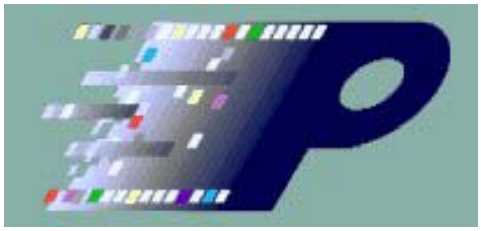

CPhyScopeDecoder User's Manual

Table of Contents

Contacting The *Moving Pixel* Company

Phone	+1.503.626.9663	US Pacific Time Zone
Fax	+1.503.626.9653	US Pacific Time Zone
Address	**The *Moving Pixel* Company** 4905 SW Griffith Drive, Suite 106 Beaverton, Oregon 97005 USA	
Email	information@movingpixel.com	
Web site	http://www.movingpixel.com	

Documentation

1 Overview

The Moving Pixel Company CPhyScopeDecoder software is a single-lane CPhy and CSI2 protocol decoder from CPhy signal acquisitions from an oscilloscope.[1] The software runs on any WinXP or Win7 host that is connected to the oscilloscope via a LAN, using the remote-control capability of the scope to control real-time acquisition.

The software's main functions are to:

- Provide real-time scope acquisition and control of one CPhy lane using three channels. Alternatively, saved binary waveform files can be loaded and disassembled.
- Post-process the acquisition data to provide DPhy/DSI/CSI2 protocol disassembly views of communication on the link.[2] The views provided are similar in look-and-feel to a logic analyzer type display.
- Provide extensive functions and manipulations for viewing, filtering, and searching captured data.
- Build video frames from decoded packets, including a frame summary listing that provides statistics, navigation, viewing and saving of images.
- Check and report many types of errors, including illegal state transitions, invalid symbol sequences, packet header and payload CRC errors, etc.
- Correlate any event in the disassembly back to acquired waveforms on the scope using the zoom window and cursors.

[1] The software supports both Agilent's Infiniium 90000 and compatible oscilloscopes as well as Tektronix oscilloscopes.

[2] While, in principle, the software can support DSI acquisition and disassembly, in practice, DSI is not yet defined (and may never be defined) for CPhy. The manual text, originally written for The Moving Pixel Company's DPhy decoder product, still refers to DSI-centric information that is not relevant to the current software.

2 Setup & Installation

This section describes the steps for installing CPhyScopeDecoder on the host computer, which should be a Windows machine running Windows XP or Windows 7 (32-bit or 64-bit). As CPhyScopeDecoder is memory and processing intensive, the computer should have a fast processor and lots of memory (at least 4 GB are recommended).

2.1 Application Installation

While the application can be installed directly on the oscilloscope, this is not a preferred configuration. The user experience is much improved having a larger screen display available for disassembly as well as the ability to view both the scope trace and disassembly at the same time.

To install CPhyScopeDecoder, simply execute the setup.exe file provided and step through the installation windows. The CPhyScopeDecoder application is installed by default in the **c:\Program Files\TMPC\CPhyScopeDecoder** directory (or **c:\Program Files (x86)\TMPC\CPhyScopeDecoder** on a 64-bit machine).

After the installer has run, shortcuts are added to the Start menu under Start->All Programs->CPhyScopeDecoder. Three shortcuts are installed:

1. **CPhyScopeDecoder.exe**: shortcut to start the application
2. **Uninstall.bat**: shortcut to uninstall the application
3. **CPhyScopeDecoderUsersManual_x_x**.pdf: shortcut to this document.

2.2 Tektronix TekVisa Installation

If a Tektronix scope is being used and the application has been installed on a separate host, this machine also must have the Tektronix TekVisa application installed on it. This provides the low-level communication path for scope discovery and control over a LAN.

For your convenience, a copy of TekVisa is provided on the CPhyScopeDecoder installation disk (TekVisa.exe). Optionally, there may be a newer version you can download using the following link:

http://www.tek.com/oscilloscope/tds7054-software/tekvisa-connectivity-software-v400

Please run this executable to install TekVisa on your host machine.

After installation, the Tektronix OpenChoice Instrument Manager can be used to configure remote visibility of Tektronix devices to the host machine. In addition to the Start menu, there is an icon installed in the desktop tray with a yellow Visa icon that can be used to launch the Instrument Manager.

For the CPhyScopeDecoder software to recognize available scopes, their address must appear in the Instrument list. Press the Update button to refresh the instrument list.

The "Search Criteria" button can be used for configuring scope discovery. Assuming access to the scope is via Ethernet, under the "LAN" tab, make sure the "Search LAN" and "Auto Discovery" check-boxes are checked.

2.3 Agilent IO Libraries Installation

If an Agilent scope is being used and the application has been installed on a separate host, this machine also must have the Agilent IO Libraries Suite installed on it. This provides the low-level communication path for scope discovery and control over a LAN.

For your convenience, a copy of the IOLibSuite is provided on the CPhyScopeDecoder installation disk (named IOLibSuite_16_3_17218.exe). Optionally, there may be a newer version you can download using the following link:[3]

http://www.agilent.com/find/iolib

Please run this executable to install IOLibSuite on your host machine.

After installation, the Agilent Connection Expert software can be used to configure and check remote visibility of Agilent devices to the host machine. In addition to using the Start menu, there is an icon installed in the destop tray with a dark-blue IO icon that can be used to launch the Agilent Connection Expert software.

For the CPhyScopeDecoder software to recognize available scopes, their address must appear in the Instrument list. Press the Refresh All button to refresh the instrument list.

2.4 Licensing

CPhyScopeDecoder requires a special USB key (dongle) for full operation, provided with purchase of the software. If this key is not present on the host machine, the application will still launch, load, and decode saved trace files but it will not communicate with a scope or process saved scope binary waveform files. Thus, multiple copies of the software may be installed on multiple machines, but only those running on a machine with the USB key plugged in will be fully functional.

2.5 Scope Setup

In CPhy, a lane uses three conductors labeled A, B, and C, intended to be measured independently by the CPhy receiver in low-power mode and differentially in high-speed mode. Accordingly, three scope channels are used to acquire signals on A, B, and C, which should be connected as follows:

- Channel 1: wire-A minus wire-B
- Channel 2: wire-B minus wire-C
- Channel 3: wire-C minus wire-A

[3] Note this package is fairly large, ~245 MB and may take some time to download.

This probing methodology is equivalent to the measurement connections used by CPhy receiver hardware. However, note that this usage does not allow CPhyScopeDecoder to distinguish the two LP states -- LP111 and LP000 – from each other. Thus, these states can only be inferred from other signal activity and not necessarily very accurately. For example, in the case of HS-Exit, after the HS postamble symbols, the software is unable to determine when exactly (or even if) the bus transitions back to LP111 from LP000. The software assumes after 100 ns (minimum HS-Exit time) that the bus is transmitting LP111.

Figure 1 - Sample CPhy Scope Capture

Figure 1 shows an example scope capture of a looping CPhy HS burst (Frame Start packet). The software configures channel scaling from -0.6 V to +0.6 V, causing LP voltages to clip at maximum. This is done to allow greater dynamic range and measurement of HS signals.

The zoom window shows the start of a CPhy HS burst where the LP001 state at the far left is indicated by channel 1 in yellow (A-B) at zero, channel 2 in blue (B-C) at minimum (-0.6V) and channel 3 in purple (C-A) at maximum (+0.6V). Where all three

channels converge to zero volts, this is inferred as LP000. After HS Prepare time, HS signals begin sending the preamble symbols, sync symbols, packet symbols, and finally postamble symbols.

Setting up the scope for acquisition is straightforward. Use the controls in the Scope tab of the control below the disassembly listing in the main window. Based user settings, the software can configure the horizontal and vertical scope settings, trigger position and level, sample rate, etc., if desired.

An easy way to configure the scope, then, is to do the following:

1. Connect channels 1, 2, and 3 to probes to measure A-B, B-C, and C-A from the DUT.
2. Bring up the CPhyScopeDecoder application, selecting the scope address from the connection window that appears.
3. In the Scope tab of the main window, type in the HS symbol rate and select the number of samples to acquire, and click on the "Config Scope" button.
4. Run the scope using the "Run Continuousand set the trigger position and level as desired. While not required, a common trigger point is the rising edge of channel 1 at around one volt, which will generally indicate entry into the LP111 state.

Certain settings are critical such as channel positions (all centered) and channel scaling (-0.6 V to 0.6 V). Also, oversampling relative to the HS symbol rate should be at least 5. Trigger position is not important to decode, though the acquisition must have an LP001 state and the first LP001 is always the starting point for disassembly.

3 Using CPhyScopeDecoder

3.1 Connecting to a scope

When the application is started, a connection dialog (see Figure 2) is shown for the user to select an oscilloscope powered-on and reachable via LAN. This dialog can also be brought up later to connect to a scope using the Control menu option "Connect to Scope...".

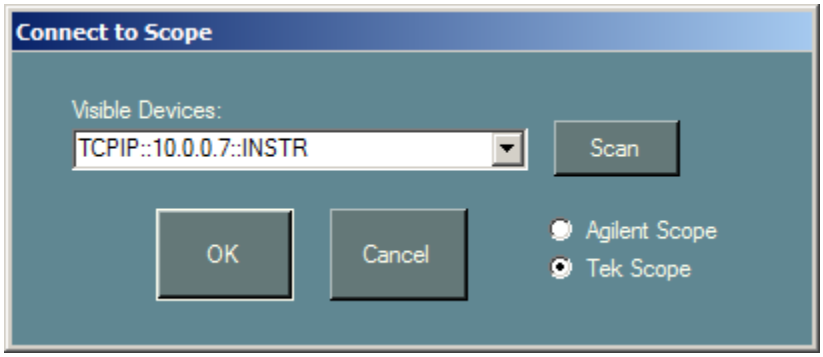

Figure 2 – Connection Dialog

In the connection dialog, a drop-down box is provided with reachable scope names as well as an option called "Offline". Offline mode allows the application to be used without a scope connection, useful for demonstration of the GUI, post-processing saved waveform files, and reviewing saved CPhyScopeDecoder traces.

Click on the desire scope type option button to show either Tektronix or Agilent available instruments. The Scan button updates the visible device list, including any new scopes that have come online since the dialog was brought up.

To connect to an instrument, simple select the appropriate scope name and click OK.

3.2 Main Window

Once connected, the main window is shown (see Figure 3). Note that name of the currently connected scope (or "Offline") appears in a right pane of the status bar in the main window. Almost all user interaction with the application is through the main window, whose functions include:

- Scope configuration
- Data acquisition
- Disassembly display with numerous view and decoding options
- Advanced predicate-based searches
- Advance predicate-based record filtering

- Video analysis and frame display
- Error summaries
- Packet summaries.

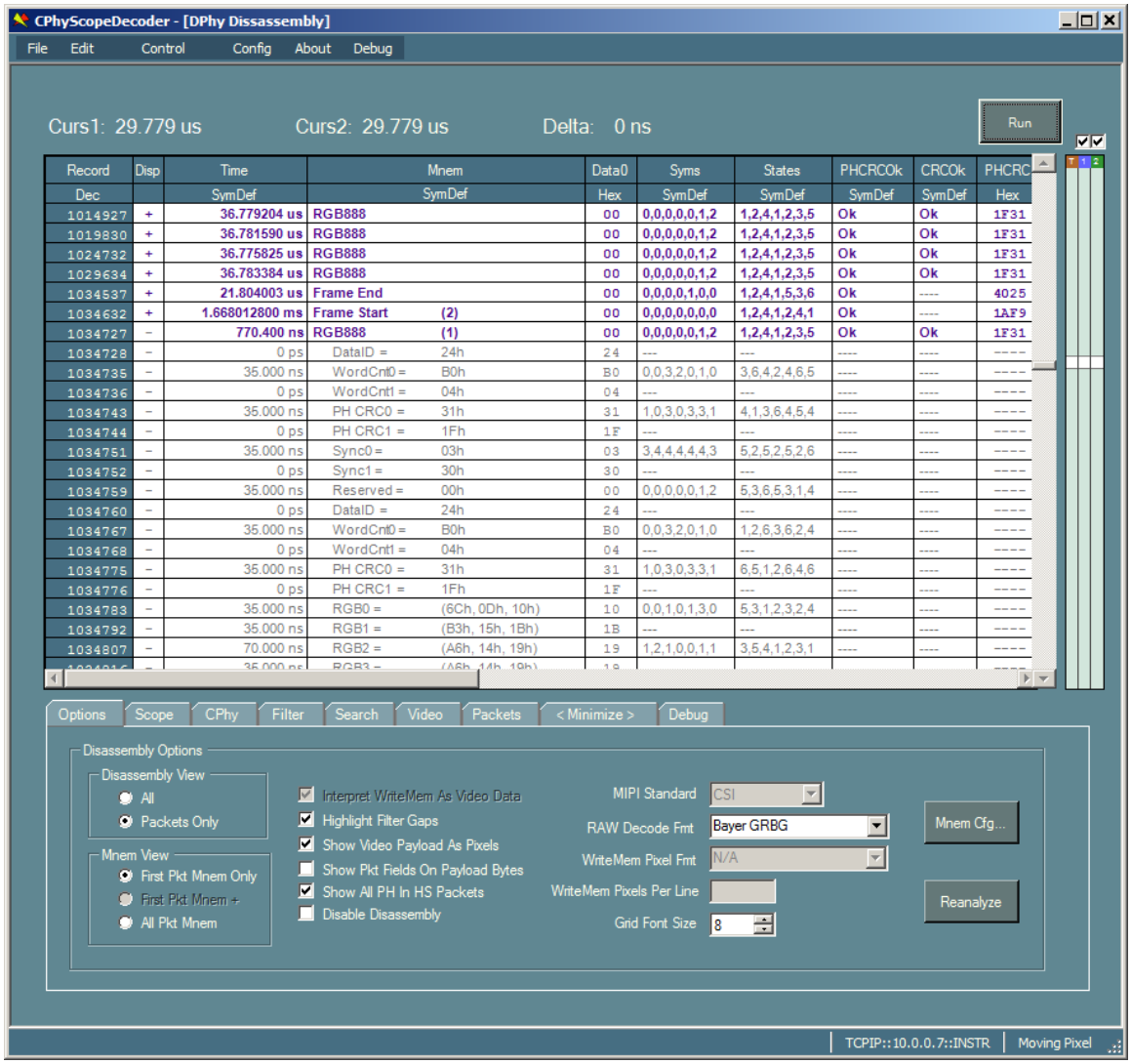

Figure 3 – Main Window

The main window has several areas of controls, roughly from top to bottom as follows:

- **Cursor Status Display** – The top part of the window displays a text readout of the two cursor locations (time relative to start of trace). The time delta between the two cursors is also displayed.
- **Run/Stop Button** – Depending on the current processing state, the label of this button at the top-right of the main window may display "Run", "Stop", or "Abort". Basically, this button is the user mechanism to start/stop an acquisition and disassembly processing.

