Goebelyzer Analyzer

User Manual

The Goebel Company

Revision: 1.0 **July 20, 2010**

Purpose

This manual describes the software for the Goebelyzer System offered by The Goebel Company. This includes all operation of the analyzer normally performed by a user.

Notice

Information in this manual has been carefully reviewed and is believed to be accurate. The Goebel Company shall not be liable for errors contained herein. The Goebel Company reserves the right to make changes or additions to the software described herein.

Contact

For technical or other inquiries contact:

email: info@GoebelEtc.com

The Goebel Company
12486 Prowell
Leavenworth WA 98826
USA
Phone: 206-601-6010

Table of Content

1. Introduction

This manual discusses topics that are unique to the Goebelyzer System. Most other Goebelyzer functionality will be as described in the ethereal user's guide. The ethereal user's guide should be the first place to search for information. This document will be a supplement to the ethereal user's guide.

The AFDX,A429, P2P, CAN, and ALLDEV interfaces are unique to the Goebelyzer System. The steps necessary to capture data on these interfaces will include special options/conditions to perform a capture described in the guide.

This document will start with a quick description of the GUI layout including the terms and layout. Next an outline for performing a capture. Sections on capture options for each interface, and information pages displayed during a live capture. Finally, a chapter on various interface data file formats.

The Goebelyzer works with many unique avionics interfaces and provides the ability to defined the format of the payload data received on an interface. This functionality will allow the user to create meaningful names and display data in formats as described in an Interface Control Document.

2. Applicable Documents/Web Sites

The following lists contain information that is useful in learning to use the Goebelyzer System.

1. Ethereal User Guide – V2.0.2 for Ethereal 0.10.12 (or latest) written by Richard Sharpe, Ed Warnicke, and Ulf Lamping. Http://www.ethereal.com/docs
2. Capture Filters. – Defines the syntax of capture/trigger filters http://wiki.ethereal.com/CaptureFilters, .
3. Mike Horn Capture Filter Tutorial. Provides a good tutorial on use of capture filters. http://home.insight.rr.com/procana
4. Mark 33 Digital Information Transfer System (DITS) Part 1, ARINC Specification 429P1-15 Published September 1, 1995.
5. SJA1000 Data Sheet 2000 Jan 04 Philips Semiconductor.

3. Quick GUI Layout.

This section is a quick description of some terms and the layout of the GUI. The ethereal document contains more information on the GUI at http://www.ethereal.com/docs.

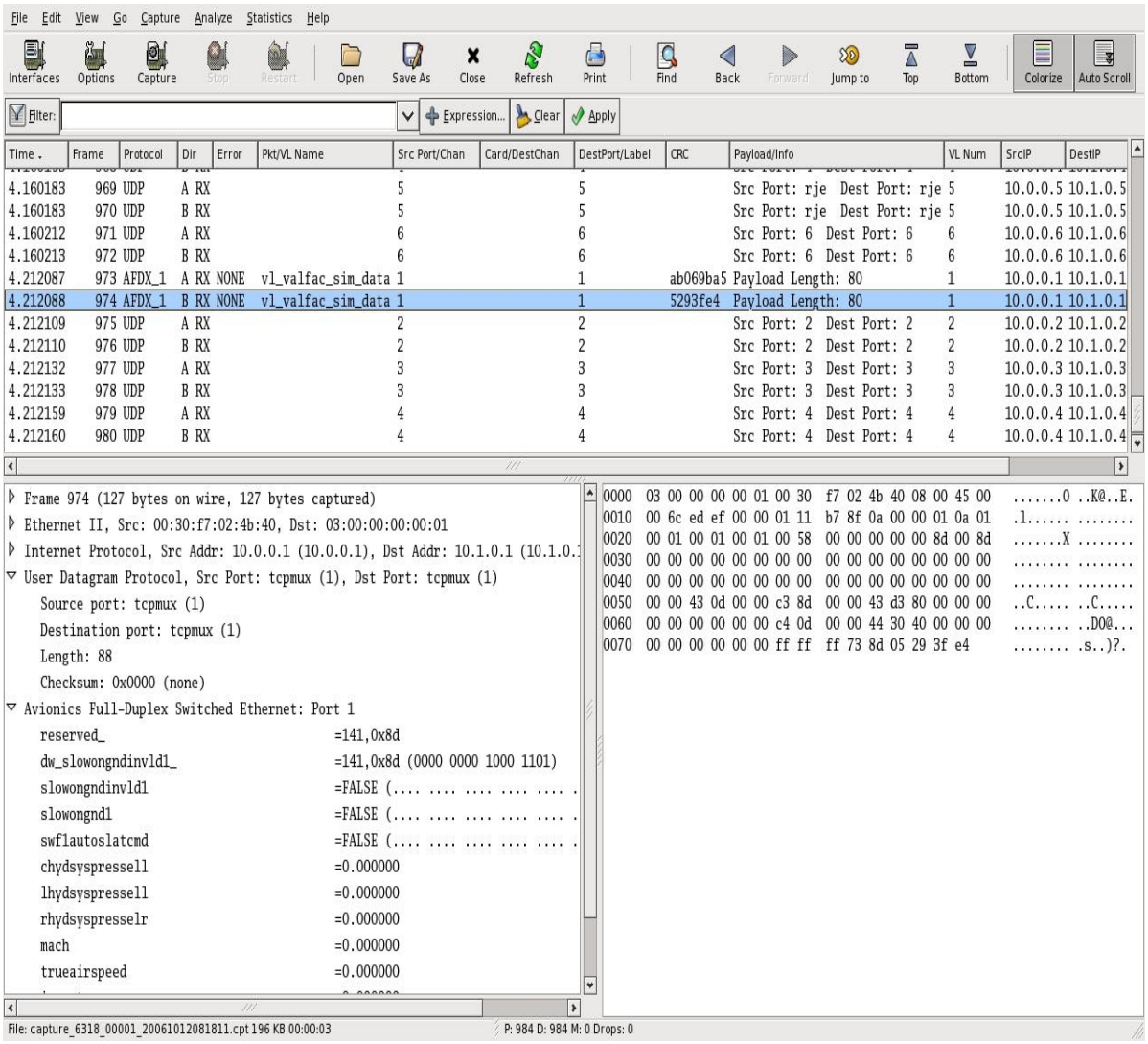
At the top, is a menu bar. It has pulldowns for File (read/write capture files), Edit (various packet find/mark operations), View (what is displayed in the GUI), Go (go to a specific packet), Capture (start/stop capture), and Analyze (enable/disable protocols, create display filters). The next layer down is the toolbar, it contains buttons to perform commonly executed functionality. The most used buttons are Options (display options menu), Capture (Execute a capture), Stop (Stop Execution of a capture). These toolbar buttons correspond to operations in the menu bar but are more convenient to the user.

The next hortizontal bar is used to create/execute display filters. Display Filters are used to eliminate packets not matching the result of the display filter boolean expression. For Example: only display packets with vl=1212 (eth.vl==1212) or udp destination port=2323 (udp.dstport==2323) or both conditions (eth.vl==1212 and udp.dstport==2323) are displayed with these filters. This will allow the user to only view packet meeting the defined condition. The expression button will display a dialog used to create expression conditions. Display filters are described in more detail in a later chapter and the ethereal user's guide at http://www.ethereal.com/docs.

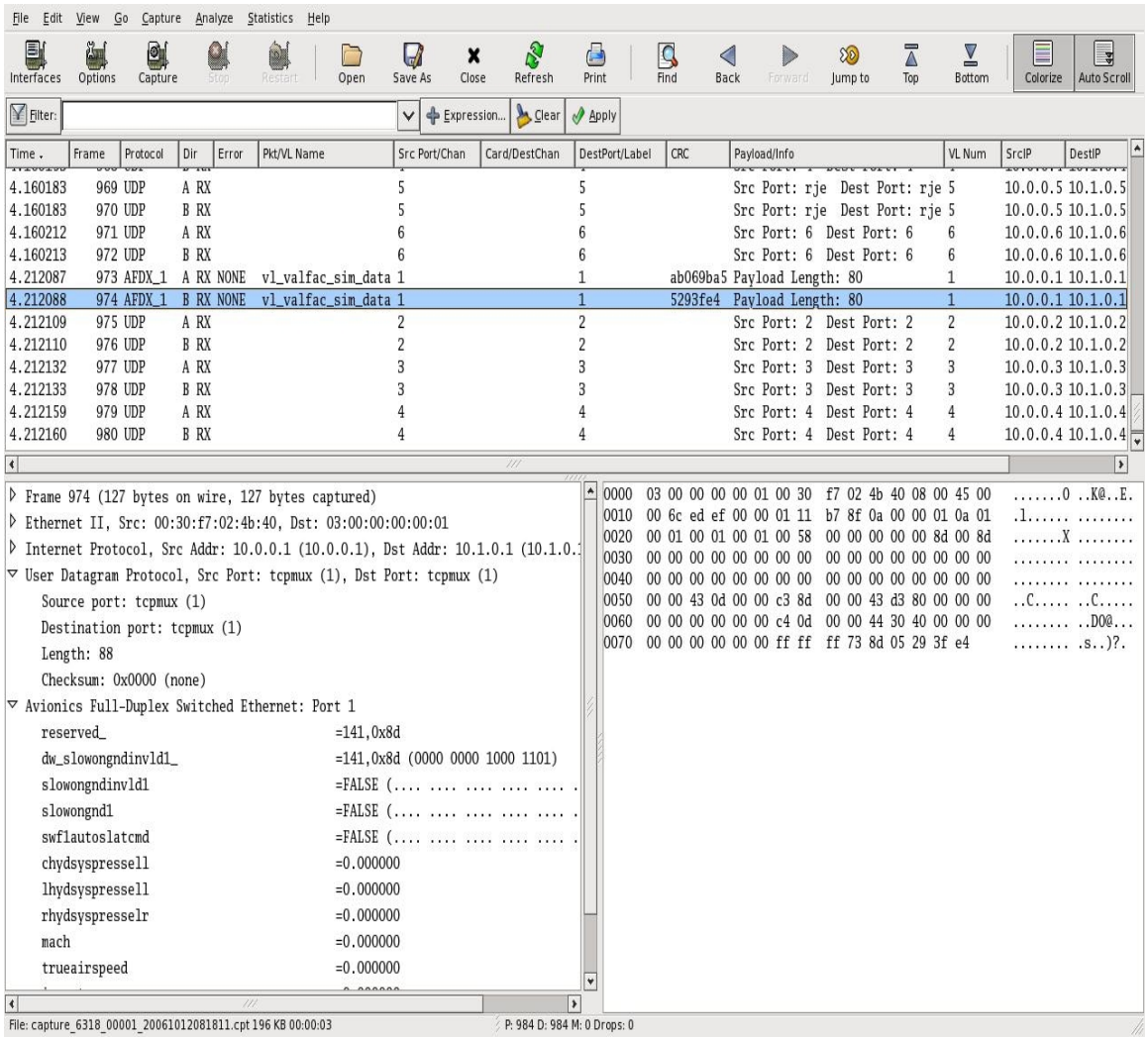
The next window section down is the packet display list. It is the list of packets received on the device sorted in time with various columns to provide unique information related to the packet and interface. The columns may include data such as time, packet names, port numbers, or IP addresses.

On the lower left is the packet tree window pane. When a packet is selected in the packet list, the packet tree will display a list of protocols contain in the packet. Each protocol can be expanded to view all the data items of the protocol. Using a data definition file, the user can define payload data that will be displayed and formatted in this pane.

On the lower right is the byte dump pane. When a packet is selected, the pane displays a raw byte dump of all data contained in the packet as shown for the highlighted packet.



Perform a Data Capture.



Starting the Analyzer. The default preferences of the interface will be set by double clicking on the desktop icon corresponding to the interface. There should be icons for all hardware interface devices on the system desktop. By starting execution of the analyzer in this way, the capture interface will be set, the packet viewing columns pertaining to the interface are displayed, and default interface options set.

There is a common sequence of events to performing data captures. Once the interface and options are set, the user only needs to depress the *Capture* button to restart a capture.

- Select/Verify correct interface on the general options (select menu item *Capture->Options* or depressing toolbar item *Options*) page. This is described in #5.1General Options Notebook Page.|outline
- Setup Capture Options for Interface. This is described in #5.2AFDX Options Page.|outline for AFDX, #5.3P2P Options Page.|outline for P2P, #5.4A429 Options Page.|outline for A429, and #5.5Alldev Options Page.|outline for ALLDEV .
- Start Capture by selecting menu item *Capture->Start* or depressing toolbar item *Capture.*
- Let capture execute. View any real-time packet/payload information Described in #5.6Real-Time Display Option Pages.|outline and #6.Active Data Capture.|outline.
- Stop Capture by selection menu item *Capture->Stop* or depressing toolbar item *Stop,* or when a trigger condition has been met.
- View packet trace data. Possibly using a display filter #7.Digital Filters.|outline to display only packets of interest to the user.

4. Preparing Capture Options.

The analyzer contains various interfaces which have special capture options. Theses options are contained in the notebook pages (located at the top of the dialog) of each interface. Options pages are only displayed for interface hardware contained in the system.

The following sections will describe the available options for each interface. The user must configure the interface options before the capture is started. Incorrect or no capture data is usually the result of incorrect capture options or setup. But for standard analyzer usage, the options should be set by default at startup.

4.1 General Options Notebook Page.

The general options page contains options that are not specific to an interface and general to the capture process.

4.1.1 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore value to before opening options dialog.
- Help. Depress the help button will display the help dialog. The help dialog tries to provide useful information to the user.

4.1.2 Capture Files Frame:

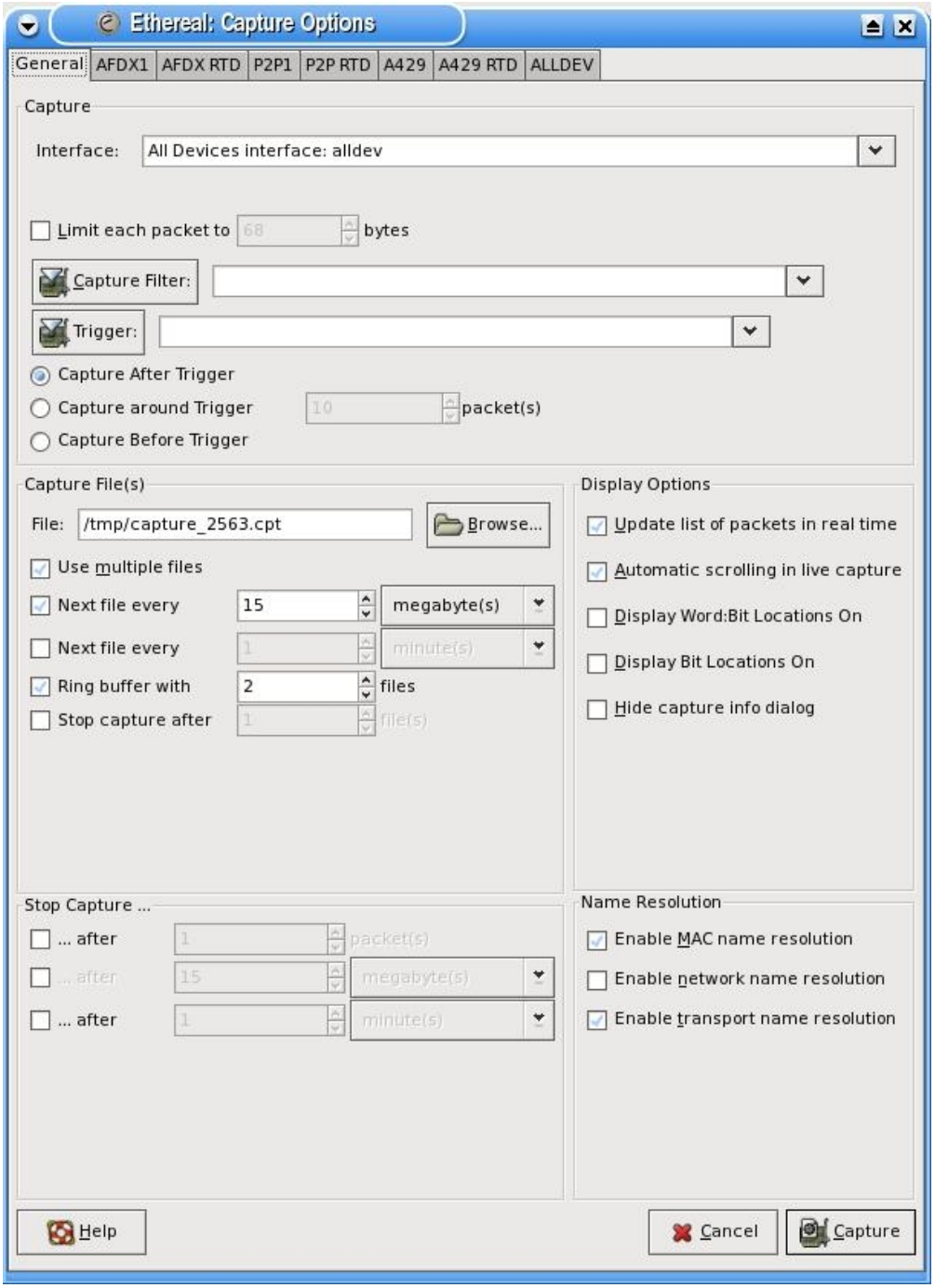
- File Entry. This entry field will allow the user to specify the name of the capture file. The default is */tmp/capture_<process_id>.cpt*. The process id is used to make each filename unique.
- Use Multiple Files. This option will allow the capture output to be saved in multiple files with limited size. When a file has exceeded defined storage, the system will move to the next file in a circular queue.

  When a single file is used for storage, the file can grow to a size larger than the available memory space of the computer. When this occurs, the system performance will produce poor response displaying capture data. With the default of 2 ring buffer file of 15MB, the analyzer can run for a longer capture intervals with no performance hits.
- Next File every X megabytes. This option will specify the amount of data stored in each file with the multiple files option enabled. This is the default with a value of 15M.
- Next File every X minutes. This option will specify the amount of time packet data is stored in each file with the multiple files option enabled.
- Ring buffer with X Files. X number of files contained in a file ring buffer (when the last file is completed, the analyzer will return the first file) with the multiple files option enabled. This option is the default with a value 2 files.
- Stop capture after X files. Stop capturing of data when X files of data have been stored. Used with the multiple files option enabled.

4.1.3 Display Options Frame.

- Update List of Packets in Real Time. This option will update the packet list pane in real time. Otherwise the packet list pane will be updated after the capture has been stopped. Using this option with a display filter at capture start, will only display packet matching the display filter in semi real time fashion.

- Automatic Scrolling in live capture. This option will scroll the packet list pane in real time. It is only available when the Update List of Packets in Real Time option is selected. Otherwise the packet list pane will display the 1st bundle of packets received.
- Display Word:Bit Locations On. This option will display the location data in the packet tree view of each payload element. The format of the data is StartWord:StartBit..EndWord:EndBit.



- Display Bit Locations On. This option will display the location data in the packet tree view of payload elements. The format of the data is StartBit..EndBit.
- Hide Capture Info Dialog. This option will allow the user to hide the capture info dialog.

4.1.4 Stop Capture Frame.

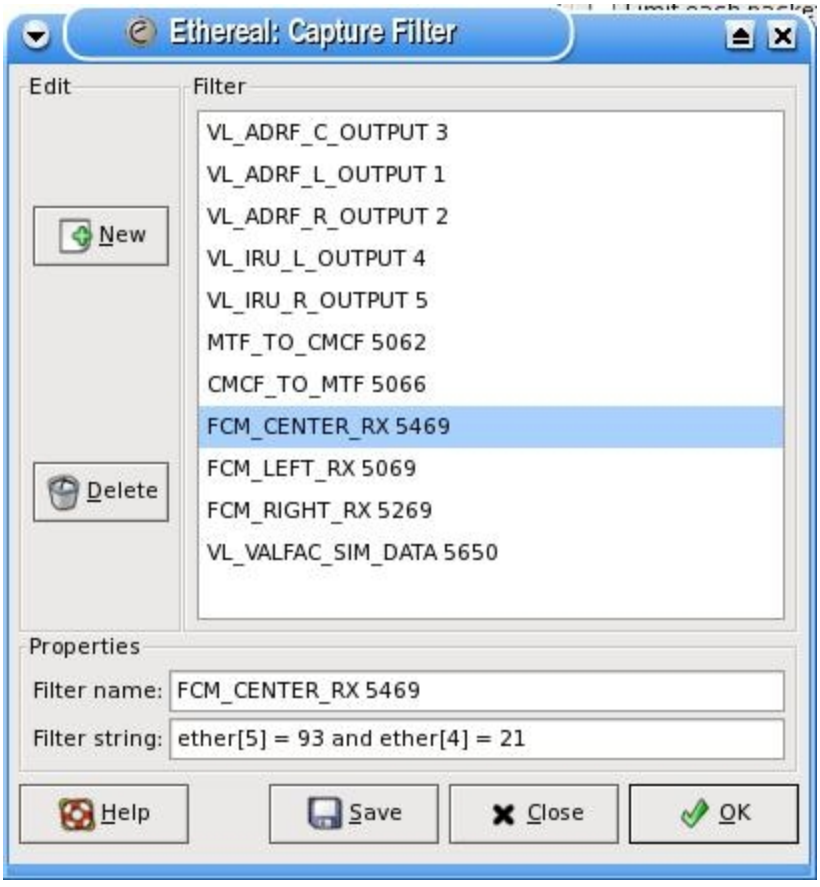
The Stop Capture Frame contains options for automatic capture completion.

