Harmonic Oxtel Protocol Specification

Version: 1.8
Author: Dan Montz
Date: 08-Aug-2017 16:36
URL: https://confluence360.harmonicinc.com/display/Spectrum/Harmonic+Oxtel+Protocol+Specification

Table of Contents

1 Overview 8
2 Communication 9
3 Harmonic Oxtel Model 10
3.1 Example Command Sequences 10
3.1.1 Loading a Template and Viewing On-Air 10
3.1.2 Loading a Template (using Preload) and Viewing On-Air 11
3.1.3 Using a Template with Updateable Text Fields 11
3.1.4 Using a Template with a 3-Point Animation 12
4 Oxtel Commands 14
4.1 Keyers 14
4.1.1 Commands and Responses 14
4.1.1.1 Fade Keyer (1) 14
4.1.1.2 Cut Keyer (3) 15
4.1.1.3 Set Transition Duration (B) 16
4.1.1.4 Set Fader Angle (@) 16
4.1.2 Unsolicited Tallies 17
4.1.2.1 Keyer Position Tally (3) 17
4.2 Templates 17
4.2.1 Commands and Responses 18
4.2.1.1 Load Image (R0) 18
4.2.1.2 Enquire Load Image (R0) 19
4.2.1.3 Preload Image (R7) 19
4.2.1.4 Enquire Preload Image (R7) 20
4.2.1.5 Erase Store (A) 21
4.2.1.6 Set Image Position (G) 21
4.2.1.7 Enquire Image Position (G) 21
4.2.2 Unsolicited Tallies 22
4.2.2.1 Image Load Tally (Y9) 22
4.2.2.2 Image Preload Tally (YA) 22
4.3 Template File Management 23
4.3.1 Commands and Responses 23
4.3.1.1 Enquire File Info (R3) 23
4.3.1.2 Query First File (R4) 24
4.3.1.3 Query Subsequent File (R5) 24
4.3.1.4 Enquire Extended File Information (R6) 25
4.3.1.5 Validate Template (RA) 26
4.3.1.6 Enable Media Tallies (YB) 27
4.3.1.7 Enquire Media Tallies (YB) 27
4.3.2 Unsolicited Tallies 28

4.3.2.1 Media Tallies (YB) 28
4.4 Animations 28
4.4.1 Commands and Responses 28
4.4.1.1 Start Animation (S0) 28
4.4.1.2 Stop Animation (S1) 29
4.4.1.3 Select Animation Frame (S2) 30
4.4.1.4 Restart Animation (S4) 30
4.4.1.5 Enable Play State Tally (YS) 31
4.4.1.6 Enquire Play State Tally (YS) 31
4.4.2 Unsolicited Tallies 32
4.4.2.1 Play State Tally (YS) - Harmonic Extension 32
4.5 EasyText 33
4.5.1 Commands and Responses 33
4.5.1.1 Update Text Field (Z0) 33
4.5.1.2 Render Box (Z3) 33
4.5.1.3 Change Image (Z4) 34
4.5.1.4 Stop Animation (Zf) 35
4.5.1.5 Pause/Restart Strap (Zg) 35
4.6 A/B Mixer 36
4.6.1 Commands and Responses 36
4.6.1.1 Cut to A (U0) 36
4.6.1.2 Cut to B (U1) 37
4.6.1.3 Fade to A (U2) 37
4.6.1.4 Fade to B (U3) 37
4.6.1.5 Cut AB (U4) 38
4.6.1.6 Fade AB (U5) 38
4.6.1.7 Set Transition Type (U6) 39
4.6.1.8 Asymmetric V-Fade AB (U8) 39
4.6.1.9 Set Absolute Mix (U9) 40
4.6.1.10 Asymmetric Transition (UA) 40
4.6.1.11 Fade to Specified Position (UC) 40
4.6.1.12 Select Mixer Input (UE) 41
4.6.1.13 Enquire Mixer Input (UE) 42
4.6.1.14 Enquire Mix Mode (Ua) 43
4.6.1.15 Color Generator Color (UZ) - Harmonic Extension 44
4.6.1.16 Enquire Color Generator Color (UZ) - Harmonic Extension 44
4.6.1.17 Enable Video Tallies (Y6) 45
4.6.1.18 Enquire Video Tallies (Y6) 45
4.6.2 Unsolicited Tallies 46
4.6.2.1 Video Tally (Y6) 46
4.7 Audio 46
4.7.1 Commands and Responses 46
4.7.1.1 Set Audio Profile (jAP) - Harmonic Extension 46

4.7.1.2 Enquire Audio Profile (jAP) - Harmonic Extension 47
4.8 System and Status 48
4.8.1 Commands and Responses 48
4.8.1.1 Enquire Latency (hLAT) - Harmonic Extension 48
4.8.1.2 Enquire Number of Graphic Layers (hNGL) - Harmonic Extension 49
4.8.1.3 Enquire System Status (M) 49
4.8.1.4 Enquire Video Layer Status (N) 50
4.8.1.5 Enquire Command Availability (X3) 51
4.8.1.6 Enquire Slave Layer Status (XA) 52
4.8.1.7 Enquire Full Version Number (Xb) 53
4.8.1.8 Enquire Product Name (Xn) 54
4.9 Health 54
4.9.1 Commands and Responses 54
4.9.1.1 Enquire Temperature (X0) 54
4.10 Scheduler 55
4.10.1 Command and Responses 55
4.10.1.1 Add Scheduled Command (i0) 55
4.10.1.2 Delete All Scheduled Commands (i2) 56
4.10.1.3 Enquire Current Time (ix) - Harmonic Extension 56
4.11 Harmonic Extensions 57
4.11.1 Locks 57
4.11.1.1 Commands and Responses 58
4.11.1.1.1 Set Session Locks (hSL) 58
4.11.1.1.2 Enquire Session Locks (hSL) 58
4.11.1.1.3 Enquire Global Session Locks (hGSL) 59
4.11.1.1.4 Enable Oxtel Lock Tally (hOLT) 60
4.11.1.1.5 Enquire Oxtel Lock Tally (hOLT) 60
4.11.1.2 Unsolicited Tallies 61
4.11.1.2.1 Oxtel Lock Tally (hOLY) 61
4.11.2 External IO 61
4.11.2.1 Commands and Responses 62
4.11.2.1.1 Enquire Number of External IO Configurations (hXNC) 62
4.11.2.1.2 Enquire External IO Configuration (hXC) 63
4.11.2.1.3 Set External IO Source (hXS) 64
4.11.2.1.4 Get External IO Source (hXS) 65
4.11.2.1.5 Set External IO Dynamic Configuration (hXDC) 66
4.11.2.1.6 Get External IO Dynamic Configuration (hXDC) 67
4.11.2.1.7 Enable External IO Tally (hXIOT) 68
4.11.2.1.8 Enquire External IO Tally (hXIOT) 69
4.11.2.1.9 Enquire External IO Supported (hEXTIO) 69
4.11.2.1.10 Enquire External Inputs (hXIN) 70
4.11.2.1.11 Enquire Outputs (hOUT) 71
4.11.2.2 Unsolicited Tallies 72

4.11.2.2.1 External IO Config Changed Tally (hXCY) 72
4.11.2.2.2 External IO Source Changed Tally (hXSY) 72
4.11.2.2.3 External IO Dynamic Config Changed Tally (hXDCY) 73
5 Timing Model 74
5.1 Latencies 75
5.2 Oxtel Command Latency 75

Revision History

Version	Date	Author	Description
1.0	Oct-2013	Dan Montz	Initial draft incorporating all information from the ChannelPort Effects API specification.
1.1	Nov-2013	Dan Montz	Incorporating feedback from QA and engineering.
1.2	Nov-2013	Dan Montz	Documenting the Z4, Zf, and Zg commands
1.3	Nov-2013	Dan Montz	Fixing a header in the Example Command Sequences
1.4	Jan-2014	Dan Montz	Adding ChannelPort 8200 key/fill pair switching latency note
1.5	Feb-2014	Dan Montz	Spaces missing in the example UE mixer command. Also, changed "fill" to "full" in the UE ARC option definition.
1.6	Nov-2015	Dan Montz	The following changes were incorporated in this revision: 1. Renaming to Harmonic Oxtel Protocol Specification 2. Added Spectrum X information 3. Added a latencies table to define latencies for ChannelPort and Spectrum X 4. Added Set/Enquire Image Position (G) command 5. Added Enquire Full Version Number (Xb) command 6. Added Enquire Product Name (Xn) command 7. Added Enquire Latency (hLAT) command
1.7	Sept-2016	Dan Montz	The following changes were incorporated in this revision: 1. Added Enquire Number of Graphic Layers (hNGL) command 2. Added Spectrum X UHD latency values

Version	Date	Author	Description
1.8	July-2017	Dan Montz	Adding External IO commands for Spectrum X 1. Modified hLAT command a. removed External In (2) source b. added 2022-6 input (4) source c. added 2022-6 output (5) source 2. Added External IO commands 3. Updated Latency image for 2022-6

1 Overview

This specification defines the automation control protocol used by the Harmonic products.

Currently, this includes the following products:

- ChannelPort - CPT-8xxx
- Spectrum X - MIP-9xxx

These products support a subset of the Miranda Oxtel Automation Protocol as well as a few extended commands to control Harmonic-specific features.

2 Communication

The Oxtel protocol can be used to control Harmonic products using serial (RS-232 or RS-422) or TCP/IP. With RS-232 or RS-422 serial communication, the link between the automation system and Harmonic product is point-to-point with a single physical connection. With TCP/IP, multiple connections are supported. Note though while network connectivity is much faster than serial connections, timeliness cannot be guaranteed without the use of scheduled commands. It is recommended that automation integrators take this into account when integrating with the Harmonic products.

All three modes of communication specified in the Miranda Oxtel protocol are supported.