- **Listing Grid** – The listing grid is the central feature of the disassembly window. It contains decoded records and their fields.
- **Cursor Control Panel** – The cursor panel is a control next to the vertical scroll bar of the listing grid (far right) that spatially indicates relevant locations in the trace, specifically: trigger location, cursor locations, and current view location. More details of its functions are provided later, but dragging or double-clicking cursor icons, or dragging/double-clicking in the panel itself is one of many ways to navigate and scroll the listing grid.
- **Disassembly Control Panel** – The tabbed control beneath the listing grid contains functional groupings of controls associated with the application. Tab group names for the application are Options, Scope, CPhy, Filter, Search, Video, Packets, and Minimize.

3.2.1 Run/Stop Button

When connected to a scope, the Run/Stop button is enabled and its label is set to "Run" when the application is idle. When clicked in this state, an acquisition is started.

While the application is processing, the Run/Stop button label is set to "Stop" or "Abort" (depending on the phase of processing). Clicking the button in this state requests an end to processing and the application will abort as soon as it is able. During data upload, for example, this is not possible, and the application waits until upload is complete to abort.

Acquisition is configured via controls on the Scope tab of the disassembly control, in particular, the sampling rate (derived from the HS Bit Rate), oversampling factor, number of samples, and trigger percent. Another check-box, called "Don't trigger scope when run" determines whether the scope is first run before data is uploaded.

After the scope has triggered and scope data is available, the application uploads the data to be processed. Upload from the scope takes from several to maybe 15 seconds, depending on the number of samples in the trace. After upload, processing begins, consisting of several phases:

- LP/HS segment partitioning
- Burst sequence analysis
- HS state-to-symbol decoding
- CPhy symbol sequence to data byte conversion
- Protocol packet decoding
- Packet header and payload CRC checking
- Video frame analysis and construction.

After processing, the disassembly listing is prepared and displayed in the Listing Grid control. Generally, this is a fairly quick process (a few seconds), though depending on the filter and disassembly settings, record content, and the number of captured records, this can take some time. Progress is indicated via the status bar, describing the current task along with a completion percentage.

3.2.2 Listing Grid

The listing grid takes up most of the main window and displays records representing events that occurred on the link. The view is similar in look-and-feel to that of a logic analyzer, where one disassembly record is displayed per line, arranged in order from earliest in time at the top to latest in time at the bottom. Generally, only a subset of records is shown in the listing at any time, based on settings in the filter tab of the disassembly control panel, disassembly options, as well as the mnemonic view state of individual records.[4]

Note that records are derived protocol components from data samples in the acquired trace. Each record is comprised of many scope data samples and there is not a fixed relationship between the number of data samples and protocol components.

Columns in the grid represent data fields associated with records, where field values may consist of direct (or near-direct) signal data acquired from the CPhy bus, derived data from that content, or other attributes created by the application. Not all fields apply to all record types, in which case a field will generally contains dashes.

The listing grid contains vertical and horizontal scroll bars for scrolling visible rows and columns in the grid. Other mechanisms are available for scrolling or selecting viewable rows in the listing including: selected keyboard combinations, clicking in the cursor panel, selecting among several context menu options, and buttons in the search and video tabs of the disassembly control panel.

3.2.2.1 Column (Field) Manipulation

The user has control over what columns are displayed, how they are ordered, and how field data are formatted in the column grid. Columns consist of a header in the first row, a radix selection in the second row, followed by field data starting with the third row. The following sections outline field- or column-related functions.

3.2.2.1.1 Selecting Columns

Single columns can be selected by left-clicking on the column header (i.e. the field name of a column). A contiguous range of columns can be selected by first selecting a column then selecting a second column while holding the shift key down. In addition, a discontinuous set of columns can be selected by clicking headers while holding the control key down. When a set of columns is selected, certain options in the column context menu will apply to each column, e.g. "Delete Columns", "Restore Select Column Width(s)"

3.2.2.1.2 Selecting Rows

(While not a column-related function, it makes sense to document this here.) Single rows can be selected by right-clicking on the row header (i.e. the Sample field of a record). A contiguous region of rows can be selected by first selecting a row then selecting a second (later) row while holding the shift key down. Once selected, rows can by copied to the clipboard with Ctrl-C.

[4] The "mnemonic view state" concept is described later in this section.

3.2.2.1.3 Add Column(s)

To add a column field in the listing grid, right-click on a column header and select the "Add Column(s)" menu option to bring up the Add Column dialog. Check the desired column fields to display and click OK.

3.2.2.1.4 Remove Column(s)

To remove a column field in the listing grid, select one or more columns and press the Delete key. Alternatively, right-click in one of the selected column headers and choose "Delete Column(s)". If multiple columns are to be deleted, the user is asked for confirmation.

3.2.2.1.5 Reordering Columns

To change the display order of visible columns, select one or more columns and drag one of the selected header cells to a new position. Alternatively, pressing the right-arrow or left-arrow keys will move the selected columns right or left respectively. Note that selected column groups are always made contiguous after moving.

3.2.2.1.6 Setting Column Widths

Column widths are automatically set to accommodate the widest string or value to be displayed and are updated whenever the field radix is changed. However, the user may adjust column width by selecting the right edge of a column header and dragging it. A column's width may be restored to its default using the column context menu option "Restore Selected Column Width(s)" or simply double-clicking on the column header.

3.2.2.1.7 Setting the Field Radix

Most fields are numeric and have an underlying value associated with it (Boolean fields are considered numeric and are associated with a 0 or 1 value). Numeric fields can be displayed as binary, decimal, or hexadecimal by right-clicking a column sub-header cell (second row in grid) to bring up the radix context menu and selecting the appropriate option. In addition, some fields have a default symbolic association – i.e. a built-in number-to-string mapping -- that may be available in the menu. Finally, a user can define his own symbol file to use for number-to-string translation (see section 3.2.2.1.9).

3.2.2.1.8 Setting the Time Field Reference

The time field displays the time-stamp of a record, indicating when the data was acquired relative to the start of trace capture (see section 3.2.2.2.4). CPhyScopeDecoder supports viewing time-stamps relative to a global time reference selected by the user via the radix context menu associated with the time field. Records occurring before the global time reference have a negative time field value and records occurring afterwards have a positive time field value.

The following time field display options are provided in the Time radix context menu:

- Relative to Start – display time-stamps relative to the first record.
- Relative to Trigger – display time-stamps relative to the trigger record
- Relative to Cursor1 – display time-stamps relative to cursor1
- Relative to Cursor2 – display time-stamps relative to cursor2
- Relative to Previous – display time-stamps relative to the preceding *visible* record

3.2.2.1.9 Defining and Using a Symbol File

To associate a radix file to a numeric field, select the "<New Symbolic File>" option in the radix context menu. This brings up an open file dialog to browse for and select the new symbol file. Once the symbol file has been selected, the context menu will enable the "Symbolic (File)" option (and it will be checked).
Symbol files are text files containing number-to-string mappings, where each mapping consists of a boolean predicate equation and a string. During disassembly, the equation from each mapping is evaluated in order for each field value and the string from the first mapping whose equation evaluates true is displayed.

Predicate equations consist of one or more terms, one term per line. In multi-term equations, all but the last term in the equation are followed by "AND" or "OR" forming a boolean equation from the terms. Finally, the last term of a predicate equation is followed by a colon, and then the symbolic string to associate with the equation enclosed in quotes.

Each predicate term consists of a relational operator and a value. Relational operators are ">", "<", ">=", "<=", "==", and "!=". Values are interpreted as decimal unless an 'h' or a 'b' is appended to denote hexadecimal or binary respectively. In addition, for hex or binary values, the character 'x' may be used as a "don't care" digit (if the first character is an 'x' it represents don't care digits up to the width of the field)

For example, the following lines define a predicate equation to detect values between 0x10 and 0x40 or if a value is odd:

```
// predicate equation
> 10h AND
< 40h OR
== x1b : "Range between (10h, 40h) or odd"
```

For values in the disassembly where this equation evaluates true, the string "Range between (10h, 40h) or odd" will be displayed.

Some notes about symbol file usage:

- Note that blank lines and lines that begin with "//" (which can be used for comments) are ignored by the parser.
- Predicate terms are interpreted as a sum-of-products equation and so AND has higher precedence than OR. Thus, predicates that describe the equation "a AND b OR c AND d" is parsed "(a AND b) OR (c AND d)".
- White space is **required** between the relational operator, the value, the "AND", "OR" and ":" delimiters, and the string.
- Mappings are evaluated in order, so subsequent predicate equations do not need to exclude values that would have already satisfied prior mappings.
- If not mapping qualifies for a value, a string with "?" is displayed. To avoid this, a default mapping can be appended to the end of the symbol file with the catch-all predicate ">= 0".

3.2.2.2 Field Descriptions

3.2.2.2.1 Mnem Field

The mnemonic field is a symbolic-only field that conveys the record type. The string displayed in this field can be customized to show certain other fields in the record and their values. This behavior is configured using the "Mnem Cfg…" button on the Options tab of the disassembly control panel. Using this feature is simply another way to view fields in a record, using the Mnem field cell to display the field rather than using a separate column in the listing.

3.2.2.2.2 Sample Field

The Sample field represents a record's ordinal number in the listing starting with 0 and incrementing by one for each possible logical record position. There is an approximate (but not exact) mapping from Sample value to HS symbol / LP state position in the acquisition. In particular, the Sample number increments with each LP state change or HS symbol, except when HS packet data is displayed. For each HS packet byte, the Sample number increments by 8.

3.2.2.2.3 Disp Field

The Disp field is a column used for showing the mnemonic view state of packet headers. This field is blank for non-packet header records but contains one of three indicators for a record containing a packet header: "+", "++", or "-". These indicators correspond to a mnemonic view state of "closed", "mnem-fields", "mnem- and packet-fields" respectively. When clicked, the mnemonic view state of the packet cycles to the next state (see section 3.2.2.7). Note that is no packet fields are selected to be included in the mnemonic field, the "++" display state is skipped.

3.2.2.2.4 Time Field

The Time field represents displays a record's time-stamp, which the user may choose to display relative to a number of time references (e.g. start of trace, cursor or trigger position, etc.) Please see section 3.2.2.1.8 for more information.

By default, the time-stamp is displayed with a radix of Symbolic (Default), which displays the time as a string, e.g. "1.300 ns", but alternatively it may be displayed as a numeric value (hex or decimal), which converts time to an integer number of picoseconds. This allows the time field to be used in search and filter criteria, for example, in restricting record display or search to a limited time range.[5]

3.2.2.2.5 Param Fields

In the MIPI CSI-2 and DSI protocols, some packet types are associated with one, two or three named parameter fields (as opposed to larger blocks of generic data such as pixel data, blanking bytes, LUT tables, etc.). The disassembly window organizes these packet parameters into the fields: "Param1", "Param2" and "Param3".

For example, the Generic Short Write packet can have 0, 1, or 2 parameters. For zero-parameter writes, all Param fields will have dashes displayed in them. For one-parameter

[5] Currently, searching or filtering on the time-stamp field when in symbolic mode does not behave as desired, as comparisons are made alpha-numerically in this radix.

writes, "Param1" will display the value of the single-byte parameter. And for two-parameter writes, "Param1" and "Param2" will display the single-byte parameters.

Another example is the DCS Set Column Address command that has two 16-bit parameters. In this case, "Param1" and "Param2" will display the 16-bit StartColumn and EndColumn parameters respectively.

Finally, note that when the Param fields are added as mnemonic fields using the "Mnem Cfg…" button, the actual parameter names are displayed in the mnemonic. For this reason, in general, the user may always want to have the Param fields selected as fields to display in the mnemonic.

3.2.2.2.6 LPSeq Field

The LPSeq field indicates the LP sequence state of the CPhy Decoder at the time the record was recorded. The LP sequence state represents a position in decoding CPhy LP signaling, for example, the state LP-320 records having seen the sequence LP11, LP10, LP00 (the first three signaling states of a BTA). The following LP sequence states are used:[6]

State	LP Seq
Stop	LP11
LP-Rqst	LP11, LP10
LP-320	LP11, LP10, LP00
LP-3202	LP11, LP10, LP00, LP10
BTA	LP11, LP10, LP00, LP10, LP00
BTA-Exit	LP11, LP10, LP00, LP10, LP00, LP10
LP-3201	LP11, LP10, LP00, LP01
LP-Esc	LP11, LP10, LP00, LP01, LP00
HS-Rqst	LP11, LP01
HS-Go	LP11, LP01, LP00 (In HS Burst)

3.2.2.2.7 PktLen Field

The PktLen field applies to records that contain the first header byte of a packet and indicates the total number of bytes associated with the packet (including header and CRC fields).

3.2.2.2.8 VC Field

The VC field applies to records that contain the first header byte of a packet and displays the 2-bit virtual channel field of the packet. In CSI, note that the value in this field is selected from a PH with a correct CRC, if one exists in the header. Otherwise, the value displayed from the first PH in the packet.