- After X packets. The user can stop the capture after a specific number of packets.
- After X Megabytes. The user can stop the capture after a specified amount of disk space has been stored.
- After X minutes. The user can stop the capture after a specified amount of time has past.

If no options are enabled, the capture will need to be stopped by completion of a trigger event or depress the toolbar item *Stop* or menu item *Capture->Stop*.

4.1.5 Capture Frame.

- Interface Pulldown. The interface pulldown will allow the user to select an interface. At this time, the available interfaces are *afdx*, *p2p*, *a429*, *can* and *alldev*. The interface will be displayed in the pulldown when hardware for an interface is contained in the system. The analyzer will allow for multiple interfaces of the same type with a board number appended to the end of the name. The *alldev* is a pseudo interface that will receive data from all physical interfaces.
- Limit each packet to X bytes. This option will limit the size of a received packet to X bytes.
- Capture Filter Button. The Capture Filter button will open a dialog box as shown below. The dialog box will allow the user to create or select a saved capture filter. The capture filters are saved in the *cfilter* file contained in the analyzer configuration directory. When a capture filter is selected, it will be displayed in the capture filter pulldown. The capture filter pulldown will contain the active analyzer capture filter. If the capture filter entry is an empty string, this will allow all packets through the filter.



A new capture is created by depressing the *New* button and entering a capture name and actual capture Berkeley Packet Filter (BPF) string. The *Delete* button is used to remove the selected capture filter. The capture filter data is saved to file with the *Save* button. The *OK* button will only keep the data for the length of this capture session.

- Capture Filter Entry. Capture Filters are used to filter out unnecessary packets at capture time. This is done to reduce the size of the resulting capture output file and is especially useful on high traffic networks or for long term capturing. The analyzer uses the Berkeley Packet Filter language to describe capture filters. This language syntax is explained in the tcpdump man page (http://www.tcpdump.org) and http:/home/insight.rr.com/procana. Note: This capture filter language is different from the one used for the analyzer display filters.
- AFDX Capture Filter Syntax: The following is a short description of the capture filter language syntax. A capture filter takes the form of a series of primitive expressions, connected by conjunctions (and/or) and optionally preceded by not:

[not] primitive [and|or [not] primitive ...]

A primitive is simply one of the following:

[src|dst] host <host>

This primitive allows you to filter on a host IP address. You can optionally precede the primitive with the keyword src|dst to specify that you are only interested in source or destination addresses. If these are not present, packets where the specified address appears as either the source or the destination address will be selected.

ether [src|dst] host <ehost>

This primitive allows you to filter on Ethernet host addresses. You can optionally include the keyword src|dst between the keywords ether and host to specify that you are only interested in source or destination addresses. If these are not present, packets where the specified address appears in either the source or destination address will be selected.

[tcp|udp] [src|dst] port <port>

This primitive allows you to filter on TCP and UDP port numbers. You can optionally precede this primitive with the keywords src|dst and tcp|udp which allow you to specify that you are only interested in source or destination ports and TCP or UDP packets respectively. The keywords tcp|udp must appear before src|dst. If these are not specified, packets will be selected for both the TCP and UDP protocols and when the specified address appears in either the source or destination port field.

ether|ip|udp|afdx [offset : 1|2|4]

This primitive will point to the specific area of the buffer. With the offset used to index into the ether,ip,udp,afdx area with a size 1 (byte), 2 (short word), or 4 (long word). Default size is 1.

less|greater <length>

This primitive allows you to filter on packets whose length was less than or equal to the specified length, or greater than or equal to the specified length, respectively.

<expr> relop <expr>

This primitive allows you to create complex filter expressions that select bytes or ranges of bytes in packets. Please see the tcpdump man pages for more details.

AFDX Examples:

ether[4:2]==12345 – capture all traffic on vl 12345.

host 192.168.0.10 -- capture all traffic to and from the IP address 192.168.0.10.

host dst 192.168.0.10 -- only destination address.

host src 192.168.0.10 -- only source address.

udp port 5280 -- capture all traffic to and from the UCP port 5280.

udp src port 5280 -- only with source port of 5280.

udp dst port 5280 -- only with destination port of 5280.

ip[2] == 0x10 -- will capture if 2nd byte in IP header is 0x10.

ip[10] == 0x10 -- will capture if 10th byte in IP header is 0x10.

Verify the value of data in the UDP payload use the **afdx** primitive.

afdx[0] == 0x10 -- will capture if 1st byte of payload is 0x10.

afdx[0:2] == 0x2000 -- will capture if 1st short word of payload is 0x2000.

afdx[0:4] == 0x20002000 -- will capture if 1st long word of payload is 0x20002000.

ether[4:2]==12345 && udp port 5280 – capture all traffic with vl of 12345 and udp port of 5280.

udp port 5280 and ip[2] == 0x10 and (afdx[0] == 0x10 or afdx[4:4] < 12) -- and or used to make more complex expressions.

afdx[0] == 0x10 and afdx[2:2] != 0x2000 and afdx[4:4] < 0x10. capture data when the (payload byte 0 is 0x10) AND (short word at payload byte 2 is 0x2000) AND (long word at payload byte 4 is less than 0x10).

P2P Capture Filter Syntax: The following is a short description of the p2p capture filter language syntax. A capture filter takes the form of a series of primitive expressions, connected by conjunctions (and/or) and optionally preceded by not:

[not] primitive [and|or [not] primitive ...]

The only primitive for the P2P interface is **P2P** .

p2p [offset : 1|2|4]

This is the only primitive. It will point to the beginning of the buffer. With the offset used to index into the buffer with a size 1 (byte), 2 (short word), or 4 (long word). Size will default to 1 if not written.

p2p[8] == 0x10 and p2p[10:2] != 0x2000 and p2p[12:4] < 0x10

This example will capture when p2p byte 8 is 0x10 and short word at byte 10 is

0x2000 AND long work at byte 12 is less than 0x10.

P2P Examples:

p2p[0:2] == 5280 -- capture all traffic to and from the port 5280.

p2p[2:2] == 8 -- capture all traffic with a length of 8

Verify the value of data in the P2P payload, payload data starts at byte 4. The message payload is located at P2P[4]. The P2P header contains 4 bytes. NOTE: a payload data item located at byte 4 would use a P2P[8] location in the filter.

p2p[8] == 0x10 -- will capture if 1st byte of payload is 0x10

p2p[8:2] == 0x2000 -- will capture if 1st short word of payload is 0x2000

p2p[8:4] == 0x20002000 -- will capture if 1st long word of payload is 0x2000

p2p[0:2] == 5280 and p2p[2] == 0x10 and (p2p[8] == 0x10 or p2p[12:4] < 12) -- and or used to make more complex expressions

A429 Capture Filter Syntax: The A429 capture filter syntax is same as the p2p with keyword p2p changed to a429.

A429 Examples:

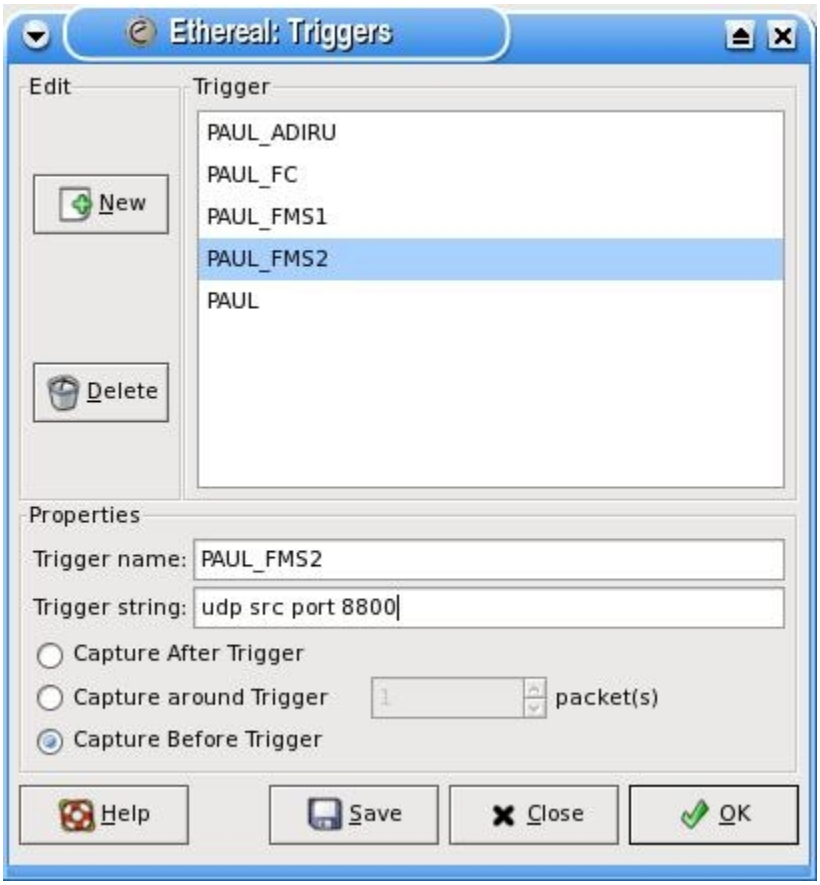
a429[3:1] == 0123 -- capture all traffic with octal label 123.

a429[3]==0333 && ((a429[0:4]&0x3000000) == 0x1) – capture all traffic with label of octal 333 and ssm of 1.

a429[3]==0333 && ((a429[0:4]&0x300) == 0x1) – capture all traffic with label of octal 333 and sdi of 1.

a429[3]==0333 && ((a429[0:4]&0x400) == 0x1) – capture all traffic with label of octal 333 and bit 11 set to 1.

Capture Trigger Button. The Capture Trigger button will open a dialog box as shown below. The dialog box will allow the user to create or select saved capture triggers. The capture triggers are saved in the *trigger* file contained in the analyzer configuration directory. When a capture trigger is selected it will be displayed in the capture trigger pulldown. The pulldown will contain the active analyzer capture trigger. The operations of the trigger dialog are similar to the capture filter.#5.1.2Capture Files Frame:|outline



- Capture Trigger Entry. Trigger on specific packet to start/stop the capture. Triggers are used to capture packets with respect to a trigger event (after trigger packet, around trigger packet, before trigger packet. The analyzer uses the BPF language for triggers. This language is explained in the tcpdump man page http://www.tcpdump.org or http://home.insight.rr.com/procana. Note: This trigger language is different from the one used for the analyzer display filters. But the same as capture filters described in previous sections.

- Trigger Event Type.

AROUND , AROUND NUMBER: selection of the AROUND trigger type, will capture packets waiting for the trigger event to occur. When the trigger event has occurred, the trigger packet is saved and the "AROUND NUMBER" of capture packets are saved. The capture will be stopped when the "AROUND NUMBER" packet is received.

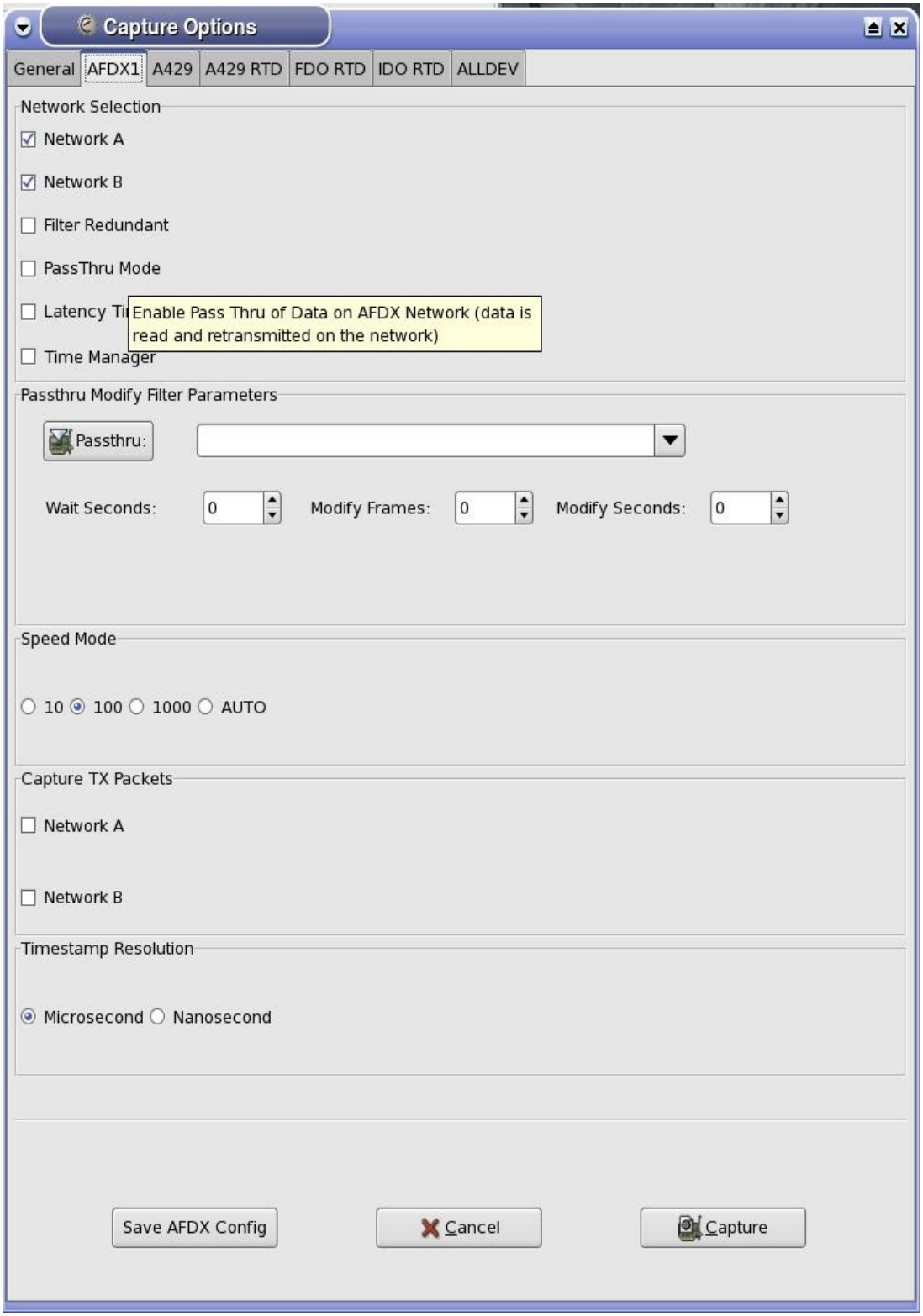
AFTER: selection of the AFTER trigger type, will wait for the AFTER trigger event. When the event is triggered, all capture packets are saved including the trigger packet event. The capture will continue running and the operator must stop the capture.

BEFORE: selection of the BEFORE trigger type, will capture packets waiting for the BEFORE trigger event. When the BEFORE trigger event is decoded, this capture packet is saved and the capture is stopped.

- Trigger Syntax. The trigger language syntax is the same as the capture filter syntax described above except that a event type must also be selected.

4.2 AFDX Options Page.

This page contains options that are specific to the afdx interface. The notebook pages are named AFDX1..AFDXN where N is the number of afdx interface board in the system.



4.2.1 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.

- Save AFDX Config. The Save AFDX Conf button will save the state of channel and device configuration data for the duration of the program execution. After the AFDX configuration is saved, these will become the default AFDX settings when the dialog is opened in the future. Permanent options can be save to the preferences file using the *Edit->Preferences->AFDX HW* option. The file is located in the analyzer configuration directory.

4.2.2 Network Selection Frame.

- Network A. This option will select Network A data to be received by the analyzer. This option is set by default.
- Network B. This option will select Network B data to be received by the analyzer. This option is set by default.
- Filter Redundant. This option will keep only the first physically received packet from the A and B networks. The later packet will be thrown away. When filter redundant is selected, the Network A and B buttons will be forced to the off state. Also, Filter Redundant operation is exclusive (only one of the 3 button can be set) of Passthru and Latency Timing.
- Pass Thru Mode. This option will enable the operation of the passthru mode. An analyzer will be connected inline with an LRU. Packets received on the A network port will be output on the B network port. The opposite will also true. This will allow for a sniffer type operation to verify all data going in and out of an LRU. The passthru filter is described below will allow for a packet data item to be changed during the passthru of data from the A to B networks ports.
- Latency Timing. TBD.
- Time Manager. TBD.

4.2.3 Passthru Modify Filter Parameters Frame.

- Passthru Button. The Passthru Button will open a dialog box shown as below. The dialog box will allow the user to create or select saved passthru filters. The passthru filters are saved in the *passthru* file contain in the analyzer configuration directory. When a passthru is selected it will be displayed in the passthru pulldown. The pulldown will contain the active analyzer passthru filter. The operations of the passthru dialog are similar to the capture filter dialog

- Pass thru Entry. Passthru Filters are used to filter on specific packets and update data elements in the packet for retransmission. This could be useful to create errors in transmission of data on the physical interface. The syntax of the passthru expression is:

<bpf expr> modify <passthru assignment>

The passthru expression <bpf expr> uses the same primitives as described in capture filter/trigger section #4.1.5Capture Frame.|outline. So if the <bpf expr> contains a TRUE value, the passthru assignment <passthru assignment> will be executed. If FALSE, no assignment is performed. This will allow the user to be very specific when selecting a packet for passthru assignment.

The format of the passthru assignment is:

ether | ip | udp | afdx [pkt offset : 1|2|4] := expr | value

Passthru Syntax Examples:

ether[4:2]==1500 modify afdx[0:4] := afdx[0:4] | 0x7 – If vl is 1500, set first long word of payload to afdx[0:4] | 7.

udp port 5280 modify afdx[10:2] := 10 – If UDP port is 5280, set value of 10 at short word offset 10 byte from afdx payload.

udp src port 5280 modify udp[20:4] := -0x12345678 passthru udp[10:4] := 0x12345678 – If UDP source port is 5280, set value of 0x12345678 at long word offset 20 bytes from UDP header and set value of 0x12345678 at long word offset 10 bytes from UDP header..

udp port 5280 and udp[12] == 0x10 and udp[16:4] < 12 modify udp[44:2] := 10 – If UDP port is 5280 and data at UDP[12] is 0x10 and long word at UDP[16 is less than 12, set short word at UDP[44] to value of 10.

Note: use afdx[] to access payload data.

- Wait Seconds. This will delay the execution of the passthru modify by the number of seconds specified.
- Modify Frames. This spin button will specify the number of frames the modify passthru filter will set the requested data.
- Modify Seconds. This spin button will specifiy the number of seconds the modify passthru filter will set the requested data.

If both the modify frames and modify seconds parameters are specified, the filter will stop when the first condition is meet. Also, setting the modify values to zero, will have the modify filter execute for the length of the capture.

4.2.4 Speed Mode Frame.

A selection of radio buttons allows the user to pick the speed of the AFDX network. The possible values are 10MBits/sec, 1000MBits/sec, 1GBits/sec, or AUTO. The analyzer will negotiate with other end items to select a speed when AUTO is selected.