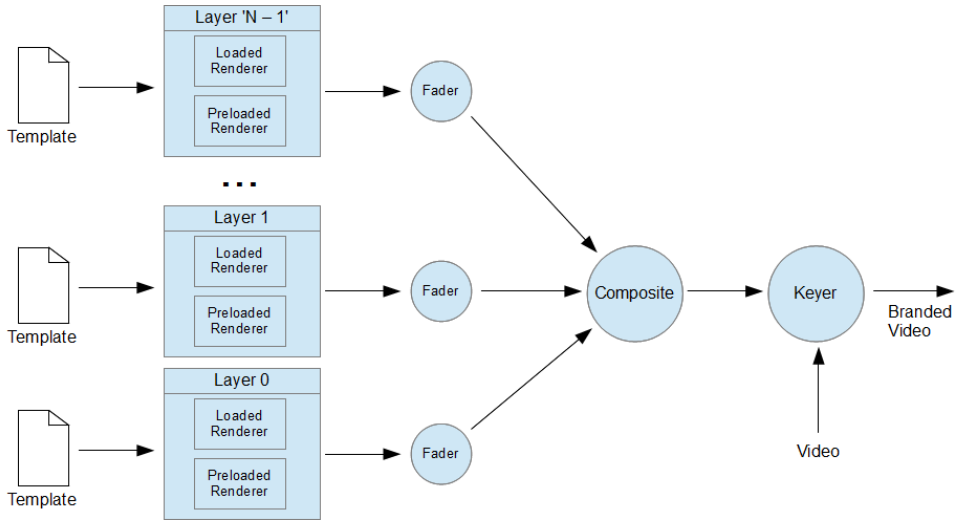
1. Serial Automation over RS-422. This is the STX, Command List, CRC sequence of bytes over RS-422.
2. TCP/IP Automation (Serial Format). This is the same as Serial Automation over RS-422 except over TCP instead of RS-422. The Miranda Oxtel specification specifies port 5006; however, Harmonic products may support a number of ports. Reference the Oxtel Port Mapping section for more details.
3. TCP/IP Automation (Network Format). This is the simple Command List, Command Terminator sequence of bytes over TCP. The Miranda Oxtel specification specifies port 5007; however, Harmonic products may support a number of ports. Reference the Oxtel Port Mapping section for more details.

The TCP modes of communication support multiple connections to the same channel. Multiple connections to the same channel are all treated equally so it is up to the clients to coordinate commands if necessary. TCP connections also have a timeout after some number of minutes of inactivity. This is to protect Harmonic products from misbehaving clients that open a connection and go away without closing the connection.

3 Harmonic Oxtel Model

The Miranda model uses keying layers to load multiple templates where each keying layer supports one template such as an image, animation or text strap. These layers are implemented as keyers and are numbered from 0 to N-1 where higher layer numbers layer on top of lower layer numbers. There is additional control per layer such as fade, masking, positioning, etc.

In Harmonic products, templates are loaded or preloaded into a layer. When a layer is loaded or preloaded with a template, a specialized renderer is launched to handle the template type. Many different renderer types are supported: .swf, .flv, .mp4, .png, .tga, .jpg, .tiff, .gif, .bmp, .wav, .aiff, .ekf, .and zip. Once the renderer generates a graphic image, a global alpha is applied by the Fader. All graphic images are then composited together (higher layers are composited over lower layers) into a single graphic image and then keyed over the background video.



3.1 Example Command Sequences

This section describes some example Oxtel command sequences. Other command sequences are possible.

3.1.1 Loading a Template and Viewing On-Air

1. Fade the layer down using the Fade Keyer (1) or Cut Keyer (3) commands. If the layer is not faded down, then the template will be visible immediately after the load completes.
2. Load the template on a layer using the Load Image (R0) command. Any previous template loaded on this layer is unloaded. By default, since the layer is faded down in the step above, the template does not start its animation until the fade angle is not 0.

3. Once the template is loaded, fade the layer up using the Fade Keyer (1) or Cut Keyer (3) commands. This will trigger the template to start playing its animation.

Example Oxtel Commands:

```
=> 30 0                          // Cut layer 0 down
=> R00720p60-pillar.swf          // Load the 720p60-pillar.swf template on layer 0

Wait for the load to complete.

=> 10 1 1e                       // Fade layer 0 up in 30 fields/frames
```

3.1.2 Loading a Template (using Preload) and Viewing On-Air

This command sequence allows automation to preload the template into an off-air location and then frame-accurately transition it to on-air.

1. Assuming that the layer already has a template loaded on this layer, preload the template on the layer using the Preload Image (R7) command.
2. Once the template is preloaded and it is time to take it to air, issue the Load Image (R0) command with the same parameters as specified in the R7 command above. This will immediately remove the previously loaded template on the layer and switch the preloaded template from the off-air location to on-air. Also, the preloaded template will start its animation at this time.

Example Oxtel Commands:

```
=> R70720p60-dve-squeezeback.swf  // Assume that the 720p60-pillar.swf template is currently
                                  // visible - preload the DVE squeezeback template.

Wait for the preload to complete.

=> R00720p60-dve-squeezeback.swf  // Load the DVE squeezeback template to take it on-air.
                                  // The template automatically starts to play since the layer
                                  // is faded up.
```

3.1.3 Using a Template with Updateable Text Fields

1. Cut or Fade the layer down and load the template using the Load Image (R0).
2. Update the text using the Update Text Field (Z0) command. This command can either replace or append text in the text field.
3. Render the new text using the Render Box (Z3) command, or set the "Render" flag in the preceding Update Text Field command.
4. Cut or Fade the layer up.

5. Additional text changes can be made with the layer faded up or down.

Example Oxtel Commands:

```
=> 30 0                                  // Cut layer 0 down
=> R00720p60-DynamicTextBanner.swf       // Load a template that contains updateable text
                                         // fields (example: 720p60-DynamicTextBanner.swf)

Wait for the load to complete.

This template contains 3 text fields (text field numbers 0-2)
=> Z00001Now Show                        // Update text field 0 (Render flag set) on
                                         // layer 0 to "Now Show"
=> Z00011Next Show                       // Update text field 1 (Render flag set) on
                                         // layer 0 to "Next Show"
=> Z00021Later Show                      // Update text field 2 (Render flag set) on
                                         // layer 0 to "Later Show"
=> 30 1                                  // Cut layer 0 up - the template will start playing
                                         // and become visible
```

3.1.4 Using a Template with a 3-Point Animation

1. Cut or Fade the layer down and load the template as indicated above. As mentioned above, by default, the template does not start its animation when it is loaded onto a layer that is faded down.
2. There are two ways to start the intro:
    a. Make the layer visible by fading it up using the Fade Keyer (1) or Cut Keyer (3) commands. As mentioned above, by default this triggers the template to start its animation because it is now visible.
    b. Alternately, the template can be authored so that a fader level does not automatically start the animation. In this case, the intro must be started explicitly with the Start Animation (S0) command.
3. The template is annotated to indicate the start and end of the middle loop section. Please refer to the Harmonic Graphics Template Authoring Guide for details on how to annotate 3-point animation templates in Flash Professional. After playing through the intro, the template plays the middle section repeatedly in a loop.
4. A Stop Animation (S1) with the ‘Complete Cycle’ flag enabled signals the template to enter the outro sequence. When the template reaches the end of the middle loop section, it continues on and plays the outro.

Example Oxtel Commands:

```
=> 30 0                                  // Cut layer 0 down
=> R00720p60-dve-squeezeback.swf         // Load a template that contains a 3-point animation
                                         // (example: 720p60-dve-squeezeback.swf)

Wait for the load to complete.

=> 30 1                                  // Cut layer 0 up to start its animation and make it visible

The template will play the intro and then loop on the middle section of the animation.

When ready to play the outro:
=> S100                                  // Stop animation with the Stop flag set to "Complete cycle,
                                         // then stop"
```

4 Oxtel Commands

Oxtel commands consist of one or more ASCII characters followed by zero or more command specific parameters. The parameters are also encoded in ASCII. All commands supported by Harmonic products are documented in this section.

String Parameters / Escaped Characters

Typically string parameters are variable length and are located at the end of the command packet. String parameters can contain 'escape' sequences to include special characters which cannot otherwise be transmitted by the Oxtel protocol. These special characters are handled using C-style escape sequences using a backslash character followed by two hexadecimal bytes to form the code for a single byte.

Character	Escape Sequence
Colon	\3A
Semi-colon	\3B
Vertical bar	\7C
Backslash	\5C

Example:

```
To send the string:           "\gfx.dir\t.swf"
The escaped string would be:  "\5Cgfx.dir\5Ct.swf"
```

4.1 Keyers

4.1.1 Commands and Responses

Fade Keyer (1)

This command fades the specified graphic keying layer up or down in the specified number of fields. When the keyer is faded up, its contents are keyed over the background video. When faded down, the background video is passed through unchanged.

If a prior fade is not finished when a reverse command is received, the fade transition will reverse direction and continue at the same rate.

The Direction parameter specifies the direction of the fade transition. If Toggle is specified, then the keyer will be faded down if it is currently faded up, and it will be faded up if currently faded down. If the fade transition is in progress, then the fade transition will be reversed.

The Rate parameter specifies the number of fields (interlaced) or frames (progressive) in the fade transition. When not specified, the value set using the Set Transition Duration (B) command will be used. The default value for the transition duration is 0.

Command Format:

Cmd	Param 1		Param 2		Param 3 (Optional)
1	%x Layer	space	%d Direction 0x0 = Down 0x1 = Up 0x2 = Toggle	space	%x Rate 0x0 - 0x3E7 (999d)

Example:

```
void sendCommand(int layer, bool, isUp, int rate)
{
    send(“1%x %d %x”, layer, (isUp) ? 1 : 0, rate); // rate specified
    send(“1%x %d”, layer, (isUp) ? 1 : 0);          // rate unspecified
}
```

Cut Keyer (3)

This command cuts the specified graphic keying layer up or down. When the keyer is cut up, its contents are keyed over the background video. When cut down, the background video is passed through unchanged.

This command is the same as the Fade Keyer (1) command executed with a Rate of 0.

Command Format:

Cmd	Param 1		Param 2
3	%x Layer	space	%d Direction 0x0 = Down 0x1 = Up 0x2 = Toggle

Example:

```
void sendCmd(int layer, bool isUp)
{
    send("3%x %d", layer, (isUp) ? 1 : 0);
}
```

Set Transition Duration (B)

This command sets the Keyer Fade duration for the specified layer. This value is used by the Fade Keyer (1) command when the Rate parameter is not specified.

The Rate parameter specifies the number of fields (interlaced) or frames (progressive).

Command Format:

Cmd	Param 1		Param 2		Param 3
B	%x Layer	space	%d Type 0x1 = Fade	space	%x Rate 0x0 - 0x3E7 (999d)

Example:

```
void sendCmd(int layer, int rate)
{
    send("B%x 1 %x", layer, rate);
}
```

Set Fader Angle (@)

This command sets the keyer fader for the specified layer to an absolute level.

Command Format:

Cmd	Param 1		Param 2		Param 3
@	%x Layer	space	%d Type 0x1 = Fade	space	%x Angle 0x0 - 0x200 (512d)

Example:

```
void sendCmd(int layer, int angle)
{
    send("@%x 1 %x", layer, angle);
}
```

4.1.2 Unsolicited Tallies

Keyer Position Tally (3)

This tally is enabled with the Enable Video Tallies (Y6) command. When enabled, it returns information about the state of the keyer positions as the keyers are cut or faded up/down using the Fade Keyer (1) and Cut Keyer (3) commands.

When this tally is first enabled, tallies for all layers are transmitted so that automation can record the initial keyer state. Subsequent tallies are transmitted only for those layers when their keyer state changes.