3.2.2.2.9 DataType Field

The DataType field applies to records that contain the first header byte of a packet and displays the 6-bit data type field of the packet. In CSI, note that the value in this field is

[6] Note that the CPhy Decoder does not distinguish between the LP000 state called HSPrepare and the HS-0 state called HSZero. The sequence state for this period is called HS-Go.

selected from a PH with a correct CRC, if one exists in the header. Otherwise, the value displayed from the first PH in the packet.

3.2.2.2.10 DCSCmd Field

The DCSCmd field applies to records that contain the first header byte of a DCS read request, short write, or long write and displays the 8-bit DCSCmd field. In DSI, note that, the value in this field for short DCS packets is selected from a PH with a correct CRC, if one exists in the header. Otherwise, the value displayed from the first PH in the packet.

3.2.2.2.11 LP Fields

Depending on the maximum lane count supported, there may be up to four LP fields: "LP0", "LP1", LP2", "LP3". These fields are present in all record types and reflect the LP lane state of lanes 0-3 respectively. Note that unused lanes (i.e. those exceeding than the maximum lane count setting - 1) always show the LP11 state, even if the lane input is disconnected. A default SymDef mapping is provided to map field values of 0-3 to LP000, LP001, LP100, and LP111 respectively. .

3.2.2.2.12 Data Fields

Depending on the maximum lane count supported, there may be up to four Data fields: "Data0", "Data1", "Data2", "Data3". These fields show the HS or LP data byte value per lane associated with the record. HS records will display bytes in the Data fields associated with active lanes. On the other hand, only Data0 may show LP data bytes, including LPDT packet bytes, as well as escape and trigger byte codes associated with the escape signaling protocol. In CSI, note that the 7 sync symbols between PH in packets are decoded to bytes (0x3003) and displayed (even though the sync sequence does not technically have a valid one-to-one mapping).

3.2.2.2.13 Trig Field

The Trig field in CPhyScopeDecoder indicates the record associated with the scope trigger position. The Cursor Panel or context menu can be used to go to this record associated with the hardware trigger. A default SymDef mapping is provided to map field values of 0 to "---" and 1 to "Trig".

3.2.2.2.14 Frame Field

The Frame field applies to records that contain the first header byte of all video-related packet types, including CSI Frame Start, Frame End, Line Start, Line End; DSI VSync Start and VSync End; blanking and video packets. Starting at one, it reflects an ordinal frame number associated with the packet. It also corresponds with frames listed in the video tab of the Disassembly Control panel. Generally, the frame number will increment with each Frame Start or VSync Start packet, but might otherwise increment if a video packet occurs with a new VC or DataType.

3.2.2.2.15 ActLine Field

The ActLine field applies to records that contain the first header byte of an active video packet, *where a preceding Frame Start or VSync Start packet has already been seen* (otherwise, the line number is unknown). Starting at one, it reflects the active line number of the packet in the frame.

3.2.2.2.16 Lane Field

The Lane field applies to all records and indicates the corresponding data lane of its data byte or symbol. For LP records, Lane field is always zero. For HS records, the Lane field can be 0 through the maximum lane count minus 1.

3.2.2.2.17 ECC Fields

For DSI, there are three ECC fields: "ECC", "ExpECC", and "ECCOk". These fields show the ECC byte contained in a packet header, the expected (computed) ECC value for the header, and a comparison result of the two respectively. These fields apply to records that contain the first header byte of a packet. ECCOk has a default SymDef mapping for the boolean comparison of 0 == "Err" and 1 == "Ok".

3.2.2.2.18 PHCRC Fields

For CSI, there are three PHCRC fields: "PHCRC", "ExpPHCRC", and "PHCRCOk". These fields show the PHCRC bytes from the PH, the expected (computed) PHCRC value for the PH, and a comparison result of the two respectively. Note that the values for these fields are taken from a PH that has a correct PHCRC, if it exists. PHCRCOk has a default SymDef mapping for the boolean comparison of 0 == "Err" and 1 == "Ok".

3.2.2.2.19 CRC Fields

There are three CRC fields: "CRC", "ExpCRC", and "CRCOk". These fields show the CRC bytes from the packet, the expected (computed) CRC value for the packet, and a comparison result of the two respectively. CRCOk has a default SymDef mapping for the boolean comparison of 0 == "Err" and 1 == "Ok".

3.2.2.2.20 BusOwn Field

The BusOwn field applies to all DSI records and indicates which end of the bidirectional link is transmitting (i.e. owns the bus). A default SymDef mapping for the boolean value is 0 == "Dev" and 1 == "Host".

3.2.2.2.21 States Field

The States field applies to every other decoded byte in a HS packet and shows the 7-state sequence associated with the current and next data bytes in the packet. The value displayed is a string and "SymDef" is the only radix available for this field. Each state can have the value of 1-6 and represents the differentially received A-B, B-C, and C-A levels on a CPhy lane.

3.2.2.2.22 Syms Field

The Syms field applies to every other decoded byte in a HS packet and shows the 7-symbol sequence associated with the current and next data bytes in the packet. The value displayed is a string and "SymDef" is the only radix available for this field. Each symbol can have the value of 0-4 and represents the state transition associated with the previous state and the current state as defined by CPhy based on the functions [Flip, Rotate, Polarity]. If the state transitions defining the symbol sequence are invalid, the value displayed for this field is "Illegal Seq".

3.2.2.2.23 Sym Fields

Depending on the maximum lane count supported, there may be up to four Sym fields: "Sym0", "Sym1", "Sym2", "Sym3". The value displayed in these fields is the decoded symbol associated with the record for lanes 0-3 respectively. These fields apply to

records other than packet data bytes, including LP states, HS preamble, HS sync, and HS postamble records.

3.2.2.2.24 State Fields

Depending on the maximum lane count supported, there may be up to four State fields: "State0", "State1", "State2", "State3". The value displayed in these fields is the decoded wire-state associated with the record for lanes 0-3 respectively. These fields apply to records other than packet data bytes, including LP states, HS preamble, HS sync, and HS postamble records.

3.2.2.3 PH Field Display

In CSI, the mnemonics displayed for PH fields include "Reserved", "DataID", "Param1" and "Param2", "WordCnt0", "WordCnt1", "PH CRC0", "PH CRC1", "Sync0". In the mnemonic string for these PH fields, the value that is displayed depends on the setting of the "Show All PH In HS Packets" check-box option (Options tab).

If this option is checked, then all PH field mnemonics will be displayed and the values shown will be the actual data values decoded from the PH. On the other hand, if this option is not checked, the values shown will be taken from a PH that has a correct PHCRC, if one exists (otherwise, values from the first PH are shown).

For the column fields themselves that are associated with the PH, note that these fields ("VC", "DataType", "PHCRC", "ExpPHCRC", and "PHCRCOk") always use values from a PH with a correct PHCRC, if possible.

3.2.2.4 Trigger and Cursors

The hardware trigger record and two cursor records can be displayed in the listing grid, changing the background shading of their associated records. The trigger record indicates the location of the hardware trigger for capture and is a static fixed position that cannot be changed but can be used as a reference for time measurement and Goto operations. Its Trig field will be set to 1.

Cursors are enabled by checking the associated checkbox in the cursor panel to the right of the listing grid and can also be controlled from the cursor panel control (see Cursor Panel section for more details). The current cursor positions and the delta time between them are displayed above the listing grid, using the current global time reference set in the time field radix context menu.

Cursor related functions are also available in the column context menu available by right-clicking in any grid cell (except the radix sub-header row). In particular, the user can select "Goto Cursor1" or "Goto Cursor2" to scroll the listing grid to the record location of the indicated cursor. Also, if the context menu is brought up from a non-header cell, the "Move Cursor1 Here" and "Move Cursor2 Here" options are also available.

3.2.2.5 Views

A main feature of the disassembly window is to allow the user to set criteria for filtering records, only displaying a subset of acquired records from the disassembly view. This is done through several mechanisms:

- Selecting the Disassembly View
- Setting the global Mnemonic View (or altering the mnemonic view of individual packets)
- Setting and applying packet and record filtering criteria
- Setting various disassembly options in the Options tab

Each mechanism defines criteria for what records are shown (or not shown) in the disassembly listing.

3.2.2.6 Disassembly View

The Disassembly View setting is set via option buttons in the Options tab.

All	Shows all record types, including non-packet traffic such as LP states, LP signaling transitions in Escape and HS bursts, and HS preamble, sync, and postamble records.
Packets Only	Shows only records that contain packet bytes, including HS, LPDT, and escape-mode commands.

When a view is selected, the listing is updated to display records only applicable to the new view. In the case of packet records, the mnemonic view state determines whether only the first header byte is shown or all packet bytes are shown (see next section).

3.2.2.7 Mnemonic View

Mnemonic view state is an attribute of records that contain the first byte of a protocol packet. Its setting determines whether only the first byte of a packet is displayed, or all bytes of the packet are displayed, and whether the user-specified mnemonic fields associated with the packet are displayed.

There are three mnemonic view states:

First Pkt Mnem Only	Only the first header byte of the packet is displayed.
First Pkt Mnem +	Only the first header byte of the packet plus its mnemonic fields are displayed. Mnemonic fields are fields selected by the user to be displayed with the mnemonic via "Mnem Cfg…" button).
All Pkt Mnem	All packet bytes are displayed. Mnemonic fields are also shown for the first header byte of a packet.

The mnemonic view state of a packet is shown in the Disp field in the record using the indicators "+", "++", or "-" (corresponding to the states in order in the table above). To cycle an individual packet's mnemonic view state, click on the indicator in the Disp column. To reset all packet mnemonic view states at once, click a Mnem View radio button in the Options tab of the Disassembly Control Panel.

When a packet is in the "-" state, records associated with header and payload bytes are shown for the packet. In this state, all packet records will show a "-" in their Disp column. Clicking on this field for any packet record, will close the packet, cycling its mnemonic view back to "+".

3.2.2.8 Column Context Menu

The column context menu is brought up by right-clicking in any cell except for the radix row (second row) of the listing grid. Depending on the cell location clicked to bring up the menu, certain options will be enabled or disabled. Here is a summary of all the functions in the column context menu:

Function	Description	Shortcut
Add Column(s)	Brings up dialog to add columns (i.e. make fields visible) to the listing grid.	
Delete Column(s)	Deletes selected columns.	
Restore Selected Column Width(s)	Restores selected columns to their default width.	
Select All	Selects all rows and all columns (usually to save to a file)	
Goto Cursor1	Scrolls the listing grid to the cursor1 record.	Ctl-1
Goto Cursor2	Scrolls the listing grid to the cursor2 record.	Ctl-2
Goto Trigger	Scrolls the listing grid to the hardware trigger record.	Ctl-T
Goto…	Brings up a dialog to enter a sample number to scroll the listing grid to (the nearest visible record is used).	Ctl-G
Move Cursor1 Here	Moves the cursor1 position to the current record.	
Move Cursor2 Here	Moves the cursro2 position to the current record.	
Search Frame Start	Fills in the search criteria to search for Frame Start for CSI or VSync Start for DSI. Then, the Next Search function is performed. .	Ctl-Alt-F
Search Same Mnem	Fills in the search criteria to search for records that match the mnemonic value associated with the current record. Then, the Next Search function is performed. Note that the current cell can be any field type.	Ctl-Alt-M
Search Same Val	Fills in the search criteria to search for records that match the current cell's field value. Then, the Next Search function is performed.	Ctl-Alt-V
Search Same Mnem And Val	Fills in the search criteria to search for records that match the mnemonic value associated with the current record and the current cell's field value. Then, the Next Search function is	Ctl-Alt-B

	performed.	
Next Search	Scrolls to the next record satisfying the search criteria	Ctl-S
Next Packet	Scrolls to the next packet start.	Ctl-P
Next Frame Start	Scrolls to the next CSI Frame Start or DSI VSync Start packet.	Ctl-F
Next Same Mnem	Scrolls to the next record that matches the mnemonic value associated with the current record.	Ctl-M
Next Same Val	Scrolls to the next record that matches the current cell's field value.	Ctl-V
Next Same Mnem And Val	Scrolls to the next record that matches the mnemonic value associated witht the current record and the current cell's field value.	Ctl-B
Prev Search	Scrolls to the previous record satisfying the search criteria	Alt-S
Prev Packet	Scrolls to the previous packet start.	Alt-P
Prev Frame Start	Scrolls to the previous CSI Frame Start or DSI VSync Start packet.	Alt-F
Prev SameMnem	Scrolls to the previous record that matches the mnemonic value associated with the current record.	Alt-M
Prev Same Val	Scrolls to the previous record that matches the current cell's field value.	Alt-V
Prev Same Mnem And Val	Scrolls to the previous record that matches the mnemonic value associated witht the current record and the current cell's field value.	Alt-B
Exclude All But Frame Start	Sets and applies the filter criteria to filter all records except for CSI Frame Start of DSI VSync Start packets.	Ctl-X, F
Exclude All But Same Mnem	Sets and applies the filter criteria to filter all records except those that match the mnemonic value associated with the current record.	Ctl-X, M
Exclude All But Same Val	Sets and applies the filter criteria to filter all records except those that match the current cell's field value.	Ctl-X, V
Exclude All But Same Mnem and Val	Sets and applies the filter criteria to filter all records except those that match the mnemonic value associated with the current record and the current cell's field value.	Ctl-X, B
Modify Filter From Current Cell	Contains a sub-menu with three items: ▪ Replace Filter Term ▪ Add OR Filter Term ▪ Add AND Filter Term Selecting one of these menu items will perform the requested function based on the current cell's	

	field name and value. The operator used is "==".	
Modify Search From Current Cell	Contains a sub-menu with three items: ▪ Replace Search Term ▪ Add OR SearchTerm ▪ Add AND Search Term Selecting one of these menu items will perform the requested function based on the current cell's field name and value. The operator used is "==".	
Close Current Mnem View	Sets the mnemonic view of the current packet to "Closed". This option is only enabled when the current mnemonic view is "Mnem- And Packet-Fields".	

