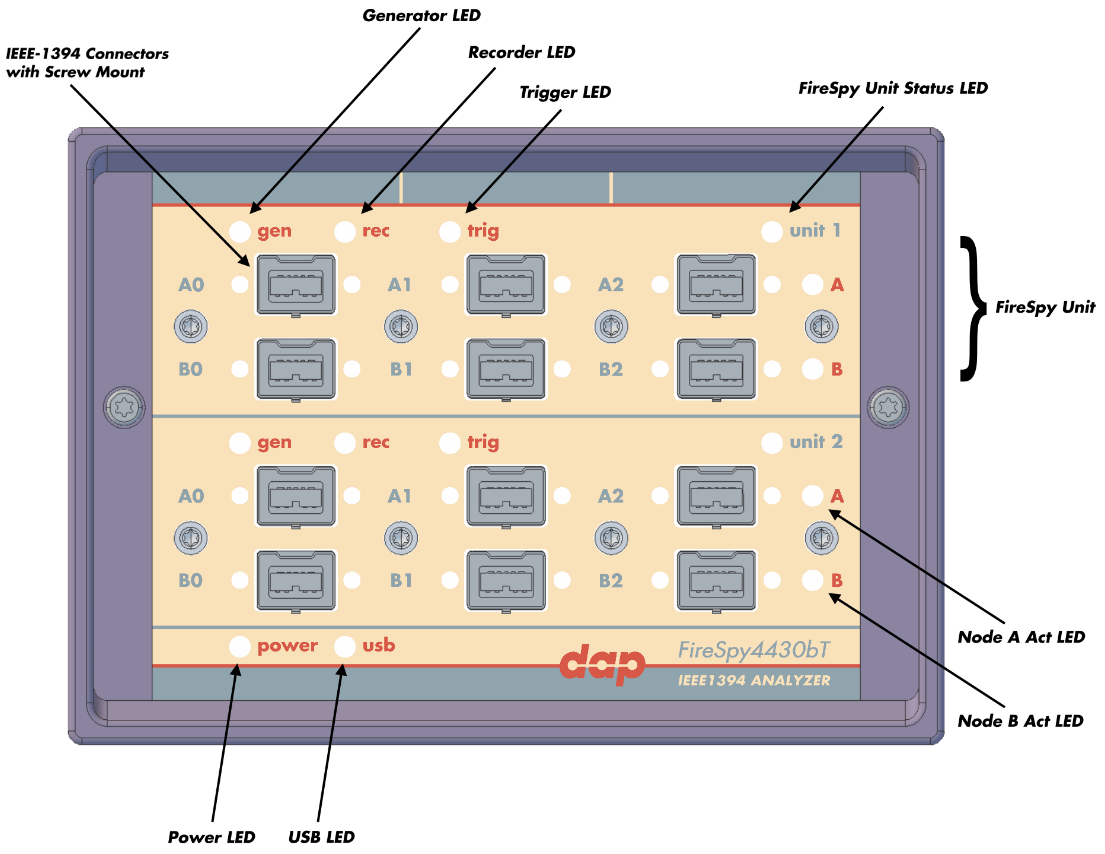
Dimensions:	125 mm x 96 mm x 301 mm
Weight:	2070 g
Operating Range:	0 – 45 C
Power Requirements:	12 V, 40Watt max.
Compliance:	FCC Class A
Connections:	• USB 3.0 connector for host-computer • 12x IEEE-1394 connectors with screw holes • Auxiliary connector • Power connector
Indicators:	• Multi-colored LEDs for: Unit status, Recorder, Generator, Trigger, Active (per node) • Green LEDs for: USB, Power
Switches:	Toggle switch for Power On/Off
Package Content:	FireSpy4x30xx Power Adapter (12V, 5A) USB 3.0 Cable
Product warranty:	36 months limited warranty
Part Number:	FS4430bT or FS4430bTAS5643 analyzer with AS5643 SW protocol package
Optional Configuration:	N/A
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package AMI-C protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.1.5.3. FireSpy Front



Generator LED
This LED will light up green while the FireSpy Generator is active. See Generator.

Recorder LED
This LED will light up red while recording is in progress. After the recording stops it will be orange during the post-processing phase. As soon as data is ready for download to the host, the LED will turn green. See Recorder.

Trigger LED
This LED will light up green while the Trigger/Sequencer is active but not yet triggered. When the Trigger occurs, the LED turns red. See Filter/Trigger.

FireSpy Unit
This FireSpy model consists of two double FireSpy Analyzer Units to form a total of 4 analyzer nodes. Each Unit has its own section on the front panel with connectors and status LEDs. Units are numbered top-down from 1 to 2. The FireSpy application can be used to control one unit at a time or multiple units simultaneously.

FireSpy Unit Status LED
This led will light up orange when the FireSpy Unit is switched on but not yet configured. The bootloader will configure the device with the firmware version last used. When configuration is successful the LED will turn green. A red LED indicates an error.

Node A and B Act LEDs
(Not yet implemented) The function of these LEDs can be configured through the settings dialog of the FireSpy Application. These LEDs can also be controlled by the FireSpy Scriptor.

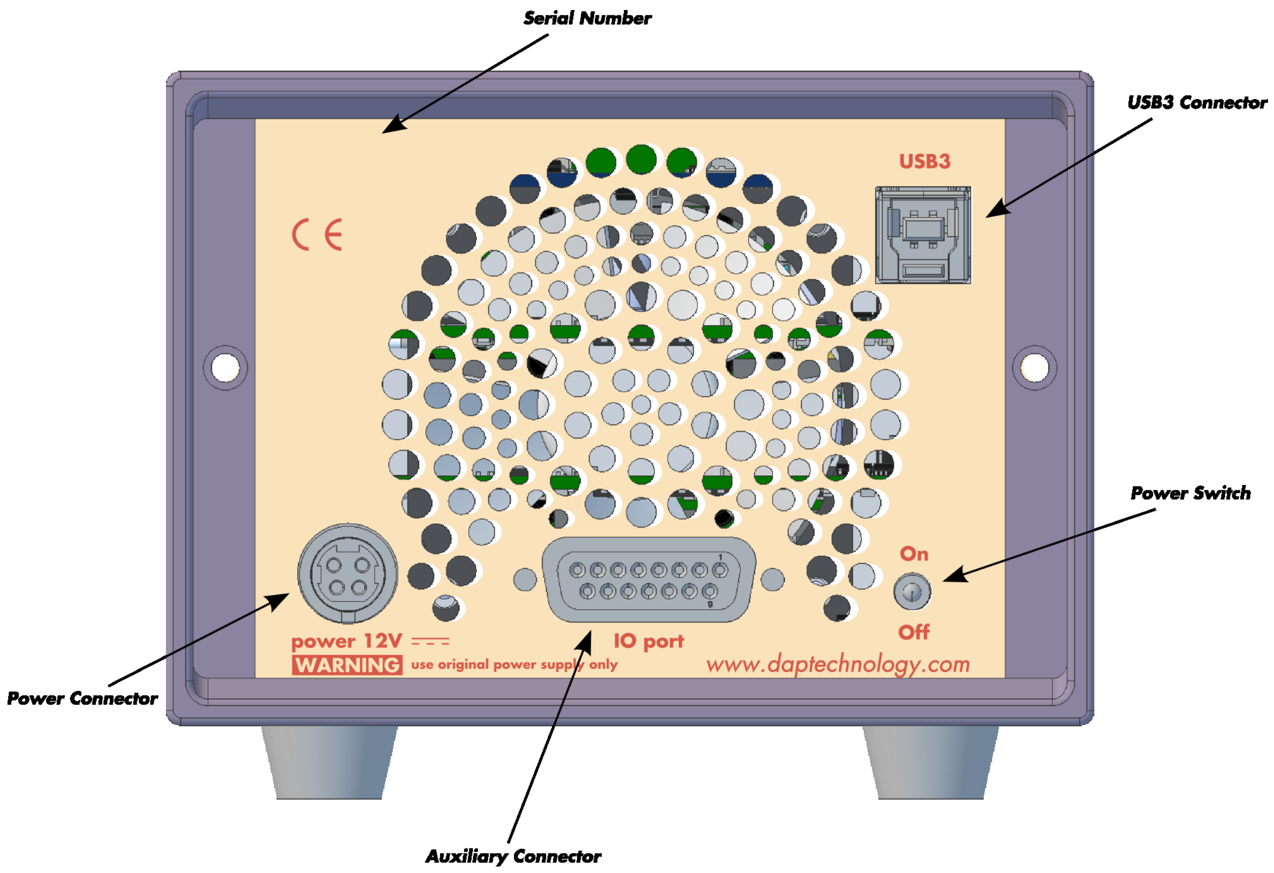
Power LED

The Power LED will light up red when power is supplied to the power connector on the back. Please be aware that this LED also lights up when the Power Switch on the back is in the Off position.

USB LED
The USB LED will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

IEEE-1394 Connectors
With these connectors the FireSpy can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. The IEEE1394b ports are (top-down, left-right) A0, A1, A2, B0, B1, B2.
Depending on the exact model, each connector is transformer coupled.

24.1.5.4. FireSpy Rear



Power Switch
Using this switch the FireSpy can be switched on and off. When switched on, the Unit LEDs on the front panel will light up orange to indicate they are starting up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

Power Connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used. Whenever power is supplied to the analyzer, the Power LED on the front panel will light up red. (Even when the Power Switch is in the Off position).

Auxiliary Connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

USB3 Connector
The FireSpy must be connected to the computer using this connector. A USB3 cable, which is part of the FireSpy-package, is connected between this connector and the USB3 port of the computer.

Serial Number

Each FireSpy has an 11-character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. The software will only work when a valid license certificate is installed for the serial number of the currently connected FireSpy. See License Manager for more information on license certificates.

24.2. Advanced Series

The “Advanced Series” of FireSpys redefines 1394 data analysis! Built on top of the proven “Basic Series” this brand-new line features better performance due to a more powerful on-board processor, bus power provider capability, new connectivity interfaces (USB 2.0 and Ethernet, PCI), enhanced Scriptor capabilities and more data capture memory.

The Advanced Series uses an more powerful architectural design with a 400MHz on-board PowerPC. A standard off-the-shelf Physical Layer chip connects to a highly optimized FPGA which not only acts as a Link layer but also incorporates the data recording functionality, Trigger and Filter engines as well as the Data Monitor. Recorded data is directly captured into onboard memory of 512MB (1GB for the FS850/FS450b).

The user can easily connect the analyzer to his PC (laptop) via a USB2.0 or Ethernet connection. All data transfer to the PC is offline, so there is no potential for the loss of data.

FireSpys from the Advanced Series are also available in PCI (FSx50y) and very soon in CompactPCI (FSx70y) configurations.

24.2.1. FireSpy 410

The FireSpy410 is a small, compact instrument that is equipped with 512 MB internal memory. The unit offers extensive hardware filtering and trigger possibilities due to efficient programmable logic and an on-board processor. It supports up to 400Mb transfer rate and is fully IEEE 1394-2000 compliant. Two standard 1394 ports allow for convenient connection to the system under test.

The FireSpy410 may be connected to a host computer via the USB 2.0 interface. Furthermore, an Ethernet port allows for easy network integration and remote control. The graphical user interface runs on Windows™ XP and Windows™ 7. It is self-intuitive and
offers a user-friendly interface. Additionally, the included FireSpy API even allows you to build your own control software by using its interfaces to several programming languages like C++, LabVIEWTM and LabWindowsTM.

24.2.1.1. Main Feature Summary

General

- IEEE 1394-1995 and 1394a-2000 compliant
- Supported S100, S200, S400 (Legacy)
- Connects to host using USB 2.0 interface or to LAN via 10/100 BaseT
- Electrical isolation between IEEE 1394 and host (USB)
- Optional Bus Power: 2.8 Watts at 12 Volt
- 480 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- External Trigger Input, positive edge
- AUX connector for:
  - Trigger input and output functions
  - Recording external event
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets

    - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets
- Measurement of bus power voltages

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
    - Packet type
    - Transmission speed
    - Boolean combination of 4 programmable packet sets
    - Data payload patterns
    - Error conditions
    - Various status events
    - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
    - Time View, displays all packets on a time line, including the prefix
    - Packet View, displays packets as list plus selected packet options
    - Transaction View, displays transactions as list or flow graph
    - Topology View, graphical topology displays as is during recording
    - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 iso streams
    - Graphically programming of stream transmit block
    - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation

Scriptor

- Script Editor
    - C-like scripting language
    - Function Library
    - Macros to automatically generate blocks of code
    - Syntax coloring
    - Integrated Debugger
    - Floating point data types
- Data Editor
- Control Panel
    - Graphical display elements for data value representation
    - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.2.1.2. Specifications

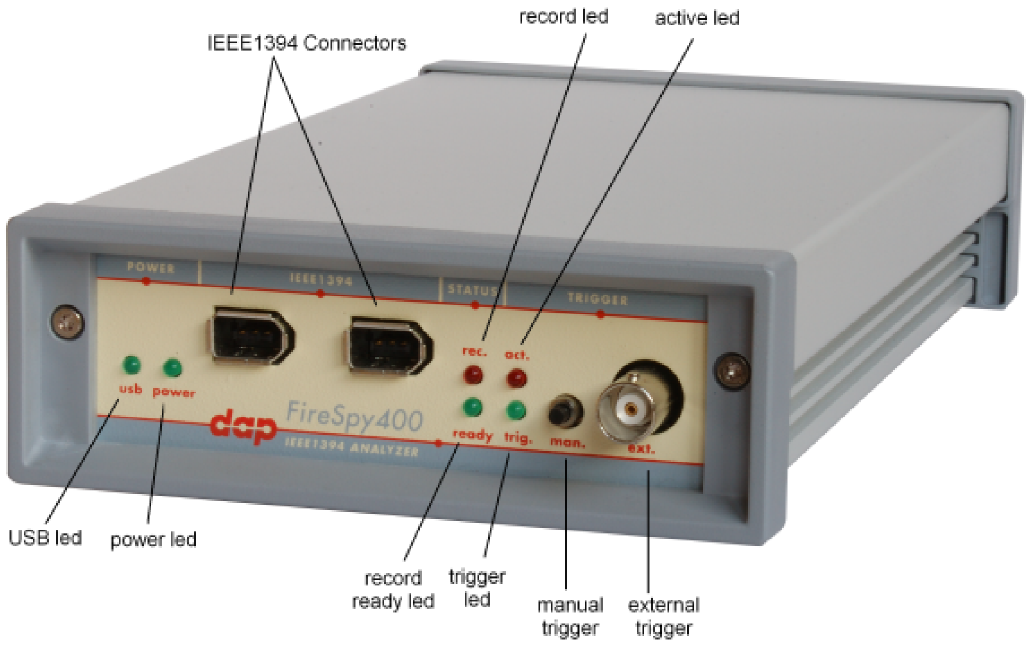
Dimensions:	125 mm x 48 mm x 224 mm
Weight:	760 g
Power Requirements:	12 V, 10 Watt maximum (without providing 1394 bus power)
Compliance:	FCC Class A
Connections:	USB2.0-connector for host-computer RJ45 Ethernet connector 2 IEEE 1394-connectors (bilingual/Beta) BNC-connector for external trigger-input
Indicators:	Green LEDs for: USB, Power, Ethernet, Trigger Red LEDs for: Record, Scriptor Active, Generate Buzzer
Switches:	Tumble switch for Power On/Off Push button for manual triggering
Package Contents:	FireSpy410 Power Adapter (12V, 1250mA) USB Cable 2.0 1394a Cable (6 pin – 6 pin) Trigger Cable
Product warranty:	36 months limited warranty
Part Number:	FS041
Optional Configuration:	-
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.2.1.3. FireSpy Front



USB led
This green led will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy 400 Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

power led
This green led will light up when the power supply is connected to FireSpy (see 'FireSpy 400 Rear' below) and the FireSpy is switched on. It indicates the availability of power from the power supply.

IEEE1394 connectors (2x)
With these connectors the FireSpy 400 can be connected to the IEEE1394a bus to be analyzed. One or both connectors may be connected to the bus (as long as no loops are created).

record led
This red led will light when recording is in progress. See Recorder.

record ready led
This green led will light up when the FireSpy Recorder is ready. See Recorder.

active led
This red led will light up when the FireSpy Generator is active. See Generator.

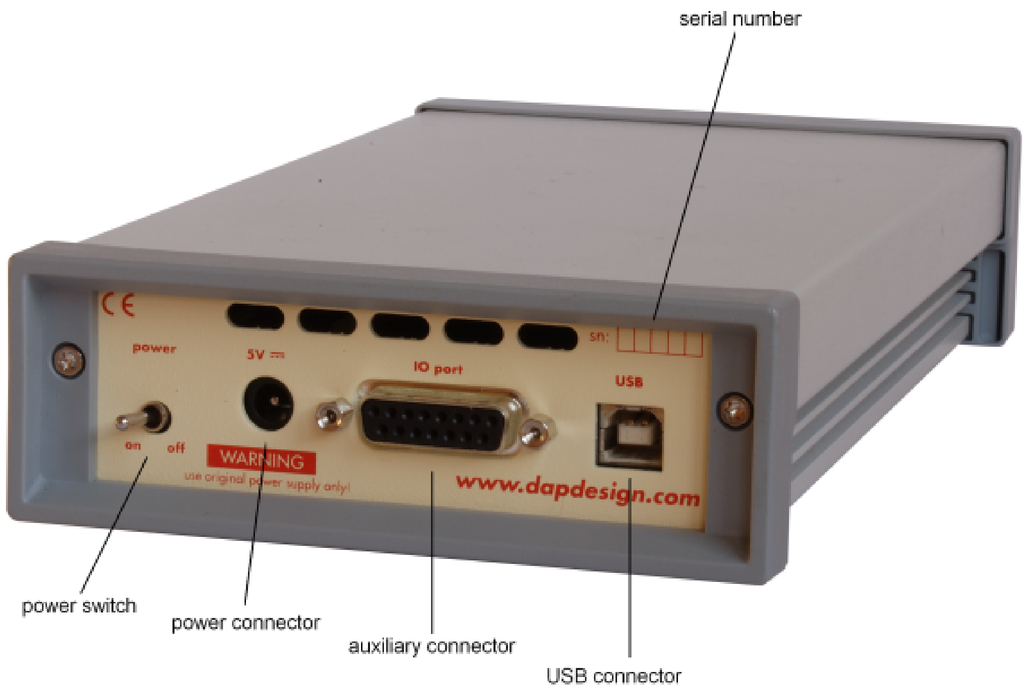
trigger led
This green led will light up when the FireSpy Recorder has been triggered. See Recorder.

manual trigger
Pressing this button will trigger the Recorder when recording is in progress and the Recorder is not yet triggered.

external trigger
Using this connector, the FireSpy can be triggered with an external electrical signal when recording is in progress and the Recorder is not yet triggered.