Tally Format:

Cmd	Param 1		Param 2
3	%1x Layer	space	%1d Direction 0x0 = Off (Down) 0x1 = On (Up) 0x2 = In Transition

Example:

```
When the tally is enabled:                  // Enable video tallies with Y61

The state of all layers are transmitted:
=> 30 0                                     // Layer 0 is cut down
=> 31 1                                     // Layer 1 is cut up
=> 32 2                                     // Layer 2 is transitioning
=> …                                        // The tally is transmitted for all supported layers

When Layer 0 is cut down:                   // Using “30 0”
The following tally will be transmitted:
=> 30 0                                     // Layer 0 is cut down

When Layer 0 is faded up:                   // Using “10 1”
The following tallies will be transmitted:
=> 30 2                                     // Layer 0 is transitioning
=> 30 1                                     // Layer 0 is up
```

4.2 Templates

Harmonic products support the following template types:

- .swf - Flash v10.1

- .flv - Flash Video
- .mp4
- .png
- .tga
- .jpg, .jpeg
- .tif, .tiff
- .gif
- .bmp
- .wav
- .aiff
- .ekf - External Key/Fill
- .zip - packaged templates with playlists.txt to control playback

In the future, Harmonic may introduce support for additional file formats. It is strongly recommended that automation systems always specify file extensions in the below commands.

4.2.1 Commands and Responses

Load Image (R0)

This command loads a template on the specified layer. Once the template has been completely loaded, the associated keying layer can be cut or faded up to reveal the graphics on-air.

If the keying layer was already faded up, then the template will be on-air once the template load completes.

If another template is loaded on the specified layer, then the current template will be unloaded before the new template is loaded.

If a "non-existent" filename is specified in the Filename parameter, the layer is unloaded; however, the preloaded template is not unloaded.

Load times are dependent on the template size and the load on the system. Automation should take care to make sure that adequate time is allowed for the template to completely load before taking the template on-air. It is recommended that automation monitor the Load Image Tally (Y9) in order to know when the load completes.

Command Format:

Cmd	Param 1	Param 2
R0	%1x Layer	%s Filename Relative to the configured graphics directory

Example:

```
void sendCmd(int layer, char *filename)
{
    send(“R0%1x%s”, layer, filename);        // Will load /fs0/gfx.dir/<filename>
}
```

Enquire Load Image (R0)

This command queries the template file that is currently loaded into the specified layer.

Command Format:

Cmd	Param 1
R0	%1x Layer

Example:

```
void sendCmd(int layer)
{
    send(“R0%1x”, layer);
}
```

Response Format:

Cmd	Param 1	Param 2
R0	%1x Layer	%s Filename

NOTE: When no template is loaded on the specified layer, the filename will be "> Empty <".

Preload Image (R7)

This command preloads a template on the specified layer. Once the preload is complete, the template can be swapped with the on-air template in a frame-accurate manner.

A preloaded template is loaded in an off-air location. When the preload is complete and a subsequent Load Image (R0) command is issued with the same layer and filename, the preloaded template is immediately swapped onto the on-air location. If a different valid filename is requested in the Load Image (R0) command, then the preloaded template is cleared.

Load times are dependent on the template size and the load on the system. Automation should take care to make sure that adequate time is allowed for the template to completely preload before commanding the Load Image (R0) command. It is recommended that automation monitor the Preload Image Tally (YA) in order to know when the preload completes.

Command Format:

Cmd	Param 1	Param 2
R7	%1x Layer	%s Filename

Example:

```
void sendCmd(int layer, char *filename)
{
    send(“R7%1x%s”, layer, filename);
}
```

Enquire Preload Image (R7)

This command queries the template file that is currently preloaded into the specified layer.

Command Format:

Cmd	Param 1
R7	%1x Layer

Example:

```
void sendCmd(int layer)
{
    send(“R7%1x”, layer);
}
```

Response Format:

Cmd	Param 1	Param 2
R7	%1x Layer	%s Filename

NOTE: When no template is loaded on the specified layer, the filename will be "> Empty <".

Erase Store (A)

This command unloads the template from the specified layer.

> NOTE: Any template preloaded on the specified layer will not be unloaded. This allows for the Erase Store command to be used in conjunction with preload functionality.

Command Format:

Cmd	Param 1
A	%x Layer

Example:

```
void sendCmd(int layer)
{
    send("A%x", layer);
}
```

Set Image Position (G)

This command sets the position of the loaded template relative to the origin. The origin (x=0, y=0) is defined is the upper left-hand corner of the screen. Positive values move the template right and down. Negative values move the template left and up.

Command Format:

Cmd	Param 1		Param 2		Param 3
G	%x Layer	space	%x X offset	space	%x Y offset

Example:

```
void sendCmd(int layer, int xOffset, int yOffset)
{
    send("G%x %x %x", layer, xOffset, yOffset);
}
```

Enquire Image Position (G)

This command enquires the position of the loaded template relative to the origin.

Command Format:

Cmd	Param 1
G	%x Layer

Example:

```
void sendCmd(int layer)
{
    send(“G%x %x %x”, layer);
}
```

Response Format:

Cmd	Param 1		Param 2		Param 3
G	%x Layer	space	%x X offset	space	%x Y offset

4.2.2 Unsolicited Tallies

Image Load Tally (Y9)

This tally is enabled with the Enable Video Tallies (Y6) command.

Once enabled, it returns information about template filenames as they are loaded/unloaded into the keying layers using the Load Image (R0) or Erase Store (A) commands.

When this tally is first enabled, tallies for all layers are transmitted so that automation can record the initial state. Subsequent tallies are transmitted only for those layers when their state changes.

Tally Format:

Cmd	Param 1	Param 2
Y9	%1x Layer	%s Filename

NOTE: When a layer is unloaded, the template filename will be "> Empty <".

Image Preload Tally (YA)

This tally is enabled with the Enable Video Tallies (Y6) command.

Once enabled, it returns information about template filenames as they are preloaded into the keying layers using the Preload Image (R7) command.

Tally Format:

Cmd	Param 1	Param 2
YA	%1x Layer	%s Filename

NOTE: When a layer is unloaded, the template filename will be "> Empty <".

4.3 Template File Management

4.3.1 Commands and Responses

Enquire File Info (R3)

This command queries information about the existence of the template on the system's file system. The folder location searched is defined by the Effects configuration section in System Manager.

Command Format:

Cmd	Param 1
R3	%s Filename

Example:

```
void sendCmd(char *filename)
{
    send(“R3%s”, filename);
}
```

Response Format:

Cmd	Param 1	Param 2
R3	%1x File exists 0x0 = No 0x1 = Yes	%s Filename

Query First File (R4)

This command queries the name of the first file within the specified folder name alias. In order to enumerate the files on the system, this command should be issued first followed by calls to Query Subsequent File (R5). The order of filenames returned is "oldest file first".

Command Format:

Cmd	Param 1
R4	%s Folder name alias **$VIDEO** = Template folder

Example:

```
void queryFirstFile
{
    send(“R4$VIDEO");
}
```

Response Format:

Cmd	Param 1	Param 2
R4	%1x End of directory reached 0x0 = No 0x1 = Yes	%s Filename

Query Subsequent File (R5)

This command is used in conjunction with Query First File (R4) to enumerate the names of all files within the media folder. It should be used following a single Query First File (R4) command using the same folder name alias. The order of filenames retrieved is "oldest file first".

Once all files have been enumerated, subsequent changes to file can be tracked using Media Tallies (YB).

Command Format:

Cmd	Param 1
R5	%s Folder name alias **$VIDEO** = Template folder

Example:

```
void querySubsequentFile
{
    send(“R5$VIDEO");
}
```

Response Format:

Cmd	Param 1	Param 2
R5	%1x End of directory reached 0x0 = No 0x1 = Yes	%s Filename

Enquire Extended File Information (R6)

This command queries for information about the specified template. In the response, only the "File exists" and "Filename" fields are valid. If no extension is specified, the command is treated as if the request was for the file with one of the supported extensions.

Command Format:

Cmd	Param 1
R6	%s Filename

Example:

```
void sendCmd(char *filename)
{
    send("R6%s", filename);
}
```

Response Format:

Format	Field	Description
%c%c	Cmd	'R6'
%1x	File exists	0x0 = No 0x1 = Yes
%03x	X-position	0x000 - not supported
%03x	Y-position	0x000 - not supported

Format	Field	Description
%03x	Width	0x000 - not supported
%03x	Height	0x000 - not supported
%03x	Clip	0x000 - not supported
%03x	Gain	0x000 - not supported
%03x	Transparency	0x000 - not supported
%02x	Image Type	0x00 - not supported
%04x	Frames	0x0000 - not supported
%1x	Animation Mode	0x0 - not supported
%02x	Load Time	0x00 - not supported
%1x	Associated Audio	0x0 - not supported
%s	Filename	Confirmation of the filename

Validate Template (RA)

This command validates the presence/absence of the specified template. Assets referenced by the template are not checked.

Command Format:

Cmd	Param 1
RA	%s Filename

Example:

```
void sendCmd(char *filename)
{
    send("RA%s", filename);
}
```

Response Format:

Format	Field	Description
%c%c	Cmd	'RA'

Format	Field	Description
%s|	Template name	Filename (followed by a pipe separator)
%1x	File exists	0x0 = No 0x1 = Yes
%04x	Missing assets	0x0000 - not supported

Enable Media Tallies (YB)

This command enables or disables media tallies for the connection on which the command was received.

Media tallies are used to track media management as files are added, deleted, or modified on the file system.

Command Format:

Cmd	Param 1
YB	%06x Template type (bitwise) 0x000001 = Images

Example:

```
void sendCmd()
{
    send("YB000001");
}
```

Enquire Media Tallies (YB)

This command queries the enable/disable state of the media tallies for the connection.

Command Format:

Cmd
YB

Example:

```
void sendCmd()
{
    send("YB");
}
```

Response Format:

Cmd	Param 1
YB	%06x Template type (bitwise) 0x000001 = Images

4.3.2 Unsolicited Tallies

Media Tallies (YB)

This tally is enabled with the Enable Media Tallies (YB) command.

Once enabled, it returns information about template filenames on the device's hard disk as they are added, deleted, and renamed.

Automation should use the R4 and R5 commands to enumerate through the template file list, and then use media tallies to track subsequent changes.

Tally Format:

Cmd	Param 1	Param 2	Param 3
YB	%0x6x Media Type 0x000001 = Images	%1x Action 0x0 = Deleted 0x1 = Added 0x2 = Modified	%s Filename

4.4 Animations

The animation control commands are used to control the playout of templates which have been loaded into a layer using the Load Image (R0) command. For Flash templates, they can be control 3-point animations as well as selecting specific animation frames. All other templates types only support the Start Animation (S0) and Stop Animation (S1) commands.