3.2.2.9 Radix Context Menu Summary

The radix context menu is brought up by right-clicking in the sub-header row (second row) of the listing grid and is used to set the display format of the current column. The functions in the menu have been described in earlier sections. Here is a summary of all the functions in the radix context menu (except for the context menu for the Time field):

- **Hexadecimal** – sets the radix to hexadecimal (enabled for numeric fields)
- **Decimal** – sets the radix to decimal (enabled for numeric fields)
- **Binary** – sets the radix to binary (enabled for numeric fields)
- **Symbolic (Default)** – sets the radix to symbolic using the default translation internal to the application.
- **Symbolic (File)** – sets the radix to symbolic using the translation defined by the symbolic file specified by the user using the "<New Symbolic File>" menu option.
- **<New Symbolic File>** – brings up an open file dialog to select a symbol file to associate with the current column.

In addition, the time field radix context menu has several more options to set the desired time field reference, which determines how the time field is displayed. One of these options is always checked:

- **Relative to Start** – sets the global time field reference to the first acquired record
- **Relative to Previous** – sets the time field reference for each record to the previous displayed record (i.e. displays delta time between records)
- **Relative to Trigger** – sets the global time field reference to the trigger record
- **Relative to Cursor1** – sets the global time field reference to the cursor1 record
- **Relative to Cursor2** – sets the global time field reference to the cursor2 record

3.2.3 Cursor Control Panel

The cursor panel is a custom control with three narrow columns on the right side of the listing grid. Each narrow column is similar to a vertical scroll bar, each associated with either a cursor or the trigger (for the purposes of this description, the trigger can be thought of as a special immovable cursor). Cursors represent a specific record position in

the listing, similar to a scroll bar, where the top of the panel represents the start of the listing and the bottom of the panel represents the end of the listing.

The first column in the cursor panel represents the trigger and, if enabled, is drawn as a red square labeled with a 'T'. The trigger cursor is always enabled. The second and third columns represent cursor1 and cursor2 respectively and, if enabled, are drawn as a blue square labeled with a '1' or a blue square labeled with a '2'. A check box above each cursor column can be used to enable or disable each cursor. Disabled cursors are drawn with a gray background in the cursor panel and are not visible in the disassembly listing.

Like a scroll-bar, the cursor panel is meant to represent the entire vertical span of the listing windows. Cursor positions within the panel symbolically indicate where each lies within the listing, i.e. a cursor toward the top of the panel indicates a record position near the start of the listing and a cursor toward the bottom of the panel indicates a record position near the end of the listing. The times associated with cursor1 and cursor2 positions and the difference between them are displayed in readouts above the listing grid.

In addition to the cursor symbols, a horizontal white region is drawn across the cursor panel. This region represents the currently viewed region of the listing, having a position and thickness corresponding to the current viewed region relative to the total listing.

The cursor panel supports the following operations for setting cursor positions and scrolling the listing:

- Left-clicking and dragging a cursor square moves the cursor position. While dragging, the current cursor position (Sample #) is displayed in a pop-up box.
- If the Shift key is pressed and held while dragging, the listing will scroll with the cursor position, otherwise the listing will not scroll.
- If the Control key is pressed and held while dragging, the cursor will scroll more slowly, allowing for finer control.
- Double clicking a cursor in the panel scrolls the listing window to the cursor position.
- Double-clicking in the cursor panel (not on a cursor) scrolls the listing window to the position clicked.

3.2.4 Disassembly Control Panel

The Diassembly Control Panel is located at the bottom part of the disassembly window and has five tabs: Options, Scope, CPhy, Filter, Search, Video, Packets, <Minimize>. The functions and controls provided by each tab are described in the following sections.

3.2.4.1 Options Tab

The Options tab contains controls for configuring decoding and disassembly display (see Figure 4). Some options are non-protocol specific:

- **Grid Font Size** – this up/down control increases or decreases the font size of the grid display. Values can range from 6 to 12.

- **Highlight Filter Gaps** – this check box sets whether or not filtered records cause a double-thick horizontal grid-line to be displayed between visible rows.
- **Disable Disassembly** – this check box disables disassembly after acquisition and upload. It is generally only useful for debugging, allowing a trace that causes a catastrophic error in decoding to be saved and sent to the Moving Pixel Company for analysis.
- **Reanalyze** – this button restarts decoding and disassembly on the current trace data. This may be needed if initial decoding was aborted before decoding was complete (in which case, only a partial disassembly is displayed). Click on the Reanalyze button to restart and complete a full decoding.

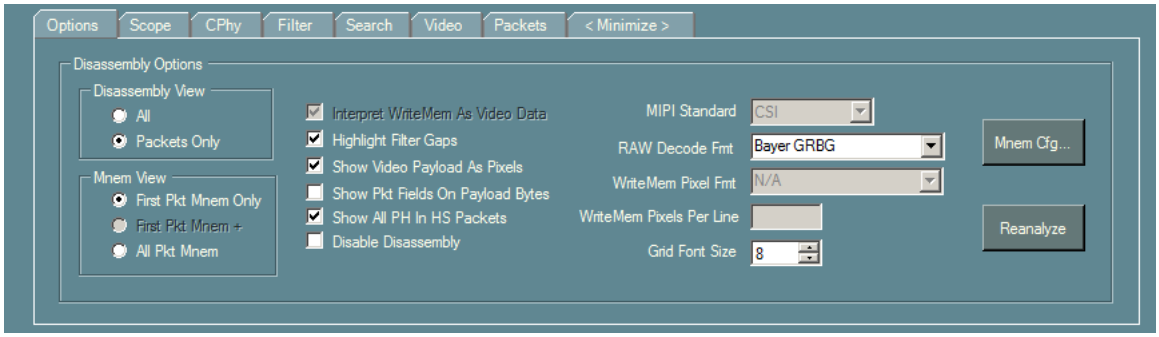

Figure 4 – Options Tab

The remaining controls in the Options window are protocol-related options, some applicable to both CSI and DSI and some applicable only to one or the other standard. Those applicable to both standards are:

- **Disassembly View** – two option buttons are provided to select the view mode: "All" or "Packets Only". "All" mode allows non-packet-byte specific records to be visible whereas "Packets Only" shows only packet-byte specific records. Please refer to section 3.2.2.5 for more information.
- **Mnem Cfg…** – this button bring up the Mnemonic Configuration dialog, allowing the user to select fields to display with the mnemonic in the listing as described in section 3.2.2.5,
- **MIPI Standard** – drop-down box to select the current standard, either CSI or DSI. (Currently, this control is disabled as only CSI is available.)
- **Show Video Payload As Pixels –** this check box configures how payload data for video packets is displayed in the disassembly. Without this option checked, each payload byte is displayed one-byte-per-record with no pixel decoding (just as all other non-video packets). However, with this option checked, pixels are decoded and presented, one pixel per record, with actual color components values displayed.[7]

[7] This discussion assumes that the mnemonic view of the video packet is "All Pkt Mnem". Otherwise, payload bytes are not visible.

- **Show Pkt Fields On Payload Bytes** – this check box determines whether packet fields such as VC, DataType, CRC, Frame, etc. are shown for other than the first byte in a packet. In some cases, is convenient ot exclude these fields from payload bytes, in particular during searches, where only the first packet bytes are desired for matches.

CSI has a couple standard-specific options:

- **Show All PH In HS Packets** – the check box enables/disables showing multiple records with mnemonics from the PH. That is, when this option is checked, only one copy of records with PH fields are shown for packets. As described in section 3.2.2.3, when this option is checked, the PH field values displayed are from a PH with a valid PHCRC, if one is found in the packet.
- **RAW Decode Fmt (CSI) –** this combo box indicates the format to use for decoding RAW video packets.

Bayer GRBG	Decodes lines 1,3,5… as alternating green, red pixels and lines 2,4,6… as alternating blue, green pixels.
Bayer RGGB	Decodes lines 1,3,5… as alternating red, green pixels and lines 2,4,6… as alternating green, blue pixels.
Bayer BGGR	Decodes lines 1,3,5… as alternating blue, green pixels and lines 2,4,6… as alternating green, red pixels.
Bayer GBRG	Decodes lines 1,3,5… as alternating green, blue pixels and lines 2,4,6… as alternating red, green pixels.
Monochrome	Decodes raw pixels as luminence values.

Finally, DSI has a few standard-specific options:

- **Interpret WriteMem As Video Data (DSI)** – this check box indicates that WriteMemoryStart and WriteMemoryContinue commands should be interpreted as containing video data. WriteMemoryStart packets always start a new frame. The data format and line length are set by "WriteMem Pixel Fmt" and "WriteMem Pixels Per Line" respectively. Note that if this option is checked and both WriteMemory and Packed Pixel packets are present in the trace, only WriteMemory frame data will be catalogued in the Video tab.
- **WriteMem Pixel Fmt (DSI) –** this combo box indicates the pixel format to use when decoding WriteMemory commands as video data. Options are:

Packed RGB 565	Packed RGB 121212
Packed RGB 666	Loose 20-bit YCbCr 422
Loose RGB 666	Packed 16-bit YCbCr 422
Packed RGB 888	Packed 24-bit YCbCr 422
Packed RGB 101010	

- **WriteMem Pixels Per Line (DSI) –** this text box indicates the number of pixels per line when decoding WriteMemory commands as video data.

3.2.4.2 Scope Tab

The Scope tab contains options and controls for configuration and interacting with the scope (see Figure 5).

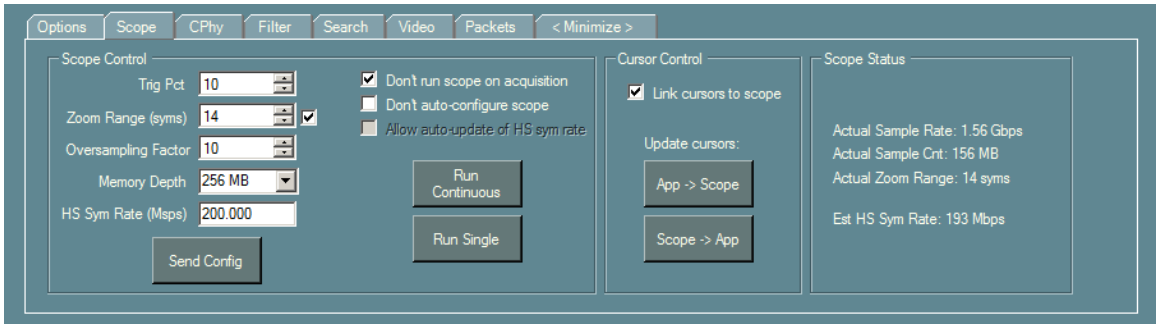

Figure 5 – Scope Tab

The following controls are provided on the scope tab:

- **Trig Pct** – this numeric control selects the horizontal trigger position for the scope, as a percent across the screen from left-to-right. Thus, a setting of 0 sets the trigger point at the left edge of the scope display and 100 sets the trigger point at the right edge of the scope display.
- **Zoom Range** – this numeric control sets the number of symbols to display when showing the zoom window on the scope.
- **Oversampling Factor** – this numeric control determines how to set the scope sampling rate relative to the HS bit rate. The setting describes how many scope samples should be used to acquire one HS bit. Generally, a range of 4 through 10 are reasonable settings for an acquisition. Note that if this value is set too small, decoding may be susceptible to noise and non-ideal waveform imperfections while larger values reduce the maximum number of symbols that can be acquired.
- **Memory Depth** – this setting indicates the number of samples to use for acquisition. If it is set to a value larger than the scope can support for the desired sample rate, the maximum supported value is used. Note that the number of samples actually used is displayed in the scope status pane of the scope tab (far left).
- **HS Sym Rate** – this text box is used to indicate the HS symbol rate of CPhy traffic on the link. This value is used to set the scope sampling rate (in conjunction with the Oversampling factor). If this value is set too high, decoding will still likely be correct. However, if this value is set too low, the scope will undersample the signal and the disassembly will likely have many errors.

  Note that this setting will be modified from it initial setting if "Allow auto-update of HS bit rate" is checked, reflecting the measured value for the HS bit rate during disassembly.

- **Don't run scope on acquisition** – this check box determines whether the scope is "Run" when the application's "Run" button is pressed. Uncheck this option before pressing Run, if you want to process a previously acquired waveform from the scope.
- **Don't auto-configure scope** – this check box prevents updating the scope settings when "Trig Pct", "Zoom Range", "Oversampling Factor", "Samples" and "HS Bit Rate" are set. Check this option if you want to use the current scope configuration for your acquisition and disassembly.
- **Allow auto-update of HS bits rate** – this check box enables/disables the HS Bit Rate setting to be updated after decoding to reflect the measured HS bit rate in the acquisition (currently, this function is disabled).
- **Run Continuous –** this button runs the scope in continuous mode. This button is provided as a convenience to the user.
- **Run Single –** this button runs the scope for a single acquisition. It is not necessary to use this button as the Run button will automatically acquire a new trace if "Don't run scope on acquisition" is unchecked. However, if you want to validate the acquisition before processing, you may want to use this button. In this case, check "Don't run scope on acquisition" before clicking the Run button to process the already acquired waveform.
- **Send Config** – this button sends the current configuration settings to the scope.
- **Cursor Control: Link cursors to scope** – this check box enables/disables a connection between application cursors and scope cursors. If checked, the scope cursors are updated whenever the application cursors are set.
- **Cursor Control: (App->Scope)** – this button updates the scope cursors from the current application cursor positions.
- **Cursor Control: (Scope->App)** – this button updates the application cursor positions from the current scope positions.

3.2.4.3 CPhy Tab

The CPhy tab contains controls for performing arbitrary user conversions between data bytes, CPhy symbols, and CPhy states. These conversions are provided to help engineers automate low-level CPhy conversion functions difficult to perform by hand.