The FireSpy is triggered when it detects a positive pulse of minimal 3 Volt (resulting in a current of about 1.6 mA) and a duration of typical 10 uS.

24.2.1.4. FireSpy Rear



power switch
Using this switch the FireSpy can be switched on (powered) and off. When switched on the 'power led' on the front will light up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

power connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used.

auxiliary connector
The Auxiliary port is a programmable. Using the settings window, some special functions can be assigned to the aux0 through aux3. The auxiliary port will be discussed in a separate chapter. See Aux-Port.

USB connector
The FireSpy must be connected to the computer using this connector. A USB cable, which is part of the FireSpy-package, is connected between this connector and the USB port of the computer.
The USB interface is electrically isolated from the IEEE1394 logic to prevent ground loops.

serial number
Each FireSpy has a 5 character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireSpy. See License Manager for more information on license keys.

24.2.2. FireSpy 410b(T)(1), 810

FireSpy 810

The FireSpy810 bus analyzer is the flagship in the second generation of FireWire analyzers from

DapTechnology. Based on the industry leading FireSpy800 the all new and enhanced architecture of the FireSpy810 has raised the bar in IEEE 1394 test equipment! It comprises a significantly more powerful on-board processor, more memory and improved connectivity to the host.

The FireSpy810 is a small, compact instrument that is equipped with 512 MB internal memory. The unit offers extensive hardware filtering and trigger possibilities due to efficient programmable logic and an on-board processor. It supports up to 800Mb transfer rate and is fully IEEE 1394b compliant. Two bilingual 1394b ports allow for convenient connection to the system under test.

The FireSpy810 may be connected to a host computer via the USB2.0 interface. Furthermore, an Ethernet port allows for easy network integration and remote control. The graphical user interface runs on Windows™ XP and Windows™ 7. It is intuitive and offers a user-friendly way of data presentation and user control. Additionally the included API even allows you to build your own controlling software or interface to LabViewTM.

The seamless integration of the SAE AS5643 protocol makes the FireSpy810 the preferred tool for many Aerospace & Defense development tasks. DapDesign has taken considerable efforts to fully support the SAE AS5643 protocol in all major functional areas of the FireSpy810 and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

FireSpy 410b(T)(1)

The FireSpy410b bus analyzer is a variation of the FireSpy810. It utilizes the TSB41BA3 Phy Chip from Texas Instruments and supports transmission speeds up to S400 Beta Mode. Optionally, the FS410b is available in a Transformer coupled configuration.

The FireSpy410b(T)(1) is a small, compact instrument that is equipped with 512 MB internal memory. The unit offers extensive hardware filtering and trigger possibilities due to efficient programmable logic and an on-board processor. It supports up to 400Mb transfer rate and is fully IEEE 1394b compliant. Two Beta 1394b ports allow for convenient connection to the system under test.

The FireSpy410b(T)(1) may be connected to a host computer via the USB2.0 interface. Furthermore, an Ethernet port allows for easy network integration and remote control. The graphical user interface runs on Windows™ XP and Windows™ 7. It is intuitive and offers a user-friendly way of data presentation and user control. Additionally the included API even allows you to build your own controlling software or interface to LabViewTM.

The seamless integration of the AS5643 protocol makes the FireSpy410b(T)(1) the preferred tool for many Aerospace & Defense development tasks. DapDesign has taken considerable efforts to fully support the SAE AS5643 protocol in all major functional areas of the FireSpy410b(T)(1) and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.2.2.1. Main Feature Summary

General

- IEEE 1394-1995, 1394a-2000 and 1394b-2002 compliant
- Supported Speeds and Modes:

		FS410b	FS410bT	FS410bT1	FS810
Port Modes	S100	Beta		Beta	Legacy
	S200	Beta	Beta	Beta	Legacy
	S400	Beta	Beta		Bilingual
	S800	-	-		Bilingual
Connectors		Beta	Beta	Beta	Bilingual
Transformer		-	Yes	Yes	-

- Connects to host using USB 2.0 interface or to LAN via 10/100 BaseT
- Electrical isolation between IEEE 1394 and host (USB)
- Optional Bus Power Provider: 2.8 Watts at 12 Volt (except FS410bT(1))
- 480 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions

- External Trigger Input, positive edge
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets
- Measurement of bus power voltages (except FS410bT(1))

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 isochronous streams
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies

- Sending of user definable packets

24.2.2.2. *Specifications*

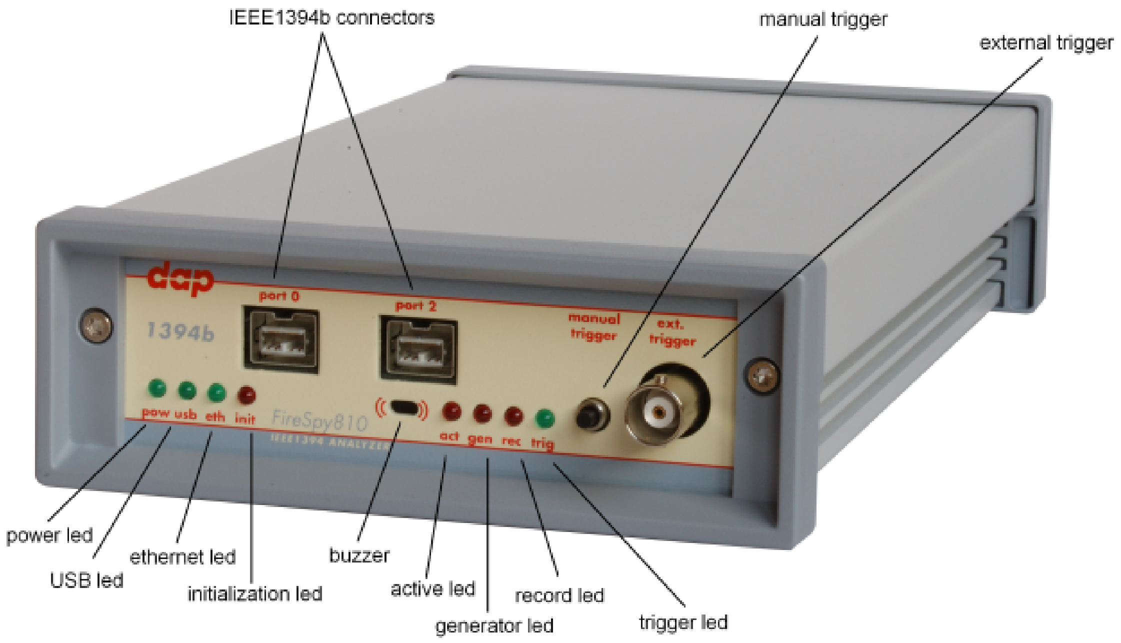
Dimensions:	125 mm x 48 mm x 224 mm
Weight:	760 g
Power Requirements:	12 V, 10 Watt maximum (without providing 1394 bus power)
Compliance:	FCC Class A
Connections:	USB2.0-connector for host-computer RJ45 Ethernet connector 2 IEEE 1394-connectors (bilingual/Beta) BNC-connector for external trigger-input
Indicators:	Green LEDs for: USB, Power, Ethernet, Trigger Red LEDs for: Record, Scriptor Active, Generate Buzzer
Switches:	Tumble switch for Power On/Off Push button for manual triggering
Package Content:	FireSpy Power Adapter (12V, 1250mA) USB Cable 2.0 1394b Cable (Beta9 – Beta9) 2x 1394b/1394a Cable (Bilingual9 – 6pin) Trigger Cable
Product warranty:	36 months limited warranty
Part Number:	FS081 FS081AS5643 (w. AS5643 SW protocol package)
Optional Configuration:	FS041b (TSB41BA3) FS041bT (TSB41BA3 and S200+ transformers) FS041bT1 (TSB41BA3 and S100-S200 transformers)
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.2.2.3. FireSpy Front



power led
This green led will light up when the power supply is connected to the FireSpy (see 'FireSpy 810 Rear' below) and the FireSpy is switched on. It indicates the availability of power from the power supply

USB led
This green led will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy 810 Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

ethernet led
This green led will light up when the FireSpy is connected through its ethernet port (see 'FireSpy 810 Rear' below) and the network is active.

The initialization led
This red led will light up when the FireSpy is performing an initialization of the firmware.

IEEE1394b connectors (2x)
With these connectors the FireSpy 810 can be connected to the IEEE1394 bus to be analyzed. One or both connectors may be connected to the bus.
Both connectors are bilingual. This means that they can be connected to a 1394b (beta) port or to a 1394a (legacy) port, by using the correct cables.

buzzer opening
Behind this hole a buzzer is installed.

active led
This red led will light up when the Scriptor is active. See Scriptor.

generator led
This red led will light up when the FireSpy Generator is active. See Generator.

record led

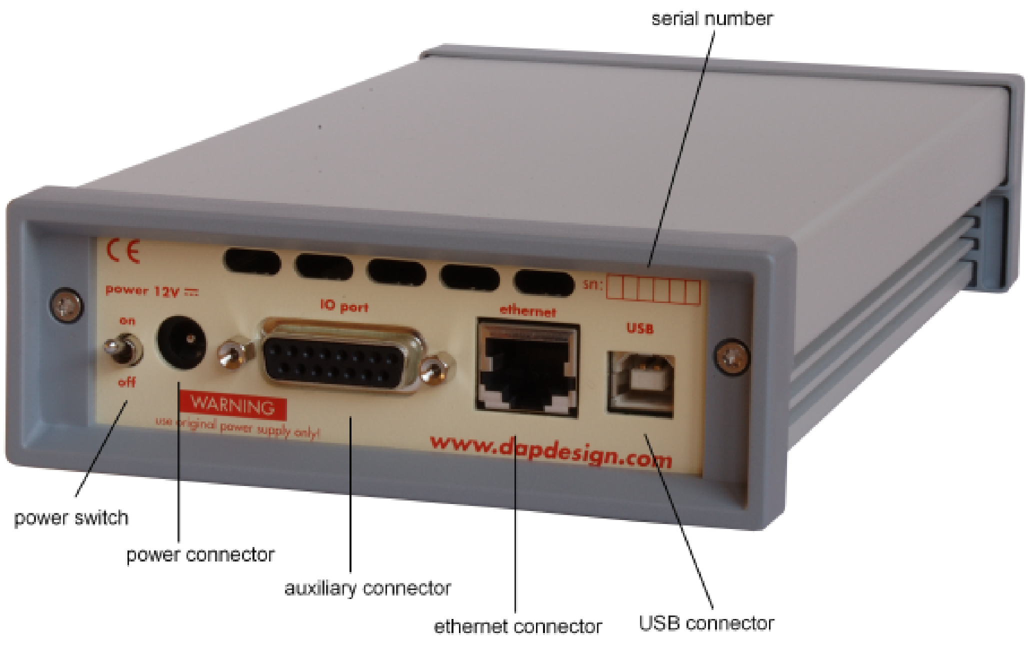
This red led will light when recording is in progress. See Recorder.

trigger led
This green led will light up when the FireSpy Recorder has been triggered. See Recorder.

manual trigger
Pressing this button will trigger the Recorder when recording is in progress and the Recorder is not yet triggered.

external trigger
Using this connector, the FireSpy can be triggered with an external electrical signal when recording is in progress and the Recorder is not yet triggered.
The FireSpy is triggered when a low to high voltage edge is detected on this pin. A low is defined as lower then 0.6 Volt. A high is defined as higher than 2.8 Volt. The maximum voltage that may be applied is 24 Volt.

24.2.2.4. FireSpy Rear



power switch
Using this switch the FireSpy can be switched on (powered) and off. When switched on the 'power led' on the front will light up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

power connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used.

auxiliary connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

ethernet connector
This is a standard ethernet port. it is used for network access to the FireSpy 810.

USB connector

The FireSpy must be connected to the computer using this connector. A USB cable, which is part of the FireSpy-package, is connected between this connector and the USB port of the computer.
The USB interface is electrically isolated from the IEEE1394b logic to prevent ground loops.

serial number
Each FireSpy has a 5 character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireSpy. See License Manager for more information on license keys.

24.2.3. FireSpy 450b(T)(1) / 850

The FireSpy850 and FireSpy450b bus analyzers complete the second generation of FireWire analyzers offered by Dap-Technology. Based on the industry leading FireSpy800, the all new and enhanced architecture of the FireSpy810, as well as the form factor advantages introduced with the FireSpy3850, make these two new PCI form factor analyzer cards the most compelling package for card-based 1394 analysis solutions found in the industry.

The PCI format takes flexibility to a higher level. This solution enables users to install the FireSpy hardware in any PC that has at least one PCI slot available. This allows for installation of the FireSpy Analyzer in developers' existing PC's, rugged field service laptop or lunchbox computer.

Both the FireSpy850 and FireSpy450b are equipped with 1 GB of on-board memory. The units offer extensive hardware filtering and trigger possibilities due to efficient programmable logic and an on-board processor. They support up to 800Mb (400Mb for the FS450b(T), 200Mb for the FS450bT1) transfer rate and are fully IEEE 1394b compliant. Three bilingual or Beta 1394 ports (depending on model) allow for convenient connection to the system under test. As an added feature the FS450b uses dip-switches to conveniently allow the changing of PHY port speed/mode settings.

The graphical user interface runs on Windows™ XP and Windows™ 7. It is intuitive and offers a user-friendly way of data presentation and user control. Additionally the included API even allows you to build your own control software or interface to LabViewTM. The seamless integration of the SAE AS5643 protocol makes the FireSpy850 or the FS450b an ideal tool for many Aerospace & Defense development tasks. For long distance applications DapTechnology also offers the FS450b in an impedance matching transformer configuration, i.e. the FireSpy model FS450bT(1).

24.2.3.1. Main Feature Summary

General

- IEEE 1394-1995, 1394a-2000 and 1394b-2002 compliant
- Supported Speeds and Modes:

		FS450b	FS450bT	FS450bT1	FS850
Port Modes	S100	Beta		Beta	Legacy
	S200	Beta	Beta	Beta	Legacy
	S400	Beta	Beta		Bilingual
	S800	-	-		Bilingual
Connectors		Beta	Beta	Beta	Bilingual
Transformer		-	Yes	Yes	-

- PCI 2.1 compliant
- Optional Bus Power (2.8 W at 12V) (except FS450bT(1))
- 992 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

- Measurement of bus power voltages (except FS450bT(1))

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 iso streams
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.2.3.2. Specifications

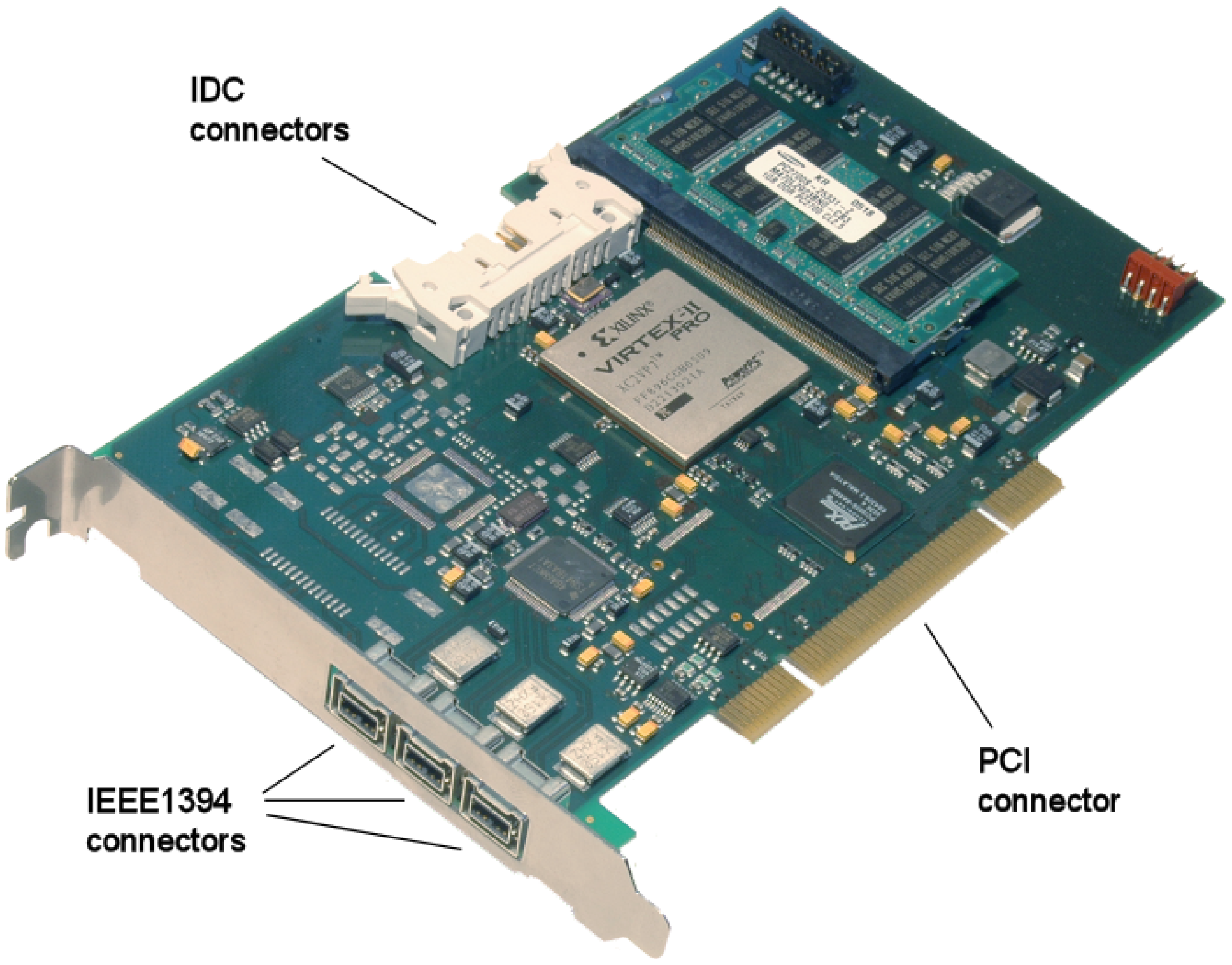
Dimensions:	PCI: half length, 125 x 48 x 209 mm
CPCI:	210 x 20 x 129 mm
Weight:	150 g / TBD
Operating Range:	0 – 70 C
Power Requirements:	12V, 10 Watt maximum (without providing 1394 bus power)
Compliance:	FCC Class A
Connections:	32bit/33MHz PCI connector, universal keyed (for 3.3V and 5V slots) 3 IEEE 1394-connectors (bilingual or Beta)
Indicators:	-
Switches:	Dip-switches for PHY port Speed/Mode Selection, (FS450b and FS450bT(1) only)
Package Content:	FireSpy850 1394b Cable (Beta9 – Beta9) 2x 1394b/1394a Cable (Bilingual9 – 6pin)
Product warranty:	36 months limited warranty
Part Number:	FS85 or FS85AS5643 (w. AS5643 SW protocol package)
Optional Configuration:	• FS45b or FS45bAS5643 with TSB41BA3 • FS45bT or FS45bTAS5643 with TSB41BA3 and S200+ transformer • FS45bT1 or FS45bT1AS5643 with TSB41BA3 and S100-S200 transformer
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.2.3.3. The PCI board



