

CSI Video Player RPC Software



Table Of Contents

1	<i>Overview</i>	3
1.1	Introduction.....	3
1.2	RPC	3
1.3	Macros.....	3
2	<i>CSI Video Player GUI.....</i>	5
2.1	Frame Sequence Format	6
2.2	Video Line Format.....	7
2.3	Line Format Options	8
2.4	Other Configuration Options.....	8
2.5	Frame Configuration Controls	9
2.6	Port Connection Controls.....	10
2.7	Configuration Files	10
3	<i>CSI Video Player Code</i>	11
3.1	Settings Class.....	11
3.2	CSI Class.....	12
3.3	New RPC Commands	12
3.3.1	LOAD_FRAME.....	12
3.3.2	DEALLOC_FRAME	12
3.3.3	MARK_ZERO_POS.....	12
3.3.4	LP_DELAY_TO_POS.....	13
3.3.5	MEASURE_MACRO.....	13
3.3.6	PIXEL_STREAM_x.....	14
3.4	CSIVideoPlayerForm Frame Construction	14
3.4.1	Prepare Routine.....	14
3.4.2	DeriveSettings Routine	14
3.4.3	SendFrame Routine.....	15
3.4.4	AddSingleFrame	15
3.4.5	AddConcentricSequentialFrames	16
3.4.6	AddConcentricInterleavedFrames	16
3.4.7	AddActiveLines	16
3.4.8	AddInterleavedActiveLinesDistrib	16
3.4.9	AddVBlanking	16
3.4.10	AddActiveLine.....	17

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>

1 Overview

1.1 Introduction

The CSI Video Player is an example Visual Studio Project written in C# to generate several CSI frame constructs supported by the standard. In particular, several variations of two frames having different VC, different data types, and/or different dimensions are supported, including sequential and interleaved frame structures.

The frame constructs are defined as macros and sent via RPC to be played to The Moving Pixel Company (TMPC) generation software (PGRemote or PGRemoteForP338). It is assumed users of this software already have familiarity with using PGRemote or PGRemoteForP338 to send MIPI commands using the PG and a P331, P332, or P338 probe.¹ It is also assumed that users understand the concepts of using RPC to communicate with PGRemote. Further documentation is available in the PGRemote User's Guide.

This software is freely available for download, modification, and use. In addition to providing customers TMPC with the complex features supported by this software, the intent is for customers to be able to use the code base as a reference and example for how RPC calls to PGRemote can be used to send commands on the MIPI bus. Moreover, because of the myriad variations of video frame testing needs by customers, this software allows the user the ability to enhance and modify the code to suit his needs.

1.2 RPC

RPC is a generic acronym that stands for Remote Procedure Call. When used in the context of TMPC's MIPI product, it refers to .NET messages that can be sent to PGRemote from another program using a provided DLL called PGRemoteClient.DLL. There are more than a hundred RPC messages supported by PGRemote, many for configuring parameters used by the application to send MIPI commands. Others messages send specific MIPI commands themselves, either for the CSI or DSI protocol.

Several new RPC commands were developed to support frame construction. In particular, commands now provide the ability to load image files into frame buffers and specify *relative* timing positions in a stream under construction (when emitting the LP11 state). Also, video commands themselves were modified to support using frame buffer data, by including frame buffer reference, byte offset and byte length parameters. These commands are described later in this document.

1.3 Macros

A macro is a list of RPC commands that are compiled into a single stream of data to be sent on the MIPI bus. Only the SEND_MIPI_CMD RPC call can be used in a macro,

¹ Except where expressly indicated, references to PGRemote in this document shall apply to PGRemoteForP338 as well.

where different parameters indicate the command type to send on the bus and define its arguments. Generally, each SEND_MIPI_CMD corresponds to a DPhy, CSI, or DSI packet on the bus. However, there are several special variants that don't correspond to a protocol packet, but that direct some other action by the PGRemote application.

2 CSI Video Player GUI

With the exception of minor support dialogs such for saving and loading configurations, browsing for image files, and the About dialog, the CSI Video Player GUI consists of a single main window (see Figure 1).