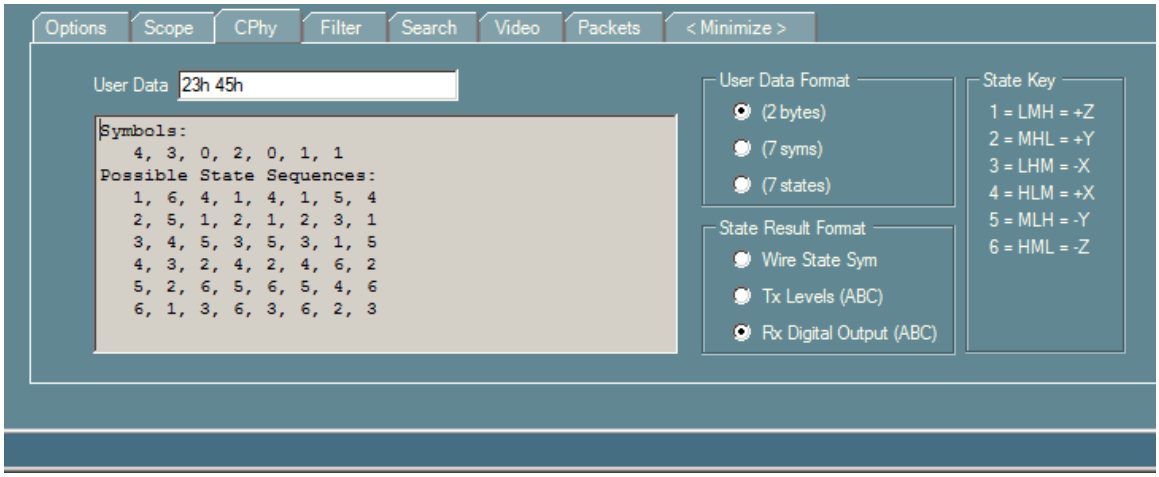

Figure 6 – CPhy Tab

The following controls are provided:

- **User Data** – this text box accepts the entry of user data, whether 2 data bytes, 7 symbols, or 7 states. The expected data type and format is dictated by the option button selection representing the User Data Format.
- **Results** – this unlabeled text box below the User Data text box displays conversion result.
- **User Data Format** – this section contains 3 option buttons labeled "(2 bytes)", "(7 syms)", or "(7 states)" and indicate the expected data type and format of the User Data text box.
- **State Result Format** – this section contains 3 options buttons labeled "Wire State Sym", "Tx Levels (ABC)" and indicate the display format of states in the Results text box.

Wire states can take on values of 1-6. These numeric values are displayed when the State Result Format is set to "Rx Digital Output". Another representation of wire-state is the CPhy names given to the Tx states: +X, -X, +Y, -Y, +Z, -Z, This display representation is used when "Tx Levels (ABC)" is selected. Finally, shorthand strings can be used to convey the A, B, and C wire voltage levels of a state. In this case, "H" represents a high-voltage, "M" represents a medium voltage, and "L" represents a low-voltage.

To perform a conversion, type data into the User Data text box and hit return (or tab). Values can be separated by spaces or commas as desired. When data bytes are entered, decimal values can be entered as normal, or hex values can be appended with 'h'. Based on the option button selections, a conversion is performed to the remaining forms and displayed.

When states are entered, the Results text box shows all possible conversions assuming various initial states. For each possible initial state, the resulting symbol sequence and data bytes are displayed. If the state sequence is invalid, "Invalid sequence" is displayed.

3.2.4.4 Filter Tab

The filter tab provides controls to filter or show records based on user criteria (Figure 7). Specifically, most controls in this tab help to define and apply a filter equation that is evaluated against each record to determine whether or not the record is displayed. Note that the application of the filter result is against the visible records defined by the current global view and packet mnemonic views.

Some controls such as the Filter, Show, and Disable radio buttons and the View combo box update the disassembly listing as soon as they are used. On the other hand, changes in the Mnemonic Type list box and the Filter Term grid do not take effect until the Apply button is clicked.

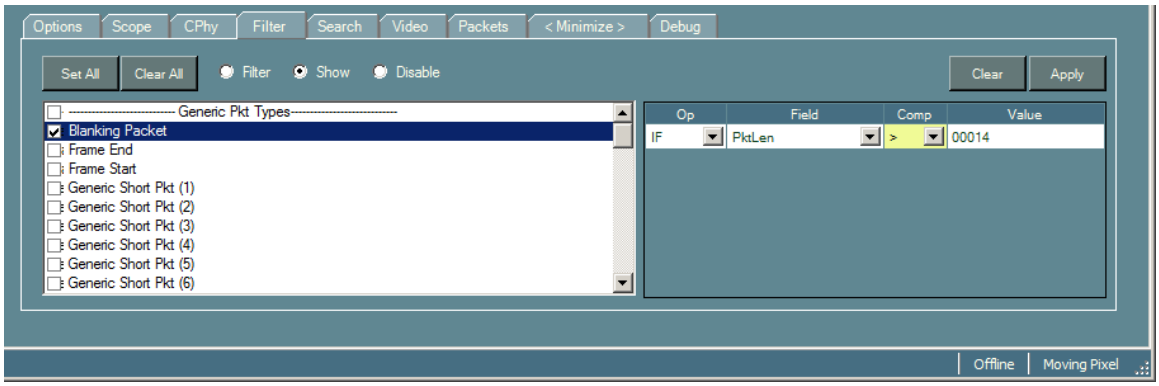

Figure 7 – Filter Tab

Here is an overview of the controls contained in the filter tab:

- **Mnemonic Type List Box** – contains a list of all mnemonic types defined for the current view. The user checks packet-types and record-types to use as criteria for the overall filter equation.
- **Filter Term Grid** – defines predicate terms of the filter equation based on packet fields.
- **Set All Button** – checks all mnemonic types in the list box.
- **Clear All Button** – clears all selected mnemonic types in the list box.
- **Filter Radio Button** – enables record filtering and uses the filter equation to determine which records should not be displayed (i.e. if the filter equation evaluates true for a record, it is not displayed).
- **Show Radio Button** – enables record filtering and uses the filter equation to determine which records should be displayed (i.e. if the filter equation evaluates true for a record, it is displayed).
- **Disable Radio Button** – disables record filtering.
- **Clear Button** – clears all terms in the filter term grid.
- **Apply Button** – Applies any changes to the filter criteria, updating the disassembly listing.

The Mnemonic Type list box and the Filter Term grid together define the overall filter equation. The list box defines a set of mnemonic types that qualify records for further evaluation by the filter terms. That is, records that satisfy the filter equation must have at least one lane with a mnemonic type than is checked in the list box. So, for example, at the extremes, if no types are checked, the filter equation will evaluate false for all records. And if all types are checked, the filter equation for each record will evaluate solely to the result from the Filter Term grid.

The Filter Term grid defines zero or more Boolean terms (one per row) which compare a record field to a constant value. Terms are logically connected via AND or OR operations to build a Sum-Of-Products equation used in the overall filter equation. The

AND operation takes precedence over the OR operation so A AND B OR C AND D is evaluated as (A AND B) OR (C AND D).

The filter grid defines four columns as follows:

- **Op** – defines how the term is logically connected to its predecessor, either via AND or OR (the first term, which has no predecessor is fixed to show IF). In addition to showing the logical operation associated with the term, this field is a combo box that provides options for adding or deleting terms. For AND terms, two options are available: add another AND term to the product or delete the current AND term. For IF or OR terms, two additional options are available: add another OR term to the equation or delete the current OR term (which includes all AND terms of the product).
- **Field** – combo box to select a record field name
- **Comp** – combo box to select a comparison operator (==, !=, <, <=, >, >=)
- **Value** – text box to enter a comparison value

To define a filter equation, begin with filling in the Field, Comp, and Value fields of the first term in the Filter Term grid. Next, select the Op drop-down and select "<Insert OR term>" or "<Insert AND term>" to add a new term to the equation. Fill in the fields of that term and continue inserting terms until the equation is fully defined.

Another method is to right-click in the disassembly grid and select one of the menu options that begin with "Exclude All But…". Alternatively, each of these menu options has a keyboard shortcut sequence that begins with Ctl-X. For example, if you right-click on a cell in the PktLen field that has a value of 4 and select "Exclude All But Same Val", the filter list box will check all packet types and enter the term"IF PktLen == 00004" in the filter grid. Then, the Show radio button will be selected and then the disassembly will update.

Figure 8 shows the results of this operation for a video capture. In particular, filtering has eliminated all packets except for Line Start and Line End. Note that many rows have a double-line dividing them. This is an optional highlighting method to indicate there are packets in the view that have been filtered. This highlight is enabled by checking "Highlight Filter Gaps" in the Options tab.

Numeric field values are assumed to be in decimal unless appended with an 'h' for a hexadecimal value or a 'b' for a binary value. When using hex or binary values, an 'x' may be used as a "don't care" placeholder. Thus, the value "1xh" will match values from 10h to 1fh.

Symbolic field comparison symbolic is supported, but the user should understand that values are compared using string-comparison and so results may not be as intended. For example, if the user constrains the filter equation to select a time range when the Time field is set to symbolic, he might set up the following equation:

IF Time > 100 us AND Time < 200 us

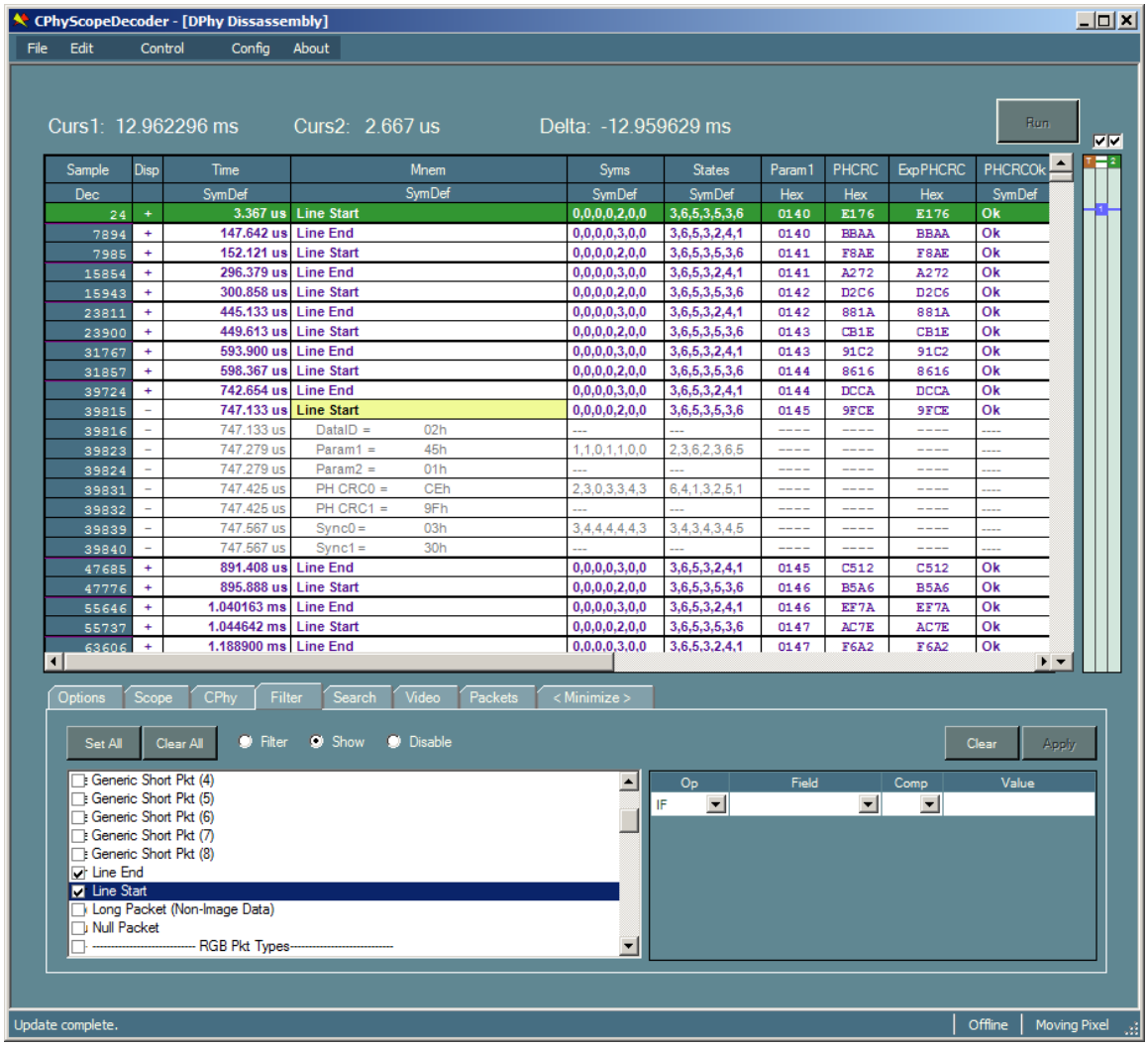

Figure 8 – Filter Example

However, the result of the string comparisons against Time values such as "120.888 ns" and "190.666666666 ms" will not be as desired. In general, the "==" and "!=" comparison operations are most predicatble, though be aware that comparisons are case-sensitive.

Note that changing the radix of a field from numeric to symbolic or symbolic to numeric *after defining filter or search terms* can affect the term's validity. Comparisons are performed on field values in the radix they are displayed (not the radix at the time the predicates were built). An error will occur if filtering or searching using a non-numeric predicate value on a field that is currently being displayed as a numeric value (though hex, decimal, and binary conversions are automatic).

3.2.4.5 Search Tab

The search tab contains controls for searching and optionally highlighting records that satisfy given criteria (Figure 9). Statistics about filtered and matched search records are also displayed.