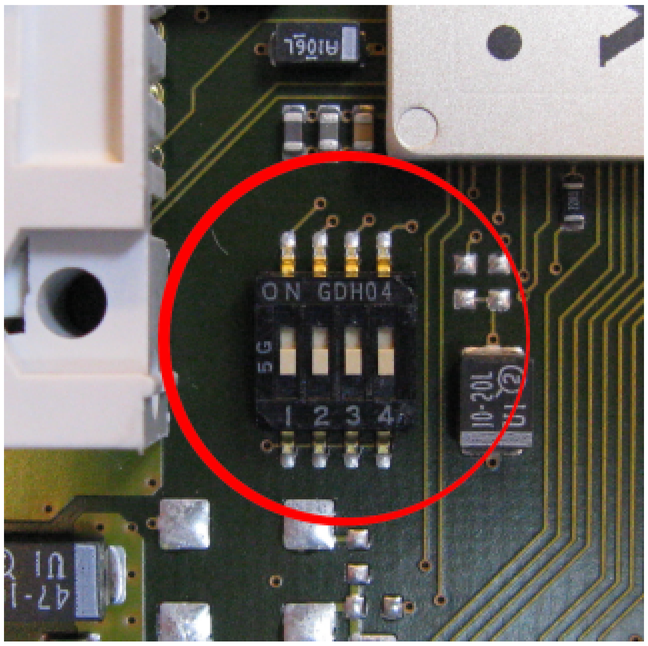
This FireSpy is a PCI card. It must be build into a PC.

IDC connector
The IDC connector is an auxiliary port.The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports. .

IEEE1394b connectors (3x)
With these connectors the FireSpy can be connected to the IEEE1394 bus to be analyzed.
For the FireSpy450b, both ports are beta only. For the FireSpy850, both ports are bilingual.
The FireSpy450bT(1) has 3 beta only ports that are transformer coupled.

PCI connector
The PCI connector is the connection between this FireSpy and the computer it is to work with. Place the connector into an empty PCI slot in the computer.

DIP Switches
The FS450b(T)(1) card contains DIP Switches that can control the mode of the PHY chip. The following modes can be set: S100, S200 and S400 (all Beta only).



SW1	SW2	SW3	SW4	
0	0	0	0	S400b only, data strobe S400, S200, S100
0	1	0	0	All Ports Beta S100
1	0	0	0	All Ports Beta S200
1	1	0	0	All Ports Beta S400

24.3. Triple Series

Utilizing the restructured hardware architecture of the FS810, both the FS3x10 and FS385x offer the industry's first 3 channel data analysis capability. And multiple units can be chained together to build a synchronized 3, 6 or 9 channel system.

The Triple Series uses an more powerful architectural design with a 400MHz on-board PowerPC. Three off-the-shelf Physical Layer chips connect to a highly optimized FPGA which not only acts as Link layers but also incorproate three completely independent data recording functions, Trigger and Filter engines as well as Data Monitors. Recorded data are directly captured into onboard memory (1GB).

The user can easily connect the analyzer to his PC (laptop) via a USB2.0 or Ethernet connection (or a PCI interface). All data transfer to the PC is offline, so there is no potential for the loss of data.

FireSpys from the Triple Series are also available in a PCI (FS385x) and in CompactPCI (FS3470) configuration.

24.3.1. FireSpy 3810, 3811,3410bT(1)

The FireSpy3810 bus analyzer is the first in a new generation of FireWire analyzers from DapTechnology. Based on the industry leading FireSpy800 and the new FireSpy3810 architecture the FireSpy3810 is the most advanced IEEE 1394 test equipment in the market. The FireSpy3810 in fact combines three FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host. With the new scripting engine the FireSpy3810 can also be used as a stand-alone analyzer.

The FireSpy3810 has three 1394 nodes connected to three synchronized analysis engines. They are controlled by a RISC processor running at 400 MHz. Each node is connected to three FireWire ports. One port of each node is connected to an individual standard bilingual FireWire connector. The other two ports of each node are connected to one high density, high speed connector and include active transformer coupling. A cable converting to 6 Beta connectors is provided.

The FireSpy3810 is equipped with 1024 MB internal memory and extensive hardware filtering and trigger possibilities. The analyzer can be connected to a host computer using the USB or Ethernet interface. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way; or you can use the API to program your own control software.

24.3.1.1. Main Feature Summary

General

- IEEE 1394-1995, 1394a-2000 and 1394b-2002 compliant
- FS3810,FS3811: Supports 100(L), 200(L), 400(L&B) and 800(B) Mbps connection speeds
- FS3410bT: Supports 200(B) and 400(B) Mbps connection speeds
- FS3410bT(1): Supports 100(B) and 200(B) Mbps connection speeds
- Connects to host using USB2.0 interface or to LAN via 10/100 BaseT
- Electrical isolation between IEEE 1394 and host (USB)
- Optional Bus Power: 2.8 Watts at 12 Volt (for each bilingual port)
- 992 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external event
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets
- Measurement of bus power voltages (bilingual ports)

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 iso streams

    - Graphically programming of stream transmit block
    - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
    - C-like scripting language
    - Function Library
    - Macros to automatically generate blocks of code
    - Syntax coloring
    - Integrated Debugger
    - Floating point data types
- Data Editor
- Control Panel
    - Graphical display elements for data value representation
    - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.3.1.2. Specifications

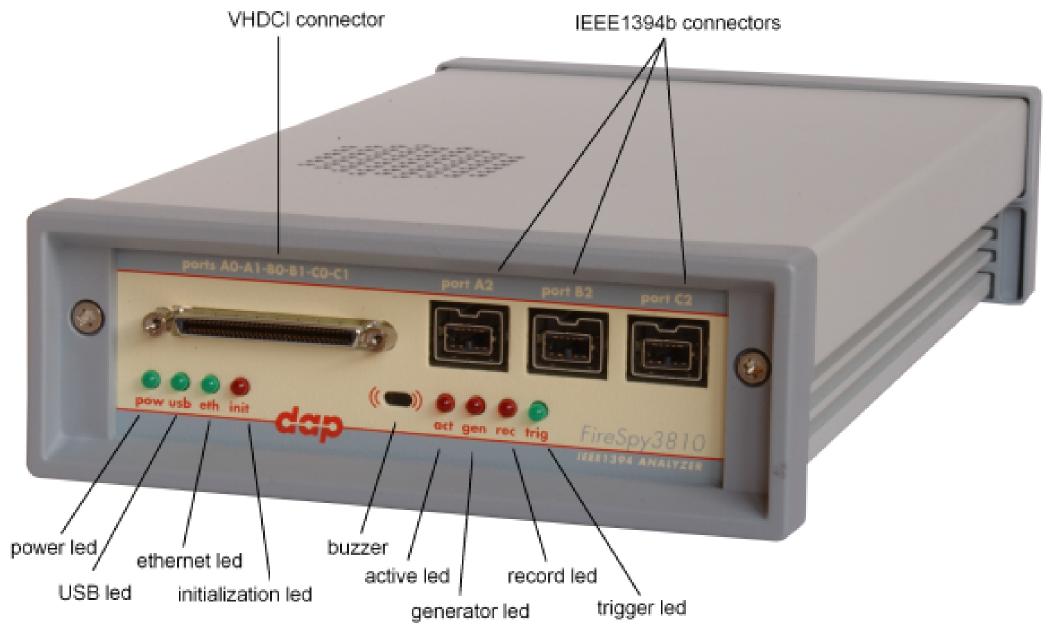
Dimensions:	125 mm x 48 mm x 209 mm
Weight:	800 g
Operating Range:	0 – 45 C
Power Requirements:	12 V, 10 Watt maximum (without providing 1394 bus power)
Compliance:	FCC Class A
Connections:	USB2.0-connector for host-computer RJ45 Ethernet connector For FS3810, FS3410bT(1): • high-density connector (VHDCI) for 6 transformer coupled 1394Beta connections • 3 IEEE 1394-connectors (bilingual) For FS3811: • 6 IEEE 1394-connectors (bilingual)
Indicators:	Green LEDs for: USB, Power, Ethernet, Trigger Red LEDs for: Record, Scriptor Active, Generate Buzzer
Switches:	Tumble switch for Power On/Off Push button for manual triggering
Package Content:	FireSpy Power Adapter (12V, 2500mA) USB Cable 2.0 For FS3810, FS3410bT(1): • 3x 1394b Cable (Beta9 – Beta9) • Adapter Cable (VHDCI – 1394Beta (x6)) For FS3811: • 3x 1394b Cable (Bilingual 9 – 9)
Product warranty:	36 months limited warranty
Part Number:	FS381 or FS381AS5643 (S800 w. AS5643 SW protocol package) FS341 or FS341AS5643 (S400 w. AS5643 SW protocol package)
Optional Configuration:	FS3811 (no transformers, 6 bilingual ports) FS3410bT1 (S100-S200 transformers)
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.3.1.3. FS3810 and FS3410bT(1) Front



VHDCI connector
This connector is a 68 pins connector. It encompasses the transformer coupled IEEE1394b ports A0, A1, B0, B1, C0, C1 and a 6 aux port signals.
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

power led
This green led will light up when the power supply is connected to the FireSpy (see 'FireSpy 3810 Rear' below) and the FireSpy is switched on. It indicates the availability of power from the power supply

USB led
This green led will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy 3810 Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

ethernet led
This green led will light up when the FireSpy is connected through its ethernet port (see 'FireSpy 3810 Rear' below) and the network is active.

The initialization led
This red led will light up when the FireSpy is performing an initialization of the firmware.

IEEE1394b connectors (3x)
With these connectors the FireSpy 3810 can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. Each connector is not connected to either of the other two ports.
The IEEE1394b ports are (from left to right) port A2, port B2 and port C2.
Each connector is bilingual. This means that they can be connected to a 1394b (beta) port or to a 1394a (legacy) port (FS381x only), by using the correct cables.

buzzer opening
Behind this hole a buzzer is installed.

active led

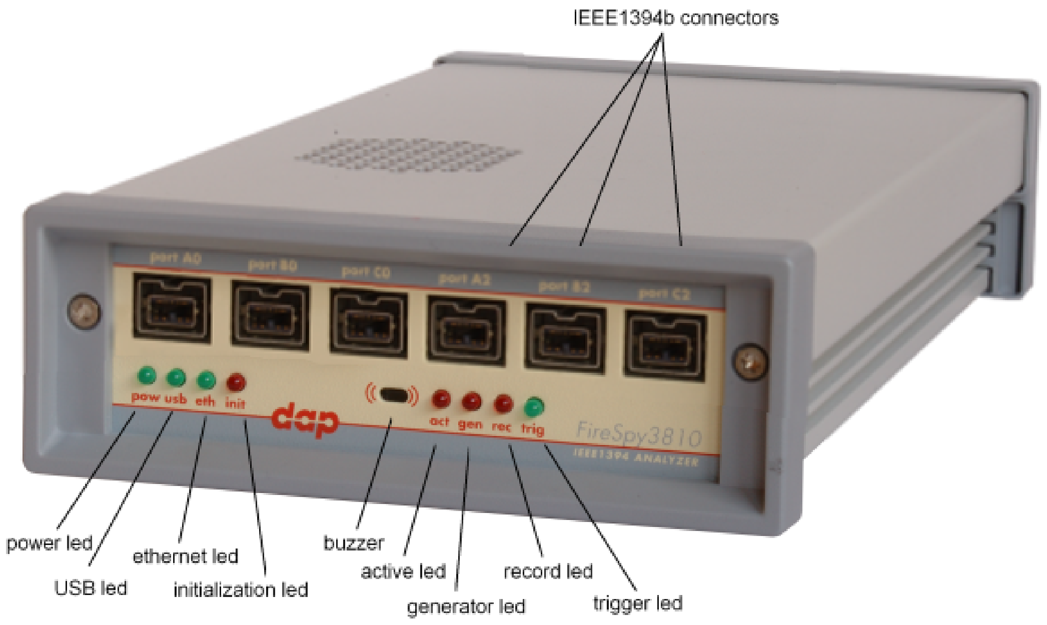
This red led will light up when the Scriptor is active. See Scriptor.

generator led
This red led will light up when the FireSpy Generator is active. See Generator.

record led
This red led will light when recording is in progress. See Recorder.

trigger led
This green led will light up when the FireSpy Recorder has been triggered. See Recorder.

24.3.1.4. FS3811 Front



power led
This green led will light up when the power supply is connected to the FireSpy (see 'FireSpy Rear' below) and the FireSpy is switched on. It indicates the availability of power from the power supply

USB led
This green led will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

ethernet led
This green led will light up when the FireSpy is connected through its ethernet port (see 'FireSpy Rear' below) and the network is active.

The initialization led
This red led will light up when the FireSpy is performing an initialization of the firmware.

IEEE1394b connectors (6x)
With these connectors the FireSpy 3811 can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. The IEEE1394b ports are (from left to right) A0, A1, B0, B1, C0, C1.

Each connector is bilingual. This means that they can be connected to a 1394b (beta) port or to a 1394a (legacy) port, by using the correct cables.

buzzer opening
Behind this hole a buzzer is installed.

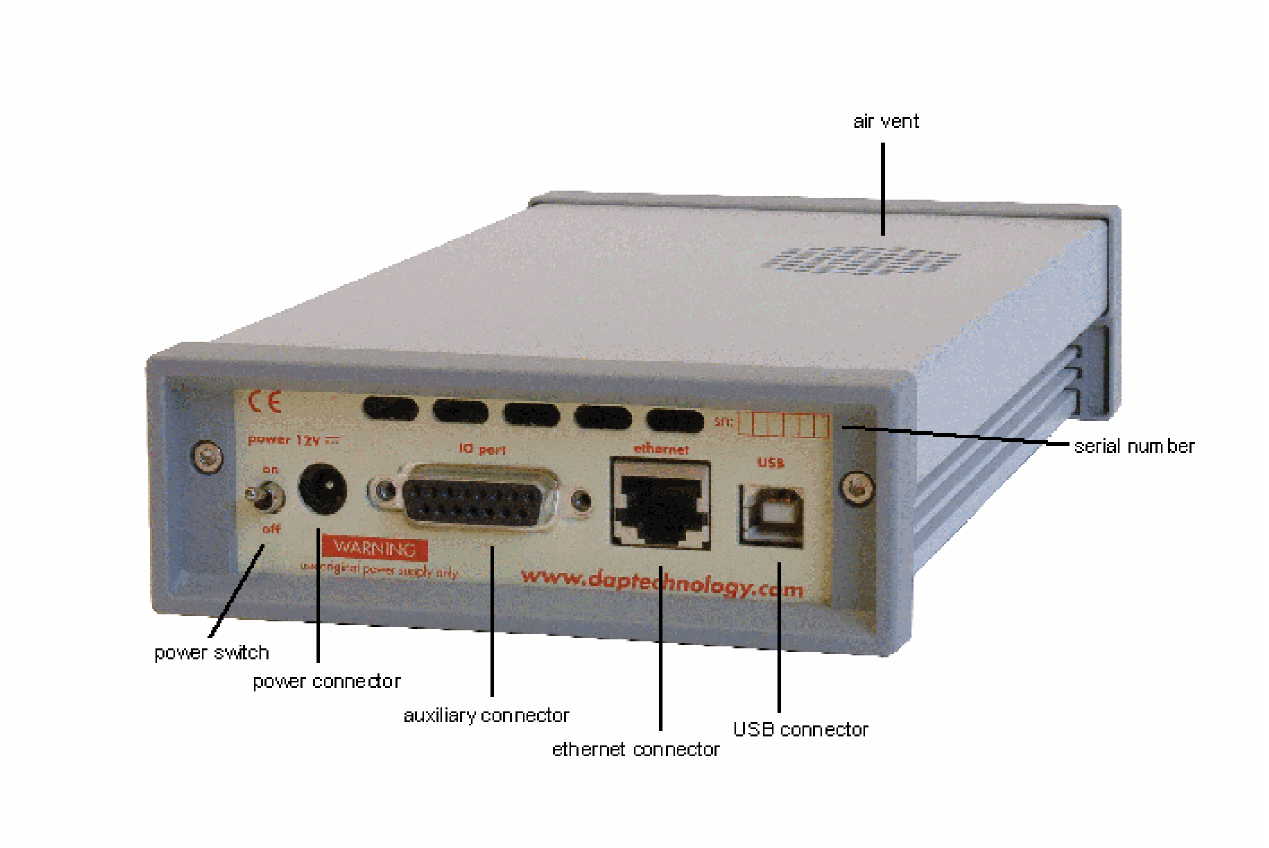
active led
This red led will light up when the Scriptor is active. See Scriptor.

generator led
This red led will light up when the FireSpy Generator is active. See Generator.

record led
This red led will light when recording is in progress. See Recorder.

trigger led
This green led will light up when the FireSpy Recorder has been triggered. See Recorder.