4.4.1 Commands and Responses

Start Animation (S0)

This command starts/resumes the animation on the specified layer.

Command Format:

Cmd	Param 1
S0	%1x Layer

Example:

```
void sendCmd(int layer)
{
    send("S0%1x", layer);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate startTemplateAnimation method. Override this function, if needed, to change the default behavior.

```
function startTemplateAnimation():Boolean;
```

Stop Animation (S1)

This command stops the animation on the specified layer.

If the Immediate flag is set, the animation halts immediately at the current frame. If the Complete Cycle flag is set, then the animation completes a cycle before stopping. The Complete Cycle flag should be used in conjunction with Flash 3-point animation templates.

Cmd	Param 1	Param 2
S1	%1x Layer	%1x Stop flag 0x0 = Complete cycle, then stop 0x1 = Immediate Stop

Example:

```
void sendCmd(int layer, int stopFlag)
{
    send("S1%1x%1x", layer, stopFlag);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate stopTemplateAnimation method. Override this function, if needed, to change the default behavior.

```
function stopTemplateAnimation(immediate:Boolean):Boolean;
where
  immediate = false = Complete cycle, then stop
  immediate = true  = Immediate stop
```

Select Animation Frame (S2)

This command sets the template animation frame on the specified layer.

This command is only applicable to templates authored in Flash. For Flash templates authored with the main Timeline, this command will automatically work. For templates authored completely in ActionScript, it is the developer's responsibility to implement this functionality by overriding the IHarmonicTemplate.gotoFrame method.

Command Format:

Cmd	Param 1	Param 2
S2	%1x Layer	%4x Frame

Example:

```
void sendCmd(int layer, int frame)
{
    send("S2%1x%04x", layer, frame);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate gotoFrame method. Override this function, if needed, to change the default behavior.

```
function gotoFrame(frame:int):Boolean;
```

Restart Animation (S4)

This command restarts the template animation from the beginning on the specified layer.

This command is only applicable to templates authored in Flash. For Flash templates authored with the main Timeline, this command will automatically work. For templates authored completely in ActionScript, it is the developer's responsibility to implement this functionality by overriding the IHarmonicTemplate. restartTemplateAnimation method.

Command Format:

Cmd	Param 1
S4	%1x Layer

Example:

```
void sendCmd(int layer)
{
    send("S4%1x", layer);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate restartTemplateAnimation method. Override this function, if needed, to change the default behavior.

```
function restartTemplateAnimation():Boolean;
```

Enable Play State Tally (YS)

This command enables or disables the template play state tally for the connection on which the command was received.

When first enabled, tallies for all layers are returned so that the client can record the initial state of each layer.

Command Format:

Cmd	Param 1
YS	%1x Tally enable 0x0 = No 0x1 = Yes

Example:

```
void sendCmd(int enable)
{
    send("YS%1x", enable);
}
```

Enquire Play State Tally (YS)

This command queries the enable/disable state of the Play State tally;

Command Format:

Cmd
YS

Example:

```
void sendCmd()
{
    send("YS");
}
```

Response Format:

Cmd	Param 1
YS	%1x Tally enable 0x0 = No 0x1 = Yes

4.4.2 Unsolicited Tallies

Play State Tally (YS) - Harmonic Extension

This tally is transmitted when the animation state of the template changes.

Tally Format:

Cmd	Param 1	Param 2
YS	%1x Layer	%1x State 0x0 = Stopped 0x1 = Playing

4.5 EasyText

4.5.1 Commands and Responses

Update Text Field (Z0)

This command updates the text in the specified text field.

If the 'Render' flag is set, then the specified text string will update on screen. If the 'Render' flag is not set, then changes will not appear on screen until the next call to Render Box (Z3) or Update Text Field (Z0) with the 'Render' flag set.

If the 'Append' flag is set, then the specified text string is appended to the existing text. This allows long test strings to be defined over several command packets.

Command Format:

Cmd	Param 1	Param 2	Param 3	Param 4
Z0	%1x Layer	%02x Text Field Num	%1x Flags (bitwise) 0x1 = Render 0x2 = Append	%s String

Example:

```
void sendCmd(int layer, int textFieldNum, int flags, char *str)
{
    send("Z0%1x%02x%1x%s", layer, textFieldNum, flags, str);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate updateTextField or appendTextField method. Override this function, if needed, to change the default behavior.

```
function updateTextField(fieldNum:int, text:String, render:Boolean):Boolean;
or
function appendTextField(fieldNum:int, text:String, render:Boolean):Boolean;
```

Render Box (Z3)

This command updates the specified text field.

Command Format:

Cmd	Param 1	Param 2
Z3	%1x Layer	%02x Text Field Num 0x00 - 0xFE = Specific text field 0xFF = Update all text fields

Example:

```
void sendCmd(int layer, int textFieldNum)
{
    send("Z0%1x%02x", layer, textFieldNum);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate.renderField or renderAllFields method. Override this function, if needed, to change the default behavior.

```
function renderField(fieldNum:int):Boolean;
or
function renderAllFields():Boolean;
```

Change Image (Z4)

This command changes the image associated with a text field to be replaced with another image on disk.

The new settings take effect when the Render Box (Z3) command is issued.

Command Format:

Cmd	Param 1	Param 2	Param 3
Z4	%1x Layer	%02x Text Field Num 0x00 - 0xFE	%s Image Filename

Example:

```
void sendCmd(int layer, int textFieldNum, char *str)
{
    send("Z4%1x%02x%s", layer, textFieldNum, str);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate updateImageField method. Override this function, if needed, to change the default behavior.

```
function updateImageField(fieldNum:int, imagePath:String):Boolean;
```

Stop Animation (Zf)

This command stops an animation from playing in the specified text field on the specified layer.

This command behaves similarly to the Stop Animation (S1) command; however, it includes a text field number parameter to allow finer control of the text field in the template.

Command Format:

Cmd	Param 1	Param 2	Param 3
Zf	%1x Layer	%02x Text Field Num 0x00 - 0xFE	%1x Stop flag 0x0 = Complete cycle 0x1 = Stop immediately

Example:

```
void sendCmd(int layer, int textFieldNum, int stopFlag)
{
    send("Zf%1x%02x%1x", layer, textFieldNum, stopFlag);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate stopFieldAnimation method. Override this function, if needed, to change the default behavior.

```
function stopFieldAnimation(fieldNum:int, immediate:Boolean):Boolean;
```

Pause/Restart Strap (Zg)

This command pauses or restarts the specified text field on the specified layer.

Command Format:

Cmd	Param 1	Param 2	Param 3
Zg	%1x Layer	%02x Text Field Num 0x00 - 0xFE	%1x Flag 0x0 = Restart 0x1 = Pause

Example:

```
void sendCmd(int layer, int textFieldNum, int flag)
{
    send("Zf%1x%02x%1x", layer, textFieldNum, flag);
}
```

For Flash Template development, this command will invoke the IHarmonicTemplate stopFieldAnimation method. Override this function, if needed, to change the default behavior.

```
function restartFieldAnimation(fieldNum:int, fromBeginning:Boolean):Boolean;
```

4.6 A/B Mixer

The following commands control the A/B mixer.

At initialization, the mixer will be cut over to the A input and the inputs will be configured as follows:

- A Input = 0 = Player/Player A
- B Input = 1 = External In/External In 1

The default Transition Type is V-Fade.

4.6.1 Commands and Responses

Cut to A (U0)

This command cuts the A/B mixer immediately to the A input of the mixer.

Command Format:

Cmd
U0

Example:

```
void sendCmd()
{
    send("U0");
}
```

Cut to B (U1)

This command cuts the A/B mixer immediately to the B input of the mixer.

Command Format:

Cmd
U1

Example:

```
void sendCmd()
{
    send("U1");
}
```

Fade to A (U2)

This command fades the A/B mixer to the A input of the mixer over a specified number of fields (interlaced) or frames (progressive).

Command Format:

Cmd	Param 1
U2	%03x Duration (fields or frames) 0x000 - 0x3E7 (999d)

Example:

```
void sendCmd(int duration)
{
    send("U2%03x", duration);
}
```

Fade to B (U3)

This command fades the A/B mixer to the B input of the mixer over a specified number of fields (interlaced) or frames (progressive).

Command Format:

Cmd	Param 1
U3	%03x Duration (fields or frames) 0x000 - 0x3E7 (999d)

Example:

```
void sendCmd(int duration)
{
    send("U3%03x", duration);
}
```

Cut AB (U4)

This command cuts the A/B mixer to the opposite input from the one currently visible.

Command Format:

Cmd
U4

Example:

```
void sendCmd()
{
    send("U4");
}
```

Fade AB (U5)

This command fades the A/B mixer to the opposite input from the one currently visible over the specified number of fields (interlaced) or frames (progressive).

The transition type used is defined by the Set Transition Type (U6) command.

Command Format:

Cmd	Param 1
U5	%03x Duration (fields or frames) 0x000 - 0x3E7 (999d)

Example:

```
void sendCmd(int duration)
{
    send("U5%03x", duration);
}
```

Set Transition Type (U6)

This command selects the A/B mixer transition type to be used in the U2, U3, U5, and UA commands.

Command Format:

Cmd	Param 1
U6	%02x Transition Type 0x01 = V-Fade 0x03 = X-Fade 0x05 = Cut

Example:

```
void sendCmd(int transitionType)
{
    send("U6%02x", transitionType);
}
```

Asymmetric V-Fade AB (U8)

This command instructs the A/B mixer to V-Fade from the current input to the other through the V-Fade color (black) over the specified number of fields (interlaced) or frames (progressive).

Command Format:

Cmd	Param 1	Param 2
U8	%03x Down Duration (fields or frames) 0x000 - 0x3E7 (999d)	%03x Up Duration (fields or frames) 0x000 - 0x3E7 (999d)

Example:

```
void sendCmd(int downDuration, int upDuration)
{
    send("U8%03x%03x", downDuration, upDuration);
}
```

Set Absolute Mix (U9)

This command sets the A/B mix position to the specified absolute value. The absolute mix value can range from 0 (A = 100%, B = 0%) to 512 (A = 0%, B = 100%).

Command Format:

Cmd	Param 1
U9	%3x Absolute Value 0x000 - 0x200 (512d)

Example:

```
void sendCmd(int absoluteValue)
{
    send("U9%3x", absoluteValue);
}
```

Asymmetric Transition (UA)

This command performs an A/B mixer transition to the destination specified.

The transition type used is defined by the Set Transition Type (U6) command. For Cut transitions, the duration parameters are ignored.

