FinishLynx Remote Control Specification

Revised 2022/05/18

Written by Kirk Sigel

179 Ward Hill Avenue
Haverhill, MA 01835

Purpose:

This specification is for remotely controlling FinishLynx by issuing commands through a serial port or TCP socket connection.

Overview:

An application that wishes to direct certain behavior or request certain actions within FinishLynx can do so by connecting to a predetermined serial port or TCP port that FinishLynx listens on. Once connected, an application can send data packets that specify an action FinishLynx is to perform. FinishLynx will send a successful reply to these data packets when the action is complete or will send an error reply if there is a problem.

Data Format Description:

A "request packet" is a data packet sent TO FinishLynx. A "reply packet" is a data packet sent FROM FinishLynx in response to a "request packet". Request packets and reply packets consist of one or more name/value pairs separated by semicolons and optionally quoted. The name and value in a name/value pair are separated by an equal sign. A packet is terminated by a carriage return and line feed. All characters in a packet are human readable ASCII with the exception of the carriage return and line feed. If a carriage return and line feed are sent without any other characters sent since the last request packet then the last request packet is executed again.

Note that all characters sent to FinishLynx are echoed back. Anything you send to FinishLynx will be sent back to you before the reply is sent. The reason that the protocol is human readable and that all characters are echoed back is so that the interface can be tested "by hand" using a simple serial port communications program or a simple Telnet program. If you set FinishLynx to listen on a certain TCP port, for instance, you can then Telnet to that port and enter commands from the keyboard and watch FinishLynx execute the commands. This allows you to experiment with how the protocol works before modifying any of your own software. Note that FinishLynx does not provide any line editing functions, meaning that, for instance, you cannot use the backspace key to correct errors in a hand typed request packet.

If a remote application does not want request packets echoed and/or reply packets sent then XOFF(13h)/XON(11h) characters can be used to control when FinishLynx sends characters. For example, beginning every request packet with an XOFF character will prevent FinishLynx from sending any characters.

Data Format Specification:
==========

```
<Request Packet>   := <Command Pair>[<Option Pair>]*<Newline>
<Repeat Packet>    := <Newline>
<Reply Packet>     := <Reply Pair>[<Option Pair>]*<Newline>
<Command Pair>     := Command=<Command Value>;
<Command Value>    := varies
<Reply Pair>       := Reply=<Reply Value>;
<Reply Value>      := Ok|Error|Unknown
<Option Pair>      := <Option Name>=<Option Value>;
<Option Name>      := varies
<Option Value>     := varies
<Newline>          := <0x0d><0x0a>
```

Note: Any part of the specification contained in brackets and followed by an asterisk can be present zero or more times. If a plus sign follows the brackets instead of an asterisk then that part of the specification must be present one or more times.

Event Commands:
==========

EventOpen

Request Packet: Command=EventOpen;[<Option Pair>]*<Newline>

Request Packet Options:

File=<filename>;
  <filename> should have a .evn extension.

Reply Packet: Reply=Ok;<Newline>

Example:

```
Send: Command=EventOpen;File=sample.evn;<Newline>
Recv: Reply=Ok;<Newline>
```

Start Commands:
==========

StartCreate

Request Packet: Command=StartCreate;[<Option Pair>]*<Newline>

Request Packet Options:

Time=<time>;
  <time> specifies the Time Of Day (TOD) of the start. Note that the precision of the start is dictated by the precision of the time given. If this option is omitted a manual start is created at the current time.

Offset=<time>;

  <time> specifies the start offset.

Key=<value>;
  <value> specifies the start ID value.

Reply Packet: Reply=Ok;<Newline>

Examples:

Send: Command=StartCreate;Time=12:10:00.0000;<Newline>
Recv: Reply=Ok;<Newline>
  A start is created with a time of ten minutes past noon.

Send: Command=StartCreate;Offset=5.0;<Newline>
Recv: Reply=Ok;<Newline>
  A manual start is created with a time of 5 seconds before current time.

Results Commands:

ResultsPrint

Request Packet: Command=ResultsPrint;<Newline>

Reply Packet: Reply=Ok;<Newline>

Example:

Send: Command=ResultsPrint;<Newline>
Recv: Reply=Ok;<Newline>
  The results are printed.

Image Commands:

The coordinate system used for images has its origin in the upper left corner of the window, with horizontal coordinates increasing towards the right and vertical coordinates increasing towards the bottom. The first pixel (in the upper left) has coordinate (0,0) and the last pixel (in the lower right) has coordinate (<image width>-1,<image height>-1).