24.3.1.5. FireSpy Rear



power switch
Using this switch the FireSpy can be switched on (powered) and off. When switched on the 'power led' on the front will light up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

power connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used.

auxiliary connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

ethernet connector
This is a standard ethernet port. it is used for network access to the FireSpy 3810.

USB connector
The FireSpy must be connected to the computer using this connector. A USB cable, which is part of the FireSpy-package, is connected between this connector and the USB port of the computer.
The USB interface is electrically isolated from the IEEE1394b logic to prevent ground loops.

serial number
Each FireSpy has a 5 character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireSpy. See License Manager for more information on license keys.

24.3.2. FireSpy 3850, 3851

The FireSpy385x bus analyzer is world's only PCI –based multichannel FireWire bus analyzer. Based on the industry leading FireSpy810 architecture the FireSpy385x is the most advanced IEEE 1394 test equipment in the market. The FireSpy385x in fact
combines three FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host.

The PCI format takes flexibility to a higher level. This solution enables users to install the FireSpy hardware in any PC that has at least one PCI slot available. This allows for installation of the FireSpy in developers existing PC's or installing the analyzer in a rugged field service laptop or lunchbox computer.

The FireSpy385x has three 1394 nodes connected to three synchronized analyzer engines. They are controlled by an onboard RISC processor running at 400 MHz. Each node is connected to three FireWire ports. One port of each node is connected to a separate standard bilingual FireWire connector.
The other two ports of each node are connected to one high speed connector and include active transformer coupling.

The FireSpy385x is equipped with 1 GByte memory and extensive hardware filtering and trigger capabilities. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way, or you can use the API and program your own control software.

The seamless integration of the SAE AS5643 protocol makes the FireSpy385x the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the SAE AS5643 protocol in all major functional areas of the FireSpy385x and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.3.2.1. Main Feature Summary

General

- IEEE 1394-1995, 1394a-2000 and 1394b-2002 compliant
- Supports 100(L), 200(L), 400(L&B) and 800(B) Mbps transfer rates
- PCI 2.1 compliant
- Optional Bus Power: 2.8 W at 12V (for each bilingual port)
- 992 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external event
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets

- several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets
- Measurement of bus power voltages (bilingual ports)

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 iso streams
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.3.2.2. Specifications

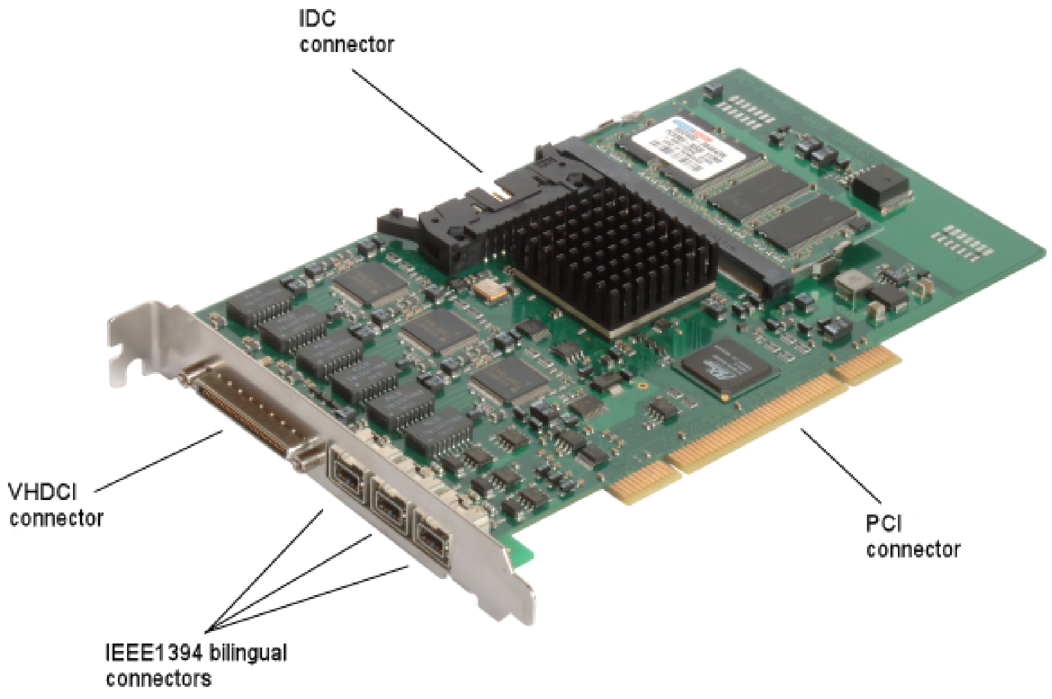
Dimensions:	Half Length PCI, 125 mm x 48 mm x 209 mm
Weight:	150 g
Operating Range:	0 – 70 C
Power Requirements:	12 V, 10 Watt maximum (without providing 1394 bus power)
Compliance:	FCC Class A
Connections:	• 32bit/33MHz PCI connector, universal keyed (for 3.3V and 5V slots) • 3 IEEE 1394-connectors (bilingual) • high-density connector (VHDCI) for 6 transformer coupled 1394Beta connections
Indicators:	-
Switches:	-
Package Content:	• FireSpy • 3x 1394b Cable (Beta9 – Beta9) • Adapter Cable (VHDCI – 1394Beta (x6))
Product warranty:	36 months limited warranty
Part Number:	FS385 or FS385AS5643 (w. AS5643 SW protocol package)
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.3.2.3. The FS3850, FS3851 PCI board



This FireSpy is a PCI card. It must be build into a PC.

IDC connector
The IDC connector is an auxiliary port.The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports. .

VHDCI connector
This connector is a 68 pins connector. It encompasses the tramsformer coupled IEEE1394b ports A0, A1, B0, B1, C0, C1 and a 6 aux port signals.
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

IEEE1394bilingual connectors (3x)
With these connectors the FireSpy 3850 can be connected to the IEEE1394 bus to be analyzed. Each connector may be connected to the bus. Each connector is not connected to either of the other two ports. The IEEE1394 ports are (from left to right) port A2, port B2 and port C2.
Each connector is bilingual. This means that they can be connected to a 1394b (beta) port or to a 1394a (legacy) port, by using the correct cables.

PCI connector
The PCI connector is the connection between this FireSpy and the computer it is to work with. Place the connector into an empty PCI slot in the computer.

24.3.3. FireSpy 3852

The FireSpy385x bus analyzer is world's only PCI –based multichannel FireWire bus analyzer. Based on the industry leading FireSpy810 architecture the FireSpy385x is the most advanced IEEE 1394 test equipment in the market. The FireSpy385x in fact
combines three FireSpy analyzers in one single instrument. It comprises a significantly more powerful on-board processor and improved connectivity to the host.

The PCI format takes flexibility to a higher level. This solution enables users to install the FireSpy hardware in any PC that has at least one PCI slot available. This allows for installation of the FireSpy in developers existing PC's or installing the analyzer in a rugged field service laptop or lunchbox computer.

The FireSpy385x has three 1394 nodes connected to three synchronized analyzer engines. They are controlled by an onboard RISC processor running at 400 MHz.Two ports of each node are connected to a circular LEMO connector and include active transformer coupling

The FireSpy385x is equipped with 1 GByte memory and extensive hardware filtering and trigger capabilities. On the host you can control the FireSpy using a graphical user interface to analyze and display the bus traffic in a user-friendly way, or you can use the API and program your own control software.

The seamless integration of the SAE AS5643 protocol makes the FireSpy385x the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the SAE AS5643 protocol in all major functional areas of the FireSpy385x and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.3.3.1. Main Feature Summary

General

- IEEE 1394-1995, 1394a-2000 and 1394b-2002 compliant
- Supports 200(B) , 400(B) and 800(B) Mbps transfer rates
- PCI 2.1 compliant
- 992 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
    - Trigger input and output functions
    - Recording external event
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
    - isochronous packets
    - all types of asynchronous packets
    - all types of PHY packets
    - all types of acknowledge packets
    - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets
- Measurement of bus power voltages (bilingual ports)

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as 'prefix only' packets
- Extensive packet/event filtering/trigger/search capabilities
    - Packet type
    - Transmission speed
    - Boolean combination of 4 programmable packet sets
    - Data payload patterns

    - Error conditions
    - Various status events
    - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
    - Time View, displays all packets on a time line, including the prefix
    - Packet View, displays packets as list plus selected packet options
    - Transaction View, displays transactions as list or flow graph
    - Topology View, graphical topology displays as is during recording
    - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 iso streams
    - Graphically programming of stream transmit block
    - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
    - C-like scripting language
    - Function Library
    - Macros to automatically generate blocks of code
    - Syntax coloring
    - Integrated Debugger
    - Floating point data types
- Data Editor
- Control Panel
    - Graphical display elements for data value representation
    - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.3.3.2. Specifications

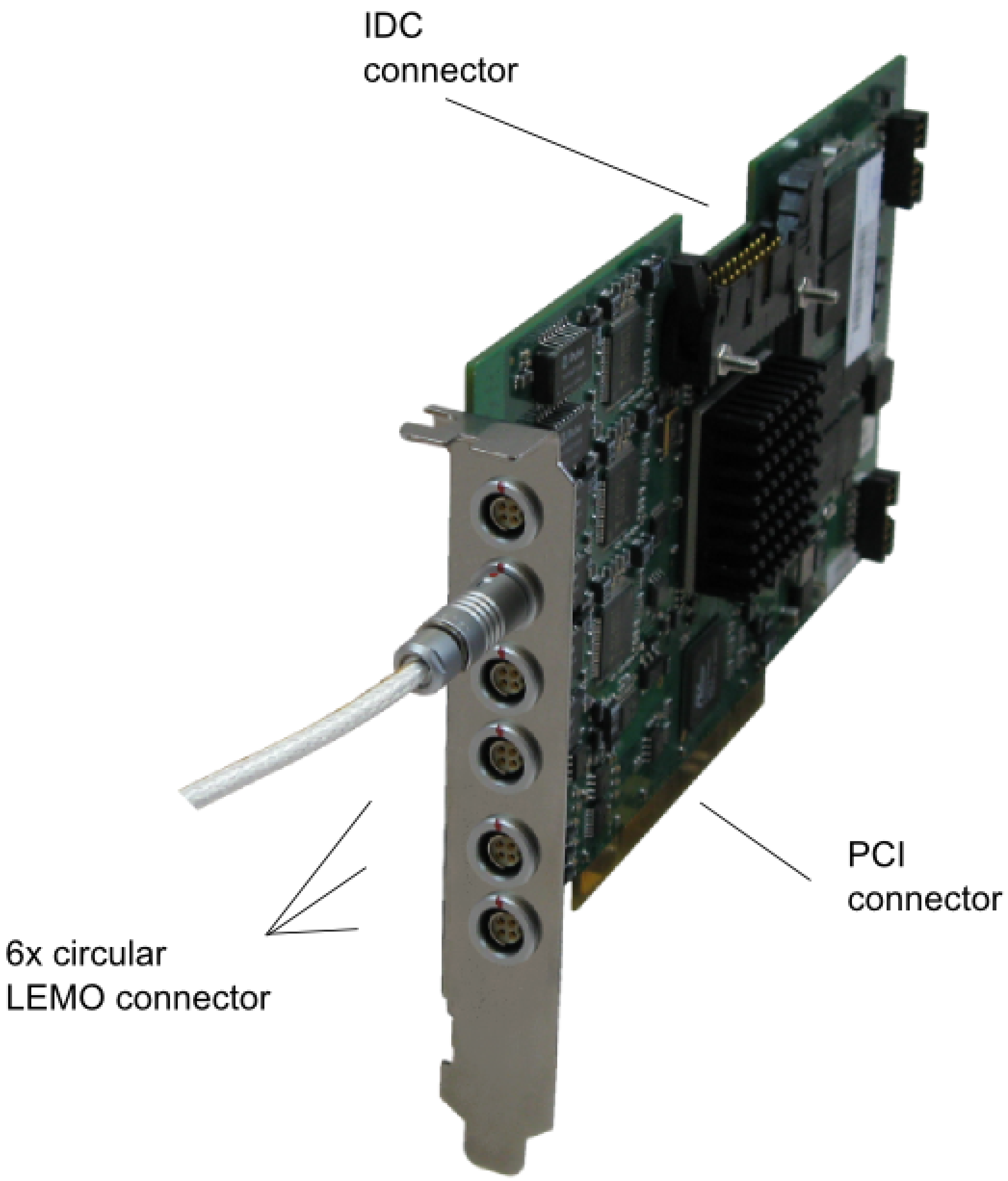
Dimensions:	Half Length PCI, 125 mm x 48 mm x 209 mm
Weight:	150 g
Operating Range:	0 – 70 C
Power Requirements:	12 V, 10 Watt maximum (without providing 1394 bus power)
Compliance:	FCC Class A
Connections:	• 32bit/33MHz PCI connector, universal keyed (for 3.3V and 5V slots) • 6x Circular LEMO connector type FGG.0B.304.CLAD42
Indicators:	-
Switches:	-
Package Content:	• FireSpy • 6x LEMO circular to IEEE-1394 bilingual
Product warranty:	36 months limited warranty
Part Number:	FS3852 or FS3852AS5643 (w. AS5643 SW protocol package)
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.3.3.3. The FS3852 PCI board



This FireSpy is a PCI card. It must be build into a PC.

IDC connector
The IDC connector is an auxiliary port.The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports. .

Circular LEMO connector (6x)
With these connectors the FireSpy 3852 can be connected to the IEEE1394 bus to be analyzed. Each FireSpy node (A, B, C) exposes two of its PHY ports through one LEMO connector per port. The FireSpy ports exposed are numbered top-down as follows: A0, A1, B0, B1, C0. C1. The exact connector model used is LEMO EGG.0B.304.CLL.

PCI connector
The PCI connector is the connection between this FireSpy and the computer it is to work with. Place the connector into an empty PCI slot in the computer.

24.3.4. FireSpy 3470

24.3.4.1. Main Feature Summary

General

- IEEE 1394-19000595, 1394a-2000 and 1394b-2002 compliant
- Supports 400(B) Mbps transfer rates
- Compact PCI PICMG 2.0 R3.0 compliant
- PXI 2.2 compliant*
- 992 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
  - isochronous packets
  - all types of asynchronous packets
  - all types of PHY packets
  - all types of acknowledge packets
  - several types of Errors
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 63 iso streams
  - Graphically programming of stream transmit block
  - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
  - C-like scripting language
  - Function Library
  - Macros to automatically generate blocks of code
  - Syntax coloring
  - Integrated Debugger
  - Floating point data types
- Data Editor
- Control Panel
  - Graphical display elements for data value representation
  - Ethernet-connected Client Panels for remote data monitoring
- Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

*remark, the PXI trigger bus is not used.

24.3.4.2. Specifications

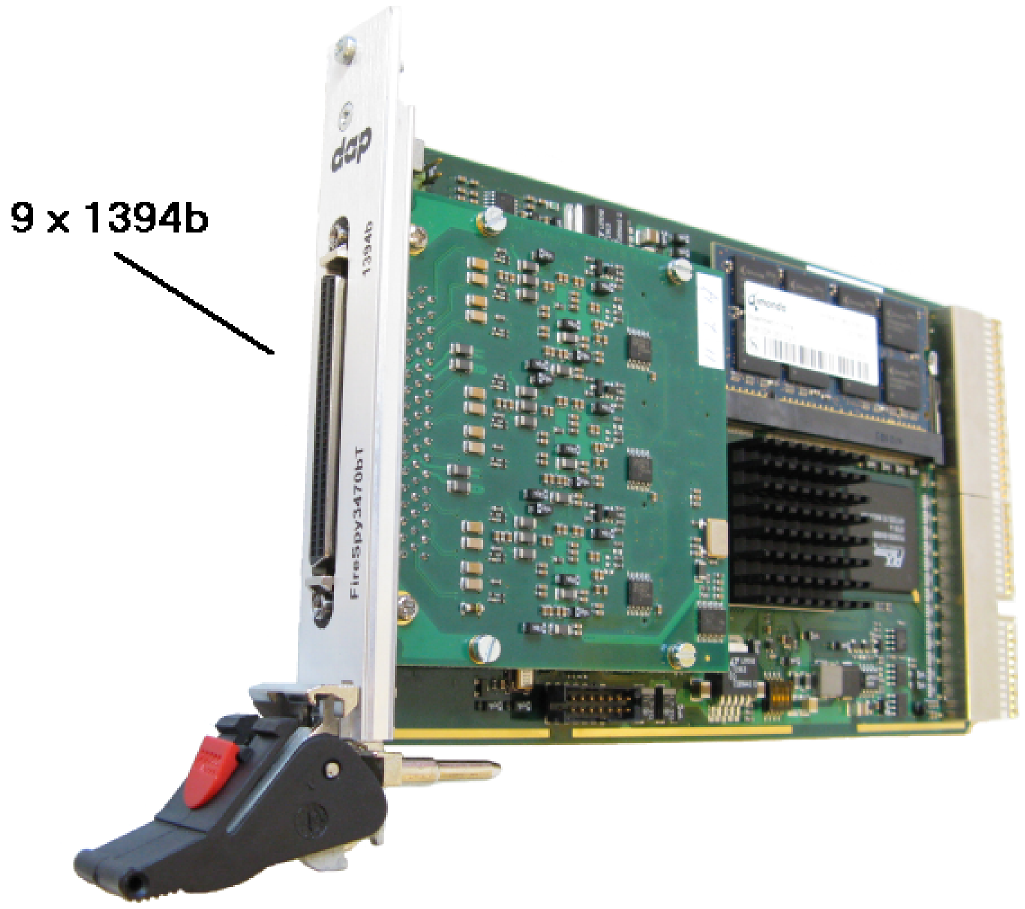
Dimensions:	213 mm x 130 mm x 20 mm
Weight:	250 g
Operating Range:	0 – 70 C
Power Requirements:	10 Watt maximum
Compliance:	FCC Class A
Connections:	• PXI J1, J2 connector • 68 pins SCSI II/III Connector for 9 transformer coupled 1394Beta connections
Indicators:	-
Switches:	-
Package Content:	FireSpy3470 Adapter Cable (SCSI2 – 1394Beta (x9))
Product warranty:	36 months limited warranty
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.3.4.3. The PXI Board



This FireSpy is a CompactPCI / PXI card, it must be fitted in a CompactPCI / PXI chassis.