Command Format:

Cmd	Param 1	Param 2	Param 3
UA	%1x Destination 0x0 = A input 0x1 = B input	%03x Down Duration (fields/frames) 0x000 - 0x3E7 (999d)	%03x Up Duration (fields/frames) 0x000 - 0x3E7 (999d)

Example:

```
void sendCmd(int destination, int downDuration, int upDuration)
{
    send("UA%1x%03x%03x", destination, downDuration, upDuration);
}
```

Fade to Specified Position (UC)

This command transitions the A/B mixer to the specified position over the given number of fields (interlaced) or frames (progressive).

The position can be from 0x000 (A input) to 0x200 (B input) and all values in-between.

Command Format:

Cmd	Param 1	Param 2
UC	%03x Destination 0x000 = A input 0x200 = B input	%03x Duration (fields/frames) 0x000 - 0x3E7 (999d)

Example:

```
void sendCmd(int destination, int duration)
{
    send("UC%03x%03x", destination, duration);
}
```

Select Mixer Input (UE)

This command is used to route video sources into the A and B inputs of the A/B mixer.

The Source parameter depends on the system and its configuration.

The UE command accepts an optional third argument as a Harmonic extension. This argument overrides the aspect ratio conversion of the video (typically the down-conversion at the SD simulcast output). The ARC only applies when the fader is showing the given input. UE commands with no third argument reset the ARC back to default. This ARC mode overrides all other settings (including ARC inferred from AFD and specified on a player at clip attach time). There is currently no way of querying the ARC mode (the UE query result is not changed by this extension).

Command Format:

Cmd	Param 1		Param 2		Param 3 (Optional)
UE	%1x Mixer Input 0x0 = A input 0x1 = B input	space	%1x Video Source 0x0 = Player /Player A 0x1 = External In /External In 1 0x6 = Player B 0x7 = External In 2 0x8 = External In 3 0x9 = External In 4 0xA = External In 5	space (include with optional Param 3)	%1x ARC 0x0 = default (same as not specifying the third argument) 0x1 = anamorphic 0x2 = 14:9 crop 0x3 = full 0x4 = letter 0x5 = pillar

Cmd	Param 1		Param 2		Param 3 (Optional)
			0xB = External In 6 0xF = Color Generator		

Example:

```
void sendCmd(int mixerInput, int videoSource)
{
    send("UE %1x %1x", mixerInput, videoSource);
}

void sendCmd(int mixerInput, int videoSource, int arc)
{
    send("UE %1x %1x %1x", mixerInput, videoSource, arc);
}
```

Enquire Mixer Input (UE)

This command queries for the current video source for the specified mixer input.

Command Format:

Cmd	space	Param 1
UE		%1x Mixer Input 0x0 = A input 0x1 = B input

Example:

```
void sendCmd(int mixerInput)
{
    send("UE %1x", mixerInput);
}
```

Response Format:

Cmd	Param 1	Param 2
UE	%1x Mixer Input 0x0 = A input 0x1 = B input	%1x Video Source 0x0 = Player/Player A 0x1 = External In/External In 1 0x6 = Player B

Cmd	Param 1	Param 2
		0x7 = External In 2 0x8 = External In 3 0x9 = External In 4 0xA = External In 5 0xB = External In 6 0xF = Color Generator

Enquire Mix Mode (Ua)

This command returns the status of the A/B mixer parameters.

Command Format:

Cmd
Ua

Example:

```
void sendCmd()
{
    send("Ua");
}
```

Response Format:

Format	Field	Description
%c%c	Cmd	'Ua'
%02x	Transition Type	0x00 = Cut 0x01 = X-Fade 0x02 = V-Fade
%03x	A/B Mix Rate	0x000 - 0x3E7 (999d)
%03x	Wipe softness	0x000 - not supported
%03x	A/B Mix Angle	0x000 = Mixer at A input 0x200 = Mixer at B input
%06x	V-Fade Color	0x000000 - not supported

Color Generator Color (UZ) - Harmonic Extension

This command sets the color of the specified Color Generator unit to the specified RGB value.

Command Format:

Cmd	Param 1	Param 2	Param 3	Param 4
UZ	%1x Color Generator Unit 0x0 = unit 0	%02x Red	%02x Green	%02x Blue

Example:

```
void sendCmd(int unit, int red, int green, int blue)
{
    send("UZ%1x%02x%02x%02x", unit, red, green, blue);
}
```

Enquire Color Generator Color (UZ) - Harmonic Extension

This command queries the color generator color from the specified Color Generator unit.

Command Format:

Cmd	Param 1
UZ	%1x Color Generator Unit 0x0 = unit 0

Example:

```
void sendCmd(int unit)
{
    send("UZ%1x", unit);
}
```

Response Format:

Cmd	Param 1	Param 2	Param 3	Param 4
UZ	%1x Color Generator Unit 0x0 = unit 0	%02x Red	%02x Green	%02x Blue

Enable Video Tallies (Y6)

This command enables or disabled the Video Tally (Y6), Image Load Tally (Y9), Image Preload Tally (YA), and Keyer Position Tally (3) on the connection on which the command was received.

Command Format:

Cmd	Param 1
Y6	%1x Tally enable 0x0 = No 0x1 = Yes

Example:

```
void sendCmd(int enable)
{
    send("Y6%1x", enable);
}
```

Enquire Video Tallies (Y6)

This command queries the state of the Video Tallies set using the Enable Video Tallies (Y6) command.

Command Format:

Cmd
Y6

Example:

```
void sendCmd()
{
    send("Y6");
}
```

Response Format:

Cmd	Param 1
Y6	

Cmd	Param 1
	%1x Tally enable 0x0 = No 0x1 = Yes

4.6.2 Unsolicited Tallies

Video Tally (Y6)

This tally is enabled with the Enable Video Tallies (Y6) command

Once enabled, it returns information about the position of the A/B mixer as it changes.

On registration, a tally is sent so that automation can record the initial state.

Tally Format:

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5	Param 6	Param 7
Y6	%1x A/B mix at 0x0 = A input 0x1 = B input 0x2 = In between	%1x Layer 0 Keyer at 0x0 = Off 0x1 = On 0x2 = In between	%1x Layer 1 Keyer at 0x0 = Off 0x1 = On 0x2 = In between	%1x Unused	%1x Unused	%02x Unused	%02x Unused

4.7 Audio

4.7.1 Commands and Responses

Set Audio Profile (jAP) - Harmonic Extension

This command sets the audio profile for the specified audio source. Valid Audio Profile values are: 0 (use default from System Manager), 1-8. Audio Profiles are configured in System Manager.

Command:

Cmd	Param 1	Param 2
jAP		%1x Audio Profile (range 0-8)

Cmd	Param 1	Param 2
	%1x Audio Source 0x0 = Player/Player A 0x1 = External In/External In 1 0x6 = Player B 0x7 = External In 2 0x8 = External In 3 0x9 = External In 4 0xA = External In 5 0xB = External In 6 0xE = Secondary Audio (from graphic template)	

Example:

```
void sendCmd(int audioSource, int audioProfile)
{
    send("jAP%1x%1x", audioSource, audioProfile);
}
```

Enquire Audio Profile (jAP) - Harmonic Extension

This command queries the audio profile for the specified audio source.

Command:

Cmd	Param 1
jAP	%1x Audio Source 0x0 = Player/Player A 0x1 = External In/External In 1 0x6 = Player B 0x7 = External In 2 0x8 = External In 3 0x9 = External In 4 0xA = External In 5 0xB = External In 6 0xE = Secondary Audio (from graphic template)

Example:

```
void sendCmd(int audioSource)
{
    send("jAP%1x", audioSource);
```

```
}
```

Response:

Cmd	Param 1	Param 2
jAP	%1x Audio Source 0x0 = Player/Player A 0x1 = External In/External In 1 0x6 = Player B 0x7 = External In 2 0x8 = External In 3 0x9 = External In 4 0xA = External In 5 0xB = External In 6 0xE = Secondary Audio (from graphic template)	%1x Audio Profile

4.8 System and Status

4.8.1 Commands and Responses

Enquire Latency (hLAT) - Harmonic Extension

This command returns the latency, in reference frames, of the specified source to the SDI output. Reference the Timing Model section at the end of this document for more details.

Command:

Cmd	Param 1
hLAT	%1x - Source 0=MCS in to SDI out 1=Oxtel in to SDI out (includes MCS) 3=External Key/Fill In to SDI out 4=2022-6 input 5=2022-6 output

Example:

```
void sendCmd(int source)
{
```

```
    send("hLAT%1x", source);
}
```

Response:

Cmd	Param 1	Param 2
hLAT	%1x Source	%1x Number of reference frames from source to SDI out

Enquire Number of Graphic Layers (hNGL) - Harmonic Extension

This command returns the number of graphic layers licensed for the channel.

Command:

Cmd
hNGL

Example:

```
void sendCmd()
{
    send("hNGL");
}
```

Response:

Cmd	Param 1
hLAT	%d Number of Graphic Layers

Enquire System Status (M)

This command queries information about the system status.

It is of limited use, but can be used by automation to poll the system periodically to make sure that the Harmonic product is still alive.

Command Format:

Cmd
M

Example:

```
void sendCmd()
{
    send("M");
}
```

Response Format:

Format	Field	Description
%c	Cmd	'M'
%1d	System Mode	0 - not supported
%03x	Version High	0x000 - not supported
%03x	Version Low	0x000 - not supported
%1d	Video Standard	0 = PAL 1 = NTSC 2 = 1080i@59.94Hz 3 = 1080i@50Hz 4 = 720p@59.94Hz 5 = 720p@50Hz 6 = 1080p@59.94Hz 7 = 1080p@50Hz 8 = 2160p@59.94Hz 9 = 2160p@50Hz
%03x	Preview Source	0x000 - not supported
%03x	Fade Rate DSK1	0x000 - not supported
%03x	Fade Rate DSK2	0x000 - not supported
%03x	FTB Rate DSK1	0x000 - not supported
%03x	FTB Rate DSK2	0x000 - not supported
%1x	System not accessed	0x0 - not supported

Enquire Video Layer Status (N)

This command returns status information about the specified layer. Only the Intuition SD/HD[+] format is supported which requires an additional parameter to identify the layer status to query.

Command Format:

Cmd	Param 1
N	%x Layer

Example:

```
void sendCmd(int layer)
{
    send("N%x", layer);
}
```

Response Format:

Format	Field	Description
%c	Cmd	'N'
%03x	Layer Fader Angle	0x000 - 0x200 (512d)
%03x	Layer FTB Angle	0x000 - not supported
%03x	unused	0x000
%03x	unused	0x000
%02x	unused	0x00

Enquire Command Availability (X3)

This command is used to determine if a particular Oxtel command is currently available on the Harmonic product for automation to use.

NOTE: This is not completely implemented - some commands allow for 1 - 4 bytes in the Command prefix.