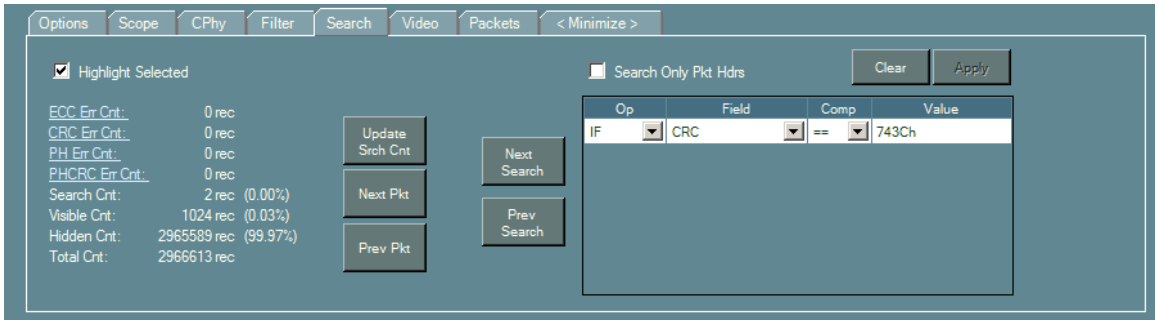

Figure 9 – Search Tab

Here is a description of the controls contained in the filter tab:

- **Highlight Selected –** when checked, records satisfying the search criteria are highlighted in the listing.
- **ECC Err Cnt** – displays the number of ECC errors detected in the trace. Clicking on this label will auto-fill the search criteria to find these records.
- **CRC Err Cnt** – displays the number of CRC errors detected in the trace. Clicking on this label will auto-fill the search criteria to find these records.
- **PH Err Cnt** – displays the number of PH errors detected in the trace. Clicking on this label will auto-fill the search criteria to find these records.
- **PHCRC Err Cnt** – displays the number of PHCRC errors detected in the trace. Clicking on this label will auto-fill the search criteria to find these records.
- **Search Cnt** – displays the number of records and percentage of all acquired records that satisfy the search criteria
- **Visible Cnt–** displays the number of records and percentage of all acquired records) that are currently visible in the listing.
- **Hidden Cnt–** displays the number of records and percentage of all acquired records that are currently hidden from view in the listing.
- **Total Cnt–** displays the number of acquired records.
- **Update Srch Cnt Button** – updates the Search Cnt readout if the initial attempt during disassembly update timed out (see description below).
- **Next Pkt Button** – scrolls to the next packet in the disassembly listing. Alternate ways of performing this function is to select "Next Packet" from the disassembly context menu or pressing Ctl-P.
- **Prev Pkt Button** – scrolls to the previous packet in the disassembly listing. Alternate ways of performing this function is to select "Prev Packet" from the disassembly context menu or pressing Alt-P.

- **Next Search Button**– scrolls the listing to the next record matching the search criteria. Alternate ways of performing this function is to select "Next Search" from the disassembly context menu or pressing Ctl-S.
- **Prev Search Button** – scrolls the listing to the previous record matching the search criteria. Alternate ways of invoking this function is the select "Prev Search" from the disassembly context menu or pressing Alt-S.
- **Search Only Pkt Hdrs** – check-box to enable/disable searching only on packet header records (i.e. LP records and HS records associated with the first byte of a packet). Checking this option speeds up searches.
- **Search Term Grid** – allows the user to define predicate terms of the search equation. This grid operates identically to the Filter Term grid described in the previous section.
- **Clear Button** – clears and filter term grid.
- **Apply Button** – Applies changes to the search term grid, updating the disassembly listing and readouts.

Note that depending on the size of the trace, the current view, and the complexity of the filter and search equations, performing a count of all search records in the trace can be time consuming. An initial attempt to update the search count is made during disassembly update, but with a ½ second timeout. If the timeout occurs, all the error counts and the Search Cnt is displayed as "<update>", indicating the user needs to click on the Update Srch Cnt button if to recomputed.

To search for and highlight records that satisfy a particular predicate equation, search criteria can be entered in to the search term grid. The user interaction with the search grid is almost identical as with the filter grid. Please see the previous section for more details.

One enhancement is the "Search Only Pkt Hdrs" check box. When this box is checked, searching is much faster, as search criteria is only evaluated against packet records and does not include payload or secondary header mnemonic records.

The search equation is built from terms defined and applied to each record in the listing when the Apply button is clicked. If the Highlight Selected checkbox is checked, records that satisfy the search equation are highlighted in the listing (note that the search equation is only evaluated against unfiltered records). The "Next Search" and "Prev Search" buttons can be used to scroll the listing to the next and previous qualifying records respectively.

3.2.4.6 Video Tab

The Video tab of the Disassembly Control panel contains information about video frames decoded from the trace (see Figure 10)

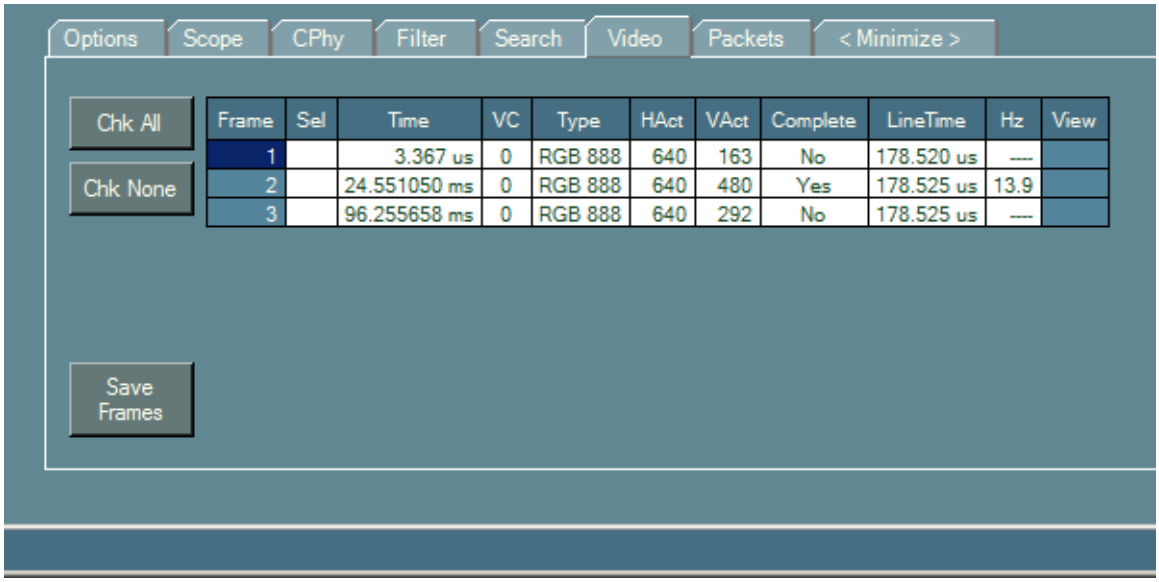

Figure 10 – Video Tab

In the video tab, a grid control displays one-line-per-decoded-frame.with the following fields:

Frame	A frame number assigned to the frame
Sel	Click-area to select the frame for saving. An 'X' is alternately displayed and cleared with each click.
Time	Time stamp of first frame packet (generally Frame Start or VSync Start packet)
VC	Virtual channel
Type	Video format
HAct	Number of active pixels in a packet. If variable packet lengths were found, indicates the longest packet found.
VAct	Number of active lines
Complete	'Yes' is displayed if Frame Start and Frame End (CSI) or VSync Start and a second VSync Start (starting the next frame for DSI) was seen.
LineTime	The longest time period between active packets (CSI) or HSync/VSync Start packets (DSI).
Hz	The Frequency between Frame Start (CSI) or VSync Start (DSI) and the next Frame Start or VSync Start.
View	Click-area labeled "View" to bring up a window displaying the video frame.

3.2.4.6.1 Scrolling Disassembly to Frame Start

To scroll disassembly to the location of the first packet in the video frame, click on the frame number in the video grid. The first packet of a video frame is usually the Frame

Start or VSync Start packet, but if these packets are missing, they could be other video packet types.

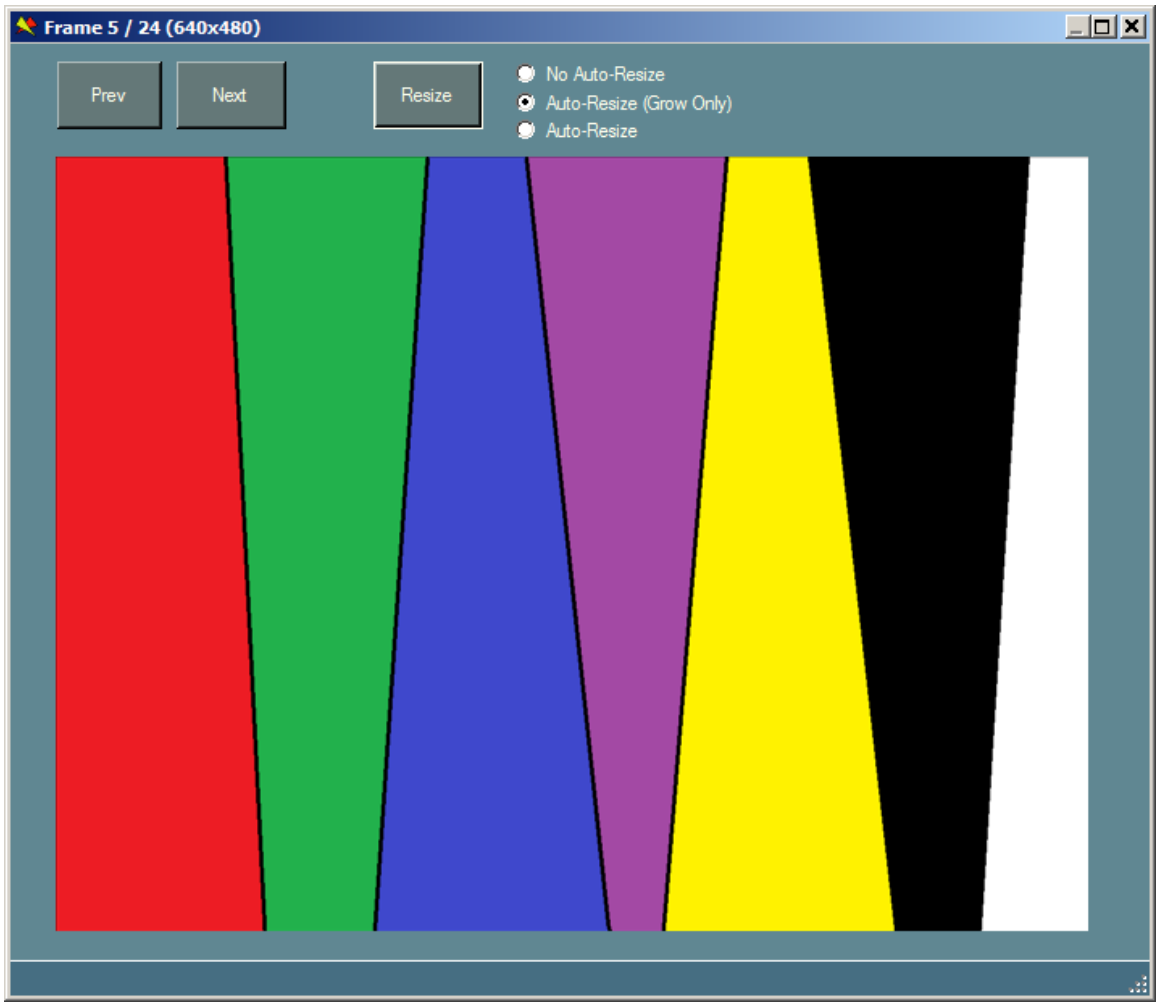

Figure 11 – Frame Display Dialog

3.2.4.6.2 Viewing Video Frames

To view a video frame, click on the "View" cell associated with the frame in the video grid (see Figure 11). This brings up the Frame Display dialog showing the captured frame. The Frame Display dialog has buttons labeled "Next" and "Prev" to step forward and backward in the frame sequence, allowing each frame to be viewed in turn. The title of the dialog indicates which frame number is being viewed and its dimensions.

The user can resize the dialog manually using the mouse by grabbing a dialog edge and dragging. If the image becomes too large for the dialog size, scroll bars appear to pan the image into view.

In addition, the dialog has a button labeled Resize and option buttons to select on of the following:

- No Auto-Resize
- Auto-Resize (Grow Only)
- Auto-Resize

These controls determine when the dialog is resized to fit the image. Note that it does not resize the image itself, just the dialog. Selecting "No Auto-Resize" means that the dialog is never automatically resized to fit the image. Whereas, "Auto-Resize (Grow Only)" causes the dialog to auto-resize if it is too small to display the image (but not too large) and "Auto-Resize" allows the dialog to always adjust to the image it displays.

3.2.4.6.3 Saving Video Frames

To save video frames to files, click on the cell labeled "View" for each frame you want to save, causing an 'X' to be displayed indicating that the frame is selected to be saved. Clicking on an 'X' unselects the frame. Alternatively, clicking on the "Chk All" button selects all frames and clicking on the "Chk None" button clears all frame selections.

Once frames have been selected, clicking the "Save Frames" button brings up a save dialog to browse and enter a file name.

The extension used for the file name indicates the format to use for saving the frame data. If the extension is "BMP", "JPG", "GIF", ".PNG", ".TIFF" (case is irrelevant), the file is saved in the requested format. Otherwise, the file is saved in binary format, exactly as received from the MIPI bus.

In saving a single frame, the file name is used unchanged. In saving multiple frames, the file name, minus extension, should end in a number. This number will be incremented for each frame saved. For example, if the save file name is "Frame123.bmp" and three frames are selected, they will be saved as "Frame123.bmp", "Frame124.bmp", and "Frame125.bmp".

3.2.4.7 Packets Tab

This tab, shows a summary of all packet types acquired in the trace (see Figure 12).