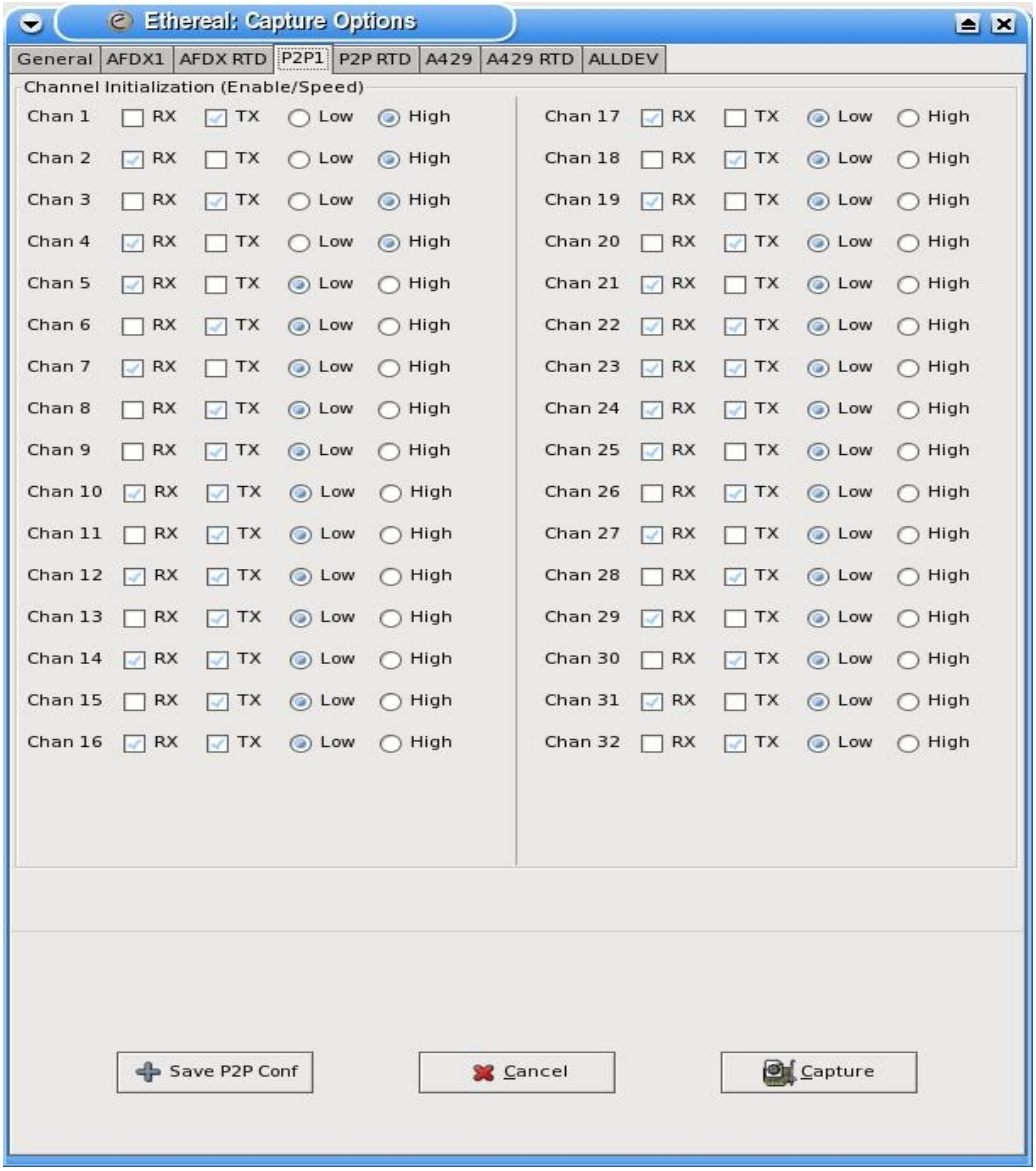
4.2.5 Capture TX Packets Frame.

- Network A. This option will allow any Network A data transmitted by the afdx card to be looped back through the analyzer.
- Network B. This option will allow any Network B data transmitted by the afdx card to be looped back through the analyzer.

4.3 P2P Options Page.

The p2p options page contains options that are specific to the p2p interface. The notebook pages are named P2P1..P2PN where N is the number of the p2p interface board in the system. Each p2p interface board has 32RX/TX channels.

4.3.1 Channel Initialization (Enable/Speed) Frame.



This frame allows the user to configure the speed and which transmit type data is input by the analyzer. There is a group of buttons for each of the 32 channels on the P2P H/W.

- RX. This option will allow RX data received from the hardware on the specified channel to be input to the analyzer.
- TX. This option will allow TX data transmitted by the hardware to be looped back to the analyzer on the specified channel.
- Low/High Speed. This option will select the speed (LO or HI) of the each P2P hardware channel.

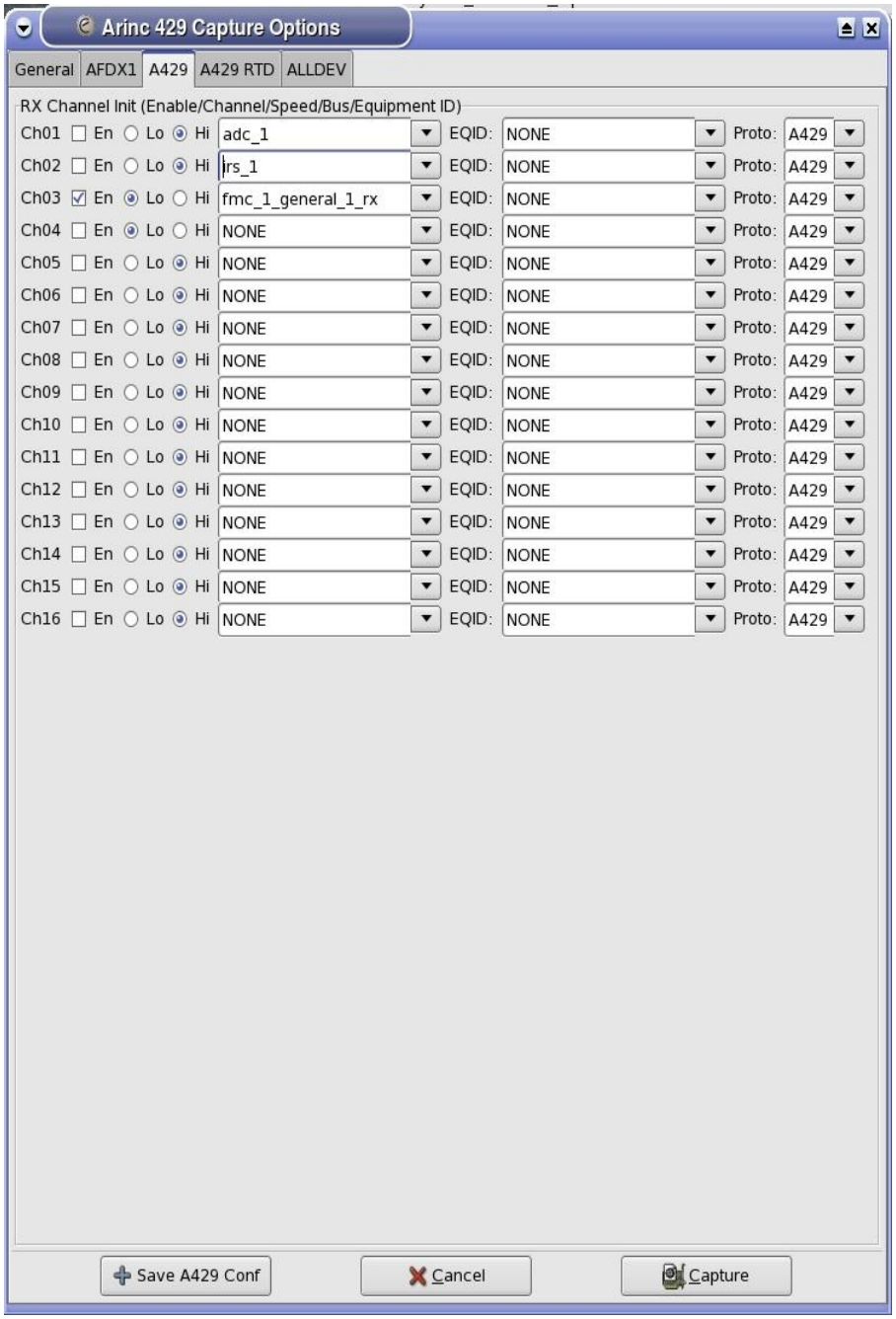
4.3.2 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.
- Save P2P Conf. The Save P2P Config button will save the state of channel and device configuration data for the duration of the program execution. After the P2P configuration is saved, these will become the default P2P settings when the dialog is opened in the future. Permanent options can be

save to the preferences file using the *Edit->Preferences->P2P HW* option. The file is located in the analyzer configuration directory.

4.4 A429 Options Page.

This page contains options that are specific to the a429 interface. There is only one A429 notebook page. Each A429 interface board has capabilities to transmit and receive on at the most 16RX/8TX channels.

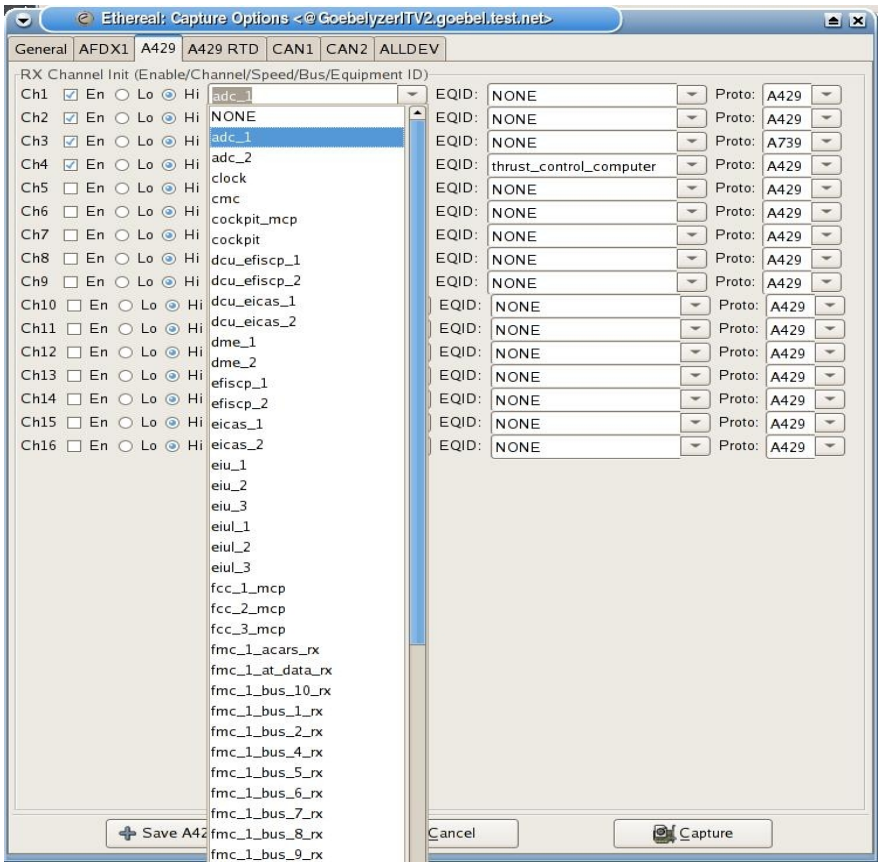


4.4.1 RX Channel Init Frame.

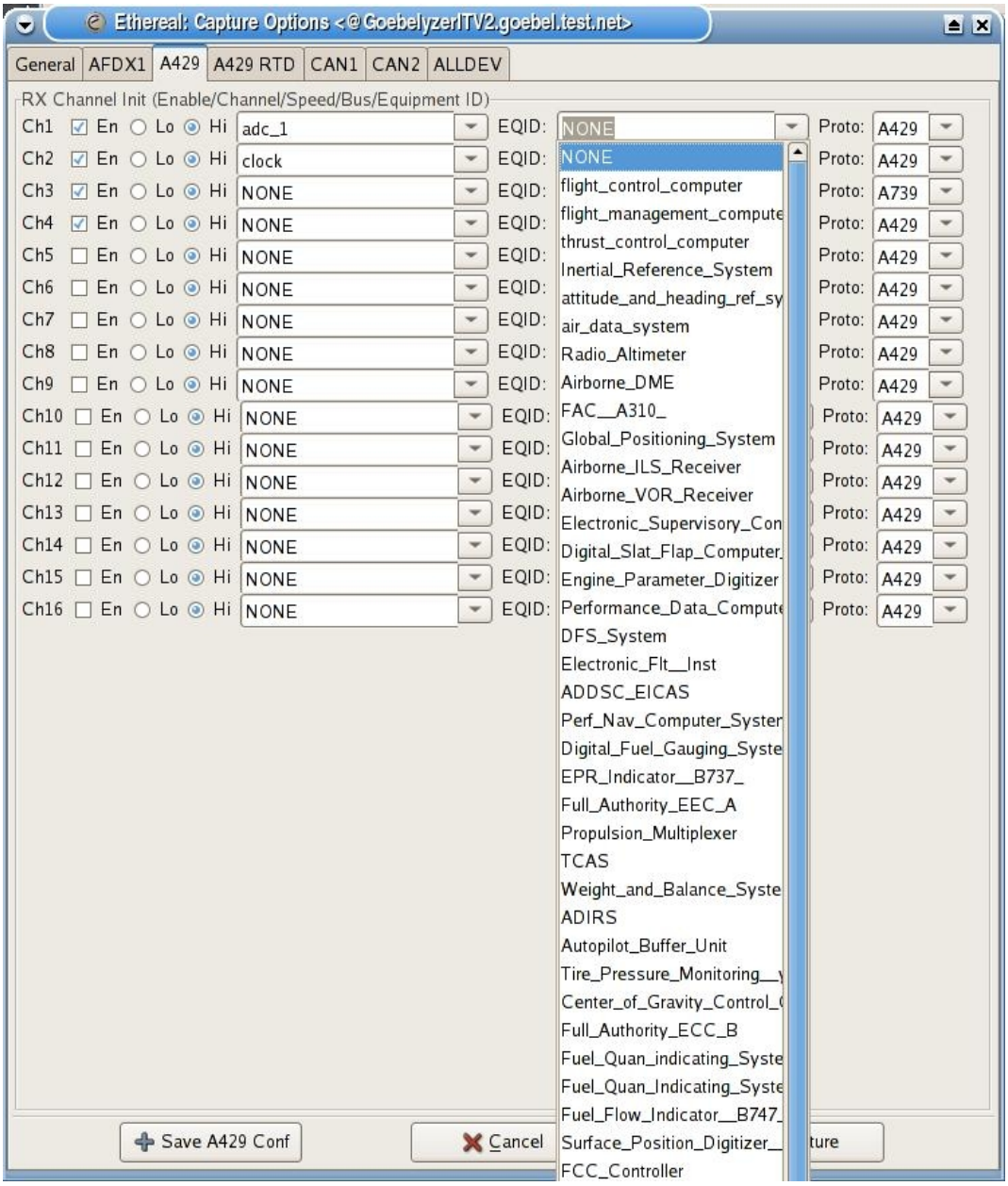
This frame allows the user to configure each A429 RX channel. There is a group of buttons for each of the RX channels on the A429 H/W.

- Enable. This option will enable/disable the input of RX data from the corresponding channel.

- RX Speed. This option will set the speed of the RX channels on the device. The speed can be set to LO (10K) or HI (100K). The speed select must match the speed of the TX channel connected to this RX channel. Due to hardware limitions, the speed is selected in pairs (1,2),(3,4).
- Bus Name Pulldown. A429 hardware can be connected between any two end items that may transmit the same labels containing different data items. The analyzer needs a way to decide which data definition to attach the label with. So, when the user selects a bus from the pulldown list, the label will be decoded based on packets for the bus name. The packet name definitions in the *packet-a429.dat* file will contain the bus name with a _w<label> appended to the end of the string. All a429 bus names are defined and stored in the *packet-a429_busses.dat* file. The NONE value is used as a default to indicate that no value have been selected or when bus name definitions do not exist.



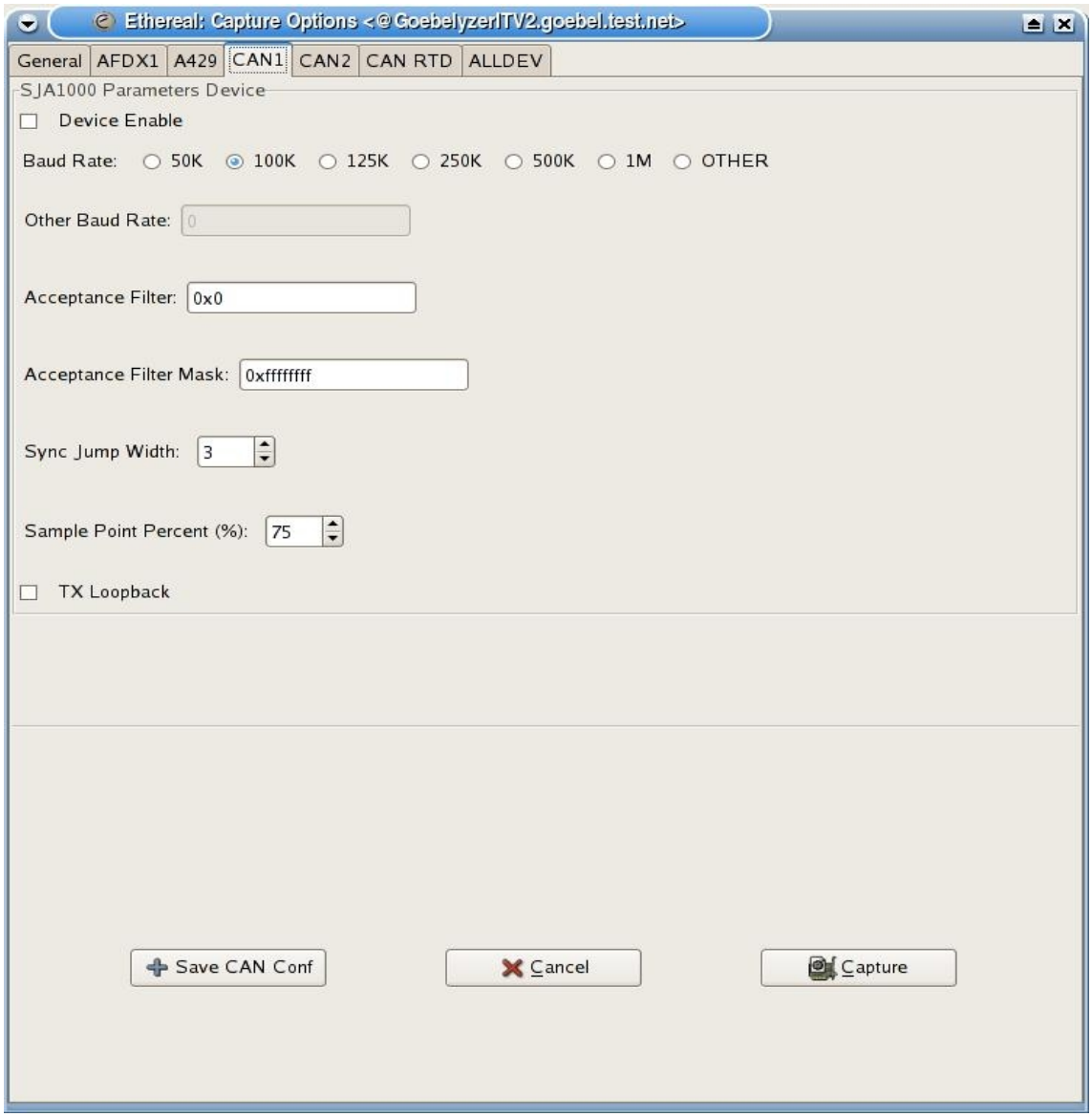
- Equipment ID Pulldown. The document ARINC 429 SPEC 429P1-15 defines a set of equipment ids (various avionics end items) and labels transmitted by the end items. These definitions are contained in the data definition file *a429-eqid.dat*. So if the user knows that data transmitted on a channel is from an equipment id in the pulldown list, the channel can be decoded with the standard label definitions for the selected id. This will also be helpful if no a429 data/bus definitions exist. If both a bus name and equipment id are selected for an RX channel, the system will try to decode with the bus name first and the equipment id second.



- Protocol Pulldown. The analyzer supports decoding A739, A429 (default), and EFIS protocols on the A429 h/w. The user can select a protocol using the pulldown menu and save the configuration.

4.4.2 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.
- Save A429 Conf. The Save A429 Config button will save the state of channel and device configuration data for the duration of the program execution. Permanent options can be save to the preferences file using the *Edit->Preferences->A429 HW* option. The file is located in the analyzer configuration directory. After the A429 configuration is saved, these will become the default A429 settings when the dialog is opened in the future.



4.5 CAN Bus Options Page.

This page contains options that are specific to the CAN bus interface. The notebook page contains options for 2 Philips SJA1000 devices which are contain on the hardware device. The user may refer to the SJA1000 Data Sheet for more information on parameter option described on this page.

