Frame Streamer Express

A Streaming USB Laser Show DAC Controller

Document last modified May 29, 2010 (by Christopher Favreau – cfavreau@fab-favreau.com)

This document describes the Frame Streamer Express V1.X, a USB based streaming laser show DAC interface. It consists of several sections: Purpose, Features, "Streaming" Details, and PC Interface Details.

Purpose:

The purpose of the FSX1 is to provide a simple interface for the existing parallel port DAC interfaces in existence. Most modern computers include USB ports. These ports provide power and a fast data interface. This interface is designed to be simple and reflect the simplicity of the parallel port DACs themselves.

Features:

1. USB Interface (for data and power)
2. X and Y 8 Bit DAC Interface
2. Controls Parallel Port DAC (How many variations are there?)
3. Programmable Output Rate (adjustable up to 30 kpps)
4. Outputs:
    - X
    - Y
    - Blanking, 1 Bit Red, 1 Bit Green, 1 Bit Blue (blanking bit is currently inferred by RGB)
    - Support extra 4 bits of misc. output (not currently supported in protocol)
5. Onboard DC to DC converter that provides +/- 12VDC @ 90mA

"Streaming" Details

The streaming nature of this interface enables it to be made with less parts and yet still allow microcontroller maintained output rate. The microcontroller maintains a very small buffer of points and does not store a whole frames worth of data. The host computer sends a frame to the controller over the USB to serial port interface. The frame is then buffered in the USB to serial chip and in the microcontroller. The microcontroller then sends out each point in the frame at the programmed output rate. Another way of thinking about it is data

is poured into the USB port via a virtual serial port interface and is regulated on the output by the microcontroller. So the whole system acts as a funnel and flow regulator for the laser show frame points.

The data packet being sent to the microcontroller consists of 3 bytes. The first byte is the control byte. It contains the packet identifier. Based on the packet identifier the following 2 bytes mean different things. The following is table of the possible packet identifiers and following bytes:

Indent. Byte	Data Byte 1	Data Byte 2	Packet Name
0xFF	0xFF	0xFF	Reset Packet, this packet resets the streaming interface and resets the packet alignment (just in case a byte is missed). This packet also tells the micro to reset the X,Y ports to 0 and the aux. port to 0. Please note that every 0xFF after the Data Byte 2 is considered part of the synchronization. When a non 0xFF Byte is sent after a reset the micro considers this byte the first byte of a new packet. Upon being reset the micro will identify itself as a Frame Streamer Express device by sending the string "FSX1.X.X" where the X's are version numbers. Version 1 is = 1.0.0. Please note that the numbers can be multiple digits.
0x10	0x##	0x##	Output Rate Setting, this packet tells the micro to set the output rate to the 16 bit value using byte 1 and byte 2. The high byte is byte 1 and the low byte is byte 2. This value is the number of points per second to output. The usable range is 1000 to 30,000.
0x2#	0x##	0x##	Output Point, this packet is a laser show frame point. Byte 1 is the X DAC value. Byte 2 is the Y DAC value. The color and blanking information is contained in the lower nibble of the identifier Byte = R is bit 1, G is bit 2, B is bit 3, blanking is implied when R,G,G = 0, bit 0 is reserved and should be set to 0.
0xE0	0x00	0x00	Send the FSX Identifier string. See the reset packet for details.
0xF0	0x00	0x00	Reset the firmware to invoke the bootloader.
ETC…			Future Packet Types

*** Any unknown packets received will result in the hardware replying with a "?". This can also indicate if the hardware is out of sync with the PC.

PC Interface Details

The interface on the PC side consists of a module that converts a conventional frame of points to point packets and writes them to the virtual serial port interface. Since the interface on the microcontroller side is based on a parallel FIFO (a type of virtual serial port interface) the microcontroller takes care of the flow control. The microcontroller regulates the FLOW with the point output timer. The faster the point output, the faster the microcontroller consumes the data. A transmit buffer empty callback can be used for notification when the write FIFO is partially emptied. When this occurs more data can be written automatically to the port. This is the main streaming mechanism on the PC side. The complete interface can be contained in a class object that handles all of the interface details and simply needs to receive a complete frame. While this has been implemented on the Windows operating system, a LINUX port is possible.