Coordinates can be specified in four different ways:

1. If the coordinate is immediately followed by an "a" (for absolute) then the value is used directly. If the value is a negative number then it is taken from the end of the legal coordinate range rather than from the beginning. For example, a coordinate of -1a for a value that has a legal range of 0..99 will translate to a coordinate value of 99. A coordinate of -5a for a value that has a legal range of 0..500 will translate to a coordinate value of 496.

2. If the coordinate is immediately followed by a percent sign then the value used is the specified percentage of the maximum legal value. For example, a coordinate of 25% for a value that has a legal range of 0..200 will translate to a coordinate value of 50.

3. If the coordinate is immediately followed by an "r" (for relative) then the value is added to the current value for that coordinate. For example, if the cross hash is currently at coordinate 50 and a value of 20r is specified then the new cross hash position will be 70.

4. If the coordinate is immediately followed by an "R" then the behavior is the same as for "r" except that it is scaled by the current zoom factor.

ImageGetInfo

Request Packet: Command=ImageGetInfo;[<Option Pair>]*<Newline>

Request Packet Options:

Window=<window number>;
  <window number> specifies which camera window to request information from, numbering starts at 1. If this option is omitted then the first window's information is returned.

Options=[<bit mask>];
  <bit mask> specifies which reply packet option(s) are included. To include multiple options sum their values. To include all options leave the <bit mask> blank. If this option is omitted then the first 7 reply packet options are included (for backward compatibility).

Reply Packet: Reply=Ok;[<Option Pair>]*<Newline>

Reply Packet Options:

Orientation=<orientation>; # Option 1
  <orientation> can be either "Left" or "Right".

Zoom=<zoom>; # Option 2
  <zoom> is the current zoom level followed by a percent sign.

ImageSize=<width>,<height>; # Option 4
  <width> and <height> are the current width and height of the image given the current zoom level.

Origin=<x>,<y>; # Option 8
  <x> and <y> specify the coordinates of the upper left pixel visible in the window.

WindowSize=<width>,<height>; # Option 16
  <width> and <height> are the width and height of the window.

Hash=<x>,<y>; # Option 32
  <x> and <y> specify the coordinates of the hash line and cross hash.

Time=<time>; # Option 64
  <time> specifies the time at the hash line.

Rate=<rate>; # Option 128
  <rate> specifies the frame rate.

FirstTime=<time>; # Option 256

  <time> specifies the time of the first frame.

LastTime=<time>; # Option 512
  <time> specifies the time of the last frame.

Examples:

Send: Command=ImageGetInfo;Window=2;<Newline>
Recv: Reply=Ok;Orientation=Left;Zoom=100%;ImageSize=1116,1000;Origin=0,105;
WindowSize=440,354;Hash=84,518;Time=14:25:29.9060;<Newline>
  The current information for window 2 is returned. (Note that a newline was inserted before the "WindowSize" option for readability. This newline is not present in the actual reply packet.)

Send: Command=ImageGetInfo;Options=768;<Newline>
Recv: Reply=Ok;FirstTime=1:22.1020;LastTime=14:31.1426;<Newline>
  The first and last frame times are returned for the first window.

ImageDraw

Request Packet: Command=ImageDraw;[<Option Pair>]*<Newline>

Request Packet Options:

Window=<window number>;
  <window number> specifies which camera window to control, numbering starts at 1. If this option is omitted then the first window is controlled. If <window number> is 0 then all windows are controlled.

Zoom=<zoom>;
  <zoom> specifies what zoom level to display the image in. <zoom> can be "Enlarge" or "Reduce", which are equivalent to selecting the "+" and "-" buttons on the image toolbar. <zoom> can also be a number followed by a percent sign to directly specify a desired zoom level.

HashTime=[<time>][,<y>];
  <time> specifies where to place the hash line. <y> specifies where to place the cross hash.

HashMove=[<x>][,<y>];
  <x> specifies where to place the hash line. <y> specifies where to place the cross hash.

Scroll=[<x>][,<y>];
  <x> specifies where to scroll horizontally. <y> specifies where to scroll vertically.

Center=<n>;
  If <n> is 1 the hash line will be centered in the window. If <n> is 0 or if the option is omitted the hash line will not be centered.

Time=[<time>][,<n>];
  For IdentiLynx, <time> specifies the frame time and <n> a frame offset.