Command Format:

Cmd	Param 1	Param 2
X3	%c Command byte 1	%c Command byte 2

Example:

```
void sendCmd(char cmdByte1, char cmdByte2)
{
```

```
    send("X3%c%c", cmdByte1, cmdByte2);
}
```

Response Format:

Cmd	Param 1	Param 2	Param 3
X3	%c Command byte 1	%c Command byte 2	%1x Supported 0x0 = No 0x1 = Yes

Enquire Slave Layer Status (XA)

This is different than the Miranda implementation. It queries the current state of the keyer for each layer (0x1 = fader angle is 0x200, 0x0 = fader angle is not 0x200).

Command Format:

Cmd
XA

Example:

```
void sendCmd()
{
    send("XA");
}
```

Response Format:

Format	Field	Description
%c%c	Cmd	XA
%1x	Layer 0 State	0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 1 State	0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 2 State	0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 3 State	

Format	Field	Description
		0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 4 State	0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 5 State	0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 6 State	0x0 = Not fully visible 0x1 = Fully visible
%1x	Layer 7 State	0x0 = Not fully visible 0x1 = Fully visible
%08x	Unused	0x00000000

Enquire Full Version Number (Xb)

This command returns the full software version number.

Command Format:

Cmd
Xb

Example:

```
void sendCmd()
{
    send("Xb");
}
```

Response Format:

Cmd	Param 1
Xb	%s Software version

The software version response string is structured as a series of integers separated with '.'.

Major	Minor	Patch	Branch	Build Number
%d	%d	%d	%d	%d

Enquire Product Name (Xn)

This command returns the product name.

Command Format:

Cmd
Xn

Example:

```
void sendCmd()
{
    send("Xn");
}
```

Response Format:

Cmd	Param 1
Xb	%s Product Name

Valid product name responses will be:

- ChannelPort
- Spectrum X
- Electra X

4.9 Health

4.9.1 Commands and Responses

Enquire Temperature (X0)

This command is implemented but always returns 0 for the temperature.

Command Format:

Cmd
X0

Example:

```
void sendCmd(int layer)
{
    send("X0");
}
```

Response Format:

Cmd	Param 1
X0	%5f Temperature - will always be 0x00000

4.10 Scheduler

Scheduled commands allow automation commands to be executed at the specified time in hours, minutes, seconds, and reference frames.

They can be used to achieve reference frame-accurate command processing when automation cannot guarantee deterministic delivery of command.

Scheduled commands are normally issued a few seconds ahead of their scheduled time, but can be scheduled for up to 23 hours in the future. NOTE: If a command is scheduled for more than 23 hours and 59 minutes in the future, then the command will be issued immediately.

Scheduled commands are volatile, so a reboot of the product or the loss of the client connection will clear the scheduled list.

The maximum number of outstanding scheduled commands is limited to 256.

The timecode parameter is referenced to VITC.

4.10.1 Command and Responses

Add Scheduled Command (i0)

This command schedules an auomation command at the specified time in hours, minutes, seconds, and frames.

Scheduled commands specify a timecode value. This value should be specified as the time at which the command should be *recognized*, i.e. Toxt + Tmcs before the command's consequences will be seen on SDI output. Reference the Timing Model section for more details about the latencies.

Command Format:

Cmd	Param 1	Param 2	Param 3	Param 4		Param 5
i0	%02d Hours	%02d Minutes	%02d Seconds	%02d Frames	;	%s Automation command

Example:

```
void sendCmd(int hours, int minutes, int seconds, int frames, char *command)
{
    send("i0%02d%02d%02d%02d;%s", hours, minutes, seconds, frames, command);
}
```

Delete All Scheduled Commands (i2)

This command deletes all scheduled commands.

Command Format:

Cmd
i2

Example:

```
void sendCmd()
{
    send("i2");
}
```

Enquire Current Time (ix) - Harmonic Extension

This command queries the current time as referenced to VITC.

Command Format:

Cmd
ix

Example:

```
void sendCmd()
{
    send("ix");
}
```

Response Format:

Format	Field	Description
%c%c	Cmd	'ix'
%1d	Field Rate	2 = 50Hz 3 = 59.94Hz 4 = 60Hz
%2d	Hours	0-23 hours
%2d	Minutes	0-59 minutes
%2d	Seconds	0-59 seconds
%2d	Frames	

4.11 Harmonic Extensions

4.11.1 Locks

Starting in Spectrum 7.4, a series of requests were submitted from customers to be able to lock graphic layers and the mixer from being changed. This feature allows an operator to lock out automation if it goes awry.

Session Locks

A Session Lock is a mechanism that Oxtel clients can use to prevent specific items from being changed via the Oxtel protocol. Any Oxtel client can enable a Session Lock. When one client has a Session Lock on a defined item, a client that does not have the same Session Lock on the item will be prevented from making changes. All clients with a Session Lock on an item will be allowed to make changes.

A Session Lock is connection-based meaning that when the Oxtel connection is closed, the lock is released.

Items that can be locked with a Session Lock include:

- Mixer
- Graphic Layers 0-7

Global Session Lock

A Global Session Lock is the logical OR'ing of all individual client session locks. This type of lock cannot be set. It can only be queried to gain an understanding of what other clients are doing with regard to Session Locks. This value is included in the Oxtel Lock Tally so that clients can be informed of changes when they occur.

Commands and Responses

Set Session Locks (hSL)

This command sets the session lock for the specified item for this connection.

Command Format:

Cmd	Param 1
hSL	%08x Session Locks (bitwise) 0x00000001 - Mixer 0x00000100 - Layer 0 0x00000200 - Layer 1 0x00000400 - Layer 2 0x00000800 - Layer 3 0x00001000 - Layer 4 0x00002000 - Layer 5 0x00004000 - Layer 6 0x00008000 - Layer 7

Example:

```
void sendCmd(int32 locks)
{
    send("hSL%08x", locks);
}
```

Enquire Session Locks (hSL)

This command queries the connection's session locks.

Command Format:

Cmd
hSL

Example:

```
void sendCmd()
{
    send("hSL");
}
```

Response Format:

Cmd	Param 1
hSL	%08x Session Locks (bitwise) 0x00000001 - Mixer 0x00000100 - Layer 0 0x00000200 - Layer 1 0x00000400 - Layer 2 0x00000800 - Layer 3 0x00001000 - Layer 4 0x00002000 - Layer 5 0x00004000 - Layer 6 0x00008000 - Layer 7

Enquire Global Session Locks (hGSL)

This command queries the global session locks. Global session locks is the logical 'OR' of all connection session locks.

Command Format:

Cmd
hGSL

Example:

```
void sendCmd()
{
    send("hGSL");
}
```

Response Format:

Cmd	Param 1
hGSL	%08x Global Session Locks (bitwise) 0x00000001 - Mixer 0x00000100 - Layer 0

Cmd	Param 1
	0x00000200 - Layer 1 0x00000400 - Layer 2 0x00000800 - Layer 3 0x00001000 - Layer 4 0x00002000 - Layer 5 0x00004000 - Layer 6 0x00008000 - Layer 7

Enable Oxtel Lock Tally (hOLT)

This command enables or disables the Oxtel Lock Tally for the connection on which the command was received.

When enabled, the Oxtel Lock Tally will be transmitted so that the client can record the initial state.

Command Format:

Cmd	Param 1
hOLT	%1x Tally enable 0x0 = No 0x1 = Yes

Example

```
void sendCmd(int enable)
{
    send("hOLT%1x", enable);
}
```

Enquire Oxtel Lock Tally (hOLT)

Command Format:

Cmd
hOLT

Example

```
void sendCmd()
{
    send("hOLT");
}
```

Response Format:

Cmd	Param 1
hOLT	%1x Tally enabled 0x0 = No 0x1 = Yes

Unsolicited Tallies

Oxtel Lock Tally (hOLY)

This tally is transmitted when the Global Session Lock changes.

Tally Format:

Cmd	Param 1	Param 2
hOLY	%08x Global Session Locks (bitwise) 0x00000001 - Mixer 0x00000100 - Layer 0 0x00000200 - Layer 1 0x00000400 - Layer 2 0x00000800 - Layer 3 0x00001000 - Layer 4 0x00002000 - Layer 5 0x00004000 - Layer 6 0x00008000 - Layer 7	%08x Unused

4.11.2 External IO

Spectrum X v8.3 has added support for a new feature known as External IO. This feature will allow customers to use IP-based networks to send and received uncompressed serial digital interface (SDI) to/from Spectrum X using the SMPTE ST 2022-6:2012 standard.

For playout, 2022-6 streams can be configured as an external input to the Spectrum X MCS router much like SDI over BNC in today's implementation. This allows the 2022-6 input stream to be used as a source for the MCS mixer or DVE. Also, the primary and secondary outputs can be configured to output 2022-6 streams as well.

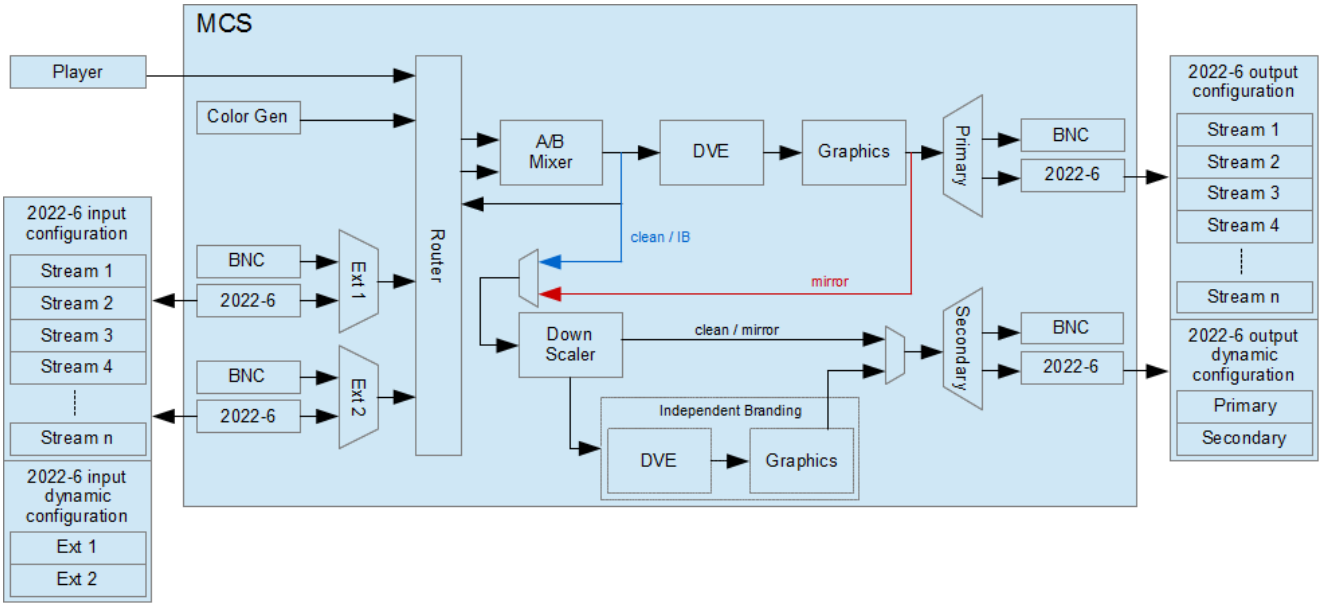
For ingest, 2022-6 can be configured to be the source for record just like the SDI over BNC is today. In this scenario the SDI over BNC input will be disabled since the 2022-6 functionality will be utilizing the same resources.