SCSI2 Connector
This connector is a 68 pins connector. It encompasses the transformer coupled IEEE1394b ports A0, A1, A2, B0, B1, B2, C0, C1 and C2

Compact PCI connector
The PCI connector is the connection between this FireSpy and the computer it is to work with. Place the connector into an empty CompactPCI slot in the computer/ PXI chassis.

24.4. Stealth Series

24.4.1. FireStealth 810bN

DapTechnology's FireStealth810bN© with its patent pending non-intrusive bus monitoring technology, is the industry' s first "Stealth Mode" IEEE 1394b analyzer. The FireStealth IEEE 1394b analyzer is a bus monitoring device that is attached to the IEEE 1394b bus leaving the existing topology unaffected. The device does not participate in the bus configuration and thus, no topology reconfiguration and node ID reassignments occur. The FireStealth's passive personality and non-intrusive behavior renders it invisible to all other devices on the IEEE 1394b bus.
Also, it does not participate in any bus management activities such as root contention, cycle start generation & transmission arbitration.

The FireStealth is a small, compact instrument that is equipped with 1 GB internal memory. The unit offers extensive hardware filtering and trigger possibilities due to efficient programmable logic and an on-board processor. It supports up to 800Mb transfer rate (beta Mode only). Two 1394b connectors allow for convenient connection to the system under test.

The FireStealth may be connected to a host computer via the USB 2.0 interface. Furthermore, an Ethernet port allows for easy network integration and remote control. The graphical user interface runs onWindows™ XP and

Windows™ 7. It is intuitive and offers a user-friendly way of data presentation and user control.

Additionally the included API even allows you to build your own control software by using its interfaces to several programming languages like C++, LabView™ and LabWindows™. The seamless integration of the AS5643 protocol (based on SAE AS5643) makes the FireStealth a great tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the SAE AS5643 protocol in all major functional areas of the FireStealth and continuously updates the analyzer functionality according to implementation requirements and ongoing standardization efforts.

24.4.1.1. Main Feature Summary

General

- Supports 100(B), 200(B), 400(B) and 800(B) Mbps transfer rates
- Connects to host using USB2.0 interface or to LAN via 10/100 BaseT
- Electrical isolation between IEEE 1394 and host (USB)
- 992 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- AUX connector for:
  - Trigger input and output functions
  - Recording external events
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets with non-Stealth analyzer

Symbol Recorder (valid module license required)
- Record RAW 1394 data at the bit level
- Advanced software analysis decodes bits into symbols
- IRIG-B122 Time synchronization
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of views

Scriptor

- Script Editor
- Data Editor
- Control Panel
- Several Sample Scripts

Topology

- Graphical Display of current Bus Topology
- Live topology updates upon Bus Reset Detection

24.4.1.2. Specifications

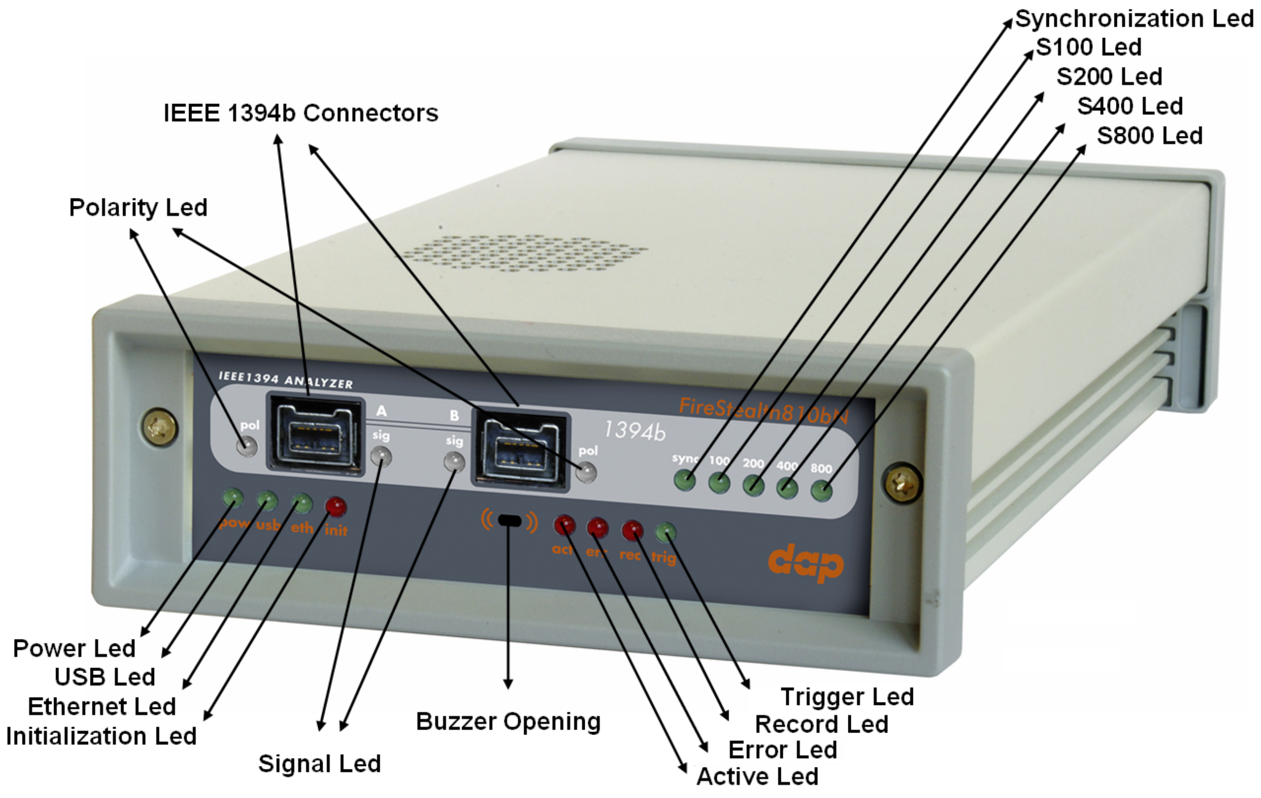
Dimensions:	125 mm x 48 mm x 224 mm
Weight:	770 g
Operating Range:	0 – 45 C
Power Requirements:	12 V, 10 Watt maximum
Compliance:	FCC Class A
Connections:	• USB2.0-connector for host-computer • RJ45 Ethernet connector • 2 IEEE 1394-connectors (Beta)
Indicators:	• Green LEDs for: Sync, Speed , USB, Power, Ethernet, Trigger • Red LEDs for: Record, Scriptor Active, Ready • Bicolor LEDs for: Port Polarity, Port Signal • Buzzer
Switches:	Tumble switch for Power On/Off
Package Content:	FireStealth810bN Power adapter (12V, 1250mA) 1394b Cable (Beta9 – Beta9), 1meter Trigger Cable
Product warranty:	36 months limited warranty
Part Number:	FS081bN or FS081bNAS5643 (w.AS5643 SW protocol package)
Optional Configuration:	-
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.4.1.3. FireStealth Front



Polarity led
This dual colored (green/red) led will light up green whenever the FireStealth has participated in the connection setup of the IEEE 1394b bus and has detected that the polarity of the cable is normal.
This dual colored (green/red) led will light up red whenever the FireStealth has participated in the connection setup of the IEEE 1394b bus and has detected that the polarity of the cable is inverted.
This dual colored (green/red) led will light up orange whenever the FireStealth has participated in the connection setup of the IEEE 1394b bus and has detected that the there is no polarity switching inside the cable although the polarity settings selected in the dialog window of the Data Pickup Logic Connection Settings are set to inverted.
This dual colored (green/red) led will not light up whenever the FireStealth has not participated in the connection setup of the IEEE 1394b bus and has not been able to detect any information about the cable polarity. (This will always be the case whenever configuration of the FireStealth is done after configuration of the 1394b bus.)

Signal led
This dual colored (green/red) led will light up green whenever the FireStealth has detected that the IEEE 1394b bus is operational.
This dual colored (green/red) led will light up red for 10 milliseconds whenever the FireStealth has detected a bus reset or an asynchronous packet on the respective port.
This dual colored (green/red) led will light up orange whenever the FireStealth has detected isochronous packets on the respective port. The intensity of the orange will range from orange to more red the more isochronous packets are detected on the respective port.

Synchronization led
This green led will light up when the FireStealth has found synchronization to the IEEE 1394b bus.

S100 led
This green led will light up when the line speed of the FireStealth has been set to S100 in the Data Pickup Logic Connection Settings dialog window.

S200 led
This green led will light up when the line speed of the FireStealth has been set to S200 in the Data Pickup

Logic Connection Settings dialog window.

S400 led
This green led will light up when the line speed of the FireStealth has been set to S400 in the Data Pickup Logic Connection Settings dialog window.

S800 led
This green led will light up when the line speed of the FireStealth has been set to S800 in the Data Pickup Logic Connection Settings dialog window.

Power led
This green led will light up when the power supply is connected to the FireStealth (see 'FireStealth Rear' below) and the 'FireStealth is switched on. It indicates the availability of power from the power supply

USB led
This green led will light up when the FireStealth is connected to the USB port of your computer (see ' FireStealth Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

Ethernet led
This green led will light up when the FireStealth is connected through its ethernet port (see 'FireStealth Rear' below) and the network is active.

The initialization led
This red led will light up when the FireStealth is performing an initialization of the firmware.

IEEE 1394b connectors (2x)
With these connectors the FireStealth can be connected to the IEEE1394b bus to be analyzed. One or both connectors may be connected to the bus.
Both connectors are beta ports. This means that they can only be connected to a 1394b (beta) port.

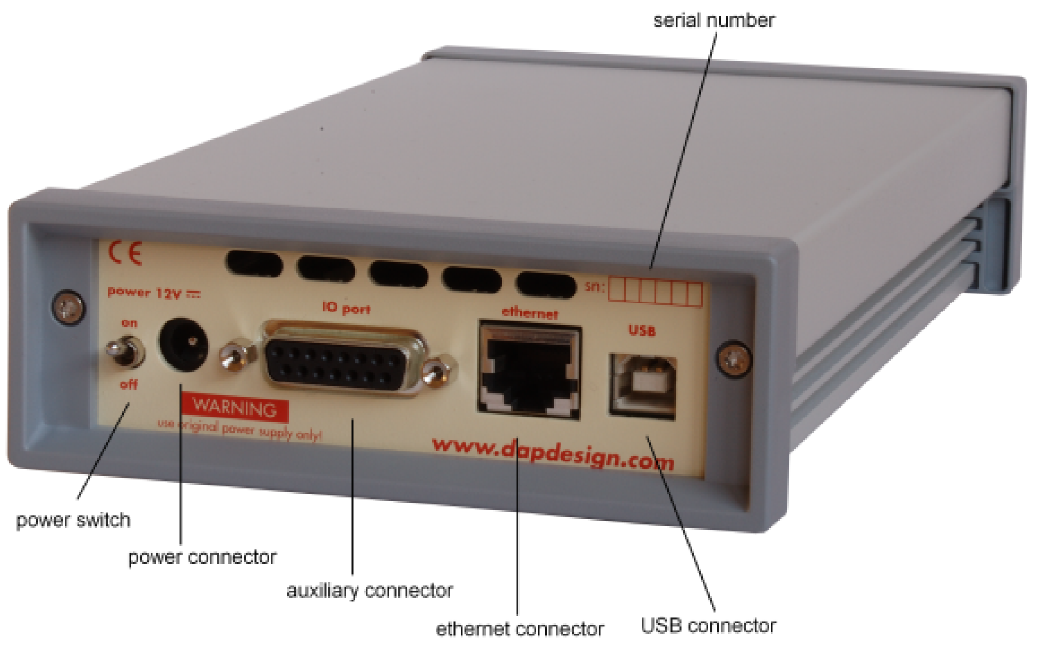
Buzzer opening
Behind this hole a buzzer is installed.

Active led
This red led will light up when the Scriptor is active. See Scriptor.

Error led
This red led will light up when a device error has occurred.

Record led
This red led will light when recording is in progress. See Recorder.

24.4.1.4. FireStealth Rear



power switch
Using this switch the FireStealth can be switched on (powered) and off. When switched on the 'power led' on the front will light up. Note that the power supply needs to be connected to the FireStealth (see below) to be able to switch the FireStealth on.

power connector
The power supply must be connected to the FireStealth, using this connector. Note that, for safety reasons, only the original power supply should be used.

auxiliary connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

ethernet connector
This is a standard ethernet port. it is used for network access to the FireStealth.

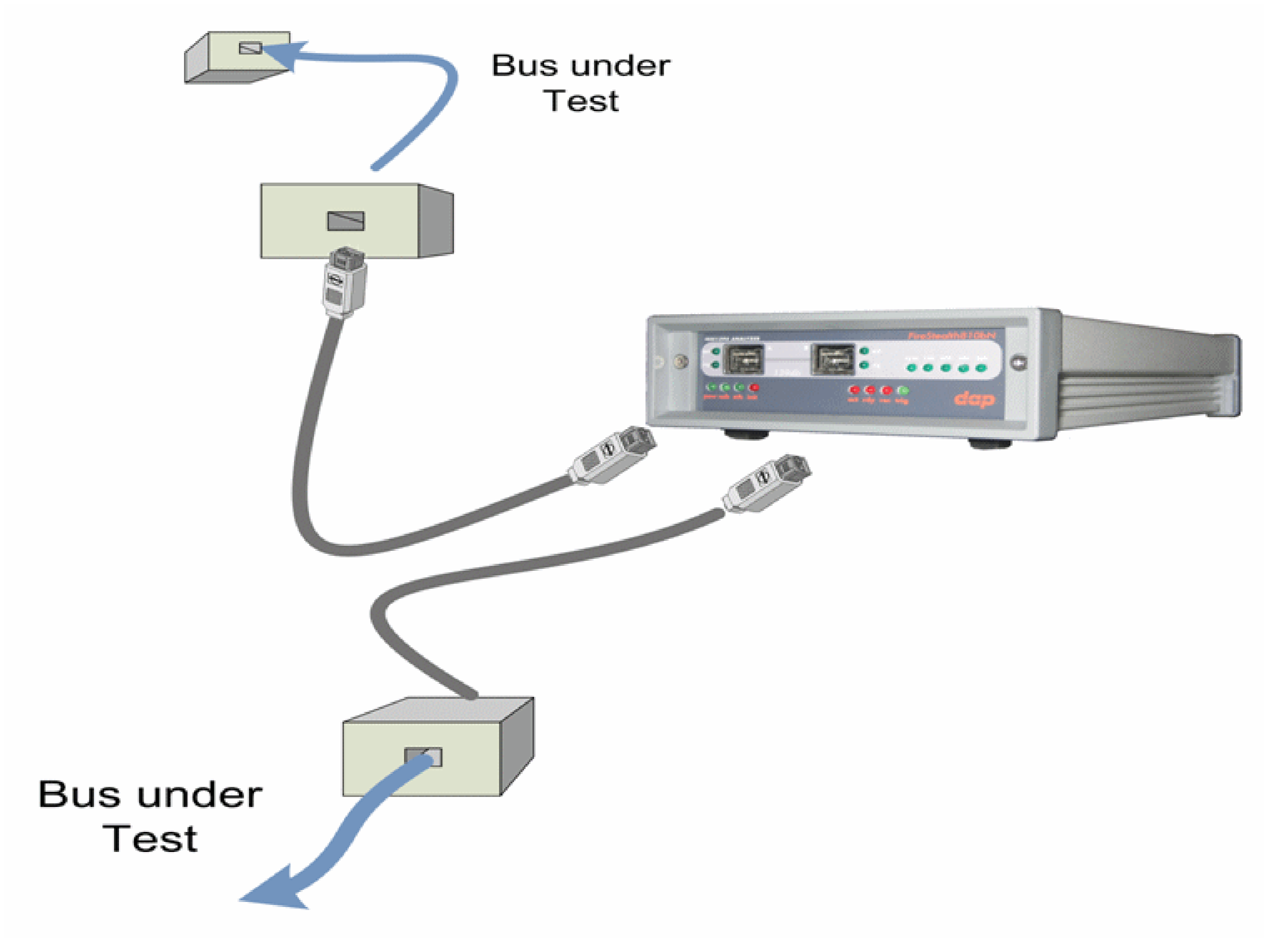
USB connector
The FireStealth must be connected to the computer using this connector. A USB cable, which is part of the FireStealth-package, is connected between this connector and the USB port of the computer.
The USB interface is electrically isolated from the IEEE1394b logic to prevent ground loops.

serial number
Each FireStealth has a 5 character serial number. This number is also programmed into the FireStealth and can be read with the License Manager of the FireStealth application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireStealth. See License Manager for more information on license keys.