4.5.1 SJA1000 Parameters Device Frame

- Baud Rate. The user can change radio buttons to select one of the commonly used baud rates. The OTHER radio button is used to select a different baud rate with the entry widget below.
- Other Baud Rate. This entry widget is used to enter a numeric value from 0 to 1000000. Values greater than 1000000 will be changed to 1000000. Invalid values such as a string entry will result in a 0 baud rate value. This entry is only allowed when the *Other* radio button is selected.
- Acceptance Filter. This entry will select a hexadecimal value of the SJA1000 acceptance filter. Invalid entries will be ignored and a value of 0x0 used. The acceptance filter will define which identifiers are allowed to be received by the SJA1000. The format of the filter is bit32 ->ID.28...bit 3-

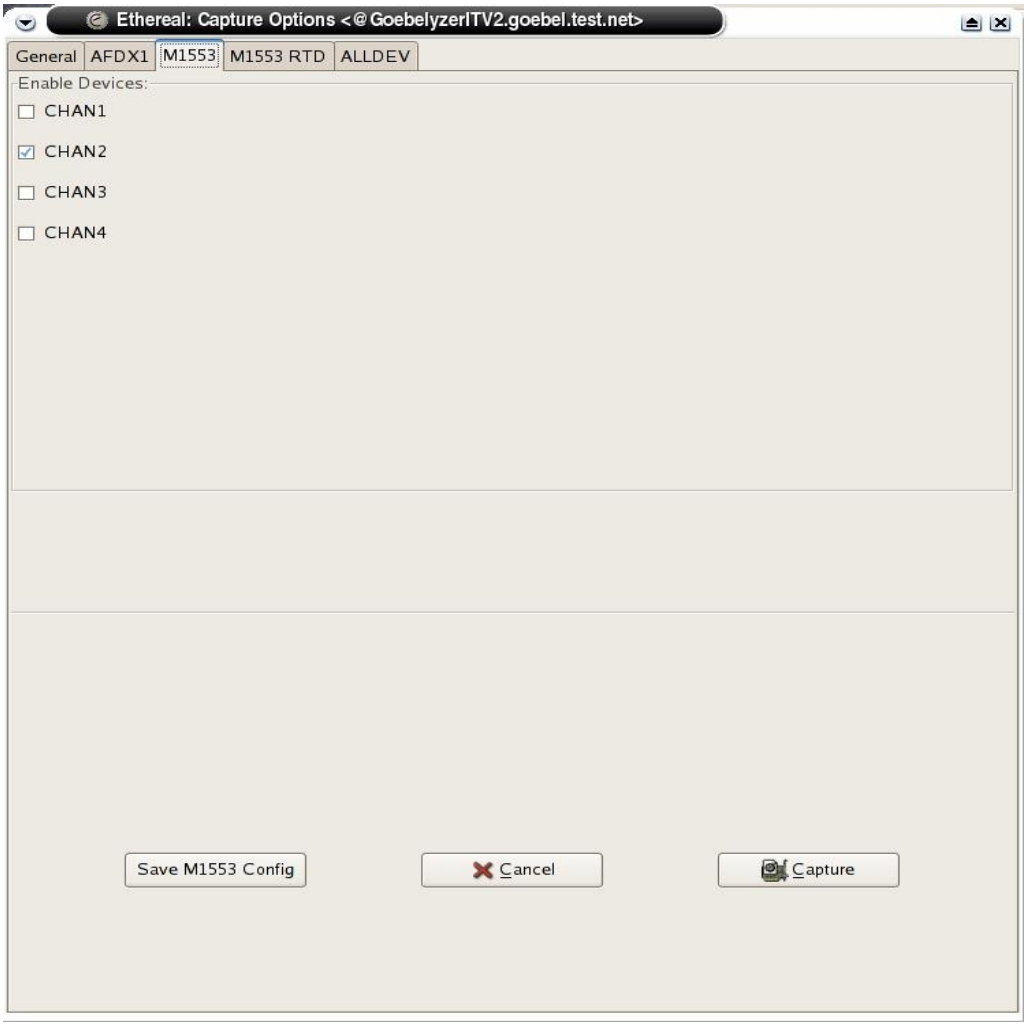
>ID.3, bit 2->RTR. When the ID and filter have matching values with a range defined by the acceptance mask. See SJA1000 Data Sheet Figure 10 for more info.

- Acceptance Filter Mask. This entry will select a hexadecimal value of the SJA1000 acceptance filter mask. Invalid entries will be ignored and a value of 0xffffffff used. The mask is used to define which bits of the received ID are matched with the acceptance filter. See SJA1000 Data Sheet Figure 10 for more info.
- Sync Jump Width. This spin button will select a value from 0-3 for the Sync Jump Width. To compensate for phase shifts between clock oscillators of different bus controllers, a bus controller must re-sync on any revelant signal edge. The sync jump width defines the maximum number of clock cycles a bit period may be shortened or lengthened by one re-sync.
- Sample Point Percent. This spin button will select a value from 0-100 for the sample point. The sample point defines the location of the sample point within a bit period. A value of 0 is at the beginning of the bit period, 50 in the middle, and 100 at the end. See SJA1000 Data Sheet Figure 13 for more info.
- TX Loopback. This button will enable the analyzer to receive TX transmissions send from this machine.

4.5.2 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.
- Save CAN Conf. The Save CAN Config button will save the state of device configuration data for the duration of the program execution. Permanent options can be saved to the preferences file using the *Edit->Preferences->CAN HW* option. The file is located in the analyzer configuration directory. After the CAN configuration is saved, these will become the default CAN settings when the dialog is opened in the future.

M1553 Options Page.



M1553 Parameters Frame.

- Channel. The user can change radio buttons to select any channel to have the analyzer listen on.

4.5.3 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.
- Save M1553 Conf. The Save M1553 Config button will save the state of device configuration data for the duration of the program execution. Permanent options can be saved to the preferences file using the *Edit->Preferences->M1553 HW* option. The file is located in the analyzer configuration directory. After the M1553 configuration is saved, these will become the default M1553 settings when the dialog is opened in the future.

4.6 FSCC Options Page.

This page contains options that are specific to the FSCC bus interface. The notebook page contains options for the two serial channels.

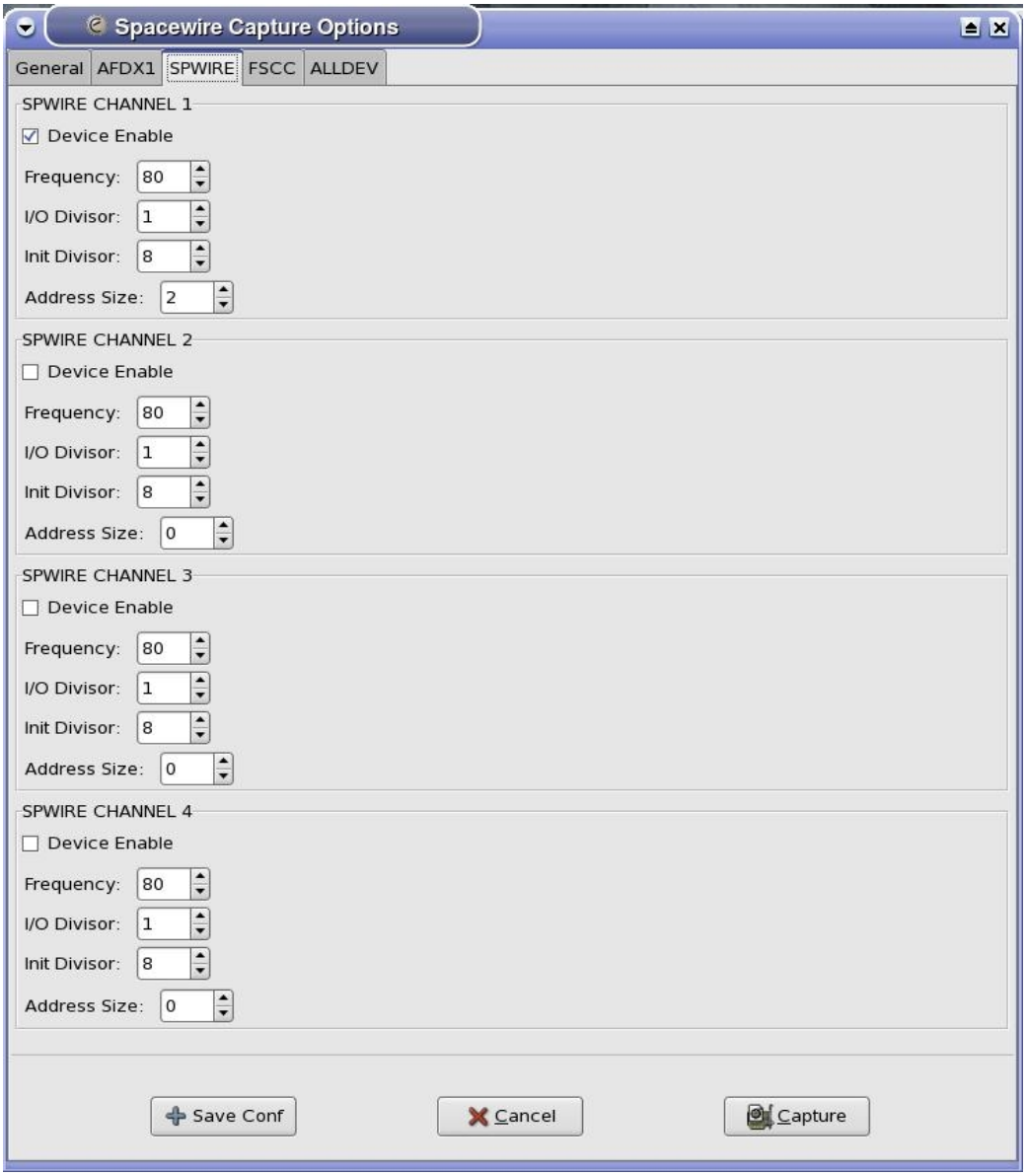
4.6.1 FSCC Parameters Channel Frame

- Interface. The user can change set the electrical interface to RS-422 or RS-485.
- Mode. The user can set the mode to synchronous or asynchrous.
- Enable. This will enable the analyzer on the selected channel.
- Frequency. The Board Frequency in the range of 100K to 30M.

- Clock PPM. Parts Per Million of the system crystal.
- Baud Rate Divisor. Divider of the Board frequency to create channel frequency.
- Address Size. The address size of received packets.
- UART Stop Bits. Number of stop bits in the asynchrous data transmission.
- UART Parity. Parity type in an asynchrous data transmission.

4.6.2 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.
- Save Conf. The Save Config button will save the state of device configuration data for the duration of the program execution. Permanent options can be saved to the preferences file using the *Edit->Preferences->FSCC HW* option. The file is located in the analyzer configuration directory. After the FSCC configuration is saved, these will become the default FSCC settings when the dialog is opened in the future.



4.7 Spacewire Options Page.

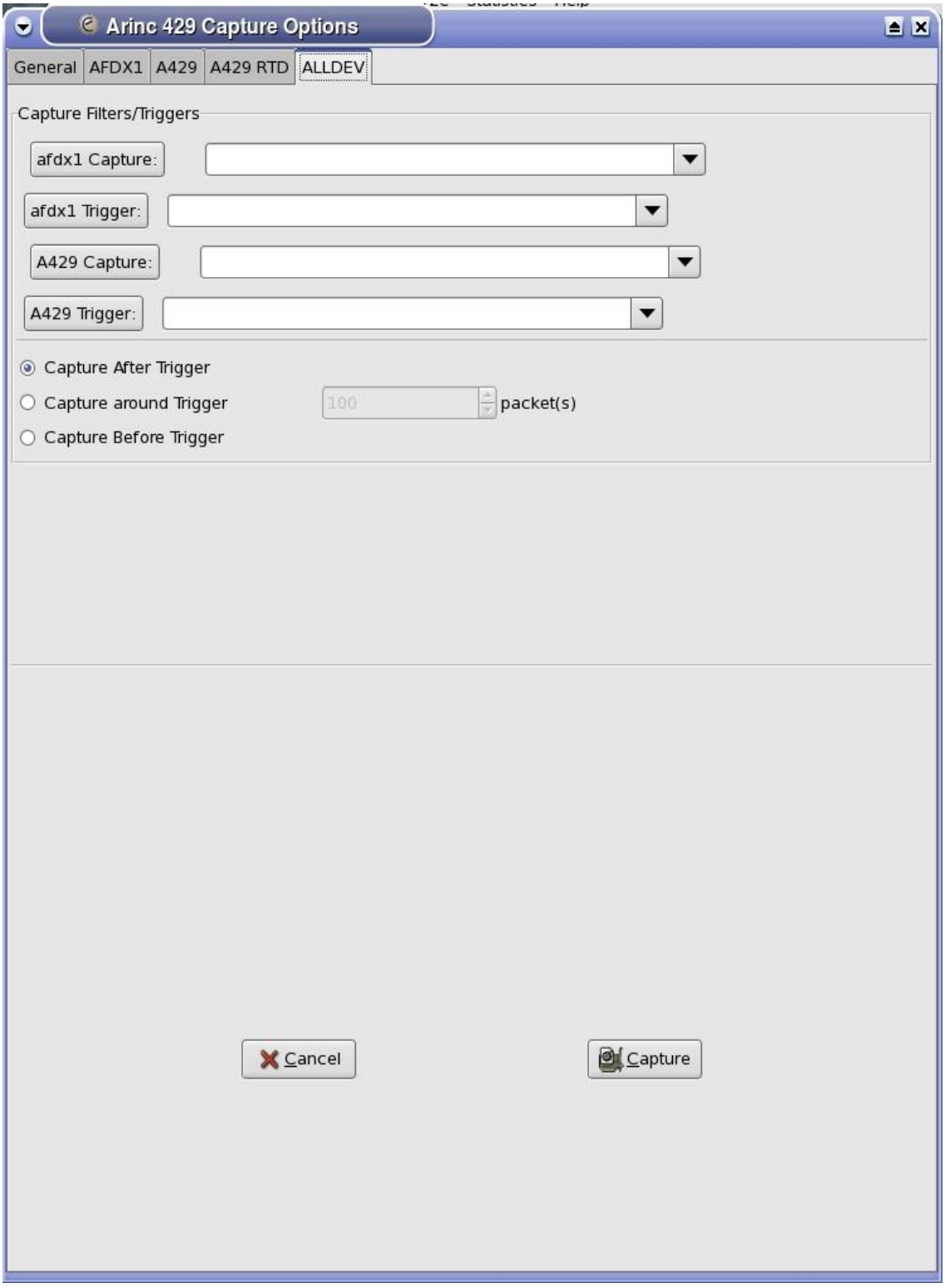
This page contains options that are specific to the Spacewire bus interface. The notebook page contains options for the two.

4.7.1 Spacewire Parameters Channel Frame

- Enable. This will enable the analyzer on the selected channel.
- Frequency. The Board Frequency in the range of 100K to 30M.
- I/O Divisor. Divider of the Board frequency to create channel frequency.
- Init Divisor. Divider of the Board frequency to creat initialization frequency of 10M.
- Address Size. The address size of received packets.

4.7.2 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.
- Save Conf. The Save Config button will save the state of device configuration data for the duration of the program execution. Permanent options can be saved to the preferences file using the *Edit->Preferences->Spacewire HW* option. The file is located in the analyzer configuration directory. After the Spacewire configuration is saved, these will become the default spacewire settings when the dialog is opened in the future.



4.8 AllDev Options Page.

This page contains options that Alldev Options Page specific to the alldev interface. The alldev is a pseudo interface which contains all hardware contained in system. So, if the system contains one afdx and one a429 board. When the alldev device is selected, time sorted packet data will be receive from both the afdx and a429 interfaces.

4.8.1 Select Devices Frame.

This frame contains a list of all hardware devices in the system. The user may remove hardware device from the alldev device by selecting a device to be removed from this list. Multiple devices selections may be made by depressing <CTRL> and left mouse buttons. The alldev device by default will contains all hardware devices in the system.

4.8.2 Capture Filters/Triggers Frame

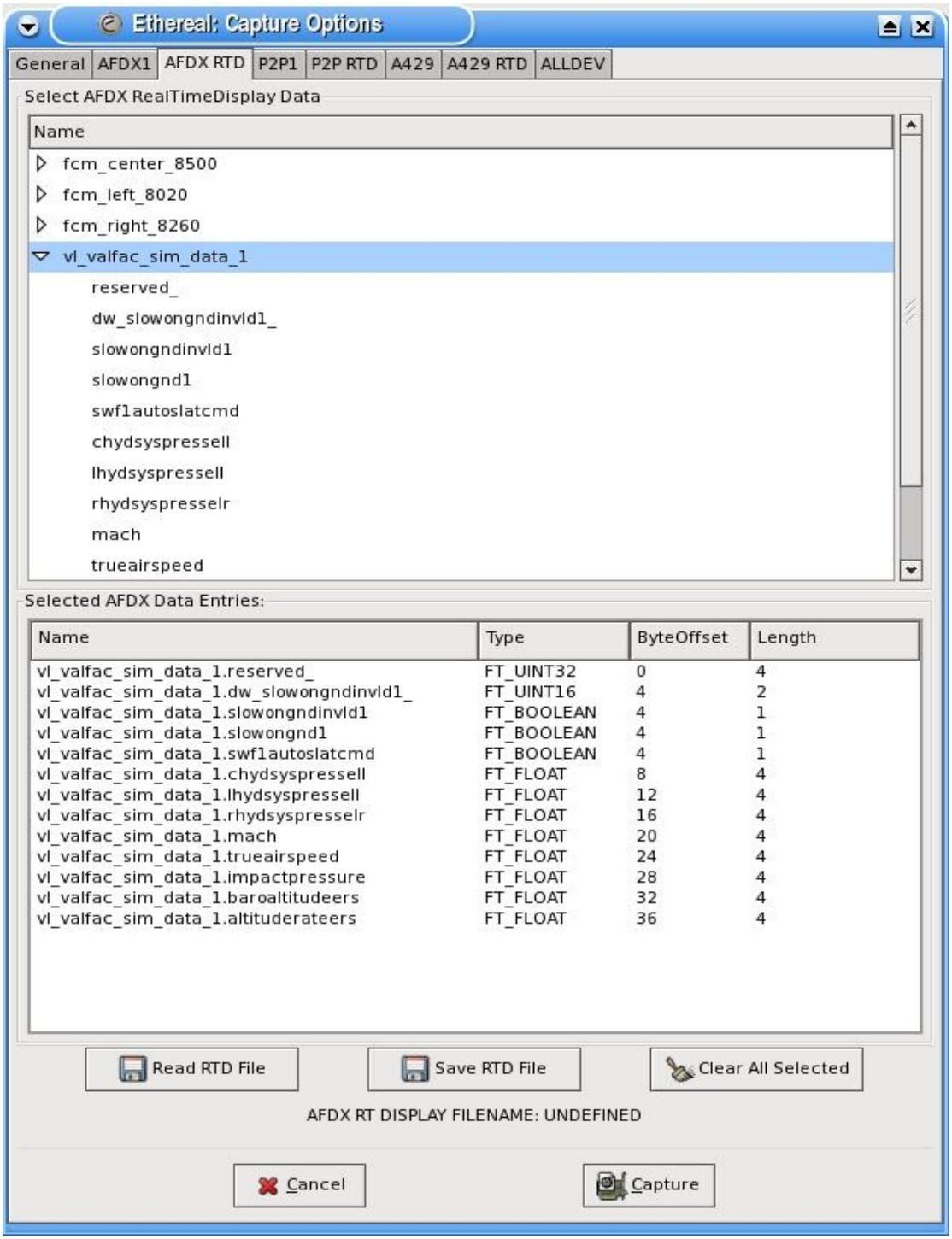
This frame contains a trigger button and pulldowns for each interface in the system. The user may enter a different trigger for each device. When the first trigger event is tripped, data will be capture on all devices and all other triggers will be disabled. Only one type of trigger event can be defined for the alldev device.

This frame contains a capture button and pulldown for each interface in the system. The user may enter a different capture filter for each device. When the capture has been started and an alldev trigger event has been tripped, each alldev capture filter will execute only for the device that it corresponds with.

Capture Filter/Trigger syntax is described in detail in section #4.1.5Capture Frame.|outline

4.8.3 Button Box.

- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.



4.9 Real-Time Display Options Page.

This page allow users to select various payload data definitions for real-time display after the capture has been started. Each interface that supports the real-time display will have a notebook tab displayed. The real-time display page is described in #5.3Data Display Page.|outline

4.9.1 Select Realtime Display Data Frame.

This frame contains a list organized as a tree of packets defined for the interface. This data is imported from the *packet-<interface>.dat* configuration file. If the packet entry is expanded, all data elements of

the packet are displayed. In the figure above, the vl_valfac_sim_data_1 packet is expanded. A list element is selected by double clicking on the entry. When an entry has been selected, a double click will clear the selection. If a packet is selected, all data elements of the packet will also be selected.