Configuration

The External IO feature is configured through System Manager. System Manager provides an interface to define:

- *Static Configuration* - the static configuration is a list of stream definitions. There is a list for inputs and a list for outputs. The input static configuration is shared by all inputs. The output static shared configuration is shared by all outputs. To switch between streams in the configuration, the Set External IO Source (hXS) command can be used.
- *Dynamic Configuration* - the dynamic configuration allows a customer to use Oxtel to configure the stream for a given input or output at run-time. This setting is volatile and will be lost after a reboot. Each input or output available for the channel can be dynamically configured using the Set External IO Dynamic Configuration (hXDC) command. Once configured, the dynamic configuration can be selected by specifying '0' for the "configId" parameter in the Set External IO Source (hXS) command.

The following picture is an illustration of how the 2022-6 feature and its configuration fits in with the current MCS functionality. For simplicity, only the Standard Channel 2 configuration is shown. It contains 2 external inputs and 2 outputs, primary and secondary.



Commands and Responses

Enquire Number of External IO Configurations (hXNC)

This command returns the number of external IO configurations for the specified type and direction.

Use this command to determine the number of configurations. To gain access to the complete configuration, use this command and the Enquire External IO Configuration (hXC) command.

Command:

Cmd	Param 1	Param 2
hXNC	%02x Type 0 = SDI 1 = 2022-6	%02x Direction 0 = input 1 = output

Example:

```
void sendCmd(int type, int direction)
{
    send("hXNC%02x%02x", type, direction)
}
```

Response Format:

Cmd	Param 1	Param 2	Param 3
hXNC	%02x Type 0 = SDI 1 = 2022-6	%02x Direction 0 = input 1 = output	%02x Number of configurations

Enquire External IO Configuration (hXC)

This command returns the external IO configuration for the specified type, direction, and index. The index is a value less than the number returned from the **hXNC** command above.

For SDI, the configuration information is internal and not user configurable.

For 2022-6, the configuration information is the same information configured via System Manager.

Command:

Cmd	Param 1	Param 2	Param 3
hXC	%02x Type 0 = SDI 1 = 2022-6	%02x Direction 0 = input 1 = output	%02x Index value < hXNC response

Example:

```
void sendCmd(int type, int direction, int index)
{
    send("hXNC%02x%02x%02x", type, direction, index)
}
```

Response Format:

When Type = SDI:

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5
hXC	%02x Type 0 = SDI 1 = 2022-6	%02x Direction 0 = input 1 = output	%02x Index	%02x Config Id	%s Name

When Type = 2022-6:

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5		Param 6		Param 7		Param 8
hXC	%02x Type 0 = SDI 1 = 2022-6	%02x Direction 0 = input 1 = output	%02x Index	%02x Config Id	%s Name	,	%s Local Interface	,	%s IP Address	,	%s Port

Set External IO Source (hXS)

This command sets the source for the specified external IO. The external IO is defined by the direction and IO id.

The **Config id** parameter is the static configuration ID specified in System Manager. '0' is always the Dynamic Config which is not specified in System Manager.

Command:

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5
hXS	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary 1 = Secondary	%02x Type 0 = SDI 1 = 2022-6	%02x Config id For SDI: N/A For 2022-6: 0 = Dynamic Config > 0 = Static Config	%02x Flags bits 0x01 = force restart of application if active

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5
		2 = Clean Primary 3 = Clean Secondary			

Example:

```
void sendCmd(int direction, int ioId, int type, int configId, int flags)
{
    send("hXNC%02x%02x%02x%02x%02x", direction, ioId, type, configId, flags);
}
```

Get External IO Source (hXS)

This command gets the source for the specified external IO. The external IO is defined by the direction and IO id.

Command:

Cmd	Param 1	Param 2
hXS	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary 1 = Secondary 2 = Clean Primary 3 = Clean Secondary

Example:

```
void sendCmd(int direction, int ioId)
{
    send("hXNC%02x%02x", direction, ioId);
}
```

Response Format:

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5
hXS	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary 1 = Secondary 2 = Clean Primary 3 = Clean Secondary	%02x Type 0 = SDI 1 = 2022-6	%02x Config id	%02x State always 0

Set External IO Dynamic Configuration (hXDC)

This command sets the dynamic configuration for the specified external IO. The external IO is defined by the direction and IO id.

For SDI, this command is not applicable.

For 2022-6:

Command:

Cmd	Param 1	Param 2	Param 3	Param 4		Param 5		Param 6		Param 7
hXDC	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External	%02x Type 0 = SDI 1 = 2022-6	%02x Flags bits 0x01 = force restart if active 0x02 = after updating config, switch to the dynamic config	,	%s Local Interface	,	%s IP address	,	%s Port

Cmd	Param 1	Param 2	Param 3	Param 4		Param 5		Param 6		Param 7
		In 6 Outputs 0 = Primary 1 = Secondary 2 = Clean Primary 3 = Clean Secondary								

Example:

```
void sendCmd(int direction, int ioId, int type, int flags, char *localif, char *ipaddr, char *port)
{
    send("hXNC%02x%02x%02x%02x,%s,%s,%s", direction, ioId, type, flags, localif, ipaddr, port);
}
```

Get External IO Dynamic Configuration (hXDC)

This command gets the dynamic configuration for the specified external IO. The external IO is defined by the direction and IO id.

For SDI, this command is not applicable.

For 2022-6:

Command:

Cmd	Param 1	Param 2	Param 3
hXDC	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary	%02x Type 1 = 2022-6

Cmd	Param 1	Param 2	Param 3
		1 = Secondary 2 = Clean Primary 3 = Clean Secondary	

Example:

```
void sendCmd(int direction, int ioId, int type)
{
    send("hXNC%02x%02x", direction, ioId, type);
}
```

Response Format:

Cmd	Param 1	Param 2	Param 3		Param 4		Param 5		Param 6
hXDC	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary 1 = Secondary 2 = Clean Primary 3 = Clean Secondary	%02x Type 1 = 2022-6	,	%s Local Interface	,	%s IP address	,	%s Port

Enable External IO Tally (hXIOT)

This command enables or disables the External IO Tally for the connection on which the command was received.

When enabled, the following tallies will be transmitted so that the client can record the initial state:

- External IO Source Changed Tally (hXSY)
- External IO Dynamic Config Changed Tally (hXDCY)

Command Format:

Cmd	Param 1
hXIOT	%1x Tally enable 0x0 = No 0x1 = Yes

Example

```
void sendCmd(int enable)
{
    send("hXIOT%1x", enable);
}
```

Enquire External IO Tally (hXIOT)

Command Format:

Cmd
hXIOT

Example

```
void sendCmd()
{
    send("hXIOT");
}
```

Response Format:

Cmd	Param 1
hXIOT	%1x Tally enabled 0x0 = No 0x1 = Yes

Enquire External IO Supported (hEXTIO)

This command returns whether or not the External IO feature is supported.

Command:

Cmd
hEXTIO

Example:

```
void sendCmd()
{
    send("hEXTIO");
}
```

Response:

Cmd	Param 1
hEXTIO	%1d supported 0 = Not supported 1 = Supported

Enquire External Inputs (hXIN)

This command returns a list of the external inputs supported by the channel. This command can be used to dynamically determine the external inputs that are currently supported for the channel configuration.

Command:

Cmd
hXIN

Example:

```
void sendCmd()
{
    send("hXIN");
}
```

Response:

Cmd	Param 1		Param 2		Param 3	
hXIN	%1x Number of external Inputs	;	%s Name	;	%d Video Source Id 1 = External In	Repeat Params 2 and 3 for all external inputs

Cmd	Param 1		Param 2		Param 3	
					/External In 1 7 = External In 2 8 = External In 3 9 = External In 4 10 = External In 5 11 = External In 6	

Example:

hXIN2;External 1;1;External 2;7:

2 Inputs:

- External 1 with id = 1
- External 2 with id = 7

NOTE: The Video Source Id is the same value as defined for the Select Mixer Input (UE) command.

Enquire Outputs (hOUT)

This command returns a list of the outputs supported for the channel. This command can be used to dynamically determine the outputs that are currently supported for the channel configuration.

Command:

Cmd
hOUT

Example:

```
void sendCmd()
{
    send("hOUT");
}
```

Response:

Cmd	Param 1		Param 2		Param 3	
hOUT	%1x Number of outputs	;	%s Name	;	%d Id 0 = Primary 1 = Secondary	Repeat Params 2 and 3 for all outputs

Cmd	Param 1		Param 2		Param 3	
					2 = Clean Primary 3 = Clean Secondary	

Example:

hOUT2;Primary;0;Secondary;1:

2 Outputs:

- Primary with id = 0
- External 2 with id = 7

Unsolicited Tallies

External IO Config Changed Tally (hXCY)

This tally is transmitted when the external IO static configuration changes.

Tally Format:

Cmd	Param 1
hXCY	%02x Type 1 = 2022-6

External IO Source Changed Tally (hXSY)

This tally is transmitted when the external IO source changes.

Tally Format:

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5
hXSY	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary	%02x Type 0 = SDI 1 = 2022-6	%02x Config id	%02x State always 0

Cmd	Param 1	Param 2	Param 3	Param 4	Param 5
		1 = Secondary 2 = Clean Primary 3 = Clean Secondary			

External IO Dynamic Config Changed Tally (hXDCY)

This tally is transmitted when the external IO dynamic configuration changes. This tally will have a different format depending on the type.