24.4.2. General Operation

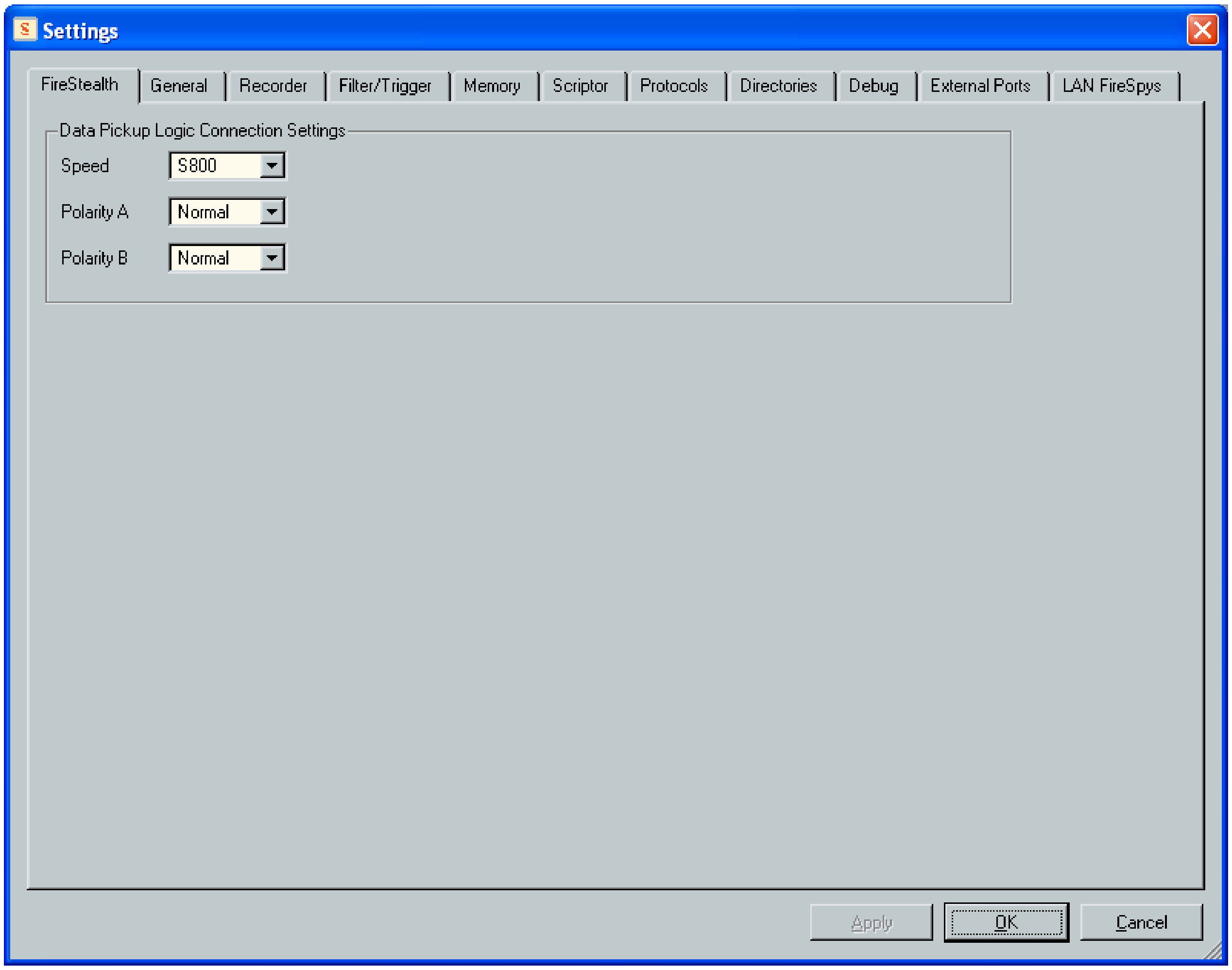
24.4.2.1. Connecting the FireStealth

The FireStealth IEEE 1394b analyzer is a bus monitoring device that is attached to the IEEE 1394b bus leaving the existing topology unaffected. The device does not participate in the bus configuration and thus, no topology reconfiguration and node ID reassignments are performed. The FireStealth's passive personality and non-intrusive behavior renders it invisible to all other devices on the IEEE 1394b bus. Also, it does not participate in any bus management activities such as root contention, cycle start generation & transmission arbitration.



24.4.2.2. Data-Pickup Logic Connection-Settings

With the application software's dedicated tab page of the settings dialog window it is possible to set the Data Pickup Logic Connection Settings.
Setting the line speed can be done by making a selection in the Speed box between S800, S400, S200 and S100.
Setting the cable polarity of the cable connected to port A of the FireStealth can be done by making a selection in the PolarityA box between Normal and Inverted.
Setting the cable polarity of the cable connected to port B of the FireStealth can be done by making a selection in the PolarityB box between Normal and Inverted.
Note that any change in the Data Pickup Logic Connection Settings will result in a resynchronization attempt of the FireStealth to the IEEE 1394b bus.



Data Pickup Logic Connection Settings

Speed S800

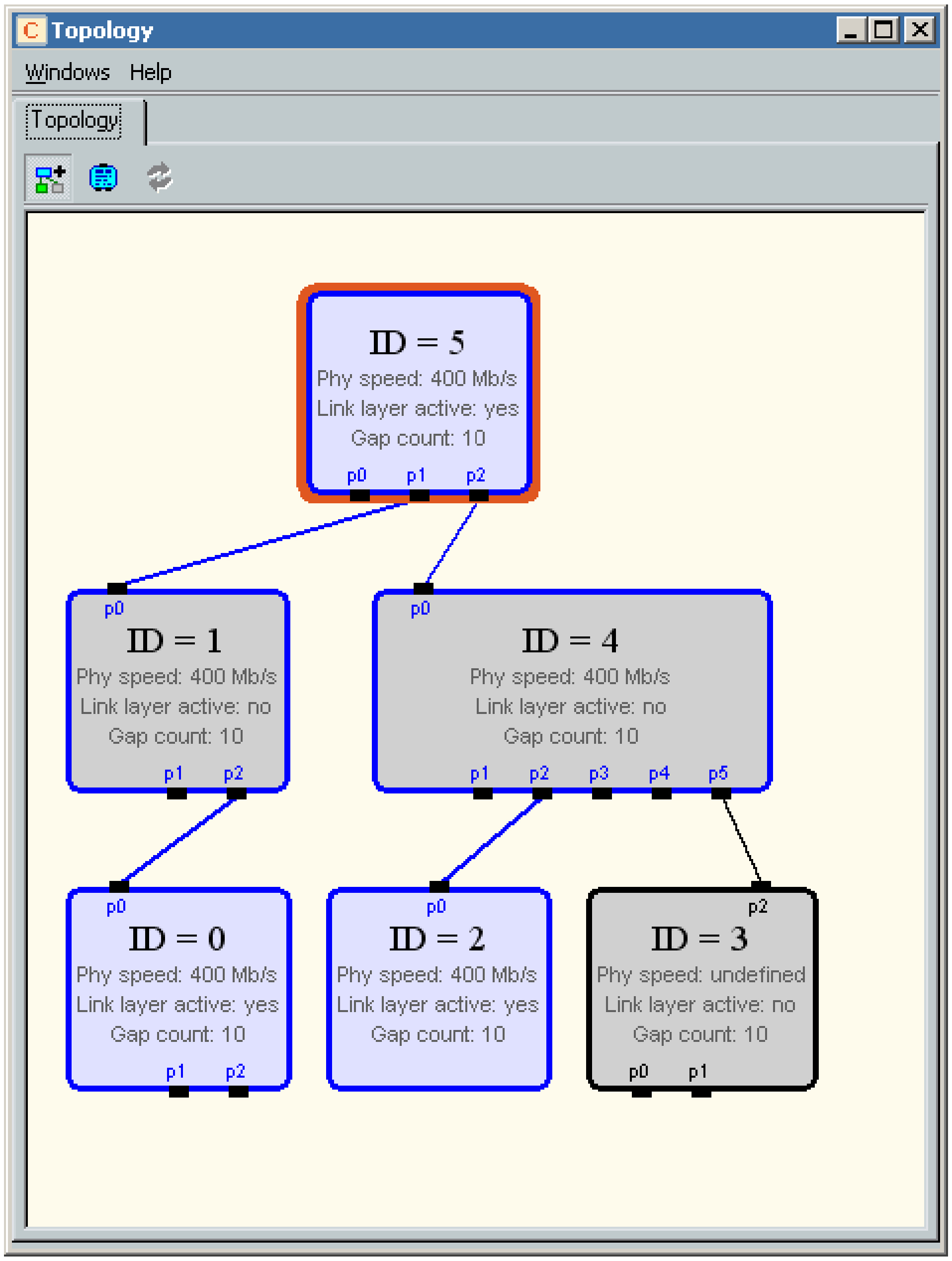
Polarity A Normal

Polarity B Normal

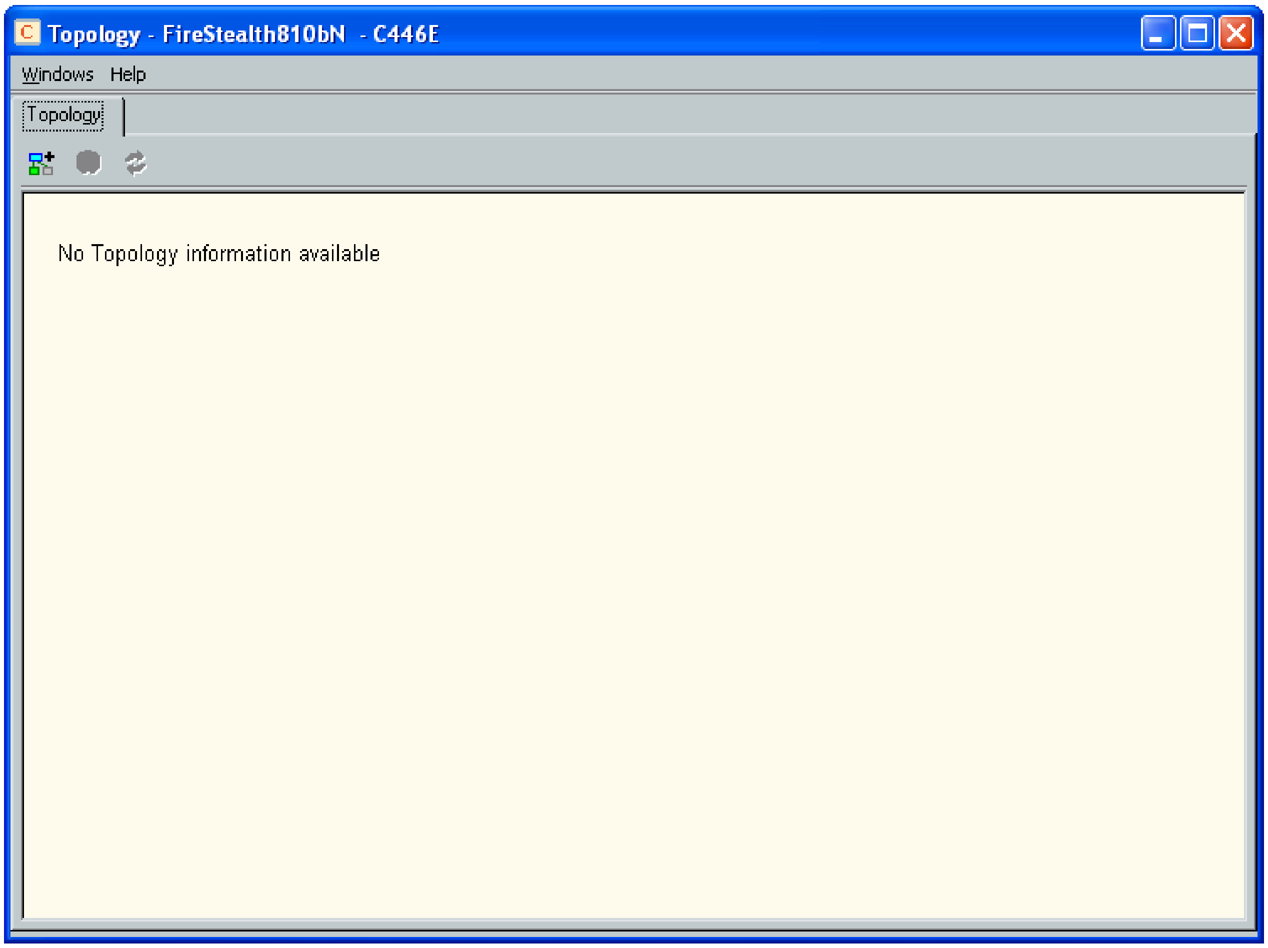
24.4.2.3. Configuration of the FireStealth

24.4.2.3.1 Bus Topology information

Whenever configuration of the FireStealth is done before configuration of the IEEE 1394b bus, the FireStealth will be able to analyze the IEEE 1394b bus configuration and will gather the bus topology information.



Whenever configuration of the FireStealth is done after configuration of the 1394b bus, the FireStealth will have missed the IEEE 1394b bus configuration and will not be able to gather the bus topology information.



No Topology information available

24.4.2.3.2 Cable Polarity Information

Whenever configuration of the FireStealth is done before configuration of the IEEE 1394b bus, the FireStealth will be able to detect, during connection setup of the IEEE 1394b bus, if the cable polarity is inverted compared to the polarity settings in the dialog window of the Data Pickup Logic Connection Settings.
For each port the information on the status of the cable polarity is indicated by the polarity led.
Whenever configuration of the FireStealth is done after configuration of the IEEE 1394b bus, the FireStealth will have missed the connection setup of the IEEE 1394b bus and will not be able to make an indication on the status of the cable polarity.

24.5. Third Generation Analyzers

In order to support the higher speeds this new generation of 1394 bus analyzer has been completely redesigned. And the 1394 interface is entirely based on DapTechnology's Firewire IP solution FireCore™ which combines both the Physical as well as the Link Layer functionality in one IP block.

Important!
Do not connect or disconnect the FireSpy when the computer is powered on.
Please turn on the connected FireSpy 1600/3200 before starting the computer and do not turn it off until after the computer is powered down.

24.5.1. Main Feature Summary

General

- S1600 (S3200 for FireSpy3200) capable 1394 PHY is implemented in FPGA allowing more low level analyzer flexibility
- Supports S800 and S1600 (and S3200 on FireSpy3200) port speeds
- Host connection via PCI Express x4

- Firmware field upgradeable to enable future expansions
- External Trigger Input: 2.5V, (not implemented yet)
- AUX connector for:
  - Trigger input and output functions
  - Recording external event (not implemented yet)
- Software runs on Windows™ XP and Windows™ 7
- Configurable 1394 Behaviour like:
  - Configuration Rom
  - Cycle master Capable
  - IRM Capable

Monitor

- Displays bus activity:
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets

Commander

- Reading and/or writing of local and reading of remote PHY registers
- Reading and/or writing of remote memory locations (incl. CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets (currently only legal packets)
- IIDC Camera Support (under development, requires IIDC Protocol license)

Recorder

- Time stamping of all packets and status events
- Packets hidden by slower connections are visible as ‘prefix only’ packets
- Extensive packet/event filtering/trigger/search capabilities
  - Packet type
  - Transmission speed
  - Boolean combination of 4 programmable packet sets
  - Data payload patterns
  - Error conditions
  - Various status events
  - Graphical Trigger Sequencer
- Different kinds of packet display views, including:
  - Time View, displays all packets on a time line, including the prefix
  - Packet View, displays packets as list plus selected packet options
  - Transaction View, displays transactions as list or flow graph
  - Topology View, graphical topology displays as is during recording
  - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges

24.5.2. Specifications

Dimensions:	170 mm x 67 mm x 175 mm (W x H x D)
Weight:	915 g
Power Requirements:	12 V, 10 Watt maximum (Bus Power not included)
Compliance:	FCC Class A
Connections:	PCIe External interface to host-computer 3 IEEE 1394-connectors (Beta Only) BNC-connector for external trigger-input
Indicators:	Green/Red LEDs for: Active, Generating, Recording, Trigger Ready
Switches:	Tumble switch for Power On/Off Push button for manual triggering
Package Content:	FireSpy1600 Power Adapter (12V, 10.0A) PCI Express External Cable PCI Express External Host Adapter Card 2 x 1394b Cable (Beta9 – Beta9) Trigger Cable
Product warranty:	36 months limited warranty
Part Number:	FS016
Optional Configuration:	
SW Add-on modules:	IIDC protocol software package

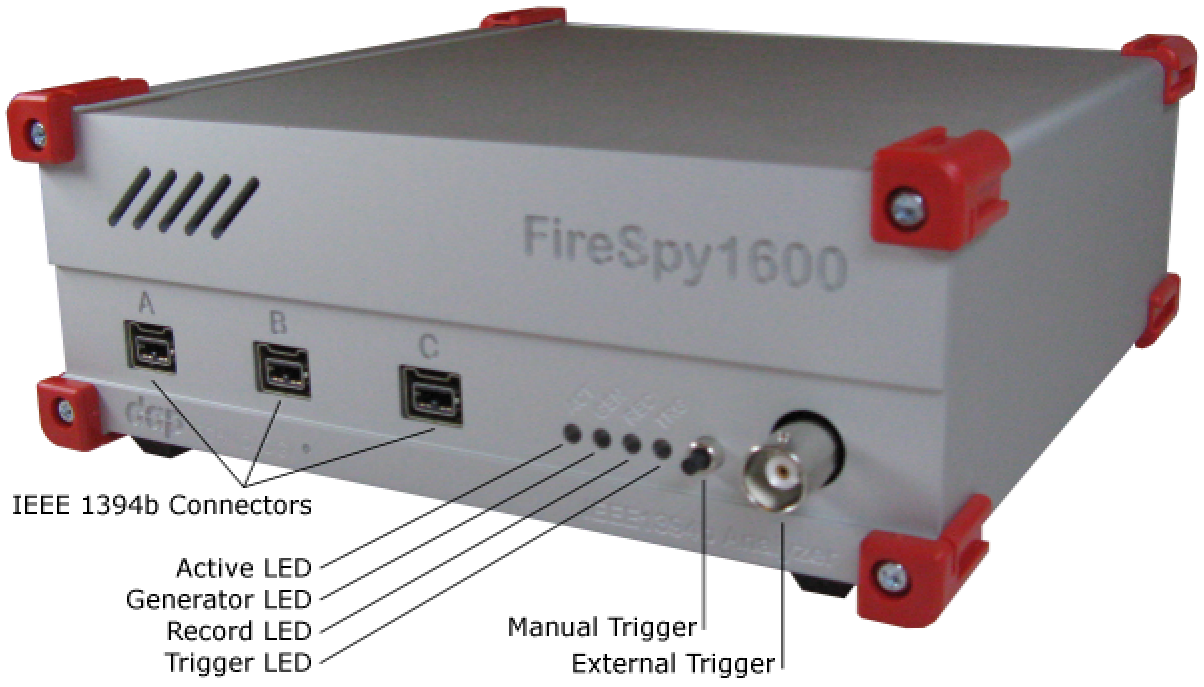
FCC Class A Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.5.3. FireSpy 1600,3200

24.5.3.1. FireSpy Front



IEEE1394b Connectors (3x)
With these connectors the FireSpy 1600 can be connected to the IEEE1394b buses to be analyzed. All connectors are beta ports. This means that they only can be connected to a 1394b (beta) port S800 or S1600.