4.9.2 Selected Real-time Display Data Frame.

The frame contains a list of entries selected for realtime display on this interface. The list displays the data type, payload byte offset, byte length. The order of the list can be changed by selecting an entry and pulling the entry to a new location in the list.

4.9.3 Button Box.

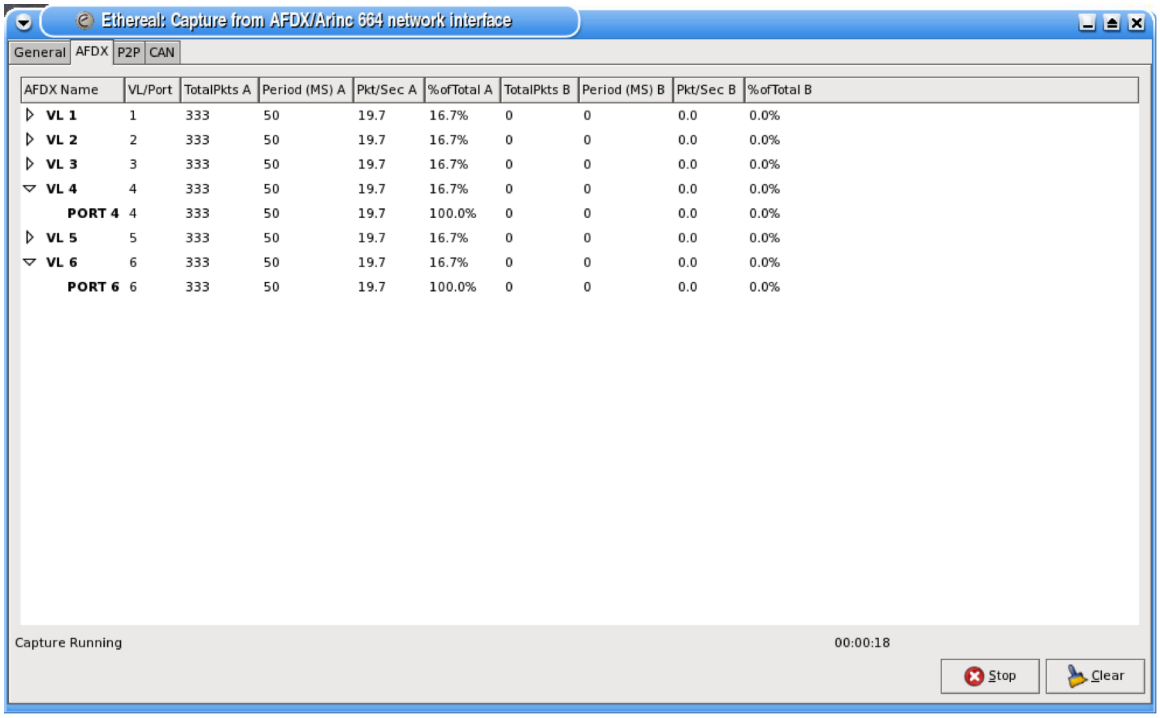
- Read RTD File. This button will open a file selection dialog. The file selected will be imported in as selected items. This will be displayed in the selected realtime displa data frame. The read file must have be saved using the Save RTD File button.
- Save RTD File. This button will open a file selection dialog. The selected realtime display items will be saved to the file selected. This will allow the user to create display lists that may be restored (Read RTD File) with a minimal amount of work.
- Clear All Selected. This button will clear all selected data items contained in the selected realtime display data frame.
- Capture. Depress the capture button to start execution of a capture.
- Cancel. Depress the cancel button to close the option dialog notebook and restore default values.

5. Active Data Capture.

This section describes the unique operations that can be performed when the capture is active and the system is receiving data. There is a general data notebook, packet info page and data element display page for each interface in the system. These are described below:

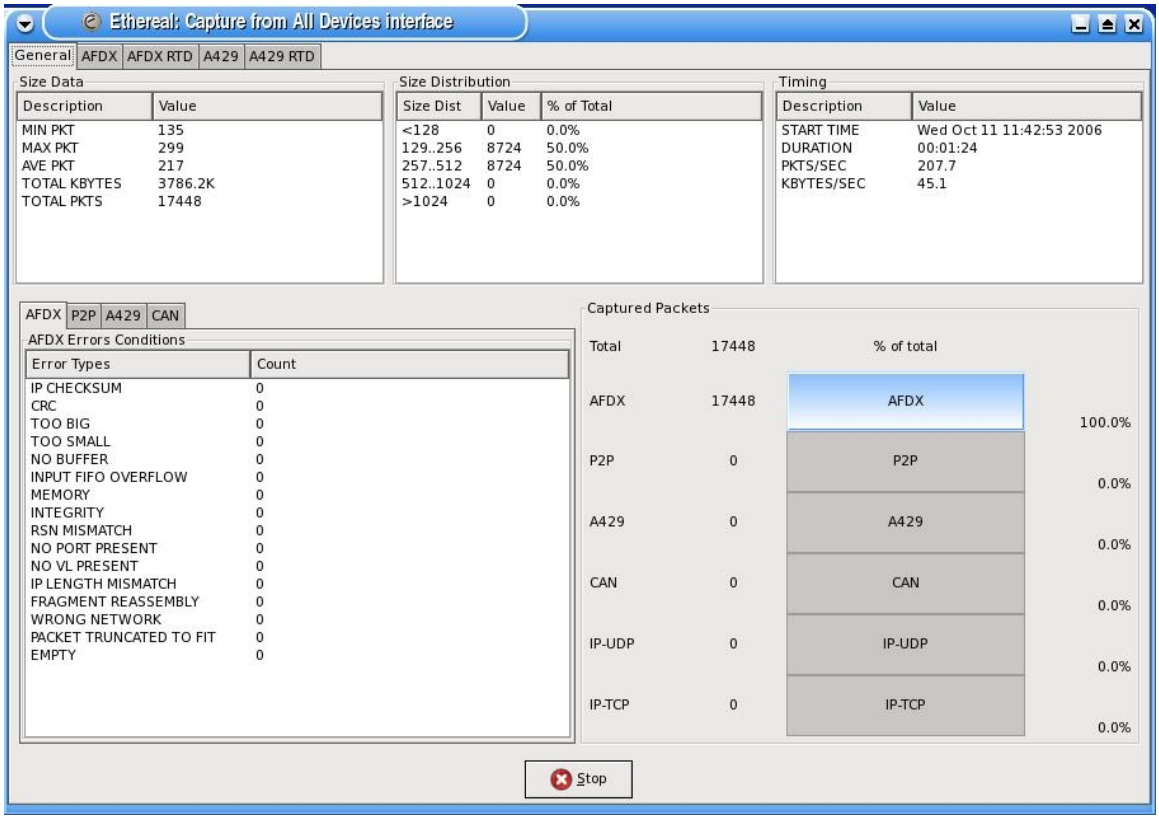
5.1 Capture Pkt Info Pages.

After starting a capture, the capture packet info page will be displayed for the selected interface. This page will display information on all active packets on the interface. Each interface will have a different appearance. But the information will be similar and include total packets, rates, percent of total as labeled in the widget header. A packet that is invalid for more than 1 second will be marked with a red foreground color.



5.1.1 Button Box.

- Stop. Depress the button to stop execution of a capture.
- Pause. Depress the button to pause display. Depress again to restart display.
- Clear. Depress the clear button to the remove all packets from list and reset counters to 0.



5.2 Capture Info Page.

The general capture info notebook will provide various general pieces of information to the user. The user will have to select the General notebook tab to display the page.

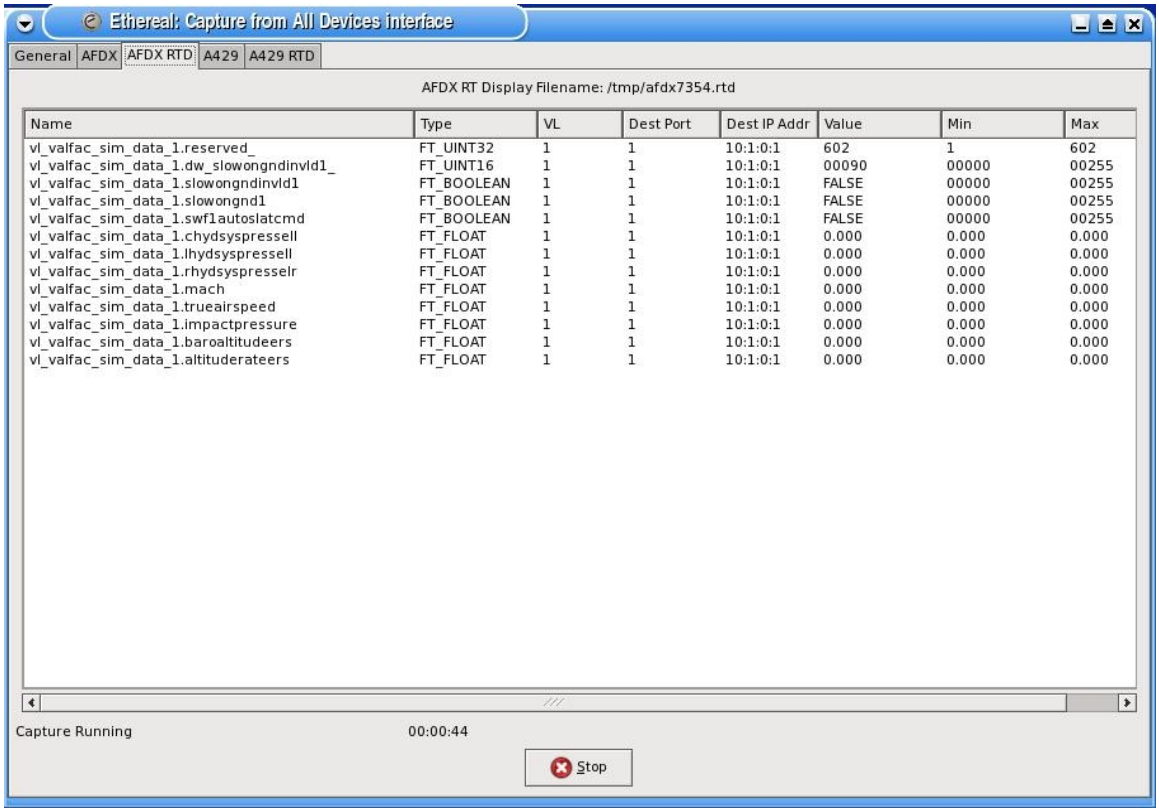
- Size Data. This frame contains general size (min,max,average) information about packets.
- Size Distribution. This frame contains general information about the number of packets contained in various size ranges.
- Timing. This frame contains time information about the capture. This includes start time, length of capture, and interface data rates.
- Error Conditions. This is a notebook containing pages for each interface. Each page contains a list of different types of receive errors related to the interface. This is the number of errors determined since the start of the capture.
- Captured Packets. This frame contains a set of progress bars for each interface. It lists the number of packets received on the interface and the percent of the total.

5.3 Packet Info Page.

This page displays higher level traffic information about the interface. The list above contains entries for each VL/Port (afdx) received by the system. A list entry will display various information like number of packets received, errors, data rates, percent of total. So, the user may view the page, to verify which VL (afdx) are being transmitted and at what rate and expanded to port contained on the VL. The a429/p2p interface have a tree structure with channels at the top and the label/ports for each channel displayed as a branch. The page information is from a live capture and updated at a 1 sec rate.

This page will contain different columns for each interface. Each interface page will look different but the concept of displaying information at a packet level is the same.

The Stop button will stop execution of the capture.



5.4 *Data Display Page.*

This page displays individual payload data items contained on the interface. The list contains entries for each data element selected from the real time option page #5.6Real-Time Display Option Pages.|outline A list entry will display various information like data type, VL/Port, engineering units value, min value, and max value. So, the user can view select payload data elements updated at an approximately 1 sec rate while the capture is executed.

This page will contain different columns for each interface. Each interface page will look different but the concept of displaying information at a payload data level is the same.

6. Digital Filters.

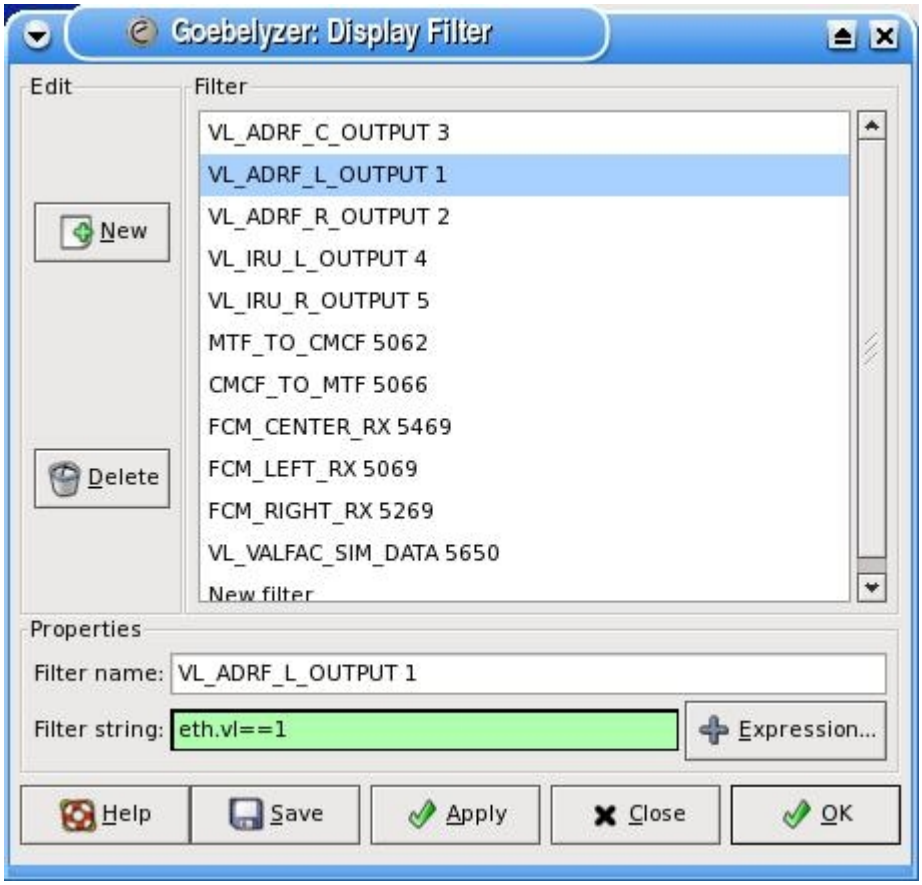
A digital filter provides a way to eliminate packets from the packet pane that do not meet the boolean condition of the digital filter. Each filter consists of one or more expressions combined with logical operators. For example: (expr1 and expr2) or expr3). Each expression consists of a protocol data field, comparison operator, and literal constant. For example: (frame.pkt_len > 10), (udp.dstport == 12312). After the user has constructed a few digital filters, the procedure will be easy to understand.

6.1 *Digital Filter Display Pane.*

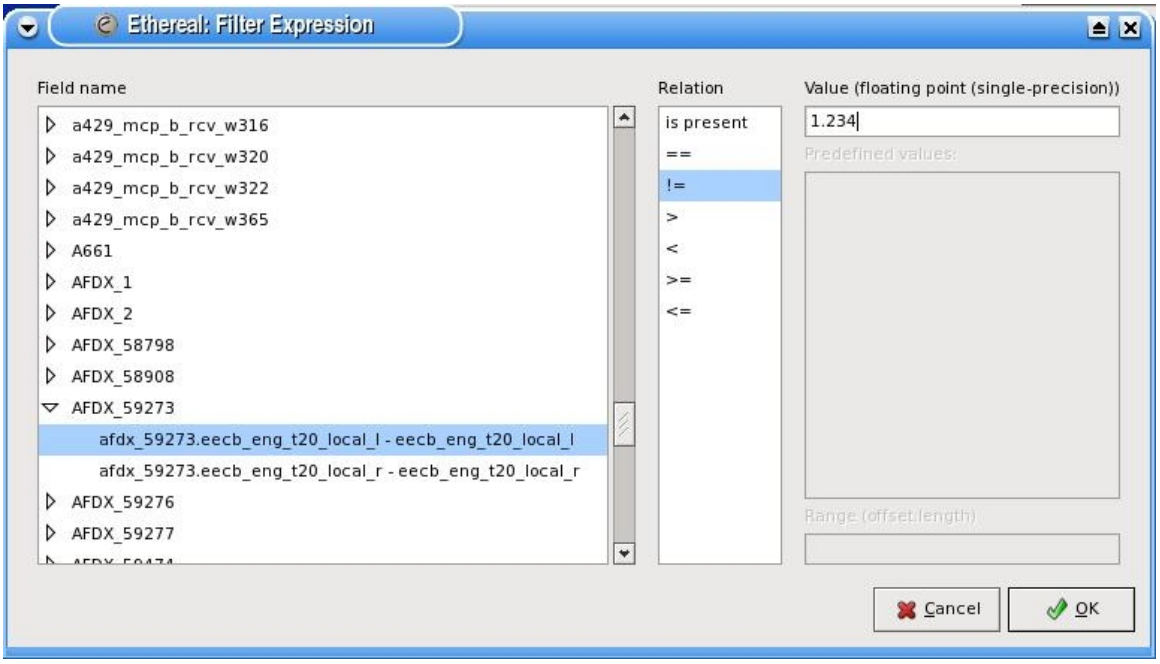
The digital filter pane consists of 4 buttons (*Filter*, *Expression*, *Clear*, and *Apply*) and an entry box pulldown.

- Filter Button. The following dialog is displayed when the filter button is depressed. The dialog box will allow the user to create or select saved digital filters. The digital filters are saved in the *dfilters* file contain in the analyzer configuration directory. When a digital filter is selected with the apply button, it will be displayed in the digital filter entry box. The user can create new filters using the new

button or remove filter with the delete button. The save button will copy all filter data in the *dfilter* file. Each entry in the *dfilter* file consists of a filter name (something that can be remembered) and the actual filter text string.



- Entry Widget. The user can enter a digital filter text string in this field. The rules for creating digital filters is described in #7.2Expression Syntax. The user can use the pulldown to select previously used digital filters. A green background indicates that the expression is valid. A red background indicates that the expression does not have the correct syntax.
- Expression Button. When the expression button is depress, the expression dialog is displayed. The dialog provides a simple way to create filter expressions. The protocol field names are listed in a tree structure containing a row for each item. The protocol can be expanded to display all field elements contained in the protocol. The user can select the field name, relation operator, and literal constant value. When the OK button is depress, the dialog is removed and the expression is displayed in the digital filter entry widget. The created string is appended to end of text in entry box. So, it might be necessary to add a relation operator between two expressions if there is text in the entry box.



- Clear Button. This button is used to clear all text from the entry box widget.
- Apply Button. This button is used to apply the digital filter contained in the entry widget and update the packet list.

6.2 Expression Syntax.

A digital filter can be broken down into the following basic blocks. This section contains details of the basic building blocks of a digital filters.

<digital filter>=<expression> <relational operator> <expression> .....

<expression> = <protocol data field> <compare operator> <literal constant>

<protocol data field>=this string is any protocol data item defined in the system. The user can depress the expression button to display a dialog with protocol data item listed in a tree format. This dialog is described in the *Expression* button section above.
<comparison operator>=There are 6 operators defined. They are used to compare a date item with a constant value. Each has an English term and C language term.

English	*C*	*Description*
	=\=	Equal (eth.vl==1212)
ne	!=	Not Equal (eth.vl != 1212)
gt	>	Greater Than (eth.vl > 1212)
lt	<	Less Than (eth.vl lt1212)
ge	>=	Greater Than or Equal (eth.vl >= 4545)
le	<=	Less Than or Equal (eth.vl le 4545)

<literal constant>=constant literal has a type as listed below:

Types	*Description*
signed/unsigned integer	These values can be decimal, octal, or hexadecimal. eth.vl == 1122 eth.vl == O366 eth.vl == 0x345
Boolean	test for existance of data item in packet.
Ethernet Address	eth.addr == 12:34:56:78:9a:bc
IPv4 Address	ip.addr == 255.255.128.128
floating point	1.23

<relation operators>=these terms are used to combine expressions with other expressions to create more complex boolean expressions. There are 4 relational operators with English and C terms.