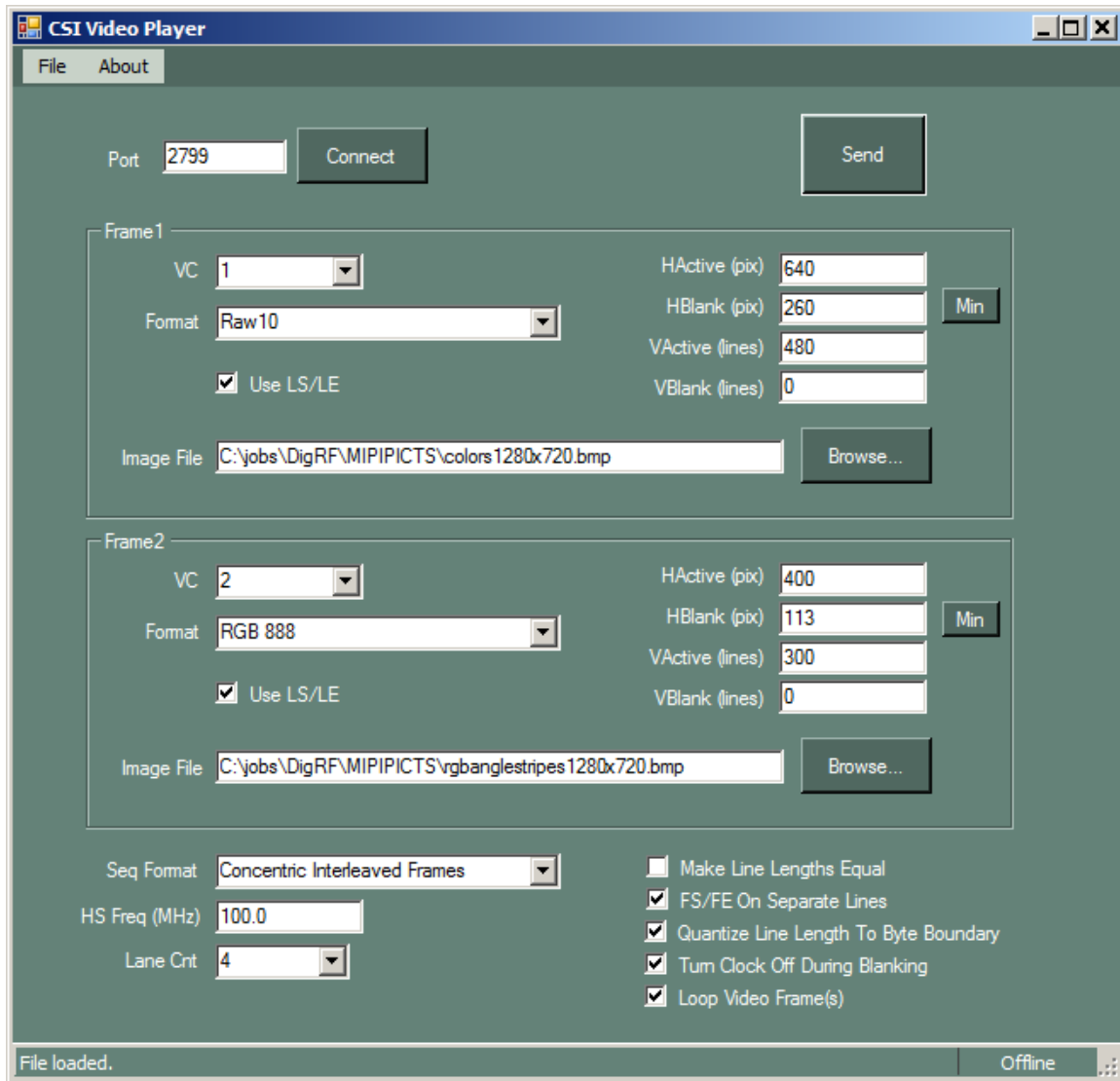


Figure 1 – CSI Video Player Main Window

The main goal of the application is to build and send a macro using RPC commands that sends two CSI video frames types out the MIPI bus. As with any macro, when played, it can be looped indefinitely or played once. It is the capability to send two frame types at once that is the additional functionality offered by this application, which is not offered by the built-in video mode capability of PGRemote.

The main window allows the user to describe the characteristics of two CSI frames to be constructed and sent out on the MIPI link, including an associated virtual channel, its video format, source image file, active and blanking dimensions. Several options are also provided, configuring variations on how the video stream should be constructed.