Active led
Off: Power is off
Red: Power on, PCIe Link not active
Red and Green: Power on, PCIe Link active
Green: Power on, Link operational

Generator LED
Reserved for future use.

Record LED
Off: Recorder not active
Green: Recorder is active
Red: Recorder buffer full. This means the host can not keep up with writing the data to hard driver during recording. Please try pointing the temporary recording file to a faster hard driver (SSD).

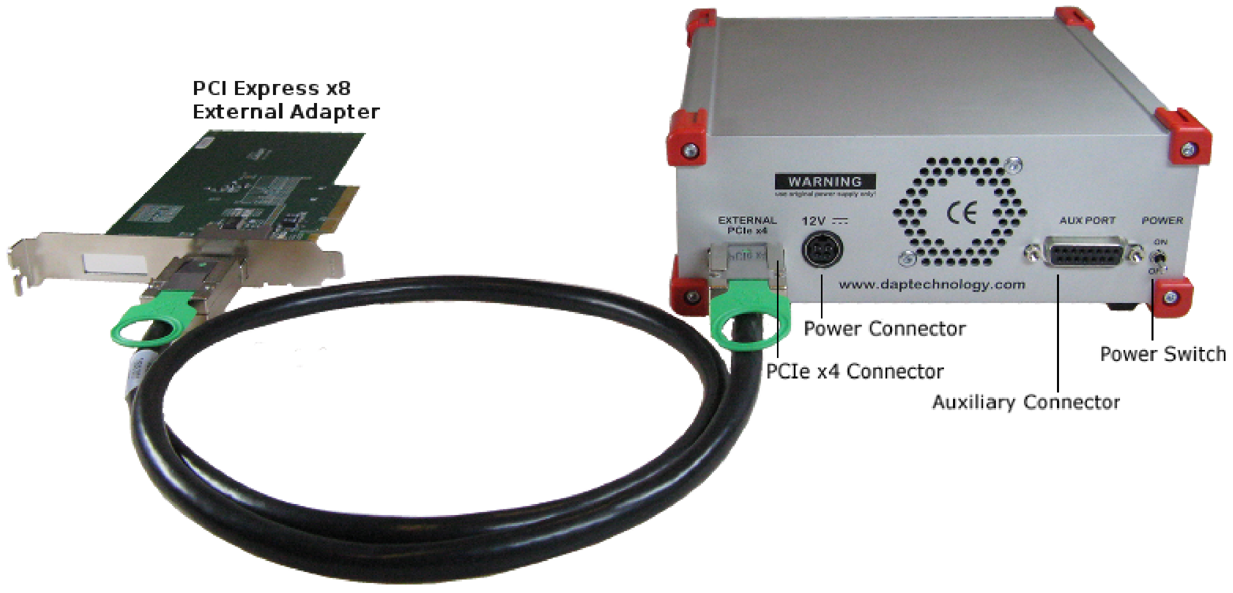
Trigger LED
Off: No trigger occurred
Green: A trigger occurred

Manual Trigger
Pressing this button during Recording marks a trigger point in the current recording

Holding this button during power-up loads the factory fallback firmware into the FPGA. Only the programmer is active, used for recovering from a faulty firmware upgrade.

External trigger
Using this connector, the FireSpy can be triggered with an external electrical signal when recording is in progress and the Recorder is not yet triggered.
The FireSpy is triggered when it detects a positive pulse .....

24.5.3.2. FireSpy Rear



External PCIe 4x Connector
The FireSpy must be connected to the computer using this connector. A PCIe 4x cable, which is part of the FireSpy-package, is connected between this connector and the PCI Express External Adapter card installed in the computer.

Power Connector
Connect only the original power supply to this connector.

Auxiliary Connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

Power Switch
Using this switch the FireSpy can be switched on (powered) and off. When turned on, the FPGA is configured from flash. This takes +/- 1 second.

Serial Number (on the bottom)
Each FireSpy has an 11 character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireSpy. See License Manager for more information on license keys.

24.6. Basic 1394 Analyzer Series

With their initial releases in 2000 and 2003 the FireSpy400 and FireSpy800 have defined new standards for 1394 data analysis. Their unmatched feature set and the outstanding price/performance ratio of the FireSpys have made them the most commonly used 1394 data analyzers in the marketplace.

The FireSpy800 data and protocol analyzer has gained tremendous market reputation in Aerospace & Defense, Consumer Electronics, Industrial Control and many other markets.

The FireSpy800 has defined an industry standard for 1394 data analysis and DapTechnology is fully committed to providing continued improvements and enhancements to its feature set and functionality.

The FireSpy800 is a small, compact instrument that is equipped with 256 MB internal memory. The unit offers extensive hardware filtering and trigger possibilities due to efficient programmable logic and an on-board processor. It supports up to 800Mb transfer rate and is fully IEEE 1394b compliant. Two bilingual 1394b ports allow for conveniently connecting to the particular system under test.

The FireSpy800 may be connected to a host computer via the USB interface. The graphical user interface runs on Windows™ XP and Windows™ 7. It is intuitive and offers a user-friendly user interface. Additionally, the included FireSpy API allows you to build your own control software by using its interfaces to several programming languages like C++, LabVIEWTM and LabWindowsTM. The seamless integration of the SAE AS5643 protocol makes the FireSpy800 the preferred tool for many Aerospace & Defense development tasks. DapTechnology has taken considerable efforts to fully support the SAE AS5643 protocol in all major functional areas of the FireSpy800 and continuously updates the analyzer functionality according to implementation requirements and ongoing tandardization efforts.

24.6.1. Main Feature Summary

General

- IEEE 1394-1995, 1394a-2000 and 1394b-2002 compliant
- Supported Speeds and Modes:

		FS400b	FS800
Port Modes	S100	Beta	Legacy
	S200	Beta	Legacy
	S400	Beta	Bilingual
	S800	-	Beta
Connectors		Beta	Bilingual
Transformer		-	-

- Connects to host using USB 1.1 interface
- Electrical isolation between IEEE 1394 and host (USB)
- 248 MByte memory for packet and data storage
- Firmware field upgradeable to enable future expansions
- External Trigger Input: 3V, positive edge, opto-coupled
- AUX connector for:
    - Trigger input and output functions
    - Recording external event
- Software runs on Windows™ XP and Windows™ 7

Monitor

- Displays bus activity:
- Counts packets according to type, speed, ack and error condition
- Counts number of bus resets
- Measurement of bus power voltages

Recorder

- Time stamping of all packets and status events with 10ns resolution
- Packets hidden by slower connections are visible as ‘prefix only’
- packets
- Extensive packet/event filtering/trigger/search capabilities
- Adjustable trigger position within programmable record buffer size
- Cyclic pre-trigger buffer management option
- Different kinds of packet display views, including:
    - Time View, displays all packets on a time line, including the prefix
    - Packet View, displays packets as list plus selected packet options
    - Transaction View, displays transactions as list or flow graph
    - Topology View, graphical topology displays as is during recording
    - Protocol View, displays packets decoded to selected protocol
- Precise time measurements
- Marking of individual packets or packet ranges
- Export format for re-generation of packets by Scriptor or API

Generator

- Simultaneous generation of up to 31 iso streams
    - Graphically programming of stream transmit block
    - Data payload import from file
- Generator and Scriptor run simultaneous for stream and asynchronous packet generation
- Special AS5643 stream generator package (optional)

Scriptor

- Script Editor
    - C-like scripting language
    - Function Library
    - Macros to automatically generate blocks of code
    - Syntax coloring
    - Integrated Debugger
- Data Editor
- Control Panel
    - Graphical display elements for data value representation
    - Ethernet-connected Client Panels for remote data monitoring
    - Several Sample Scripts

Commander

- Reading and/or writing of local and reading of remote
- PHY registers
- Reading and/or writing of remote memory locations (incl.
- CSR register space)
- Possibility to graphically view the current Topologies
- Sending of user definable packets

24.6.2. Specifications

Dimensions:	125 mm x 48 mm x 224 mm
Weight:	760 g
Power Requirements:	5 V, 5 Watt maximum
Compliance:	FCC Class A
Connections:	USB1.1-connector for host-computer 2 IEEE 1394-connectors (bilingual) BNC-connector for external trigger-input
Indicators:	Green LEDs for: USB, Power, Ready Trigger Red LEDs for: Record, Scriptor Active
Switches:	Tumble switch for Power On/Off Push button for manual triggering
Package Content:	FireSpy800 Power Adapter (5V, 2400mA) USB Cable 1394b Cable (Beta9 – Beta9) 2x 1394b-a Cable (Bilingual9 – 6pin) Trigger Cable
Product warranty:	36 months limited warranty
Part Number:	FS08 FS08AS5643 (w. AS5643 SW protocol package)
Optional Configuration:	FS04b or FS04bAS5643 (w. AS5643 SW protocol package)
SW Add-on modules:	SBP2 protocol software package IIDC protocol software package AV/C protocol software package IP1394 protocol software package AS5643 protocol software package

FCC Class A Compliance

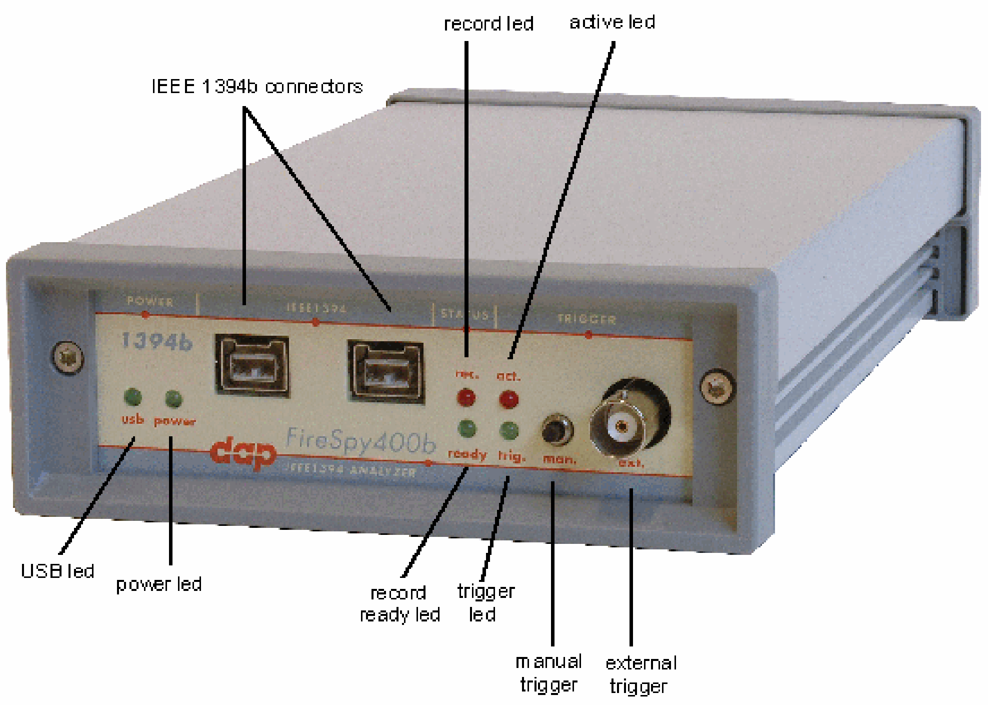
This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

24.6.3. FireSpy 400b

24.6.3.1. FireSpy Front

.



USB led
This green led will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy 400b Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

power led
This green led will light up when the power supply is connected to FireSpy (see 'FireSpy 400b Rear' below) and the FireSpy is switched on. It indicates the availability of power from the power supply.

IEEE1394b connectors (2x)
With these connectors the FireSpy 400b can be connected to the IEEE1394b bus to be analyzed. One or both connectors may be connected to the bus.
Both connectors are beta ports. This means that they only can be connected to a 1394b (beta) port.

record led
This red led will light when recording is in progress. See Recorder.

record ready led
This green led will light up when the FireSpy Recorder is ready. See Recorder.

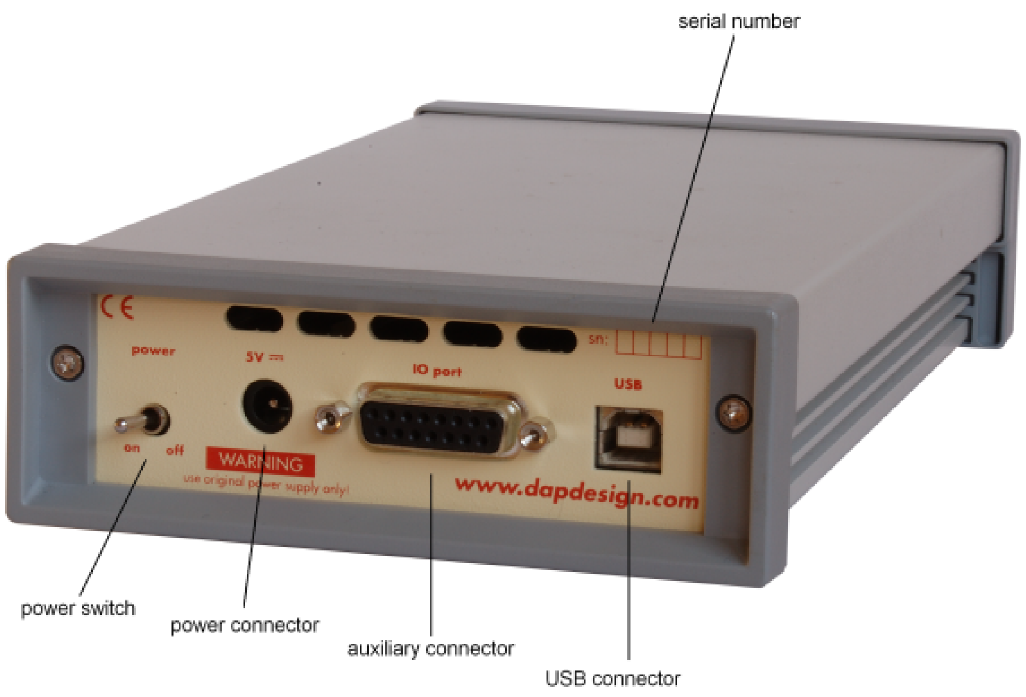
active led
This red led will light up when the FireSpy Generator is active. See Generator.

trigger led
This green led will light up when the FireSpy Recorder has been triggered. See Recorder.

manual trigger
Pressing this button will trigger the Recorder when recording is in progress and the Recorder is not yet triggered.

external trigger
Using this connector, the FireSpy can be triggered with an external electrical signal when recording is in progress and the Recorder is not yet triggered.
The FireSpy is triggered when it detects a positive pulse of minimal 3 Volt (resulting in a current of about 1.6 mA) and a duration of typical 10 uS.

24.6.3.2. FireSpy Rear



power switch
Using this switch the FireSpy can be switched on (powered) and off. When switched on the 'power led' on the front will light up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

power connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used.