English	*C*	*Description*
and	&&	Logical AND
or	||	Logical OR
xor	^^	Exclusive OR
not	!	Logical NOT

6.3 Protocol Data Formats.

The main static protocols used in filters are:

- **Frame** (contains high level information about the packet).

  frame.cap_len Capture Frame Length. Unsigned 32-bit integer.

  frame.file_off File Offset Signed 32-bit integer.

  frame.link_nr Link Number. Unsigned 16-bit integer.

  frame.marked Frame marked. Boolean. Frame is marked in the GUI.

  frame.number Frame Number. Unsigned 32-bit integer.

  frame.board Adfx board number. Unsigned 32-bit integer.

  frame.pkt_len Total Frame Length. Unsigned 32-bit integer.

  frame.protocols Protocols in frame. String. Protocols carried by this frame.

  frame.ref_time This is a Ref Time frame. No value. This frame is a Reference Time frame.

  frame.time Arrival Time. Date/Timestamp. Absolute time when this frame was captured.

  frame.time_delta Time delta from previous packet. Time duration. Time delta since previous diplayed frame.

  frame.time_relative Time since reference or first frame. Time duration. Time relative reference or first frame.

- **Ethernet** (afdx ethernet packet parameters)

  eth.addr Source or Destination Address. 6-byte Hardware (MAC) Address. Source or Destination Hardware Address

  eth.dst Destination. 6-byte Hardware (MAC) Address. Destination Hardware Address

  eth.equipment_id Equipment ID. Unsigned 8-bit integer. Equipment ID

  eth.if Interface ID. Unsigned 8-bit integer. Interface ID

eth.len Length. Unsigned 16-bit integer

eth.network_id Network ID. Unsigned 8-bit integer.

eth.src Source. 6-byte Hardware (MAC) Address. Source Hardware Address

eth.trailer Trailer Byte array. Ethernet Trailer or Checksum

eth.trailer_crc CRC. Unsigned 32-bit integer. Ethernet CRC

eth.trailer_rsn RSN. Unsigned 8-bit integer. Redundancy Sequence Number

eth.type Type. Unsigned 16-bit integer

eth.vl VL. Unsigned 16-bit integer. Virtual Link Number

- **EDE** (afdx EDE packet parameters).

ede.sn Unsigned 16 bit integer.

ede.timeu Unsigned 16 bit integer.

ede.timel Unsigned 32 bit integer.

ede.crcx Unsigned 16 bit integer.

ede.crcy Unsigned 16 bit integer.

- **IP**, (afdx IP packet parameters).

ip.addr Source or Destination Address. IPv4 address.

ip.checksum Header checksum. Unsigned 16-bit integer.

ip.checksum_bad Bad Header checksum. Boolean.

ip.dsfield Differentiated Services field. Unsigned 8-bit integer.

ip.dsfield.ce ECN-CE. Unsigned 8-bit integer.

ip.dsfield.dscp Differentiated Services Codepoint. Unsigned 8-bit integer.

ip.dsfield.ect ECN-Capable Transport (ECT). Unsigned 8-bit integer.

ip.dst Destination. IPv4 address.

ip.flags Flags. Unsigned 8-bit integer.

ip.flags.df Don't fragment. Boolean.

ip.flags.mf More fragments. Boolean.

ip.flags.rb Reserved bit. Boolean.

ip.frag_offset Fragment offset. Unsigned 16-bit integer.

ip.fragment Frame number. IP Fragment.

ip.fragment.error Defragmentation error Frame number. Defragmentation error due to illegal fragments.

ip.fragment.multipletails Multiple tail fragments found. Boolean. Several tails were found when defragmenting the packet.

ip.fragment.overlap Fragment overlap. Boolean. Fragment overlaps with other fragments.

ip.fragment.overlap.conflict Conflicting data in fragment overlap. Boolean. Overlapping fragments contained conflicting data.

ip.fragment.toolongfragment Fragment too long. Fragment contained data past end of packet.

ip.fragments No value.IP Fragments

ip.hdr_len Header Length. Unsigned 8-bit integer.

ip.id Identification. Unsigned 16-bit integer.

ip.len Total Length. Unsigned 16-bit integer.

ip.proto Protocol. Unsigned 8-bit integer.

ip.reassembled_in Reassembled IP in frame. Frame number This IP packet is reassembled in this frame.

ip.src Source. IPv4 address.

ip.tos Type of Service. Unsigned 8-bit integer.

ip.tos.cost Cost. Boolean.

ip.tos.delay Delay. Boolean.

ip.tos.precedence Precedence. Unsigned 8-bit integer.

ip.tos.reliability Reliability. Boolean.

ip.tos.throughput Throughput. Boolean.

ip.ttl Time to live. Unsigned 8-bit integer.

ip.version Version. Unsigned 8-bit integer.

- **UDP** (afdx UDP packet parameters).

udp.checksum Checksum --Unsigned 16-bit integer

udp.checksum_bad Bad Checksum --Boolean

udp.dstport Destination Port -- Unsigned 16-bit integer

udp.length Length -- Unsigned 16-bit integer

udp.port Source or Destination Port --Unsigned 16-bit integer

udp.srcport Source Port --Unsigned 16-bit integer

- **AFDX_VL<vl>_PORT<port>*** (one for afdx payload packet defined). These protocols are defined based on data contained in data definition files.
- **P2PHDR** (P2P header info parameters, at the begininning of all p2p messages).

p2phdr.crc CRC. Unsigned 16-bit integer. P2P CRC.

p2phdr.dest DEST. Unsigned 16-bit integer. P2P Dest Location.

p2phdr.id ID. Unsigned 16-bit integer. P2P ID.

p2phdr.label LABEL. Unsigned 16-bit integer. P2P Msg Label.

p2phdr.length LENGTH. Unsigned 16-bit integer. P2P Msg Length.

p2phdr.src SRC. Unsigned 16-bit integer. P2P Src Location.

- **P2P_<port>*** (one for each P2P payload packet defined). These protocols are defined based on data contained in data definition files.
- **A429** (A429 common info parameters, label,sdi,and ssm).

a429.label label. Unsigned 8-bit integer.

a429.sdi sdi. Unsigned 8-bit integer.

a429.ssm ssm. Unsigned 8-bit integer.

a429.word word. Unsigned 32-bit integer.

- **<a429_busname>_<label>*** (one for each A429 payload packets defined). These protocols are defined based on data contained in data definition files.
- **CANHDR** (CAN header info, at the beginning of all CAN messages).

canhdr.eff_id Unsigned 32-bit integer. CAN EFF Identifier.

canhdr.ff Unsigned 8-bit integer. CAN Frame Format.

canhdr.len Unsigned 8-bit integer. CAN Data Length.

canhdr.rtr Unsigned 8-bit integer. CAN Remote Transmission Request.

canhdr.sff_id SFF ID --Unsigned 16-bit integer. CAN SFF Identifier

- **CAN_<can id>*** (CAN payload packets). These protocols are defined based on data contained in data definition files.
- **A661 Header** parameters in A661 headers.

a661_hdr.blk_seq Unsigned 32-bit integer Sequence Number

a661_hdr.data_blk_size Unsigned 16-bit integer Data Block Size

a661_hdr.dest Unsigned 8-bit integer Destination

a661_hdr.extended_blk_size Unsigned 16-bit integer Extended Block Size

a661_hdr.health Unsigned 8-bit integer Health Assumed

a661_hdr.lowest_seq Unsigned 32-bit integer Lowest Sequence Number

a661_hdr.num_of_grps Unsigned 8-bit integer Number of Groups

a661_hdr.service Unsigned 8-bit integer Service Available

a661_hdr.source Unsigned 8-bit integer Source

a661_hdr.start_marker Unsigned 16-bit integer Start Marker

- **A661** parameters in A661 blocks.

a661.bb_context_num Unsigned 16-bit integer BEGIN BLOCK: Context Num

a661.bb_keywork Unsigned 8-bit integer BEGIN BLOCK: Keyword

a661.bb_layer_id Unsigned 8-bit integer BEGIN BLOCK: Layer Id

a661.bb_size Unsigned 32-bit integer BEGIN BLOCK: Size

a661.cmd_keyword Unsigned 16-bit integer Command Type

a661.cmd_size Unsigned 16-bit integer Command Structure Size

a661.cmd_widget_id Unsigned 16-bit integer Command Widget ID

a661.eb_data_grp_size Unsigned 16-bit integer END BLOCK Data Group Size:

a661.eb_keyword Unsigned 8-bit integer END BLOCK Keyword:

a661.eb_seq_num Unsigned 32-bit integer END BLOCK Sequence Number:

a661.eb_start_marker Unsigned 16-bit integer END BLOCK Start Marker:

a661.enable_array_index Unsigned 8-bit integer Enable Array Entry Index

a661.enable_array_value Unsigned 8-bit integer Enable Array Value

a661.entry_array_enable Unsigned 8-bit integer Entry Array Enable

a661.entry_array_index Unsigned 8-bit integer Entry Array Index

a661.entry_array_num Unsigned 16-bit integer Entry Array Num Of Entries

a661.entry_array_str Unsigned 16-bit integer Entry Array String Data

a661.entry_array_strlen String Entry Array String Length

a661.entry_popup_array_picture Unsigned 16-bit integer Entry PopUp Array Picture

a661.event_keyword Unsigned 16-bit integer Event Type

a661.event_origin Unsigned 16-bit integer Event Origin

a661.event_value Unsigned 32-bit integer Event Value

a661.evt_active_panel_id Unsigned 16-bit integer Event Active Tabbed Panel Id

a661.evt_button_state Unsigned 8-bit integer Check Button State

a661.evt_cursor_x Unsigned 32-bit integer Event Cursor X Position

a661.evt_cursor_y Unsigned 32-bit integer Event Cursor Y Position

a661.evt_first_visible Unsigned 16-bit integer Event First Visible Entry

a661.evt_frame_x Unsigned 32-bit integer Event Frame Position X

a661.evt_frame_y Unsigned 32-bit integer Event Frame Position Y

a661.evt_item_index Unsigned 16-bit integer Event Item Index

a661.evt_map_x Unsigned 32-bit integer Event Map X Position

a661.evt_map_y Unsigned 32-bit integer Event Map Y Position

a661.evt_num_of_inc Unsigned 32-bit integer Event Number of Increments

a661.evt_selected_entry Unsigned 16-bit integer Event Selected Entry

a661.evt_strlen Unsigned 16-bit integer Event String Length

a661.evt_sync_data1 Unsigned 32-bit integer Sync Data Value 1

a661.evt_sync_data2 Unsigned 32-bit integer Sync Data Value 1

a661.evt_sync_datatype Unsigned 8-bit integer Sync Data Type

a661.evt_sync_linkid Unsigned 16-bit integer Sync Link Id

a661.evt_value Unsigned 32-bit integer Event Value

a661.except_keyword Unsigned 16-bit integer Exception Type

a661.mapitem_axis_ratio Unsigned 32-bit integer Map Item Axis Ratio

a661.mapitem_blk_num Unsigned 8-bit integer MapItem Block Number

a661.mapitem_clear Unsigned 16-bit integer Map Item Clear Flag

a661.mapitem_color Unsigned 8-bit integer Map Item Color

a661.mapitem_crschg Signed 32-bit integer Map Item Course Change

a661.mapitem_endflag Unsigned 8-bit integer Map Item End Flag

a661.mapitem_fill_style Unsigned 8-bit integer Map Item Fill Style Index

a661.mapitem_inbnd_crs Signed 32-bit integer Map Item Inbound Course

a661.mapitem_index Unsigned 16-bit integer Map Item Index

a661.mapitem_last_blk Unsigned 8-bit integer Last Block Boolean

a661.mapitem_legend_str String Map Item Legend String

a661.mapitem_length Signed 32-bit integer Map Item Length

a661.mapitem_num Unsigned 16-bit integer Map Item Num of Elements

a661.mapitem_orientation Signed 32-bit integer Map Item Orientation

a661.mapitem_radius Signed 32-bit integer Map Item Radius

a661.mapitem_relative_pos Unsigned 8-bit integer Map Item Relative Position

a661.mapitem_styleset Unsigned 16-bit integer Map Item Style Set

a661.mapitem_symbol_type Unsigned 16-bit integer Map Item Symbol Type

a661.mapitem_sync_data1 Unsigned 32-bit integer Map Item Sync Data 1

a661.mapitem_sync_data2 Unsigned 32-bit integer Map Item Sync Data 2

a661.mapitem_sync_type Unsigned 8-bit integer Map Item Sync Type

a661.mapitem_type Unsigned 8-bit integer Map Item Type

a661.mapitem_update_num Unsigned 8-bit integer Sequence of Block

a661.mapitem_x_lat_range Signed 32-bit integer Map Item X Lat Range

a661.mapitem_x_lat_range2 Signed 32-bit integer Map Item X Lat Range 2

a661.mapitem_y_lat_bearing Signed 32-bit integer Map Item Y Long Bearing

a661.mapitem_y_lat_bearing2 Signed 32-bit integer Map Item Y Long Bearing 2

a661.notify_keyword Unsigned 16-bit integer Notitication Type

a661.parm_1byte_value Unsigned 8-bit integer SET PARAMETER: value 1 byte

a661.parm_2byte_value Unsigned 16-bit integer SET PARAMETER: value 2 byte

a661.parm_4byte_value Unsigned 32-bit integer SET PARAMETER: value 4 byte

a661.parm_8byte_value1 Unsigned 32-bit integer SET PARAMETER: value1 8 byte

a661.parm_8byte_value2 Unsigned 32-bit integer SET PARAMETER: value2 8 byte

a661.parm_enable Unsigned 8-bit integer SET PARAMETER: enable

a661.parm_entry_idx Unsigned 8-bit integer SET PARAMETER: entry index

a661.parm_id Unsigned 16-bit integer CMD SET PARAMETER: id

a661.parm_num_of_str Unsigned 16-bit integer SET PARAMETER: number of strings

a661.parm_str String SET PARAMETER: string data

a661.parm_str_idx Unsigned 16-bit integer SET PARAMETER: string index

a661.parm_str_size Unsigned 16-bit integer SET PARAMETER: string size

a661.req_keyword Unsigned 16-bit integer CMD UA REQUEST: Request Type

a661.req_widget_id Unsigned 16-bit integer CMD UA REQUEST: Widget ID

a661.sym_alignment Unsigned 8-bit integer Text Alignment

a661.sym_closed Unsigned 8-bit integer Symbol Number of Vertices

a661.sym_color_idx Unsigned 8-bit integer Symbol Color Index

a661.sym_endangle Unsigned 32-bit integer Symbol End Angle

a661.sym_endx Unsigned 32-bit integer Symbol End X Position

a661.sym_endy Unsigned 32-bit integer Symbol End Y Position

a661.sym_filled Unsigned 8-bit integer Symbol Filled

a661.sym_font_idx Unsigned 8-bit integer Symbol Font Index

a661.sym_halo Unsigned 8-bit integer Symbol Halo

a661.sym_inner_radius Unsigned 32-bit integer Symbol Inner Radius

a661.sym_line_style Unsigned 16-bit integer Symbol Line Style

a661.sym_linelength Unsigned 32-bit integer Symbol Line Length

a661.sym_num_of_vertices Unsigned 16-bit integer Symbol Number of Vertices

a661.sym_outer_radius Unsigned 32-bit integer Symbol Outer Radius

a661.sym_radius Unsigned 32-bit integer Symbol Circle Radius

a661.sym_rotation_angle Unsigned 32-bit integer Symbol Rotation Angle

a661.sym_size_x Unsigned 32-bit integer Symbol X Size

a661.sym_size_y Unsigned 32-bit integer Symbol Y Size

a661.sym_startangle Unsigned 32-bit integer Symbol Start Angle

a661.sym_startx Unsigned 32-bit integer Symbol Start X Position

a661.sym_starty Unsigned 32-bit integer Symbol Start Y Position

a661.sym_x Unsigned 32-bit integer Symbol X Position

a661.sym_x2 Unsigned 32-bit integer Symbol X2 Position

a661.sym_x3 Unsigned 32-bit integer Symbol X3 Position

a661.sym_y Unsigned 32-bit integer Symbol Y Position

a661.sym_y2 Unsigned 32-bit integer Symbol Y2 Position

a661.sym_y3 sym_y3 Unsigned 32-bit integer Symbol Y3 Position

- **EFIS Buffers**.

efis.baro_hpa_747-400 Unsigned 32-bit integer Baro HPA (747-400)

efis.baro_in_747-400 Unsigned 32-bit integer Baro In (747-400)

efis.baro_metric_747-400 Unsigned 32-bit integer Baro Metric (747-400)

efis.conic_angle Conic Definition Word (Subtended Angle)

efis.conic_init_angle Conic Definition Word (Init Angle)

efis.conic_radius Conic Definition Word (Radius)

efis.dyn_state Unsigned 32-bit integer Start of Dynamic Data

efis.eot Unsigned 32-bit integer EOT

efis.fill_in Unsigned 32-bit integer Fill-In Word

efis.fpv_747-400 Unsigned 32-bit integer FPV (747-400)

efis.label Unsigned 8-bit integer

efis.lat Vector Latitude Word

efis.lat_fine Vector Latitude Word

efis.long_fine Vector Latitude Word

efis.metric_alt_747-400 Unsigned 32-bit integer Metric Alt (747-400)

efis.mode_approach Unsigned 32-bit integer Mode Approach

efis.mode_map Unsigned 32-bit integer Mode Map

efis.mode_plan Unsigned 32-bit integer Mode Plan

efis.mode_pos Unsigned 32-bit integer Mode Pos

efis.mode_sta Unsigned 32-bit integer Mode STA

efis.mode_vor Unsigned 32-bit integer Mode VOR

efis.mode_vor_adf Unsigned 32-bit integer Mode VOR/ADF

efis.mode_word Unsigned 32-bit integer Mode Word

efis.mode_wpt Unsigned 32-bit integer Mode WPT

efis.range_dist_747-400 Unsigned 32-bit integer Range Dist (747-400)

efis.range_dist_747-8 Unsigned 32-bit integer Range Dist (747-8)

efis.range_word_747_400 Unsigned 32-bit integer Range Word

efis.range_word_747_8 Unsigned 32-bit integer Range Word

efis.sdi Unsigned 8-bit integer

efis.sot Unsigned 32-bit integer SOT

efis.sot_blk_cnt Unsigned 32-bit integer SOT Block Count

efis.sot_word_cnt Unsigned 32-bit integer SOT Word Count

efis.ssm Unsigned 8-bit integer

efis.sym_azimuth Symbol Azimuth Word (rotated symbols only)

efis.sym_dist_747-400 distance 747-400 Distance Word (747-400)

efis.sym_dist_747-8 Distance Word (747-8)

efis.sym_rwy_leng Symbol Length Word (runway symbol only)

efis.text_id String Text ID String

efis.word Unsigned 32-bit integer efis Data

efis.wxr_data_747-400 Unsigned 32-bit integer Wxr Data (747-400)

- **Arinc 739**.