Once all the user settings and options have been entered, operation is straightforward. Assuming PGRemote is running and configured to accept incoming RPC requests via the same port specified in the Port textbox, the user only needs to press the Send button to start the process. Connection to PGRemote will occur automatically (if not already connected) then the macro will be built and sent according to the user configuration. The status bar will update the progress.

The remainder of this section describes the user-settings in more detail and specifically how they affect frame construction.

2.1 Frame Sequence Format

A central option that the user selects is the frame sequence format, which describes the overall structure of how video frames are to be sent. The “Seq Format” combobox has four options:

- **Single Frame** – This option is offered for convenience, only sending one frame (frame 1) in the sequence. This is equivalent to the built-in video mode offered by PGRemote, though the options to quantize the line length automatically, put Frame Start/End on separate lines, and automatically compute the minimum HBlanking period are enhancements to this mode not available in the PGRemote GUI.
- **Sequential Frames** – This option simply plays frame 1 in its entirety then frame 2. If the macro is looped, this means frame 1 and frame 2 alternate indefinitely on the link.
- **Concentric Sequential Frames** – This option embeds the active lines of both frames between Frame Start and Frame End packets for the frames. All frame 1 active lines are sent before all frame 2 active lines. While not required, this is generally used when the Virtual Channel of both frames is the same and the pixel formats (data types) are different.
- **Concentric Interleaved Frames** – This option again embeds the active lines of both frames between Frame Start and Frame End packets for the frames.² However, active lines from frame 1 and frame 2 are interleaved, spaced out in the combined frame. For example if VActive is the same for both frames, lines will alternate. However, if VActive for frame 1 is three times VActive for frame 2, three lines of frame 1 will output for every one line of frame 2.

² Again, if the Virtual Channel for both frames is the same, only one Frame Start and one Frame End is sent.

The next table shows symbolically how the different frame sequences are constructed. (Note these sequences assume “FS/FE On Separate Lines” option is set and the VActive settings for both frames are equal.)

Sequence Format	Sequence
Single Frame	FS1, F1L1, F1L2, ... FE1, VB1
Sequential Frames	FS1, F1L1, F1L2, ..., FE1, VB1, FS2, F2L1, F2L2, ..., FE2, VB2
Concentric Sequential Frames	FS1, FS2, F1L1, F1L2, ..., F2L1, F2L2, ..., FE1, FE2, VB1, VB2
Concentric Interleaved Frames	FS1, FS2, F1L1, F2L1, F1L2, F2L2, ... FE1, FE2, VB1, VB2

The next table defines the symbols used::

Symbol	Description
FS _x	Frame x Frame Start packet
FE _x	Frame x Frame End packet
FxLy	Frame x, line y
VB _x	Frame x vertical blanking period

2.2 Video Line Format

The general format of active video lines is as follows:

[FS | HBP] [LS] VP [LE] [FE | HFP] [HB]

using the symbols in the table below::

Symbol	Description
FS	Frame Start packet
HBP	Horizontal Back Porch LP11 blanking
LS	Line Start packet
VP	Video Packet
LE	Line End packet
FE	Frame End packet
HFP	Horizontal Front Porch LP11 blanking
HB	General Horizontal LP11 blanking

Brackets means the segment is optional based on user-settings. A ‘|’ symbol means that one or the other segment type applies depending on the line number. For example, [FS | HBP] describes an optional segment that either consists of a Frame Start packet or an equivalent LP11 time with the same duration as a Frame Start packet (SOT, FS, EOT).

Criteria that control optional segments are:

- LS and LE are present if “Use LS/LE” option is checked for the frame.
- FS | HBP and FE | HFP are present if “FS/FE On Separate Lines” is *not* checked.
- If present, FS is sent on line 1 of the frame and HBP is sent on all other lines.
- If present, FE is sent on the last line of the frame and HFP is sent on all other lines

2.3 Line Format Options

As indicated in the previous section, there are several check-boxes provided in the main window that affect line configuration. They are:

- Make Line Lengths Equal
- FS/FE On Separate Lines
- Quantize Line Length To Byte Boundary
- Turn Clock Off During Blanking

When “Make Line Lengths Equal” is checked, the total line length for both frames is set to the longer of the two frames. This means that the blanking period (segment HB in the previous section) is extended for whichever of the two frames has the shorter line length. If this option is not checked, line lengths are sent as configured and may be different between frames.