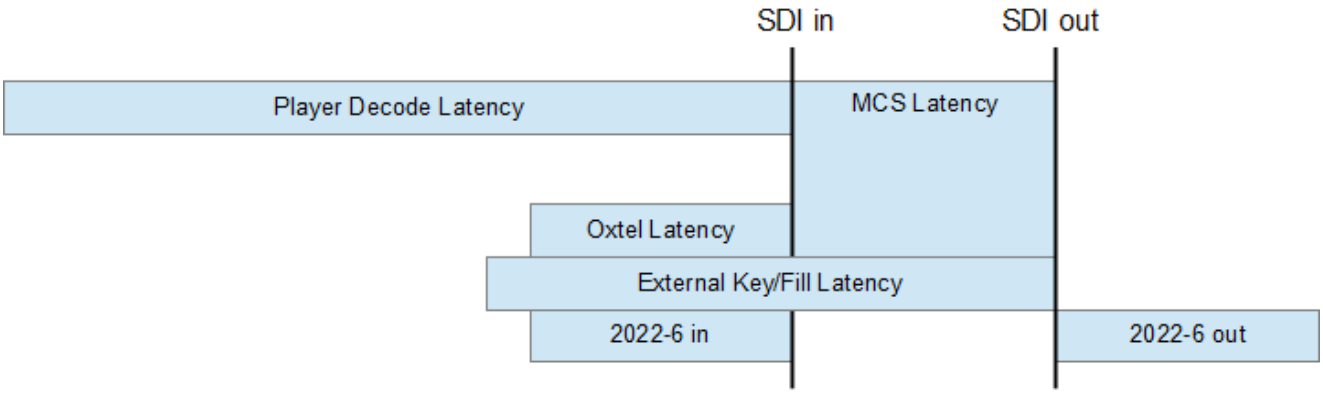
For 2022-6:

Tally Format:

Cmd	Param 1	Param 2	Param 3		Param 4		Param 5		Param 6
hXDCY	%02x Direction 0 = input 1 = output	%02x IO id Inputs 1 = External In 1 7 = External In 2 8 = External In 3 9 = External In 4 A = External In 5 B = External In 6 Outputs 0 = Primary 1 = Secondary 2 = Clean Primary 3 = Clean Secondary	%02x Type 1 = 2022-6	,	%s Local Interface	,	%s IP address	,	%s Port

5 Timing Model

The diagram below illustrates the system latencies that define which frame a command or status message will be associated with.



The traditional MediaPort decode latency is defined by the **Player Decode Latency**. This is the minimum number of fields from the time the director recognizes an automation command or player API command which affects the SDI output until the first picture which reflects that change has been decoded and converted to match the primary video output format and is available at the input of the MCS. As long as they do sufficiently far in advance, automation systems can specify to the director the field time at which they'd like player control to affect a frame that is at the input of the MCS and, because the director is aware of the **Player Decode Latency**, it will apply the changes at the appropriate time to make that happen.

The **MCS Latency** is the number of fields of latency through the master control switcher. The state of the AB mixer at the moment a set of input frames are at the inputs to the AB mixer will control how they are mixed even though it will be **MCS Latency** fields later before the effect of that mix can be observed on the SDI output.

For the Oxtel commands specified to have a fixed latency (see Oxtel Latency below) and affect the SDI output, the **Oxtel Latency** is the number of fields from the time the Oxtel command is recognized by the device until the frame which the command will take affect at the input to the MCS. For example, once an AB mixer command is recognized which alters the mix ratio it will be **Oxtel Latency** fields before that change will be applied to the MCS. Similarly, once a graphics command like Cut Up Keyer is recognized, it will be **Oxtel Latency** fields before the frame which that command will first be applied to will be at the input of the MCS (and **Oxtel Latency** + **MCS Latency** fields before the frame will be on the SDI output).

For Oxtel status commands the **Oxtel Latency**, also applies. That is, the status command doesn't return the current status at the input to the MCS (nor the SDI output), but rather the state as it will be at the input of the MCS **Oxtel Latency** fields in the future. This has the nice property that an Oxtel command which affects the state followed immediately by an Oxtel status command will return status that reflects the changes made by the previous command.

Oxtel Latency is measured relative to the time at which an Oxtel command is recognized by the device, and recognition occurs according to the following rules:

- For Oxtel commands received at any point in the first field of a reference frame the field during which it was received marks the start of the **Oxtel Latency** (as though it had been received right at top-of-reference frame).
- For Oxtel commands received at any point in the second field of a reference frame the field which represents the start of **Oxtel Latency** for that command is undefined.

The **External Key/Fill Latency** is the number of fields of latency from SDI in, through the graphics pipeline, and then through the MCS. The *control* of external key/fill via Oxtel has the same latency as any other graphics layer.

Note for ChannelPort 8200/Spectrum X modules only: Switching between Key/Fill input pair 1 and pair 2 requires two additional reference fields of latency to affect the switch.

5.1 Latencies

The following table defines the latencies for each supported product. All values are specified in fields.

Latency (fields)	ChannelPort (no Dolby)	ChannelPort (Dolby)	MIP-9xxx	MIP-9xxx (UHD)
MCS Latency	4	6	6	14
Oxtel Latency (to start of MCS)	6	6	6	6
External Key/Fill (end-to-end)	10	12	14	n/a
2022-6 in	n/a	n/a	6	n/a
2022-6 out	n/a	n/a	6	n/a

5.2 Oxtel Command Latency

Command	Prefix	Latency (Toxt) in Reference Fields
Keyers		
Fade Keyer	1	6
Cut Keyer	3	6
Set Transition Duration	B	Immediate

Command	Prefix	Latency (Toxt) in Reference Fields
Set Fader Angle	@	6
Templates		
Load Image	R0	6 (if preloaded) otherwise 6 from the time the template load completes
Enquire Load Image	R0	N/A
Preload Image	R7	N/A
Enquire Preload Image	R7	N/A
Erase Store	A	6
Set Image Position	G	6
Enquire Image Position	G	N/A
Template File Management		
Enquire File Info	R3	N/A
Query First File	R4	N/A
Query Subsequent File	R5	N/A
Enquire Extended File Information	R6	N/A
Validate Template	RA	N/A
Enable Media Tallies	YB	N/A
Enquire Media Tallies	YB	N/A
Animations		
Start Animation	S0	6
Stop Animation	S1	6
Select Animation Frame	S2	6
Restart Animation	S4	6
Enable Play State Tally	YS	N/A
Enquire Play State Tally	YS	N/A
EasyText		

Command	Prefix	Latency (Toxt) in Reference Fields
Update Text Field	Z0	6
Render Box	Z3	6
Change Image	Z4	6
Stop Animation	Zf	6
Pause/Restart Strap	Zg	6
A/B Mixer		
Cut To A	U0	6
Cut To B	U1	6
Fade to A	U2	6
Fade to B	U3	6
Cut AB	U4	6
Fade AB	U5	6
Set Transition Type	U6	Immediate
Asymmetric V-Fade AB	U8	6
Set Absolute Mix	U9	6
Asymmetric Transition	UA	6
Fade to Specified Position	UC	6
Select Mixer Input	UE	6
Enquire Mixer Input	UE	N/A
Enquire Mix Mode	Ua	N/A
Color Generator Color	UZ	6
Enquire Color Generator Color	UZ	N/A
Enable Video Tallies	Y6	N/A
Enquire Video Tallies	Y6	N/A
Audio		

Command	Prefix	Latency (Toxt) in Reference Fields
Set Audio Profile	jAP	6
Enquire Audio Profile	jAP	N/A
System and Status		
Enquire Latency	hLAT	N/A
Enquire Number of Graphic Layers	hNGL	N/A
Enquire System Status	M	N/A
Enquire Video Layer Status	N	N/A
Enquire Command Availability	X3	N/A
Enquire Slave Layer Status	XA	N/A
Enquire Full Version Number	Xb	N/A
Enquire Product Name	Xn	N/A
Health		
Enquire Temperature	X0	N/A
Scheduler		
Add Scheduled Command	i0	N/A
Delete All Scheduled Command	i2	N/A
Enquire Current Time	ix	N/A
Locks		
Set Session Locks	hSL	N/A
Enquire Session Locks	hSL	N/A
Enquire Global Session Locks	hGSL	N/A
Enable Oxtel Lock Tally	hOLT	N/A
Enquire Oxtel Lock Tally	hOLT	N/A
External IO		
Enquire Number of External IO Configurations	hXNC	N/A

Command	Prefix	Latency (Toxt) in Reference Fields
Enquire External IO Configuration	hXC	N/A
Set External IO Source	hXC	N/A
Get External IO Source	hXC	N/A
Set External IO Dynamic Configuration	hXDC	N/A
Get External IO Dynamic Configuration	hXDC	N/A
Enable External IO Tally	hXIOT	N/A
Enquire External IO Tally	hXIOT	N/A
Enquire External IO Supported	hEXTIO	N/A
Enquire External Inputs	hXIN	N/A
Enquire Outputs	hOUT	N/A

Oxtel Port Mapping

Oxtel uses TCP port 5006 for Serial Automation and TCP port 5007 for Network Automation. However, Harmonic products require more ports both because each product may have two channels, and because there can be a number of units accessed via the director they are connected to.

So we use the following port mapping:

Director TCP port	is forwarded to the product on	Protocol
9000	com0	Serial Automation
9001	com0	Serial Automation
9002	com1	Serial Automation
9003	com1	Serial Automation
9004	com2	Serial Automation
9005	com2	Serial Automation
9006	com3	Serial Automation
9007	com3	Serial Automation
9008	com4	Serial Automation
9009	com4	Serial Automation

9010	com5	Serial Automation
9011	com5	Serial Automation
9012	com6	Serial Automation
9013	com6	Serial Automation
9014	com7	Serial Automation
9015	com7	Serial Automation
9016	com8	Serial Automation
9017	com8	Serial Automation
9018	com9	Serial Automation
9019	com9	Serial Automation
9020	com10	Serial Automation
9021	com10	Serial Automation
9022	com11	Serial Automation
9023	com11	Serial Automation
9024	com12	Serial Automation
9025	com12	Serial Automation
9026	com13	Serial Automation
9027	com13	Serial Automation
9028	com14	Serial Automation
9029	com14	Serial Automation
9030	com15	Serial Automation
9031	com15	Serial Automation
9100	com0	Network Automation
9101	com0	Network Automation
9102	com1	Network Automation
9103	com1	Network Automation

9104	com2	Network Automation
9105	com2	Network Automation
9106	com3	Network Automation
9107	com3	Network Automation
9108	com4	Network Automation
9109	com4	Network Automation
9110	com5	Network Automation
9111	com5	Network Automation
9112	com6	Network Automation
9113	com6	Network Automation
9114	com7	Network Automation
9115	com7	Network Automation
9116	com8	Network Automation
9117	com8	Network Automation
9118	com9	Network Automation
9119	com9	Network Automation
9120	com10	Network Automation
9121	com10	Network Automation
9122	com11	Network Automation
9123	com11	Network Automation
9124	com12	Network Automation
9125	com12	Network Automation
9126	com13	Network Automation
9127	com13	Network Automation
9128	com14	Network Automation
9129	com14	Network Automation

9130	com15	Network Automation
9131	com15	Network Automation