a739.ack Unsigned 32-bit integer ACK Word

a739.bg Unsigned 32-bit integer Background Word

a739.cmd Unsigned 32-bit integer Command Code

a739.cntrl Unsigned 32-bit integer Cntrl Word

a739.cntrl_color Unsigned 32-bit integer Line Number

a739.cntrl_flash Unsigned 32-bit integer Flashing

a739.cntrl_init_char Unsigned 32-bit integer Init Char Position

a739.cntrl_line_num Unsigned 32-bit integer Line Number

a739.cntrl_underscore Unsigned 32-bit integer Underscore

a739.cts Unsigned 32-bit integer CTS Word

a739.data Unsigned 32-bit integer Data Word

a739.data_str String Data Word String

a739.disc Unsigned 32-bit integer Discrete Word

a739.disc_blank_screen Unsigned 32-bit integer Blank Screen

a739.disc_clr_acars Unsigned 32-bit integer Clear ACARS Request

a739.disc_clr_disp Unsigned 32-bit integer Clear Display Buffer

a739.disc_clr_fmc Unsigned 32-bit integer Clear FMC Request

a739.disc_clr_sysstem Unsigned 32-bit integer Clear Subsystem Request

a739.disc_dspy_ann Unsigned 32-bit integer DSPY Annunciator

a739.disc_exec_ann Unsigned 32-bit integer EXEC Annunciator

a739.disc_msg_ann Unsigned 32-bit integer MSG Annunciator

a739.disc_ofst_ann Unsigned 32-bit integer OFST Annunciator

a739.disc_self_test Unsigned 32-bit integer Self Test

a739.disc_ssm Unsigned 32-bit integer SSM

a739.disc_test_req Unsigned 32-bit integer MCDU Test Request

a739.eicas Unsigned 32-bit integer EICAS Discrete Word

a739.eicas_mcdu_port Unsigned 32-bit integer EICAS MCDU Input Port

a739.enq Unsigned 32-bit integer ENQ Word

a739.enq_mal Unsigned 32-bit integer ENQ MAL

a739.eot Unsigned 32-bit integer EOT Word

a739.etx Unsigned 32-bit integer ETX Word

a739.func Unsigned 32-bit integer Function

a739.label Unsigned 32-bit integer Label

a739.mal_label Unsigned 32-bit integer MAL Label

a739.max_rec_cnt Unsigned 32-bit integer MAX Record Count

a739.mcdu_id Unsigned 32-bit integer MCDU ID Word

a739.mcdu_status Unsigned 32-bit integer MCDU Status

a739.nak Unsigned 32-bit integer NAK Word

a739.pb Unsigned 32-bit integer Push Button Word

a739.pb_code Unsigned 32-bit integer Push Button Code

a739.pb_seq_num Unsigned 32-bit integer Push Button Sequence Number

a739.port1_data Unsigned 32-bit integer Port 1 Data In

a739.port1_rcv Unsigned 32-bit integer Port 1 Receiver

a739.port2_data Unsigned 32-bit integer Port 2 Data In

a739.port2_rcv Unsigned 32-bit integer Port 2 Receiver

a739.port3_data Unsigned 32-bit integer Port 3 Data In

a739.port3_rcv Unsigned 32-bit integer Port 3 Receiver

a739.rec_cnt Unsigned 32-bit integer Record Count

a739.rts Unsigned 32-bit integer RTS Word

a739.sal_label Unsigned 32-bit integer SAL Label

a739.scratchpad Unsigned 32-bit integer Scratch Pad Word

a739.seq_num Unsigned 32-bit integer Record Sequence Number

a739.stx Unsigned 32-bit integer STX Word

a739.stx_seq_num Unsigned 32-bit integer STX Sequence Number

a739.stx_wd_cnt Unsigned 32-bit integer STX Word Count

a739.subsystem_id Unsigned 32-bit integer Subsystem ID Word

a739.subsystem_id_sal Unsigned 8-bit integer Subsystem ID SAL

a739.syn Unsigned 32-bit integer SYN Word

a739.vector Unsigned 32-bit integer Vector Word

a739.word Unsigned 32-bit integer Word

a739.word_cnt Unsigned 32-bit integer Word Count

* **M1553.**

m1553.broadcast_rcv Boolean

m1553.busy Boolean

m1553.cmd_word Unsigned 16-bit integer

m1553.data_cnt Unsigned 16-bit integer

m1553.dynam_bus_cntr_accept Boolean

m1553.instr Boolean

m1553.mode Unsigned 16-bit integer

m1553.mode_code Unsigned 16-bit integer

m1553.mode_word Unsigned 16-bit integer

m1553.msg_err Boolean

m1553.msg_type Unsigned 16-bit integer

m1553.rt_addr Unsigned 16-bit integer

m1553.serv_reg Boolean

m1553.stat_rt_addr Unsigned 16-bit integer

m1553.stat_word Unsigned 16-bit integer

m1553.sub_address Unsigned 16-bit integer

m1553.subsys_flag Boolean

m1553.t_r Boolean

- **M1553_<subaddr id>_[tx|rx]*** (M1553 payload packets). These protocols are defined based on data contained in data definition files.

* **M1553 Generic.**

m1553.word[1-32] Unsigned 16-bit integer

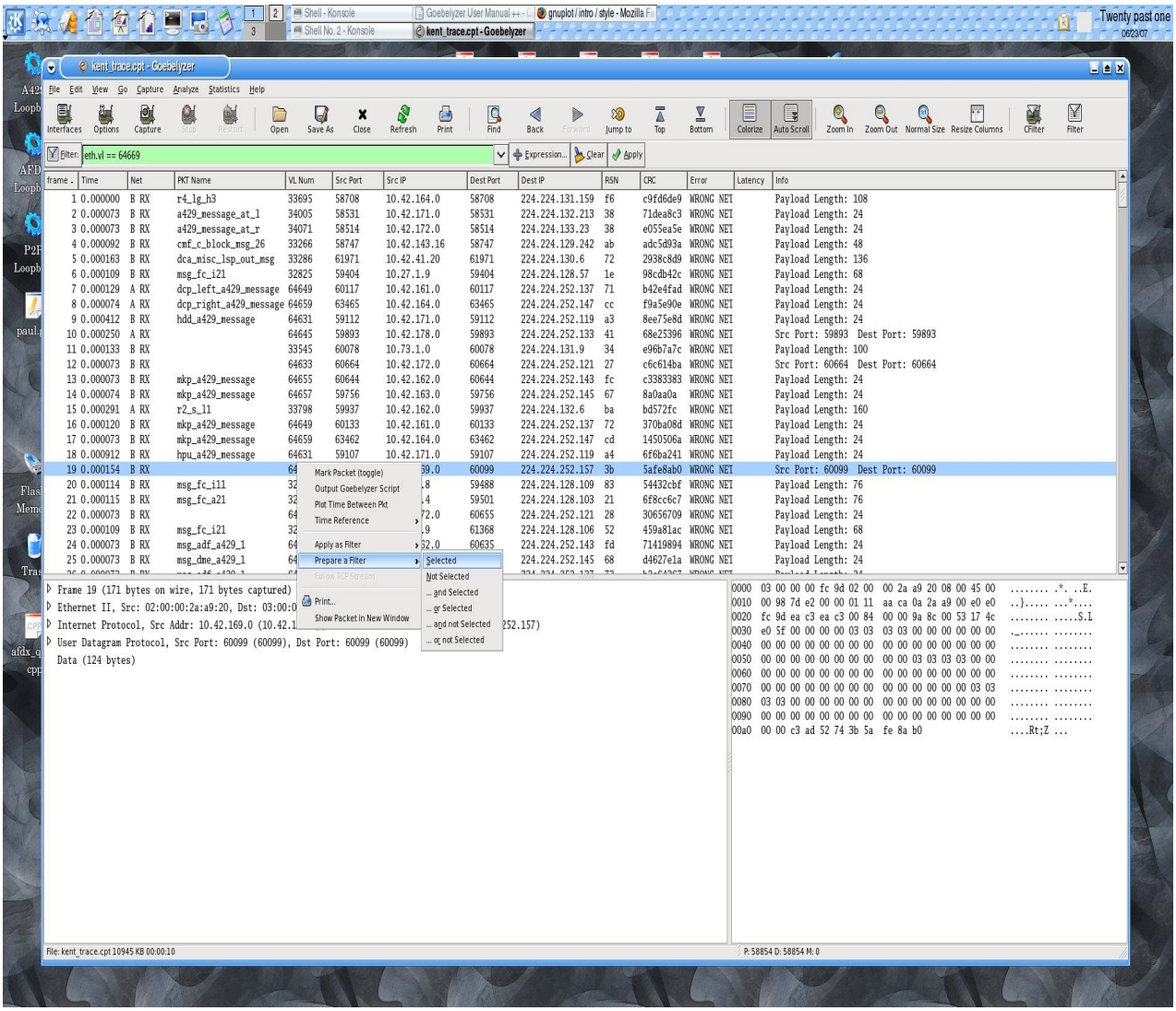
6.4 Examples:

- (eth.vl == 1212) and (udp.dstport > 2323) –filter packets such that only packets displayed will have a vl entry of 1212 and destination port greater than 2323.
- frame.pkt_len > 100 -- display only packets with a length greater than 100.
- frame.number > 10 and frame.number < 20 – display only the 10-20 packet listed entries.
- ip.addr == 123.124.125.126 and udp.srcport == 65000. -- Display all packets with an ip address of 123.124.125.126 and udp source port of 65000.
- a429.label == 0376 or a429.label == 0377 – display all a429 packets with a label of octal 376/377.
- p2phdr.label == 0x1234 – display p2p packets with a port/label of 0x1234.

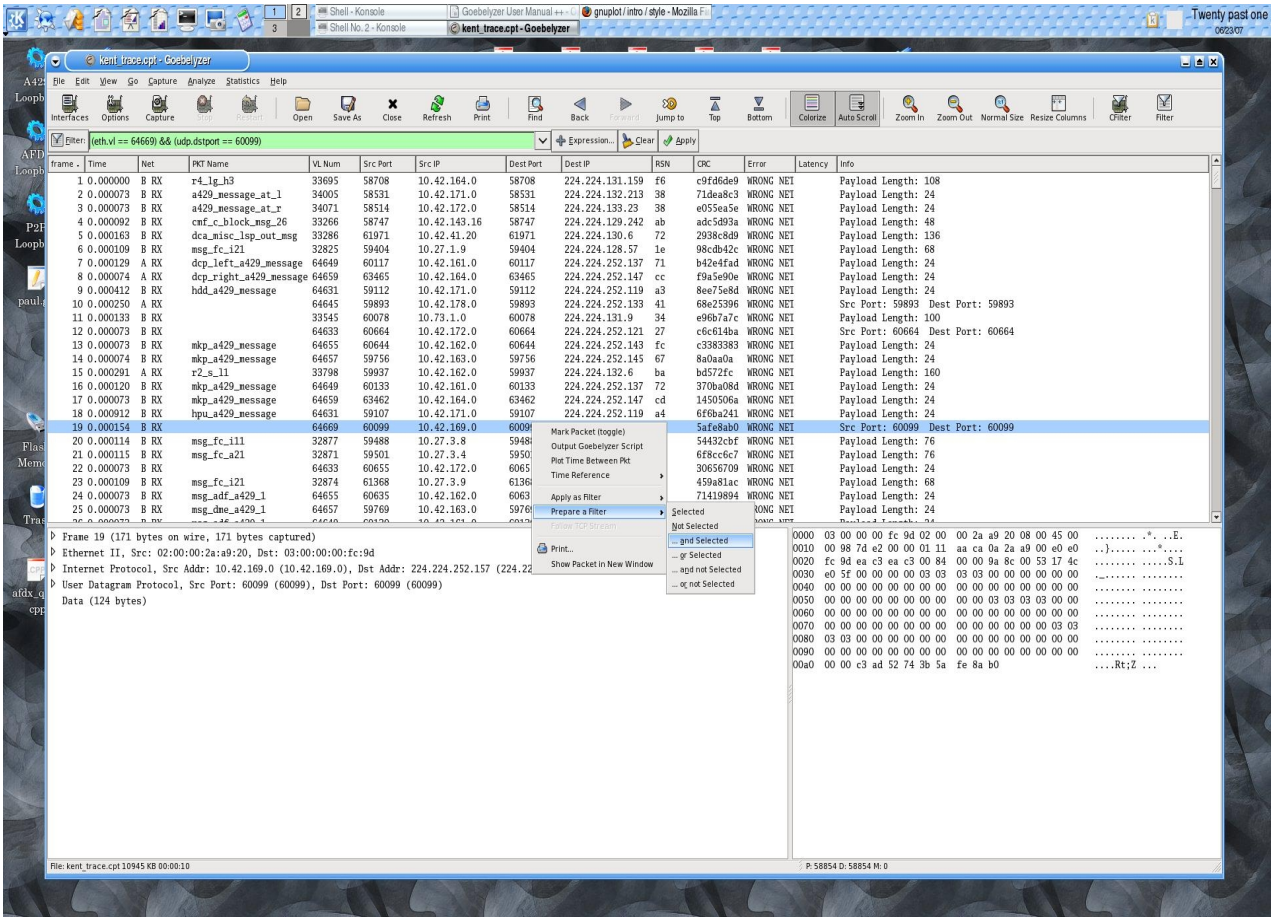
The user may right click in various columns (vl, src/dst port src/dst ip)of the packet window to create display filters. This will allow users to create display filters from data contained in the packet window.

There are 2 options: Apply as Filter, Prepare as Filter in the right click menu. The Apply option menu will execute the updates immediately (same depressing the Display Filter Apply Button). The Prepare option menu will add changes to the display filter string without execution. This will allow the user to add more expressions to the filter before execution.

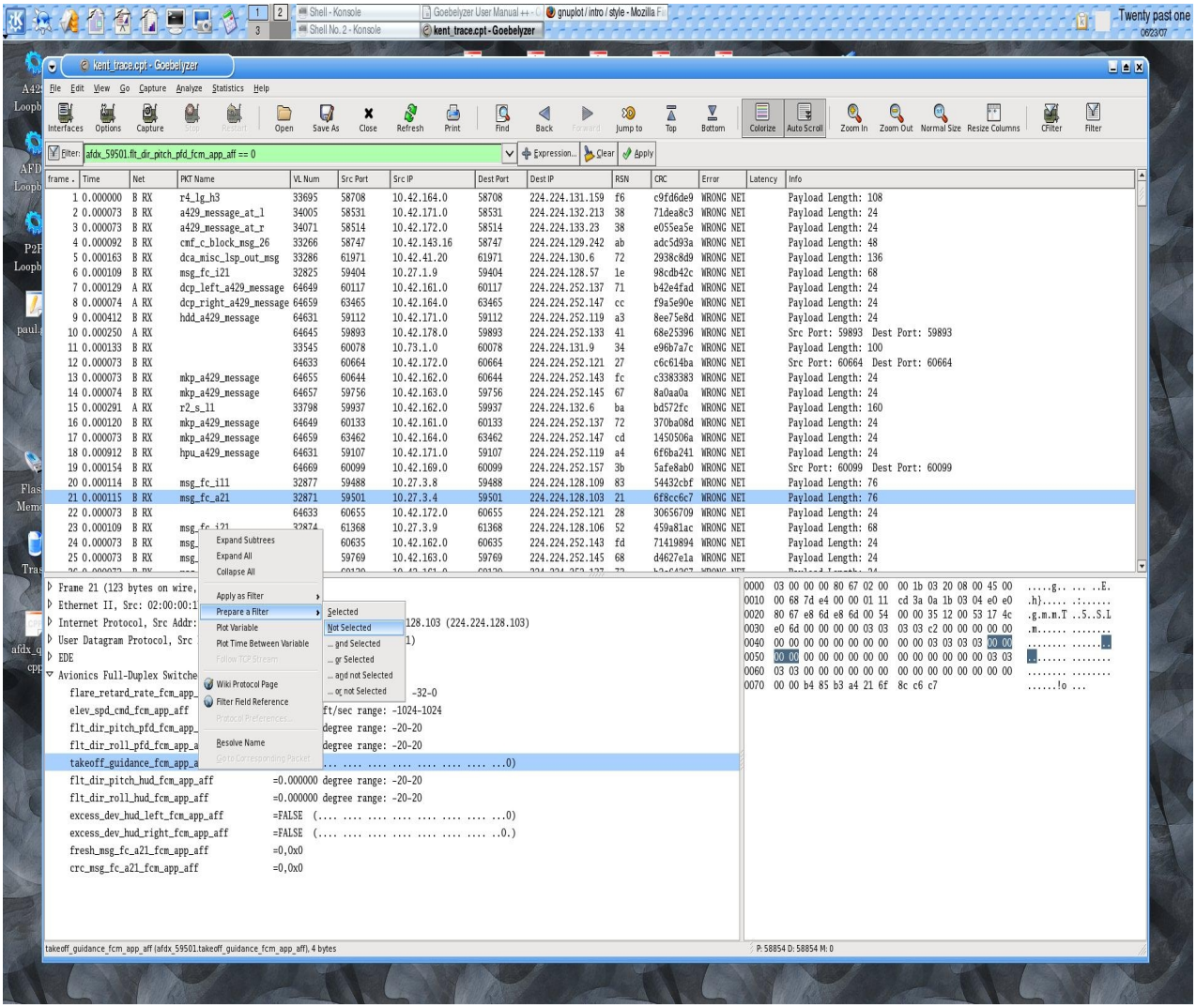
A download menu from the Apply/Prepare entries will allow the addition of relational operators NOT, AND, OR, NOT AND, and NOT OR. As shown Below. In the first figure, the user move to a vl column, right clicks and selects Prepare a Filter->Selected.



Then the user moves to a destination port column, right clicks and selects Prepare a Filter->... and Selected. At this point the filter can be executed by depressing the Apply button.



The user may also right click in the protocol tree to execute a display filter using payload protocol information.

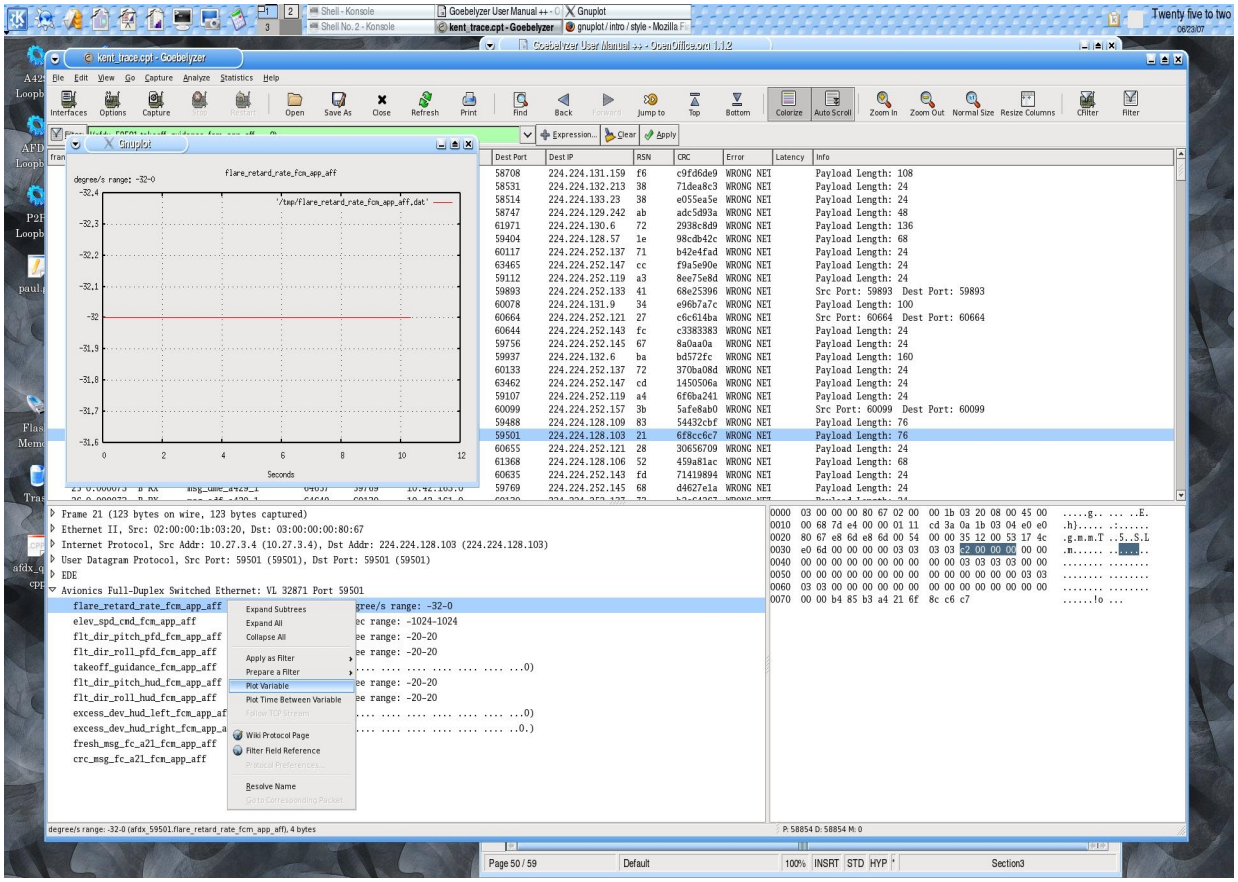


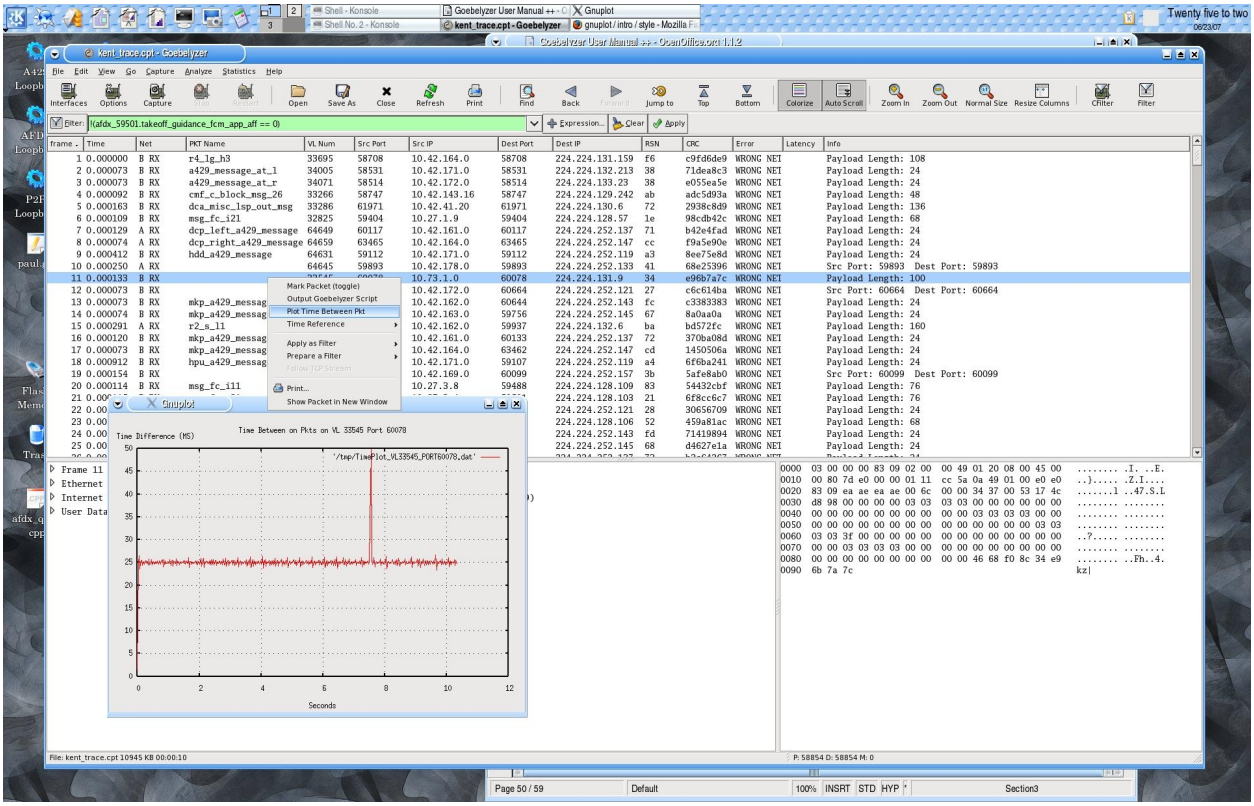
7. Plotting.