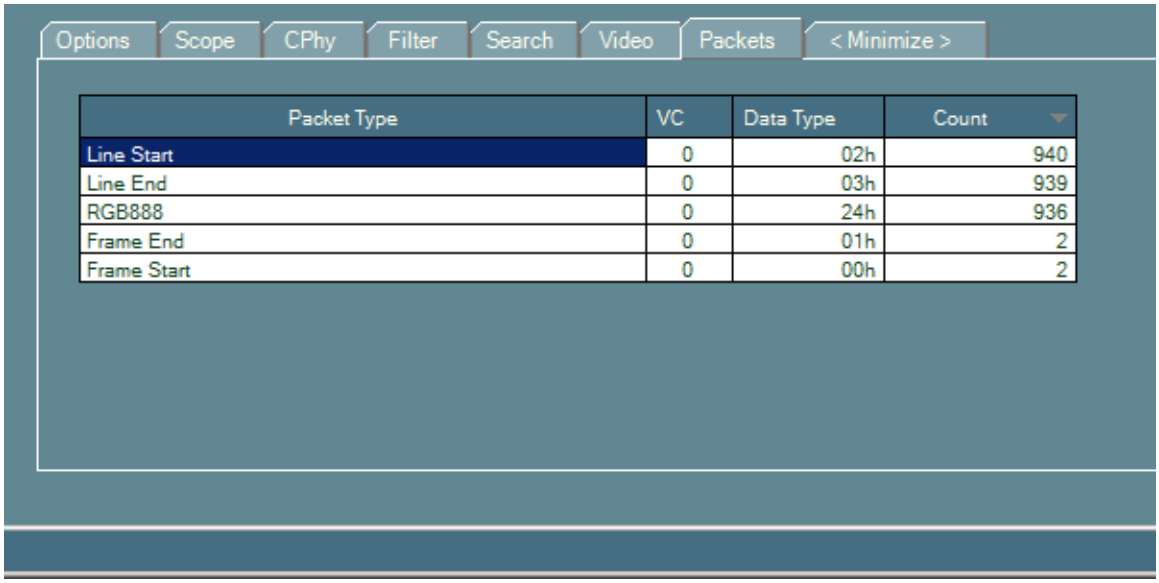

Figure 12 – Packets Tab

In this display the packet counts of each packet type, separated by Virtual Channel, are shown. Note that totals include packet types that may be hidden from display via filter criteria. Clicking on a column header alternately sorts the display in ascending or descending order based on the column content. For DSI, DCS command categories are summarized as well as individual DCS packet types.

3.2.4.8 <Minimize> Tab

The <Minimze> tab is used to collapse the Disassembly Control panel, so that only the tab labels remain. This allows the disassembly grid can be expanded using up almost all the area of the disassembly window. The Disassembly Control panel re-expands when any other tab is clicked.

3.3 Defining Custom Commands

Custom commands are DataTypes or DCS command codes that are not defined in the relevant MIPI specification. If present in the CPhy bus traffic, custom DataTypes *for long packets* are important to enumerate to ensure the CPhyDecoder does not get confused during parsing. This is because the default behavior when the instrument encounters an unknown DataType is to assume it is a short packet.

To define custom commands, a dialog is provided when "Define Custom Commands…" is selected from the Config menu (see Figure 13). This dialog lets you specify custom short packet data types, long packet data types, and DCS commands. Simply enter lists of values (separated by spaces) in the appropriate text boxes and click OK. Note that DataTypes do not include the VC field and are masked to 6-bits. Values are parsed as decimal unless appended with an 'h', in which case they are parsed as hexadecimal (i.e. 20h = 32 decimal).

Clicking the Cancel button exits the dialog without registering any changes. Clicking the Clear All button clears all the text boxes.

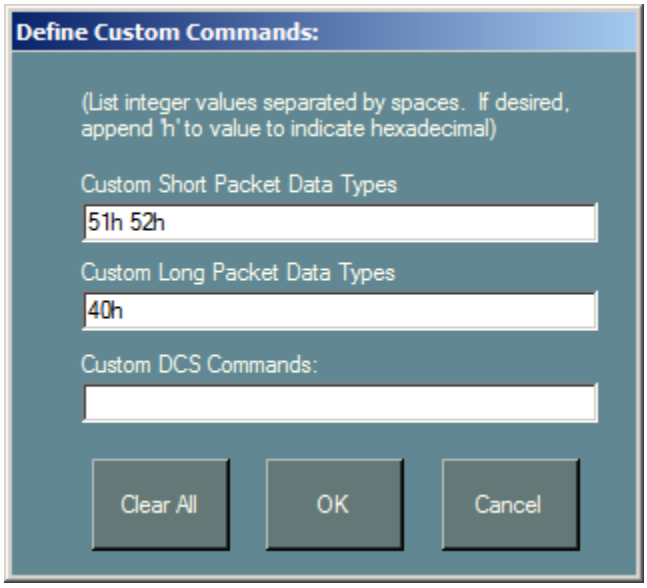

Figure 13 – Custom Command Dialog

3.4 Configuration Files

CPhyScopeDecoder has three file types that contain application configuration, user settings and captured data. They are all binary files with undocumented formatting. The first file type (.bin) is only used by the application to record global application state and cannot be explicitly generated by the user. Another file type (.cfg) contains almost all user settings and can be saved/loaded via File menu options. Finally, the last file type (.trc) contains captured trace data plus a few user settings important for disassembly. A trace file can also be saved/loaded via File menu options.

User configuration settings and application state are automatically saved in two application files when the application is closed and restored when the application is launched. These files are:

C:\ProgramData\Moving Pixel Company\CPhyScopeDecoder\AppConfig.bin

- last used firmware files
- last used instrument serial number, disassembly window
- last used disassembly window position and size
- user color scheme

C:\ProgramData\Moving Pixel Company\CPhyScopeDecoder\DefaultSettings.cfg

- all other configuration settings except for captured data
-

Note that when a trace file is loaded, a few critical user settings are also loaded and will overwrite their current values. These settings are: CPhy standard, lane count, HS frequency, and custom packet definitions.

3.5 Color Dialog

The color options dialog allows the user to adjust colors for use in the application. This dialog is experimental and only a few colors have the greatest impact on the aesthetics of the application, in particular, the Button background and the Form background.

To bring up the Color Options Dialog, select "Set Colors…" in the Options menu. To change a color, first select the color type option, for example in Figure 14, the "Form BG" (form background) is selected. Then left-click and drag in the color square, adjust the color sliders, or type in color values to set the color. The color rectangles to the left of the option buttons show the currently assigned color. Clicking on a rectangle makes the selected color the current color, i.e. fills in the color settings. This way, control colors can easily be copied to other controls.

As colors are selected, the main window colors change to reflect the new settings. When finished, click OK to keep the new colors, Cancel to discard the new colors, or Defaults to restore the colors to the application defaults.

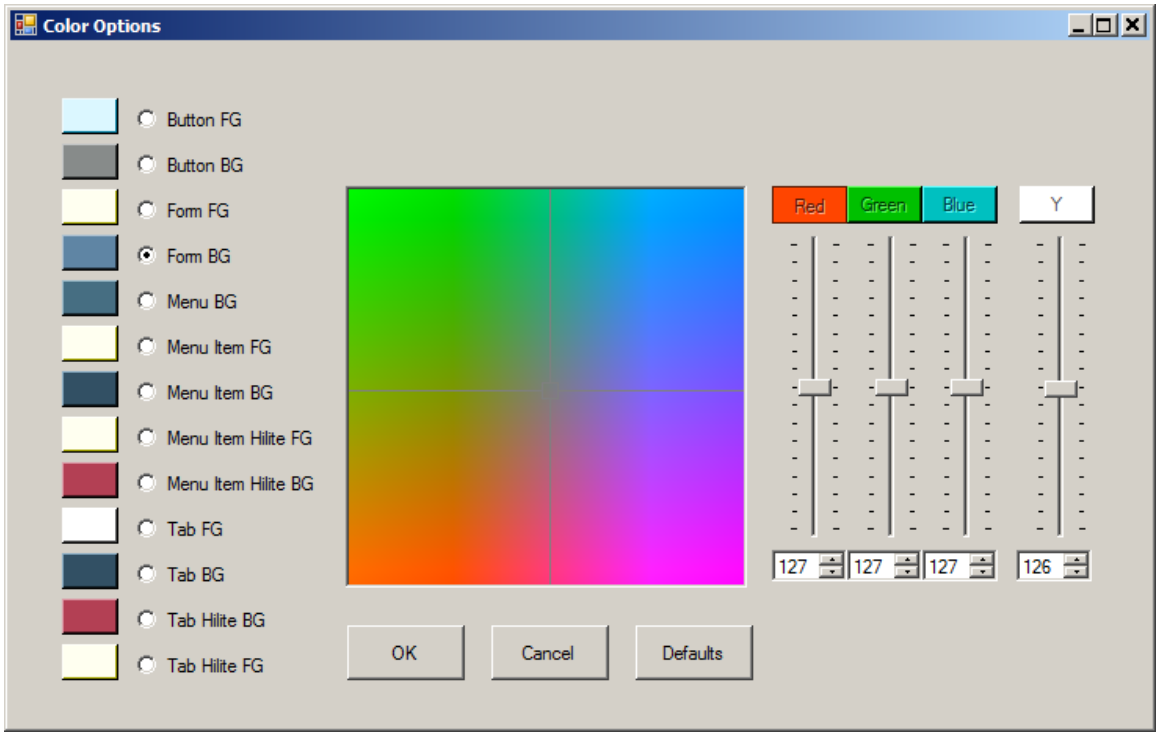

Figure 14 – Color Dialog

3.6 Menus

This section outlines the menu commands available in CPhyScopeDecoder:

- **File**:
  - **Load Cfg… –** loads a previously saved configuration file, overwriting the current configuration.

  - **Save Cfg --** saves the current configuration to a file.
  - **Save Cfg As… --** same as the **Save Cfg** command above except a dialog appears to browse and enter a new configuration file name.
  - **Load Trace** – loads a previously saved trace file, overwriting the current captured data.
  - **Save Trace As**… **--** saves the current captured data to a trace file specified through a file dialog.
  - **Import From Scope File** – loads a saved binary waveform file from the scope. This file *must contain all three waveforms* associated with A-B, B-C, and C-A CPhy signals. While the sample rate is obtained from the scope file, the user should ensure that the HS Bit rate is set reasonably or that "Allow auto-update of HS bit rate" is set. Note that currently, this function is implemented only for Agilent waveform files.
  - **<recent files>** a list of the four most recent configuration files. Selecting one of these file names loads the file.
  - **Clear Recent File List –** clears the most recent file list
  - **Exit** – exits the application
- **Edit:**
  - **Select All** – selects all records in the disassembly (usually used prior to saving selected records to a file)
  - **Save Selected Rows To PDF** – brings up a file dialog to obtain a pdf file name as a destination for writing selected rows.
  - **Save Selected Rows to CSV** – brings up a file dialog to obtain a csv file name as a destination for writing selected rows.
- **Control**:
  - **Connect to Scope…** – brings up the connection dialog (see section 3.1).
  - **Disconnect from Scope** – disconnects from the scope, putting the application in offline mode.
  - **Enable RPC** – if unchecked, brings up a dialog to set the RPC port number (and checks the option). If checked, the option is unchecked.
- **Config:**
  - **Define Custom Commands…** – brings up the custom command dialog (see section 3.3)
  - **Set Color**s **–** brings up the Color Options dialog to customize the colors of forms, buttons, and other controls.
- **About**
  - **Help** – displays this manual as a help file
  - **About** – brings up a summary window displaying the current software version. This dialog also includes a summary of release notes of changes/fixes for each version.

4 Remote Control

To facilitate automated control and testing, CPhyScopeDecoder supports receiving requests from other applications via a TCP/IP server port. This section describes details of the communication interface, supported commands, and example client code provided with installation of the CPhyScopeDecoder application. It is assumed that the user is conversant with programming in the .NET environment or otherwise knowledgeable about interfacing between his preferred programming environment (e.g. LabView, MATLAB, Python) and .NET.

4.1 Using CPhyScopeDecoder as an RPC Server

To enable CPhyScopeDecoder as an RPC server, simply select the "Enable RPC" menu option in the Control menu. If this option is not already checked, a simple dialog will appear requesting the port number to use for incoming RPC requests. You will need to use a port number that is not already in use by your system and other applications. This number should be between 1024 and 65525, though higher numbers in the range may be preferred as there are fewer port numbers at the high-end that are used by well-known applications. For more information, go to the following link:

http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

To disable CPhyScopeDecoder as an RPC server, uncheck the "Enable RPC" menu option.

When CPhyScopeDecoder is enabled as an RPC server, the user can interact with the GUI as normal. In addition, RPC calls from other applications can interact with the application as well, able to perform many of the functions programmatically that the user can perform using the GUI.

Notes:

- Currently there is no access control to prevent multiple applications from connecting and making RPC calls simultaneously to the CPhyScopeDecoder server even as the user is using the application, which, of course, may cause confusing behavior.
- Where a user initiated action would normally elicit a message box on CPhyScopeDecoder, usually a warning or a confirmation, the equivalent RPC call will not do so. Instead, the operation proceeds as if the user had responded affirmative to the message.
- RPC calls block until CPhyScopeDecoder has completed processing them. In some cases, this can take a few seconds, for example when saving and restoring large trace files. For client applications that require a responsive interface, these RPC calls should be made from a separate thread.

4.2 CPhyScopeDecoderRPC Library

To facilitate the user in building RPC client programs to interact with CPhyScopeDecoder, a DLL is provided (CPhyScopeDecoderRPC.dll). This library is written in C# and is managed by .NET and it provides classes to encapsulate error codes, command definitions, command field definitions, and a few simple calls to establish a connection to the CPhyScopeDecoder server, send commands, and read status. These classes are contained in the namespace "CPhyScopeDecoderRPC" and are described below.

Note: for those interfacing to the DPhyDecoderCltRPC DLL from a language other than C# or an environment different than .NET and you have suggestions how to make the DLL more usable for your needs, please contact the Moving Pixel Company. This may be necessary especially with respect to function signatures, which currently take advantage of C# generic types and may not be easily portable to other languages or environments.

4.2.1 Class Overview

The CPhyScopeDecoderRPC namespace contains four classes:

- CPhyScopeDecoderRPCClient – main client instance to connect and send commands to the CPhyScopeDecoder server
- RPCErrs – error code definitions
- RPCCmds – RPC command code definitions
- RPCDefs – RPC command parameter definitions

These classes are further described in the following sections.

4.2.1.1 CPhyScopeDecoderRPCClient Class

The CPhyScopeDecoderRPCClient Class represents the main class of the library, providing an interface to connect to the CPhyScopeDecoder server, send RPC commands, and obtain results.