When “FS/FE On Separate Lines” is checked, the Frame Start and Frame End packets are sent on separate lines and are not considered part of an active line. When this option is not checked, the first line of each frame will have a Frame Start packet preceding the Line Start (if present) or video packet. Also, the last line of each frame will have a Frame End packet following the Line End (if present) or video packet. Finally, so that lines all have the same length, an equivalent LP11 time is added to all other lines during these segment times (HBP and HFP segments in the previous section).

When “Quantize Line Length To Byte Boundary” is checked, the horizontal blanking period (HB) is incrementally adjusted if necessary so that it can be sent by PGRemote. This is a requirement of the Moving Pixel Company MIPI solution, that line lengths (per lane) are all a multiple of 8 HS clocks. If this option is not checked, an error will be returned by PGRemote if a video line is not an integral number of bytes per lane.

When “Turn Clock Off During Blanking” is checked, the clock is turned off at the beginning of the HB and VB segments while in the LP11 state and then turned back on again at the end of the segment. Otherwise, the clock will remain on continuously.

2.4 Other Configuration Options

Other configuration options in the main window are:

- HS Freq
- Lane Cnt
- Loop Video Format

These options mimic the same settings in PGRemote and will overwrite them when the macro is sent. Other settings such as LP Frequency, DPhy Timing settings, and P338 mode settings are not currently provided. Their current values in PGRemote are used when the macro is sent.

2.5 Frame Configuration Controls

Each frame can be independently configured using the following controls:

Control	Usage
VC	Combo box that selects the Virtual Channel to use for the frame.
Format	Combo box that selects the video format (DataType) to use for the frame.
Use LS/LE	Check box that indicates whether to use Line Start and Line End packets in the line definition.
Image File	Text box that contains the path name to the image file to use for the frame. It is filled in when the Browse... button is used.
Browse	Button that brings up a file dialog to select the image file to use for the frame.
HActive (pix)	Text box specifying the number of active pixels in a line.
HBlank (pix)	Text box specifying the number of blanking pixels in a line. Note that this number must encompass all overhead associated with the line, including (possibly) FS, LS, LE, FE, Clock Off, Clock On, and SOT/EOT of the active video packet.
VActive (lines)	Text box specifying the number of active lines in the frame.
VBlank (lines)	Text box specifying the number of lines to use for vertical blanking.
Min	Button that computes the minimum legal HBlank setting, that is, the minimum setting that covers all non-video overhead in the line.

Additional notes:

- If PGRemoteForP338 is used, the file can be any of the supported file types (BMP, JPG, TIF, GIF, PNG) or test pattern names. Otherwise, the file should be a BMP or binary file of the correct dimensions.
- If PGRemoteForP338 is used, the image will be automatically scaled to HActive by VActive. Otherwise, the image file must match the given user dimensions.
- If HBlank is too small to account for all the line overhead, PGRemote will return the error: “Error adding macro command to macro. (Insufficient delay for LP_DELAY_TO_POS. Need an additional X bytes per lane)”.

2.6 Port Connection Controls

The port connection controls consist of a Port textbox and a Connect/Disconnect button. These controls are used to connect to PGRemote before sending any commands. However, a connection is automatically made if necessary when a command is sent.

Port Number – Text box indicating the port number to use to connect to PGRemote. This number must match the “RPC Port” setting in the Connect menu of PGRemote, which must also be enabled.

Connect/Disconnect – Button that, when clicked, initiates an RPC connection to PGRemote using the given Port Number. If successful, the status window shows “Connected” in the lower-right corner and the button label changes to Disconnect. Clicking the Disconnect button closes the RPC connection to PGRemote.

2.7 Configuration Files

The File menu supports saving and loading the current configuration using the “Save Cfg” and “Load Cfg” options respectively. The file extension .cfg is used (please provide other naming clues to distinguish between PGRemote configuration files and CSI Video Player configuration files which both have the same extension).