The user can plot the value of specific payload data for the all samples in the displayed trace. The user will right click on “Plot Variable(s)” option in the protocol tree on the payload data to be display As shown below.

The user can also create multiple variable plots using the “Add Variable to Multi-Plot” option.

Variables are added using the “add variable to multi-plot” option with the last to plot selected using the “Plot Variable(s)” option.





The user can plot the time between packet samples for all packet of the same type contained in the trace. The user will right click in the packet window on specific packet executing the "Time between packets" option. This is show above.

The data for the plot is contained in a file in the /tmp directory. The .plt file is a script to display the data contained in the .dat file. The filename is created using the data name or packet data. The user can use the command *"ls -lrt /tmp"* to view the latest file contained in the directory.

8. Data Definition Files.

These files define data to display packet elements in various formats. The analyzer will use these files to display packet elements in the format defined in the file. Data definition files are located in the following directories:

- AFDX - /home/user/.ethereal.a664
- P2P – /home/user/.ethereal.p2p.
- A429 – /home/user/.ethereal.a429.
- CAN - /home/user/.ethereal.can.
- M1553 - /home/user/.ethereal.m1553.
- Spacewire - /home/user/.ethereal.spwr
- FSCC - /home/user/.ethereal.fscc
- ALLDEV – /home/user/.ethereal.alldev.

8.1 AFDX XML Data File.

This section describes the some the details of format for the~user/.ethereal.a664/packet-afdx.xml file.

The packet-afdx.xml configuration file is used to provide afdx payload definition information to the user. The format is simple and compact. Each packet definition consists a <packet definition> and a list (1..N) of <elem definitions>. The <packet definition> defines the high level information about the packet include port, vl, protocol, rate, and length. The <elem definition> defines information about a specific data element of the packet including name, mask, type, offset, length, and info. The file consists of these types in the following repeated format: <packet definition, and N <elem definition>s. This will continue until the end of the file.

8.1.1 Packet Data.

- **name** = name description of packet...string of characters...256 max length
- **port** = The packet is received on this port...number value is decimal number with range between 0 and 65536
- **vl** = The packet is received on this VL number. Format is a decimal between values between 0-65536.
- **ede**= The value of the EDE identifier. A value of 0 will indicate no EDE data is contained in the packet. This field is optional and will default to 0 when no value is specified.
- **protocol** = NONE,A661,A429BLK. This field is optional and used to defined A661 or A429BLK is defined within an afdx packet. The A429BLK is used when the packet defines an A429 messages. The format of the A429BLK packet is:

```
struct a429blk_payload {
  uint32 reserved;
  uint8 fs1;
  uint8 fs2;
```

uint8 fs3;

uint8 fs4;

...A429 blocks containing a 2 byte length, 2byte pad, array of 4 byte A429 words of the requested length.

...possibly more A429 blocks.

} a429blk_payload_t;

- Other protocols may be added in the future.
- **proto_offset** = This field is optional used when a A661 protocol is defined. Start byte location of A661 protocol within packet.

8.1.2 Element Data.

- **name** = name description of element...string of characters...256 max length.
- **mask** = mask used to retrieve data not located on a byte boundary. 0 should be used if the data type uses all bits in the type. Value is hexadecimal number. Examples are 0xffff0000 (upper 16bits of 4 byte word) 0xfffc0, 0x7ffc000, Mask has no meaning for the following: FT_FLOAT,FT_DOUBLE.
- **type=** the following types are supported for AFDX:

types	*description*
FT_FLOAT	4 byte floating point number.
FT_DOUBLE	8-byte floating point number.
FT_UINT8	1 byte unsigned value.
FT_UINT16	2 byte unsigned value.
FT_INT16	2 byte signed (2's complement) value.
FT_UINT32	4 byte unsigned value.
FT_HEX	4 byte value display hexadecimal.
FT_INT32	4 bye signed (2's complement) value.
FT_BOOLEAN	1 bit displayed within a 4 byte word.
FT_BNR	1,2 or 4 byte scaled integer.
FT_UBNR	1,2 or 4 byte unsigned scaled integer.

- **range**= float value...max range of scaling for FT_BNR,FT_UBNR. Not used for all other data types. So, fixed pt types will have one more field in the element definition.
- **offset**= offset in bytes into the payload data where the element is located. Value is decimal number.
- **length**= length in bytes of the data element sizes are listed in the type section for different types. Value is decimal number (1,2 or 4).
- **info**= string containing units of the engineering data (deg/sec, lbs, feet). Max length of 256 bytes.

AFDX XML Example File:

```
<afdxicd>

  <packet name='vl1234' port='1' vl='1' protocol='none' rate='1000' length='20' >

    <elem name='paul' mask='0xffffffff' type='FT_UINT32' offset='0' length='4'  />
```

```
    <elem name='paul1' mask='0xffffffff' type='FT_UINT32' offset='4' length='4'  />
  </packet>
  <packet name='vl2345' port='1' vl='2345' protocol='none' rate='1000' length='20' >
    <elem name='ted' mask='0xffffffff' type='FT_UINT32' offset='0' length='4'  info='deg/sec' />
    <elem name='ted1' mask='0xffffffff' type='FT_UINT32' offset='4' length='4'  />
  </packet>
  ....
</afdxicd>
```

8.2 AFDX VL XML File.

This section describes the file format for the~user/.ethereal.a664/packet-afdx_vl_db.xml file. This file is used to attach a name with a VL number.

The following describes the format of the configuration file used to provide afdx vl data definition to the user.

```
<afdxvlicd>
  <vl name='vl1' vl='2003' bag='4'  />
  <vl name='vl2' vl='2203' bag='4'  />
  <vl name='vl3' vl='2403' bag='4'  />
</afdxvlicd>
```

- **name** = vl name description of packet...string of characters...256 max number of bytes.
- **vl** = The number corresponding to the vl name value is decimal number with range between 0 and 65536.
- **bag** = bag value of vl between 0 and 256.

8.3 A429 XML Data File.

This section describes the some the details of format for the ~user/.ethereal.a429/packet-a429.xml file.

The packet-a429.xml configuration file is used to provide a429 label definition information to the user. The format is simple and compact. Each label definition consists a <label definition> and a list (1..N) of <elem definitions>. The <label definition> defines the high level information about the label include port, vl, protocol, rate, and length. The <elem definition> defines information about a specific data element of the packet including name, mask, type, offset, length, and info. The file consists of these types in the following repeated format: <packet definition, and N <elem definition>s. This will continue until the end of the file.

8.3.1 Label Data.

- **name** = name description of packet...string of characters...256 max length.
- **label** = The label number. Format is an octal value between 0-377.
- **sdi** = the sdi of the label. A value of 'XX' indicates no sdi. Otherwise in the range of 0 to 3.
- **rate** = rate of label in milliseconds.

8.3.2 Element Data.

- **name** = name description of element...string of characters...256 max length
- **mask** = mask used to retrieve data not located on a byte boundary. 0 should be used if the data type uses all bits in the type. Value is hexadecimal number. Examples are 0xffff0000 (upper 16bits of 4 byte word) 0xfffc0, 0x7ffc000.
- **type=** the following types are supported for A429:

types	*mask used*	*description*
FT_BNR	yes	4-byte scaled integer value.
FT_UBNR	yes	4-byte scaled unsigned value.
FT_UINT8	yes	1-byte unsigned value.
FT_CHAR	no	8-bit ascii value.
FT_ISO5	no	7-bit ascii value.
FT_BOOLEAN	yes	1-bit boolean value within a 4-byte word.
FT_UINT16	yes	2-byte unsigned value.
FT_UINT32	yes	4-byte unsigned value.
FT_BCD	yes	Binary coded decimal value

- **range**= float value...max range of scaling for FT_BNR,FT_UBNR. Not used for all other data types. So, fixed pt types will have one more field in the element definition.
- **offset**= offset in bytes into the payload data where the element is located. Value is decimal number
- **length**= length in bytes of the data element sizes are listed in the type section for different types. Value is decimal number.
- **info**= string containing units of the engineering data (deg/sec, lbs, feet).

A429 XML Example File:

```
<a429icd >

 <label name='paul_w203' label='203' sdi='XX' rate='50' >

  <elem name='w203_label' mask='0xff' type='FT_UINT8' offset='3' length='1' />

  <elem name='data' mask='0x1ffff800' type='FT_BNR' range='131072.000000' offset='0' length='4' info='range
-131072..131072' />

 </label>

 <label name='paul_w204' label='204' sdi='XX' rate='50' >

  <elem name='w204_label' mask='0xff' type='FT_UINT8' offset='3' length='1' />

  <elem name='data1' mask='0x1ffff800' type='FT_BNR' range='13.0' offset='0' length='4' info='range -13.0..13.0' />

</a429icd>
```

8.4 A429 Bus Data File.

This section describes the file format for the packet-a429_buses.dat file.

The packet-a429_buses.dat configuration file is used to provide a429 bus definition data to the analyzer. These values are correlated with the packet name described in the packet-a429.dat file. The packet name will consist of the A429 bus name with a "_w<label>" appended to the end. Examples include paul_a429_bus_w132, bill_a429_bus_w376.

This will allow the user to select a specific bus to be received on a selected receive channel in the A429 option notebook page.

➢ <A429 bus name>

➢ ...

➢ <EOF>

8.5 A429 Equipment ID Data File.

This section describes the file format for the packet-eqid.dat file.

The a429-eqid.xml configuration file is used to provide a429 equipment ID definition data to the analyzer. The format of the a429-eqid.xml file is same as packet-a429.xml with the label defition containing the added field of "EQ_ID". This file will be provided by the Goebel Company as part of a standard installation.

Each equipment ID decimal value corresponds to avionics unit/black box. Standard labels are defined for each equipment ID. These equipment Ids data strings are correlated with the packet name described in the a429-eqid.dat file. The packet name will consist of the A429 equipment id name with a "_w<label>" appended to the end. Examples include flight_control_computer_w132, flight_management_computer_w376.

This will allow the user to select a specific equipment id to be received on a selected receive channel in the A429 option notebook page.

8.6 P2P XML Data File.

This section describes the file format for the ~user/.ethereal.p2p/packet-p2p.xml file.

The following describes the format of the configuration file used to provide p2p data definition to the user. Each label definition consists a <label definition> and a list (1..N) of <element definitions>. The <label definition> defines the high level information about the label. The <element definition> defines information about a specific data element of the packet. The file consists of these types in the following repeated format: label definition and N element definitions. This will continue until the end of the file.

8.6.1 Label Data.

- **name** = name description of packet...string of characters...256 max length.
- **label** = The label number. Format is an 16bit hex value.
- **length** = length of packet in bytes.
- **rate** = rate of label in milliseconds.

8.6.2 Element Data.

- **name** = name description of element...string of characters...256 max length
- **mask** = mask used to retrieve data not located on a byte boundary. 0 should be used if the data type uses all bits in the type. Value is hexadecimal number. Examples are 0xffff0000 (upper 16bits of 4 byte word) 0xfffc0, 0x7ffc000, Mask has no meaning for the following: FT_FLOAT,FT_DOUBLE

- **<type>** = element type.

  The following types are supported for P2P:

types	*mask used*	*description*
FT_FLOAT	no	4 bytes floating point number.
FT_DOUBLE	no	8-byte floating point number.
FT_UINT16	yes	2-byte unsigned value.
FT_INT16	yes	2-byte signed (2's complement) value.
FT_UINT32	yes	4-byte unsigned value.
FT_HEX	yes	2-byte unsigned value displayed as hexadecimal.
FT_INT32	yes	4-byte signed (2's complement) value.
FT_BOOLEAN	yes	1 bit boolean contain in 2 byte value.
FT_BNR	yes	2-byte unsigned integer value.
FT_UBNR	yes	2 byte scaled integer value.
FT_UINT8	yes	1 byte unsigned value.

- **range**= float value...max range of scaling for FT_BNR,FT_UBNR. Not used for all other data types. So, fixed pt types will have one more field in the element definition.
- **offset**= offset in bytes into the payload data where the element is located. Value is decimal number
- **length**= length in bytes of the data element sizes are listed in the type section for different types. Value is decimal number.
- **info**= string containing units of the engineering data (deg/sec, lbs, feet).

P2P XML Example File:

```
<p2picd>
  <packet name='packet1' label='0x1f10' rate='10' length='86' >
    <elem name='data1' mask='0xff' type='FT_UINT16' offset='0' length='2' />
    <elem name='data2' mask='0xff00' type='FT_UINT16' offset='0' length='2' />
    <elem name='data3' mask='0xff00' type='FT_UINT16' offset='2' length='2' />
    <elem name='data4' mask='0xffff' type='FT_UINT16' offset='4' length='2' />
    <elem name='data5' mask='0xffff' type='FT_UINT16' offset='6' length='2' />
  </packet>
</p2picd>
```

8.7 M1553 XML Data File.

This section describes the file format for the ~user/.ethereal.p2p/packet-m1553.xml file.

The following describes the format of the configuration file used to provide m1553 data definition to the user. Each packet definition consists a <packet definition> and a list (1..N) of <element definitions>. The <packet definition> defines the high level information about the packet. The <element definition> defines information about a specific data element of the packet. The file consists of these types in the following repeated format: packet definition and N pkt element definitions. This will continue until the end of the file.

8.7.1 Label Data.

- **name** = name description of packet...string of characters...256 max length.
- **rt_addr** = remote terminal address value.
- **length** = subaddress value.
- **dir** = direction of the packet either 'rx' or 'tx'.
- **rate** = rate of label in milliseconds.
- **length** = length of the packet in bytes.

8.7.2 Element Data.

- **name** = name description of element...string of characters...256 max length
- **mask** = mask used to retrieve data not located on a byte boundary. 0 should be used if the data type uses all bits in the type. Value is hexadecimal number. Examples are 0xffff0000 (upper 16bits of 4 byte word) 0xfffc0, 0x7ffc000, Mask has no meaning for the following: FT_FLOAT,FT_DOUBLE
- **<type>** = element type.

  The following types are supported for M1553:

types	*mask used*	*description*
FT_FLOAT	no	4 bytes floating point number.
FT_DOUBLE	no	8-byte floating point number.
FT_UINT16	yes	2-byte unsigned value.
FT_INT16	yes	2-byte signed (2's complement) value.
FT_UINT32	yes	4-byte unsigned value.
FT_HEX	yes	2-byte unsigned value displayed as hexadecimal.
FT_INT32	yes	4-byte signed (2's complement) value.
FT_BOOLEAN	yes	1 bit boolean contain in 2 byte value.
FT_BNR	yes	2-byte unsigned integer value.
FT_UBNR	yes	2 byte scaled integer value.
FT_UINT8	yes	1 byte unsigned value.

- **range**= float value...max range of scaling for FT_BNR,FT_UBNR. Not used for all other data types. So, fixed pt types will have one more field in the element definition.
- **offset**= offset in bytes into the payload data where the element is located. Value is decimal number
- **length**= length in bytes of the data element sizes are listed in the type section for different types. Value is decimal number.
- **info**= string containing units of the engineering data (deg/sec, lbs, feet).

M1553 XML Example File:

```
<m1553icd>
 <packet name='subaddr_15a_rx' rt_addr='16' subaddr='15' dir='rx' rate='1000' length='64' >
  <elem name='paul1' mask='0x1' type='FT_BOOLEAN' offset='0' length='2' />
```

```
  <elem name='paul2' mask='0x2' type='FT_BOOLEAN' offset='0' length='2' />
 </packet>
</m1553icd>
```

9. Frequently Asked Questions

- ➔ Can I talk with a live person (not in a foreign country) about questions I may have? YES, the Goebel Company (support@goebeletc.com, or 206.601.6010) will be happy to answer any question that the user may have.
- ➔ How do I start the execution of the analyzer?

  On the Desktop, there will be icons for each interface and program. The title should be of the form Goebelyzer <interface> <program> where interface is afdx,a429,p2p,m1553,alldev...
- ➔ Where is the analyzer ICD configuration files located?

  The ICDs are organized in directories under the */home/user/icd* directory. Each ICD corresponds to an aircraft and data creation date.
- ➔ How do I select an ICD?

  The "ICD Selection" icon on the desktop can be used to select the data defines that the analyzer will load on startup.
- ➔ Where are the analyzer configuration information files located?

  The files are located in the home directory of user at */home/user/.ethereal.<interface>* where the interface is a429,a664,p2p,can, or alldev.
- ➔ How can I view the boolean state of bit=15 of long word=20 (80byte offset) in an AFDX packet with vl=1212 and destination port=4545 on a periodic basis?

  1. The user must have an entry in the packet-afdx.xml configuration file for this data. This is described in section #8.1AFDX Data File.|outline. It would be helpful if all data for a program was defined in the config file. But as an example, the following entry would need to be in the packet-afdx.xml file.

```
<packet name='pkt=-fms_10ms' port='4545' vl='1212' protocol='none' rate='1000' length='20' >

  <elem name='boolean_value' mask='0x8000' type='FT_BOOLEAN' offset='80' length='4'  />

</packet>
```

  2. Start execution of the analyzer.
  3. Depress the Options button to display the Options dialog. Select AFDX RTD Notebook. Double click the entry pkt_fmt_10ms->boolean_value. Start Capture. This is described in section #5.7Real-Time Display Option Pages.
  4. Select the AFDX RTD Notebook page. The value will be updated on a 1 sec interval when the packet is received. This is described in section #6.3Data Display Page..
- ➔ How can I capture data only from udp destination ports 2323 and 3434 on an AFDX device?

  1. Start the analyzer with the AFDX device selected.

  2. A capture filter of "udp dst port 2323 or udp dst port 3434" could be used. This is described in section #5.1.5Capture Frame. Then the capture button is depressed to start the capture.

3. OR perform a normal capture and enter a display filter (display filter entry is located below the menu) of udp.dstport == 2323 or udp.dstport == 3434. .Display filters are to eliminate packets from the capture display that do not return a True result to the filter expression. In this case, an udp dst port of 2323 or 3434. The display filter can be with all items defined in the configuration file. So, a user may be able to view unique payload information. Packets with airspeed < 200, or altitude > 2500 or both using and/or/not boolean terms to combine boolean conditions into a filter with multiple conditions.

➔ How can I capture all AFDX packets with a 0x12 at payload byte 4 after udp port 4545 is received with a 0x2323 at payload byte 10?

This is performed by entering a trigger entry of "udp port 4545 and udp[18:2] == 0x2323" and a capture filter of "udp[20:1] == 0x12". The afdx payload begins at 8 byte into the udp header. So, all afdx payload entries will be +8 bytes. Triggers and Capture Filters are described in section #5.1.5Capture Frame.