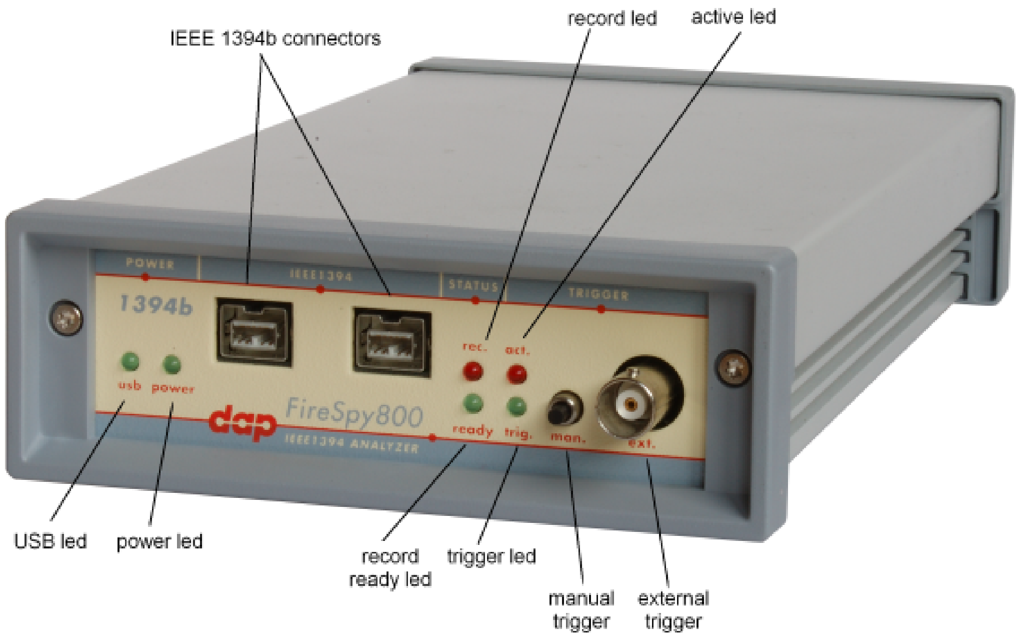
auxiliary connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

USB connector
The FireSpy must be connected to the computer using this connector. A USB cable, which is part of the FireSpy-package, is connected between this connector and the USB port of the computer.
The USB interface is electrically isolated from the IEEE1394 logic to prevent ground loops.

serial number
Each FireSpy has a 5 character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireSpy. See License Manager for more information on license keys.

24.6.4. FireSpy 800

24.6.4.1. FireSpy Front



USB led
This green led will light up when the FireSpy is connected to the USB port of your computer (see 'FireSpy 800 Rear' below) and the computer is switched on. It indicates the availability of USB bus power.

power led
This green led will light up when the power supply is connected to FireSpy (see 'FireSpy 800 Rear' below) and the FireSpy is switched on. It indicates the availability of power from the power supply.

IEEE1394bilingual connectors (2x)
With these connectors the FireSpy 800 can be connected to the IEEE1394 bus to be analyzed. One or both connectors may be connected to the bus.
Both connectors are bilingual. This means that they can be connected to a 1394b (beta) port or to a 1394a (legacy) port, by using the correct cables.

record led
This red led will light when recording is in progress. See Recorder.

record ready led
This green led will light up when the FireSpy Recorder is ready. See Recorder.

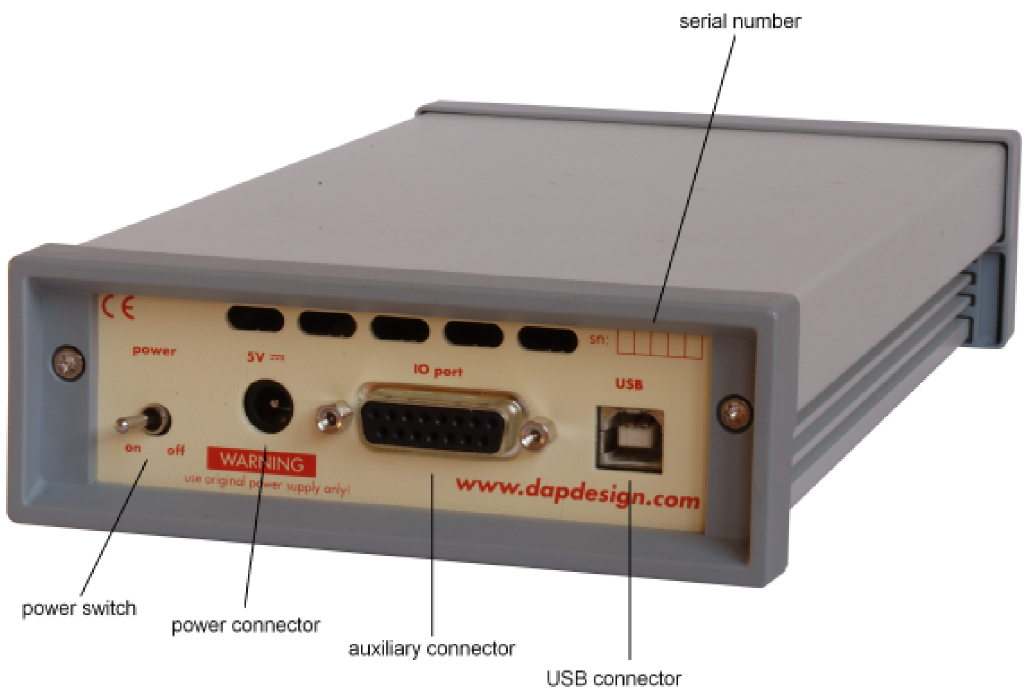
active led
This red led will light up when the FireSpy Generator is active. See Generator.

trigger led
This green led will light up when the FireSpy Recorder has been triggered. See Recorder.

manual trigger
Pressing this button will trigger the Recorder when recording is in progress and the Recorder is not yet triggered.

external trigger
Using this connector, the FireSpy can be triggered with an external electrical signal when recording is in progress and the Recorder is not yet triggered.
The FireSpy is triggered when it detects a positive pulse of minimal 3 Volt (resulting in a current of about 1.6 mA) and a duration of typical 10 uS.

24.6.4.2. FireSpy Rear



power switch
Using this switch the FireSpy can be switched on (powered) and off. When switched on the 'power led' on the front will light up. Note that the power supply needs to be connected to the FireSpy (see below) to be able to switch the FireSpy on.

power connector
The power supply must be connected to the FireSpy, using this connector. Note that, for safety reasons, only the original power supply should be used.

auxiliary connector
The auxiliary port will be discussed in a separate chapter. See Auxiliary connector ports.

USB connector
The FireSpy must be connected to the computer using this connector. A USB cable, which is part of the FireSpy-package, is connected between this connector and the USB port of the computer.
The USB interface is electrically isolated from the IEEE1394b logic to prevent ground loops.

serial number
Each FireSpy has a 5 character serial number. This number is also programmed into the FireSpy and can be read with the License Manager of the FireSpy application. Part of the software will only work when license keys are installed for the serial number of the currently connected FireSpy. See License Manager for more information on license keys.

24.7. Auxiliary connector ports

All FireSpy devices have one or more auxiliary connector ports.
The signals on these ports are:

- A number of aux signals
- Ground (for aux signals and Power if available)
- Power (not on IDC connector)

The aux signals are all open drain CMOS output with a pullup resister to 3.3 Volts. To prevent a short it is best to only pull these signals low and allowing the internal (or an extra external pullup) resistor to pull the signal high.
When you are sure some aux signal is only used as input, it may be pulled high actively, but it may never be pulled higher then 3.7 Volts.
The current that may flow into an aux pin when driven low by the FireSpy is maximum 12 mA.

All aux signals can be controlled by the Scriptor and for some of the aux signals special functions can be selected.
See the description of the 'External Ports' tab of the 'Settings Dialog' for the possible functions.

The Voltage of the Power signal and the current that can be drawn from it is FireSpy type dependant.
The table below shows the Voltage and the maximum current that can be drawn from the Power pin.

FireSpy type	**Voltage**	**Max current all ports together**
FireSpy400b	5 +- 10%	100mA (SUBD port)
FireSpy800	5 +- 10%	100mA (SUBD port)
FireSpy410b	12 +- 10%	100mA (SUBD port)
FireSpy810	12 +- 10%	100mA (SUBD port)
FireSpy381x	12 +- 10%	200mA (SUBD and VHDCI ports together)
FireSpy385x	12 +- 10%	200mA (VHDCI port)

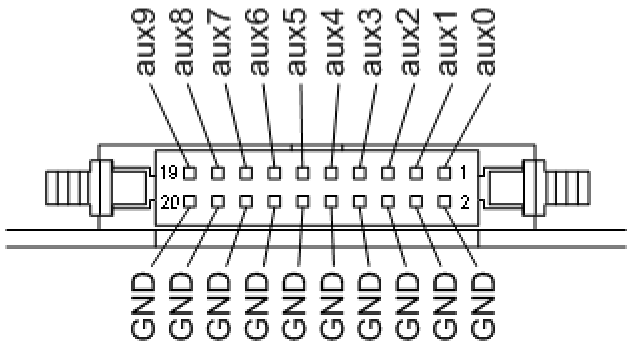
Note that the Power on the VHDCI connector can be switched off. See the General tab of the Settings Dialog.

Each aux port can be configured to have an active /TriggerOut pin. (This can be configured in the settingsdialog.) When the recorder is triggered, this pin will be active low for a duration of about 160nS.

24.7.1. The IDC connector

The 20 pin IDC connector is available for the FireSpy PCI cards.

It has 10 aux signals.



This 20 pins connector has the following signals:

Pin 1	aux0
Pin 2	Ground

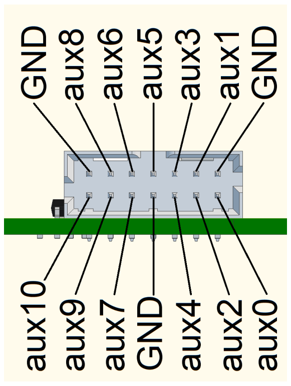
Pin 3	aux1
Pin 4	Ground
Pin 5	aux2
Pin 6	Ground
Pin 7	aux3
Pin 8	Ground
Pin 9	aux4
Pin 10	Ground
Pin 11	aux5
Pin 12	Ground
Pin 13	aux6
Pin 14	Ground
Pin 15	aux7
Pin 16	Ground
Pin 17	aux8
Pin 18	Ground
Pin 19	aux9
Pin 20	Ground

For a description of the aux signals, see above.

24.7.2. The IBC connector

The 14 pin IBC connector is available for the FireSpy PCIx cards.

It has 11 aux signals.



This 14 pins connector has the following signals:

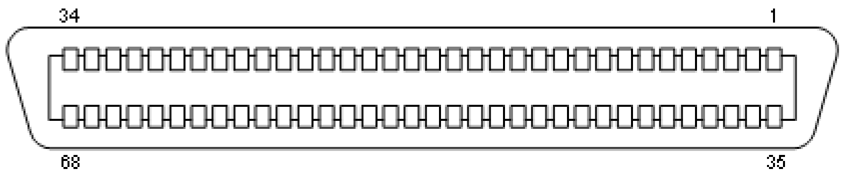
Pin 1	GND
Pin 2	aux2
Pin 3	aux6
Pin 4	aux9
Pin 5	aux10
Pin 6	aux1
Pin 7	aux3
Pin 8	GND
Pin 9	aux5
Pin 10	aux8
Pin 11	aux4
Pin 12	aux0
Pin 13	GND
Pin 14	aux7

For a description of the aux signals, see above.

24.7.3. The VHDCI Connector

The VHDCI connector is available for the FireSpy Triple Series.

This connector has 6 transformer coupled IEEE1394b ports. Two for each node in the FireSpy. These Ports are A0 and A1 for the first node, B0 and B1 for the second node and C0 and C1 for the third node. This connector also holds 6 aux signals and optional power. Note that the power can be switched on and off using the Settings Dialog.



VHDCI Pins	Name	Port
Pin 1	Not used	
Pin 2	Cable shield	
Pin 3	C1-TPA+	Port C1
Pin 4	Not used	
Pin 5	C1-TPB+	Port C1
Pin 6	Cable shield	
Pin 7	C0-TPA+	Port C0
Pin 8	Not used	
Pin 9	C0-TPB+	Port C0
Pin 10	Cable shield	
Pin 11	B1-TPA+	Port B1
Pin 12	Not used	
Pin 13	B1-TPB+	Port B1
Pin 14	Cable shield	
Pin 15	Ground	CONTROL
Pin 16	Ground	CONTROL
Pin 17	Aux4	CONTROL
Pin 18	Aux2	CONTROL
Pin 19	Aux0	CONTROL
Pin 20	Cable shield	
Pin 21	Power	CONTROL
Pin 22	Cable shield	
Pin 23	B0-TPA+	Port B0
Pin 24	Not used	
Pin 25	B0-TPB+	Port B0
Pin 26	Cable shield	
Pin 27	A1-TPA+	Port A1
Pin 28	Not used	
Pin 29	A1-TPB+	Port A1
Pin 30	Cable shield	
Pin 31	A0-TPA+	Port A0
Pin 32	Not used	
Pin 33	A0-TPB+	Port A0
Pin 34	Cable shield	
Pin 35	Not used	
Pin 36	Cable shield	
Pin 37	C1-TPA-	Port C1
Pin 38	Not used	
Pin 39	C1-TPB-	Port C1
Pin 40	Cable shield	
Pin 41	C0-TPA-	Port C0
Pin 42	Not used	
Pin 43	C0-TPB-	Port C0

Pin 44	Cable shield	
Pin 45	B1-TPA-	Port B1
Pin 46	Not used	
Pin 47	B1-TPB-	Port B1
Pin 48	Cable shield	
Pin 49	Ground	
Pin 50	Ground	
Pin 51	Aux5	CONTROL
Pin 52	Aux3	CONTROL
Pin 53	Aux1	CONTROL
Pin 54	Cable shield	
Pin 55	Power	
Pin 56	Cable shield	
Pin 57	B0-TPA-	Port B0
Pin 58	Not used	
Pin 59	B0-TPB-	Port B0
Pin 60	Cable shield	
Pin 61	A1-TPA-	Port A1
Pin 62	Not used	
Pin 63	A1-TPB-	Port A1
Pin 64	Cable shield	
Pin 65	A0-TPA-	Port A0
Pin 66	Not used	
Pin 67	A0-TPB-	Port A0
Pin 68	Cable shield	

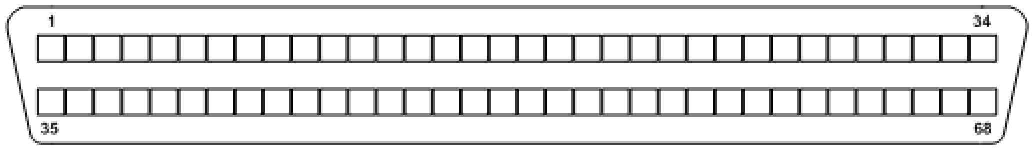
Note that the metal parts of the connector are connected to 'Cable shield'.

For a description of the aux signals and the Power signal, see above.

24.7.4. The SCSI2 Connector

The SCSI2 connector is available for the FireSpy 3470

This connector has 9 transformer coupled IEEE1394b ports. Three for each node in the FireSpy. These Ports are A0, A1 and A2 for the first node, B0, B1 and B2 for the second node and C0, C1 and C2 for the third node.



SCSI2 Pins	Name	Port
Pin 1	Cable shield	
Pin 2	TPA+	Port A2
Pin 3	TPA-	Port A2
Pin 4	Cable shield	
Pin 5	TPA+	Port A1
Pin 6	TPA-	Port A1
Pin 7	Cable shield	
Pin 8	TPA+	Port A0
Pin 9	TPA-	Port A0
Pin 10	Cable shield	
Pin 11	VCC12	
Pin 12	VCC12	
Pin 13	Cable shield	
Pin 14	TPA+	Port B2
Pin 15	TPA-	Port B2

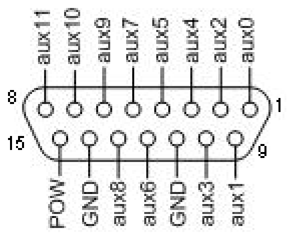
Pin 16	Cable shield	
Pin 17	TPA+	Port B1
Pin 18	TPA-	Port B1
Pin 19	Cable shield	
Pin 20	TPA+	Port B0
Pin 21	TPA-	Port B0
Pin 22	Cable shield	
Pin 23	VCC12	
Pin 24	VCC12	
Pin 25	Cable shield	
Pin 26	TPA+	Port C2
Pin 27	TPA-	Port C2
Pin 28	Cable shield	
Pin 29	TPA+	Port C1
Pin 30	TPA-	Port C1
Pin 31	Cable shield	
Pin 32	TPA+	Port C0
Pin 33	TPA-	Port C0
Pin 34	Cable shield	
Pin 35	Cable shield	
Pin 36	TPB+	Port A2
Pin 37	TPB-	Port A2
Pin 38	Cable shield	
Pin 39	TPB+	Port A1
Pin 40	TPB-	Port A1
Pin 41	Cable shield	
Pin 42	TPB+	Port A0
Pin 43	TPB-	Port A0
Pin 44	Cable shield	
Pin 45	Ground	
Pin 46	Ground	
Pin 47	Cable shield	
Pin 48	TPB+	Port B2
Pin 49	TPB-	Port B2
Pin 50	Cable shield	
Pin 51	TPB+	Port B1
Pin 52	TPB-	Port B1
Pin 53	Cable shield	
Pin 54	TPB+	Port B0
Pin 55	TPB-	Port B0
Pin 56	Cable shield	
Pin 57	Ground	
Pin 58	Ground	
Pin 59	Cable shield	
Pin 60	TPB+	Port C2
Pin 61	TPB-	Port C2
Pin 62	Cable shield	
Pin 63	TPB+	Port C1
Pin 64	TPB-	Port C1
Pin 65	Cable shield	
Pin 66	TPB+	Port C0
Pin 67	TPB-	Port C0
Pin 68	Cable shield	

Note that the metal parts of the connector are connected to 'Cable shield'.

24.7.5. The SUBD connector

The 15 pin SUBD connector is available for the external FireSpy units.

It has 12 aux signals and power.



This 15 pin SUBD connector has the following signals:

Pin	**Name (Basic & Advanced)**	**Name (Gen 4)**
1	aux0	aux0
2	aux2	aux2
3	aux4	aux4 (IRIG1)
4	aux5	aux5 (IRIG2)
5	aux7	aux7 (TIM2)
6	aux9	RS232 Receive
7	aux10	aux10 (VS-Sync)
8	aux11	RS232 Transmit
9	aux1	aux1
10	aux3	aux3 (TIM1)
11	Ground	Ground
12	aux6	aux6 (IRIG3)
13	aux8	aux8 (TIM3)
14	Ground	Ground
15	Power	Power

For a description of the aux signals and the Power signal, see above.