The client constructor takes no arguments and is simply created with the call:

```
      CPhyScopeDecoderRPCClient client = new
CPhyScopeDecoderRPCClient();
```

Next, should be a call to connect to a CPhyScopeDecoder server. The server should be running and enabled for RPC. As an example, the following call connects to a CPhyScopeDecoder server located on the local host machine and enabled for RPC at port 2799:

```
      int rc = client.Connect("", 2799);
```

When finished sending RPC commands, the client can disconnect from the server with

```
      int rc = client.Disconnect();
```

4.2.1.2 RPCErrs Class

The RPCErrs class contains definitions for the possible error codes that can occur in the CPhyScopeDecoder server. However, note that only a small subset of the errors in this class will be returned by an RPC call. No comprehensive documentation of these errors is provided currently – the names are meant to provide a more descriptive indication of the error that occurred. In many cases, the error code returned is also supplemented with an error message that is returned as well by the RPC call.

A few error codes deserve further comment:

- **FAIL**: used as a generic catch-all failure code.
- **LOCAL_FAILURE:** indicates that an error occurred on the client side (before reaching the CPhyScopeDecoder server). Often this error is associated with an error marshaling unexpected or incorrect command arguments.
- **ARGTYPE_MISMATCH**: this error is returned if the expected argument type for a command parameter does not match the argument type it received.
- **INVALID_PARAM:** this is a common error returned indicating that one or more arguments are out of range.

4.2.1.3 RPCCmds Class

The RPCCmds class contains definitions for all the possible RPC commands that can be sent to the CPhyScopeDecoder server. They are documented in Appendix A. These command codes are used with the RPCCmd and RPCQuery calls (and their variants) in the DLL.

4.2.1.4 RPCDefs Class

The RPCDefs class contains constants to be used for RPC command parameters. These command codes are documented in Appendix A

4.2.2 Sending RPC Commands

Sending an RPC command requires the use of a generic RPC call RPCCmd, which has four function signatures, accommodating 0, 1, 2, and 3 command arguments respectively:

```
public int RPCCmd(int cmdCode, ref string errMsg,
                        ref string statusMsg)

public int RPCCmd<T1>(int cmdCode, T1 arg1,
                        ref string errMsg, ref string statusMsg)

public int RPCCmd<T1, T2>(int cmdCode, T1 arg1, T2 arg2,
                        ref string errMsg, ref string statusMsg)

public int RPCCmd<T1, T2>(int cmdCode, T1 arg1, T2 arg2, T3 arg3,
```

```
                                ref string errMsg, ref string statusMsg)
```

Commands that execute successfully return 0. Those that result in an error return a negative error code from the RPCErrs class. In addition, commands that result in a negative error code may provide a description of the error in the returned string errMsg parameter.

The RPCCmd functions are also used in the case of queries, when the return type is an integer. In this case, the return code can be negative to indicate and error or otherwise represents the returned value of the query. For example, the GET_MEMORY_DEPTH command, if successful, returns the requested number of bytes of scope memory to be used for acquisition.

For queries that don't return a single integer result, the RPCQuery functions are used. These generic calls support commands that take one, two or three parameters and provide a reference argument for the return value, respVal, that takes on the expected return type for the specific call made:

```
public int RPCQuery<T1>(int cmdCode,
         ref T1 respVal, ref string errMsg, ref string statusMsg)

public int RPCQuery<T1, T2>(int cmdCode, T1 arg1,
         ref T1 respVal, ref string errMsg, ref string statusMsg)

public int RPCQuery<T1, T2, T3>(int cmdCode, T1 arg1, T2 arg2,
         ref T1 respVal, ref string errMsg, ref string statusMsg)
```

The calls that require RPCQuery are tagged in their description in Appendix A with a "(RPCQuery)" designation.

Which call should be used depends on the RPC command: how many parameters are required, and what is the return type. For example to set the prefill percentage (i.e. trigger position), the single-argument signature call is used, e.g.

```
client.RPCCmd(RPCCmds.SET_PREFILL_PCT, 50, ref eMsg, ref sMsg)
```

On the other hand, to save a video frame to a BMP file, the dual-argument signature call is used, e.g.

```
client.RPCCmd(RPCCmds.SAVE_FRAME, 1, AppDir + "frame1.bmp", ref
eMsg, ref sMsg)
```

4.2.2.1 Alternate Command Interface For Non-.NET Environments

Some languages or programming environments cannot support the generic style argument passing (which is a Microsoft C# construct). Accordingly, the CPhyScopeDecoderRPCClient DLL supports the following alternate procedure calls:

```
public int RPCCmdF(int cmdCode, float arg1,
```

```
                    ref string errMsg, ref string statusMsg);
public int RPCCmdI(int cmdCode, int arg1,
                    ref string errMsg, ref string statusMsg);
public int RPCCmdIF(int cmdCode, int arg1, float arg2,
                    ref string errMsg, ref string statusMsg);
public int RPCCmdII(int cmdCode, int arg1, int arg2,
                    ref string errMsg, ref string statusMsg);
public int RPCCmdS(int cmdCode, string arg1,
                    ref string errMsg, ref string statusMsg);
public int RPCCmdIRetS(int cmdCode, int arg1, ref string rVal,
                    ref string errMsg, ref string statusMsg);
public int RPCCmdSRetS(int cmdCode, string arg1, ref string rVal,
                    ref string errMsg, ref string statusMsg);
```

These calls have explicit parameter types, different calls for different combinations. The naming convention is to use 'I' for integer parameter type, 'F' for float parameter type, and 'S' for string parameter type. 'RetS' indicates that the rVal reference parameter is of type string.

4.3 CPhyScopeDecoderRPCTest Project

When CPhyScopeDecoder is installed, a Visual Studio 2010 project written in C# is included that demonstrates use of the RPC interface. The project is called CPhyScopeDecoderRPCTest and is located in the CPhyScopeDecoder top-level application directory:

c:\Program Files\TMPC\CPhyScopeDecoder

The Main procedure in the project is located in Program.cs and simply connects to the CPhyScopeDecoder server, which is assumed to be running on the local host and enabled for RPC at port 3333. In addition, CPhyScopeDecoder should be connected to a DPhy Decoder instrument. Once connected, the routine simply progresses through almost all of the RPC commands available. Note that this code doesn't perform anything particularly useful and is intended to be run in the debugger, stepping or running with breakpoints set at locations of interest.

Note in the References section the reference to the CPhyScopeDecoderRPC DLL, which is located in the bin/Debug directory. In addition, a type library of the DLL is also provided, which may be useful to non-.NET programmers as it contains all of the DLL routine calling signatures.

5 Appendix A: RPC Command Reference

Table 1: RPC Commands (RPCCmd class)

RPC Command	Code	Description
File I/O Commands		
LOAD_CONFIG_FILE arg1 = configuration filename (string)	0x01	Loads a configuration file from the given file.
SAVE_CONFIG_FILE arg1 = configuration filename (string)	0x02	Saves the current configuration from the given file.
LOAD_TRACE_FILE arg1 = trace filename (string)	0x03	Loads a trace file from the given file.
SAVE_TRACE_FILE arg1 = trace filename (string)	0x04	Saves the current acquisition to the given file.
Configuration Write Commands		
SET_CUSTOM_SHORT_PACKETS arg1 = list of short packet dataIDs (int[], range 0-63)	0x20	Sets the custom short dataID list for decoding.
SET_CUSTOM_LONG_PACKETS arg1 = list of long packet dataIDs (int[], range 0-63)	0x21	Sets the custom long dataID list for decoding.
SET_CUSTOM_DCS_PACKETS arg1 = list of DCS commands (int[], range 0-255)	0x22	Sets the custom DCS command list for decoding.
SET_MIPI_STANDARD arg1 = standard (int, use STANDARD_x defines)	0x23	Sets the current MIPI standard (currently unimplemented)
SET_MAX_LANE_COUNT arg1 = maximum lane count(int, range 1-4)	0x24	Sets the Max Lane Count setting (currently unimplemented)
SET_DATA_LANE_MAP arg1 = data lane mapping (int, 16-bit value)	0x26	Sets the Data Lane map setting (currently unimplemented)
Configuration Read Commands		
GET_CUSTOM_SHORT_PACKETS respVal = list of short packet dataIDs (int[])	0x40	Gets the custom short dataID list (RPCQuery)
GET_CUSTOM_LONG_PACKETS respVal = list of long packet dataIDs (int[])	0x41	Gets the custom long dataID list (RPCQuery)
GET_CUSTOM_DCS_COMMANDS respVal = list of DCS commands (int[])	0x42	Gets the custom DCS command list (RPCQuery)
GET_MIPI_STANDARD returns MIPI standard (STANDARD_x define)	0x43	Gets the current MIPI standard setting (currently unimplemented)
GET_MAX_LANE_COUNT returns Max Lane Count	0x44	Gets the current Max Lane Count setting (currently unimplemented)
GET_DATA_LANE_MAP returns Data Lane Map value	0x46	Gets the current Data Lane Map setting (currently unimplemented)
Status Read Commands		

Acquisition Control Commands

SET_PREFILL_PCT arg1 = prefill percent (int, 0-99)	0xa0	Sets the prefill buffer percentage (i.e. trigger position in acquisition buffer)
SET_MEMORY_DEPTH arg1 = memory depth (int, (1<<13) to (1 << 27))	0xa1	Sets the requested acquisition buffer size in bytes.
SET_OVERSAMPLING_FACTOR arg1 = oversampling factor (int, 4 to 20)	0xa2	Sets the scope oversampling factor
SET_HS_SYM_RATE arg1 = HS symbol rate (float)	0xa3	Sets the HS symbol rate
RUN	0xa8	Starts an acquisition
ABORT	0xaa	Stops an acquisition, upload, or disassembly.
GET_RUN_STATUS returns run status (STATUS_x defines)	0xab	Gets the current acquisition.
GET_PREFILL_PCT returns prefill percent (int)	0xac	Gets the current prefill buffer percentage
GET_MEMORY_DEPTH returns memory depth (int)	0xad	Gets the current requested acquisition buffer size.
GET_ACTUAL_SAMPLE_COUNT returns actual scope acquisition size (int, bytes)	0xae	Gets the actual acquisition buffer size supported by scope.
GET_ACTUAL_SAMPLE_RATE respVal = actual scope sample rate (double, Hz)	0xaf	Gets the actual sample rate supported by scope (RPCQuery)

Disassembly Window Commands

GET_FRAME_COUNT returns the number of frames (int)	0xf0	Gets the number of frames detected in the listing.
GET_RECORD_COUNT returns the number of records (int)	0xf1	Gets the total number of records in the acquisition.
GET_DISASM_FIELD_STRING returns the disassembly field string (string)	0xf2	Gets the string displayed in the disassembly window, given the sample number and column name. (RPCQuery)
GET_VIDEO_SUMMARY_FIELD_STRING returns the video summary field string (string)	0xf3	Gets the string displayed in the video summary tab, given the ones-based frame number and column name. (RPCQuery)
GET_FRAME_START_RECORD returns the record number of the frame start (int)	0xf4	Gets the record number of the start of a frame given its ones-based frame number.
SAVE_FRAME arg1 = ones-based frame index (int) arg2 = file name (string)	0x115	Saves the referenced frame to a file. The file type is determined by the extension (BMP, JPG, TIF, GIF) or otherwise is saved in binary format.
SET_CURRENT_RECORD arg1 = record number (int)	0x116	Sets the current record in the disassembly window (scrolls to view).

SET_MNEM_VIEW_STATE arg1 = mnem view state (int, VIEW_STATE_x define) arg2 = start record index (int, optional) arg3 = end record index (int, optional, inclusive)	0x117	Sets the mnemonic view state for the given record range. The start and end record indexes are optional. If not present, the command applies to the entire acquisition.
SET_SEARCH_STRING arg1 = search string (string) Define terms as "<IF | AND | OR> <field> <op> <value>" Use '\n' to separate terms Example: "IF CRCOk == Err\n OR ECCOk == Err"	0x118	Sets the search field equation controls in the search tab from the given string.
NEXT_PREV Arg1 = search function opcode (int) Use (NEXT_x and PREV_x defines)	0x119	Call a search function. Same as context menu search functions.
EXPORT_CSV arg1 = CSV file name (relative to CPhyScopeDecoder app) arg2 = start sample index (int, optional) arg3 = end sample index (int, optional, inclusive)	0x11a	Exports the given record range to a CSV file. The start and end record indexes are optional. If not present, the command applies to the entire acquisition.

Table 1: MIPI Standards (RPCDefs class)

STANDARD_DSI	0
STANDARD_CSI	1

Table 7: Run Status (RPCDefs class)

STATUS_IDLE	0
STATUS_CAPTURING	1
STATUS_UPLOADING	3
STATUS_DISASSEMBLING	4
STATUS_IDLE_AFTER_TIMEOUT	5
STATUS_IDLE_AFTER_ERROR	6

Table 9: Mnemonic View States (RPCDefs class)

MNEM_VIEW_FIRST_PKT_MNEM_ONLY	0
MNEM_VIEW_FIRST_PKT_MNEM_PLUS	1
MNEM_VIEW_ALL_PKT_MNEM	2

Table 10: NEXT_PREV Opcodes (RPCDefs class)

SEARCH_FRAME_START	13
SEARCH_SAME_MNEM	14
SEARCH_SAME_VAL	15
SEARCH_SAME_MNEM_AND_VAL	16
NEXT_SEARCH	18
NEXT_PACKET	19
NEXT_FRAME_START	20
NEXT_SAME_MNEM	21
NEXT_SAME_VAL	22
NEXT_SAME_MNEM_AND_VAL	23
PREV_SEARCH	25

PREV_PACKET	26
PREV_FRAME_START	27
PREV_SAME_MNEM	28
PREV_SAME_VAL	29
PREV_SAME_MNEM_AND_VAL	30
NEXT_NONNULL_FIELD_VAL	100
PREV_NONNULL_FIELD_VAL	101