Reply Packet: Reply=Ok;<Newline>

Examples:

Send: Command=ImageDraw;Zoom=100%;HashTime=1:23.45,50%;Center=1;<Newline>
Recv: Reply=Ok<Newline>
  The zoom level is set to 100%, the hash line is moved to the frame closest to time 1:23.45, the cross hash is placed halfway down the image, and the image is centered on the hash line.

Send: Command=ImageDraw;HashMove=-1r;<Newline>
Recv: Reply=Ok;<Newline>
  The hash line is moved one frame to the left.

Send: Command=ImageDraw;Zoom=Enlarge;Center=1;<Newline>
Recv: Reply=Ok;<Newline>
  The zoom level is increased and the image is centered on the existing hash line location.

Send: Command=ImageDraw;HashMove=-1a,0a;<Newline>
Recv: Reply=Ok;<Newline>
  The hash line is moved to the last frame of the image. The cross hash is moved to the top of the image.

ImageExport and ImagePrint

Request Packet: Command=ImageExport;[<Option Pair>]*<Newline>
Request Packet: Command=ImagePrint;[<Option Pair>]*<Newline>

Request Packet Options:

Window=<window number>;
  <window number> specifies which camera window to export or print, numbering starts at 1. If this option is omitted then the first window is exported or printed.

File=<filename>;
  <filename> specifies the name of the file for ImageExport. If this option is omitted then the event's standard name will be used. If <filename> does not have an extension or has an unknown extension then the extension will be set to ".jpg" and a JPEG file will be written. If the extension is ".tga" then a TARGA file will be written. If the file already exists it will be overwritten. This option is ignored for the ImagePrint command.

Time=[<time>][,[<y>][,[<time2>][,[<y2>]]]];
  <time> and <y> specify which part of the image to export or print. If <time> and/or <y> are omitted then the current hash line value is used. If this option is omitted completely then the currently visible image is exported or printed. If <time2> and/or <y2> are included then they specify the right and/or bottom coordinates and <time> and/or <y> specify the left and/or top coordinates.
  For IdentiLynx, <time> specifies the frame time and <y> a frame offset.

Area=[<left>][,[<top>][,[<right>][,[<bottom>]]]];
  <left>, <top>, <right>, and <bottom> specify which part of the image to export or print. If the "Time" option is specified then <left> and <top> default to -200r and <right> and <bottom> default to 200r. If the "Time" option is not specified then all four values default to 0r.

Reply Packet: Reply=Ok;<Newline>

Examples:

Send: Command=ImageExport;File=image;<Newline>
Recv: Reply=Ok;<Newline>
  The currently visible image is written to the file "image.jpg".

Send: Command=ImagePrint;Time=;<Newline>
Recv: Reply=Ok;<Newline>
  The region of image starting 200 pixels left and above the current hash position and extending 200 pixels right and below the current hash position is printed.

Send: Command=ImageExport;Area=50r,50r,-50r,-50r;<Newline>
Recv: Reply=Ok;<Newline>
  The currently visible image (minus a 50 pixel band around the perimeter) is exported to a file whose name is the same name as the event but with the extension ".jpg".

ImageExportVideo

Request Packet: Command=ImageExportVideo;[<Option Pair>]*<Newline>

Request Packet Options:

Window=<window number>;
  <window number> specifies which camera window to export, numbering starts at 1. If this option is omitted then the first window is exported.

File=<filename>;
  <filename> specifies the name of the file for ImageExportVideo. If this option is omitted then the event's standard name will be used. If <filename> does not have an extension or has an unknown extension then the extension will be set to ".avi" and an AVI file will be written. If the file already exists it will be overwritten.

Time=[<time>][,[<offset>][,[<time2>][,[<offset2>]]]];
  <time> and <time2> specify the start and end frames of the exported video, respectively. If <time2> is omitted then the end frame is the same as the start frame. <offset> and <offset2> specify frame offsets for the start and end frames, respectively. If the "Time" option is omitted then the currently selected frames (or all frames) will be exported.

Area=[<left>][,[<top>][,[<right>][,[<bottom>]]]];
  <left>, <top>, <right>, and <bottom> specify which part of the image frame to export. If the "Area" option is omitted then the entire image frame is exported. Using the "Area" option will force FinishLynx to recompress the video, which can cause the export to take much longer.

Reply Packet: Reply=Ok;<Newline>

Example:

Send: Command=ImageExportVideo;Window=2;Time=1:00.00,-10,,10;<Newline>
Recv: Reply=Ok;<Newline>
  The video in window 2 at 1 minute (starting 10 frames before and going until 10 frames after) is exported.