3 CSI Video Player Code

As mentioned in the overview section, the CSI Video Player is written in C# and is a Visual Studio 2008 project. The following files comprise the major components of the project:

Directory	File	Description
(1)	CSIVideoPlayer.sln	VS 2008 solution file
(2)	CSIVideoPlayer.csproj	VS 2008 project file
(2)	CSIVideoPlayer.cs	Main code file
(2)	Support.cs	CSI, DSI and Settings support classes
(2)	MsgBox.cs	MsgBox support class
(2)	About.cs	About dialog
(3)	CSIVideoPlayer.exe	CSIVideoPlayer executable
(3)	MessageBoxExLib.dll	MsgBox support DLL
(3)	PGRemoteRPCClient.dll	PGRemote RPC DLL

Where:

Directory (1) : CSIVideoPlayer\

Directory (2) : CSIVideoPlayer\CSIVideoPlayer\

Directory (3) : CSIVideoPlayer\CSIVideoPlayer \Bin\Release\

Much of the code is self-documenting and thus the focus of this section centers around routines in CSIVideoPlayer.cs. Note that PGRemoteRPCClient DLL provides the mechanism for sending RPC calls to PGRemote and is documented further in the PGRemote User's Manual. However, we do later describe several new RPC calls that are provided in the latest PGRemoteRPCClient DLL that facilitate video construction.

The CSIVideoPlayerForm class manages the CSI Video Player main window and the underlying functions to build and send the video macro to PGRemote. The functions in this class can be categorized into the following categories:

- Support Routines
- Form Handlers
- File Load/Save Routines
- Measurement and Parameter Computation Routines
- Frame Construction Routines

For the most part, only routines in the latter two categories will be discussed further in this section.

3.1 Settings Class

The CSIVideoPlayerForm class saves all user settings from controls within a global instance of the Settings class (called m_set). When preparing for frame construction, this

class is also used to store values derived from user settings or results from test macro measurements. Specific fields will be discussed throughout the discussion of the routines that compute or measure them.

3.2 CSI Class

The CSI class contains constants and routines that relate to the CSI protocol and packet formats. In particular the data types of all the pixel stream commands supported by PGRemote are available in the Formats global variable. Plus, there are the following routines:

- CSIVideoCmdString – converts a CSI command code (i.e. dataType) to a string description.
- CSIVideoCmdBitsPerPixel – returns the number of bits per pixel for a given CSI command code
- CSIVideoCmdToRPCCmd – returns the corresponding RPC command associated with the CSI command code.

3.3 New RPC Commands

As mentioned, several new RPC commands have been added to PGRemote to facilitate video construction. This section describes their functions.

3.3.1 LOAD_FRAME

The LOAD_FRAME command requests that PGRemote load an image from a file into one of four internal frame buffers.³ There are three parameters to this command:

1. Frame Buffer Number (integer, 0-3)
2. Video Format DataType (integer)
3. Image File Path or Test Pattern Name (string)

Note that this command does not use MIPICmd procedure call in PGRemoteRPCClient (which uses the SEND_MIPI_CMD message). Thus, it is not sent inside of a macro definition. Instead it uses the PGRemoteCmd call which preloads the frame into the frame buffer before the macro is sent.

3.3.2 DEALLOC_FRAME

The DEALLOC_FRAME command deallocates memory associated with a previously loaded frame (via LOAD_FRAME).

3.3.3 MARK_ZERO_POS

The MARK_ZERO_POS command takes no parameters and is used with the MIPICmd procedure call in PGRemoteRPCClient and thus can be sent in a macro definition. Its purpose is to save an internal stream pointer in PGRemote to the current location in the stream while compiling a macro.⁴ This pointer position can then be used later via the

³ In the case of PGRemoteForP338 a test pattern name can be used as well.

⁴ Note that the position is quantized to the next byte boundary in the stream.

LP_DELAY_TO_POS as a relative reference point for inserting LP11 delays into the macro for blanking. See LP_DELAY_TO_POS for more information.

3.3.4 LP_DELAY_TO_POS

The LP_DELAY_TO_POS command is similar to the LP_DELAY command except that a byte position *relative to the zero position* set by the MARK_ZERO_POS is given instead of an absolute time delay.⁵ This allows alignment of segments within lines, including the end of lines themselves.

There are three parameters to the LP_DELAY_TO_POS command:

1. Position (integer, byte offset from last zero position)
2. Driver Enable (integer, 0 == all lanes in high-impedance, 1 == all lanes driving)
3. LP State to drive (integer, 11-bit value representing lane/clock state)

The LP State setting determines the LP state of data lanes 0 – 3 and the clock lane as shown in the following table.⁶

LP State Bits	Description
D[1..0]	Lane 0 state: 0-3 correspond to LP00 – LP11
D[3..2]	Lane 1 state: 0-3 correspond to LP00 – LP11
D[5..4]	Lane 2 state: 0-3 correspond to LP00 – LP11
D[7..6]	Lane 3 state: 0-3 correspond to LP00 – LP11
D[10..8]	Clk Lane state: 0-4 correspond to LP00 – LP11, or current state

3.3.5 MEASURE_MACRO

The MEASURE_MACRO command is used in the same way as the SEND_MACRO command, in conjunction with the START_MACRO command. It allows a macro to be constructed and its length measured instead of sending it on to the MIPI bus. If the macro contains no errors, the returned value indicates the number of bytes per lane required to implement the macro. The value is rounded up to the nearest integral byte boundary if necessary.

Note that all macros implicitly begin and end in the LP11 state. Thus, the measurement will always include the SOT and EOT transition if it contains a HS packet. For example, the following sequence:

```
START_MACRO
SEND_MIPI_CMD FRAME_START ...
MEASURE_MACRO
```

⁵ The last call to MARK_ZERO_POS is used as the zero reference. If no call has been made previously in the macro, the start of the macro is used as the zero reference.

⁶ If the P338 is used in a dual-interface mode, both links drive the same state.

will return the number of bytes per lane required for the Frame Start packet, including SOT and EOT overhead.

3.3.6 PIXEL_STREAM_x

While not new, all of the PIXEL_STREAM commands now support an alternative method for referencing pixel data. In particular, special files name can be used to reference one of the internal frame buffers (rather than an image file or P338 test pattern). If this file name is used, then the arg1 and arg2 parameters provide additional indexing information.

The file name syntax for referencing an internal frame buffer is:

USERFRAME<n> where <n> is the buffer number (0-3)

For example, USERFRAME2 references internal frame buffer 2, which must have previously been loaded using the LOAD_FRAME command.

When the USERFRAME syntax is used, the arg1 and arg2 parameters of the pixel stream commands are as follows:

- arg1 – integer, starting byte offset in the frame buffer to use
- arg2 – integer, lengths of the pixel stream packet payload

3.4 CSIVideoPlayerForm Frame Construction

Frame construction begins when the Send button is clicked. The handler SendFrameButton_Click first calls the Prepare routine then the SendFrame routine.

3.4.1 Prepare Routine

The Prepare routine performs the following tasks:

- Reads in current user-settings from the GUI controls (GetControlData)
- Checks for connection to PGRemote. If not connected, connects to PGRemote (ConnectButton_Click)
- Initializes global settings in PGRemote, such as the protocol , HSFreq, lane count, and options (InitPGRemote)
- Derives, computes, and measures many parameters necessary to build the video frame (DeriveSettings).

3.4.2 DeriveSettings Routine

Many of the settings derived in DeriveSettings are associated with each frame. They are:

- The RPC video command associated with the frame format.
- The number of bits per pixel associated with the frame format.
- The total number of bits per video line associated with the frame format.

- The number of bytes per lane for a video line, quantized if allowed and necessary. Also, if “Make Line Lengths Equal” is set, we use the maximum of the line lengths between frame 1 and frame 2.
- The number of active bytes in a video line.

Other parameters we need for frame construction are measured by building simple macros in PGRemote and using the MEASURE_MACRO command. In this way we measure the number of bytes per lane required to send a short packet, the clock on protocol, and the clock off protocol.

3.4.3 SendFrame Routine

SendFrame is the top-level routine for constructing and sending the macro containing the video frame. The first task before building the macro is to load the image(s) into internal frame buffers of PGRemote. This is done using the LOAD_FRAME RPC command.

In addition, prior to loading each frame, we set the HActive and VActive timing parameters in PGRemote to the desired output frame dimensions. This is in case we are talking to PGRemoteForP338 which will rescale the input image according to these settings if the image file dimensions are different.

Next, we build the macro. After sending the “START_MACRO” RPC call we then call different routines to add RPC calls to build the frame depending on the sequence format selected by the user,. For a single frame, we call AddSingleFrame. For sequential frames, we call AddSingleFrame twice (for frame 1 then frame 2). For either of the two concentric settings, we call AddConcentricSequentialFrames or AddConcentricInterleavedFrames.

After all of the packet commands composing the macro have been sent, the SendFrame routine calls the “SEND_MACRO” command to build and sent the macro. The frame sequence either plays once and completes or loops indefinitely depending on the “Loop Video Frame(s)” user-setting.

Finally, once the macro has been sent, “DEALLOC_FRAME” is called for each frame to reclaim the frame buffer memory in PGRemote.

3.4.4 AddSingleFrame

AddSingleFrame adds packets to a macro definition to implement the sending of a single frame (frame 1). This routine basically calls AddActiveLines and then AddVBlanking to send the active and vertical blanking segments of the frame respectively.

If the option is set to send Frame Start and Frame End on separate lines (rather than on the first and last line of the frame), the AddActiveLines call is explicitly bracketed by FRAME_START and FRAME_END packets. Otherwise, AddActiveLines will add these packets when sending the first and last lines of the frame.

3.4.5 AddConcentricSequentialFrames

AddConcentricSequentialFrames calls AddActiveLines for frame 1 then again for frame 2. After the active lines have been sent, AddVBlanking is called for the combined number of lines for frame 1 and 2.

If the option is set to send Frame Start and Frame End on separate lines, the AddActiveLines calls are bracketed by Frame Start and Frame End packets for each frame, if the frames have different Virtual Channels, or just one Frame Start / Frame End if the Virtual Channels are the same. The sending of these packets is achieved via the AddConcentricFSFE routine.

3.4.6 AddConcentricInterleavedFrames

AddConcentricInterleavedFrames calls AddInterleavedActiveLinesDistrib to send interleaved lines from frame 1 and 2. After the active lines have been sent, AddVBlanking is called for the combined number of lines for frame 1 and 2.

If the option is set to send Frame Start and Frame End on separate lines, the AddActiveLines calls are bracketed by Frame Start and Frame End packets for each frame, if the frames have different Virtual Channels, or just one Frame Start / Frame End if the Virtual Channels are the same. The sending of these packets is achieved via the AddConcentricFSFE routine.

3.4.7 AddActiveLines

AddActiveLines calls AddActiveLine for each line in a frame, printing status messages as it progresses. If requested, it sets appropriate flags to have Frame Start and Frame End packets inserted into the first and last lines of the frame.

3.4.8 AddInterleavedActiveLinesDistrib

AddInterleavedActiveLinesDistrib calls AddActiveLine for each line in both frame 1 and frame 2, interleaving them according to each frame's current progress. Basically, whichever frame is least completed (measured as a percentage) determines which frame contributes the next line. Status messages are printed indicating progress.

Optionally, there is another routine that could be called instead of this one, called AddInterleavedActiveLinesOneForOne. This routine alternates lines from frame 1 and 2 until the shorter frame is exhausted, at which point the remaining lines are sent from the taller frame.

3.4.9 AddVBlanking

AddVBlanking adds the specified period of LP11 to the macro. If the user has requested that the clock be turned off during this time, the clock is accordingly turned off at the beginning of the period and turned back on again at the end of the period.

This routine makes use of the new MARK_ZERO_POS and LP_DELAY_TO_POS RPC commands to implement exact LP11 timing. First, MARK_ZERO_POS is called to set

the current location in the macro as a starting reference point. Optionally, the clock is then turned off. Then, the LP_DELAY_TO_POS command is sent giving the desired number of bytes per lane to send LP11.

If the clock is not turned off and consequently does not need to be turned back on, then this number is simply the requested duration of LP11. But, if the clock needs to be turned back on, this number is adjusted based on the time needed to turn the clock back on (measured in DeriveSettings). Then, the clock is turned back on, if requested. In this way, we can ensure the LP11 period has exactly the requested duration.

3.4.10 AddActiveLine

AddActiveLine sends all the segments necessary to create a video line. Many of the segments are optional and are included or not in the line based on user settings. Basically, the routine starts off by calling MARK_ZERO_POS to record the current line start position in the macro. Then, each segment is sent in the following order (if present in the line):

- Frame Start or LP11 equivalent period
- Line Start
- Video Packet – sent with a reference to the frame’s USERFRAME buffer having the appropriate byte offset and length.
- Line End
- Clock Off
- LP11 – to implement this, we make use of the LP_DELAY_TO_POS RPC call. If the clock does not need to be turned on afterward, we simply use the total line length (in bytes per lane) in the call. Otherwise, we adjust the delay period by the required time to turn the clock back on (as measured in DeriveSettings).
- Clock On