Presentation Interfaces and Automation

A Case Study of Development of a System

For The Father's House
Vacaville, CA
By Bill Lyons

1. **The Problem**
    a. **Evolution of a system**
        i. When TFH first "graduated" from an overhead projector in 1999, it began with one portable projector connected to one laptop computer running PowerPoint in single screen mode. Since the church was renting a community center, it all had to be portable. Operators only had to learn to connect the two together via a relatively short VGA cable, aim and focus the projector, and run PowerPoint. A simple setup, but with many limitations. One of these was the need to "freeze" or "blank" the projector in order to make any change to the presentation without the audience seeing the computer desktop.
        ii. VCR was added, thus requiring an additional cable, composite video, and meant switching inputs of the projector to select the source – causing two blinks to be visible to the audience.
        iii. A new video driver was found, enabling the laptop to run in dual screen mode. This added the advantage that the operator could modify the PowerPoint presentation without the audience seeing it. It also added the complexity of teaching operators how to run in dual screen mode, which few had ever seen before. Only a few operators could be found that were up to the challenge of multitasking under the pressure of a live presentation, as well as the weekly setup and teardown of the equipment.
        iv. In 2003, a new sanctuary was under construction. A more professional presentation was a must. A dual projector system was desired, along with multiple sources, including a DVD player and a video camera. To enhance the presentation operation the software of choice was Media Shout. In order to enable a smooth transition between sources, an Analog Way Smart Fade seamless video scaler/switcher tied all the inputs together. Because it was to be a permanent installation, the operators no longer needed to know how to setup or tear down the equipment. However, this was more than offset by having so many inputs to control.
        v. In 2005, a second video camera, video mixer in a complete video control suite, satellite receiver, and 3rd projector were added. It was desired to have the 3rd projector operate independently from the other two, with the same seamless flexibility of the two side screens. The first thought was to use another identical Smart Fade unit. However, after learning that the Smart Fade was being discontinued, and being replaced with the Octo Fade, the Octo Fade was chosen.
    b. **The resulting challenge**
        i. The operator must control three separate pieces of software (Media Shout and remote control software for the two video switchers), and use a stack of infrared (IR) remote controls for various devices, while simultaneously communicating via

intercom with a video camera operator in a separate room, and the lighting and sound operators in the same booth, and still pay attention to the contents of the service and frequent interruptions from ushers, pastors, and others.

- ii. Certain times in the service require the operator to do multiple things flawlessly and in rapid succession.
    1. **Example 1:** during the announcements, when having Media Shout on two screens, and video camera on the third, on cue the operator must
        - a. select the Smart Fade software and switch to the DVD player (mouse),
        - b. select the Octo Fade software and switch to the DVD player (mouse),
        - c. and start the DVD playing with its IR remote,
        - d. while coordinating the timing with the lighting and sound operators,
        - e. all this in 1 or 2 seconds to make a professional transition.
    2. **Example 2:** during worship, with camera 2 selected and focused on a worship leader, and lyrics on the center screen, and an overlay of lyrics and camera on the side screens, the pastor suddenly takes the microphone and starts speaking, and then wants to read from a prearranged scripture. The operator must:
        - a. Direct the camera operator to switch to camera 1 and find the Pastor
        - b. Select the Smart Fade software and fade the input to the camera output
        - c. Select the Octo Fade software and fade the input to the camera output.
        - d. Select the Media Shout software and locate the scripture and select it..
        - e. Select the Smart Fade software and fade the input to Media Shout.
- iii. The need to simplify the operation becomes very clear.

c. The Goal

- i. Ability to preprogram a timed sequence of events for repeatable playback.
- ii. A single floating (always on top) control panel allowing immediate access to all the video inputs for both video switchers.
- iii. Visual indication of the current status of each video switcher.
- iv. Simplified operation, improving presentation quality by less experienced operators.

2. The Solution

a. Girder. Please see the Girder documentation for details and instructions. This summary is only to provide the reader with enough information to understand how the pieces of the solution work together.

- **i. What is Girder?** Well, the short answer is "glue." It takes inputs, what it calls "Events", from various sources, depending on the plugins, can make decisions based on those inputs, and sends outputs, again via plugins. That may sound trivial, but when you have completely unrelated things that you want to talk to each other, it is VERY cool. I think the name "Girder" is symbolic - the internal structure that holds a building up. It's not really pretty, and is hidden when complete, but is very useful and efficient.

ii. **Commands.** A Girder command performs some sort of action. There are many different types of actions to choose from, including operating system functions, sending messages to applications, variable manipulation and decision making scripts with a language called Lua, actions with various Girder plugins, and many more.

iii. **Events.** A Girder event is a trigger causing a command or group of commands to take action. It is usually caused by some sort of input, but can also be triggered from a Lua script.

b. Media Shout

i. **Media Shout and the outside world.** Media Shout 2.5 itself does not have any direct way to send commands or raise events in other programs. At the time of this writing, version 3 is expected to have the same limitation.

ii. **Media Shout and Flash.** One workaround is to use Macromedia Flash. A Flash programmer could perform operating system commands including some of the functions of this solution. The Flash file would then be played from Media Shout. However, Flash is a language I have never had the time to learn. Even if I did, I do not believe it would have the simplicity, versatility and expandability of this solution. That is a subject for others to explore.

iii. **Media Shout and MIDI.** Since MIDI is normally used to control audio devices, synthesizers and the like, Media Shout plays MIDI files just like it plays .wav files or .mp3 files. This gives us a useful little loophole.

c. MIDI

i. **Background.** MIDI is an acronym for Musical Instrument Digital Interface. Before beginning this project, that was about as much as I knew. The following is all based on my limited research to accomplish this project, and is the sum total of what I know about MIDI now. If you are a MIDI expert, and I have some of it wrong, please forgive me.

ii. **MIDI is a “language” or “protocol”** which consists of codes symbolizing tracks, channels, ports, timings and events. These pieces work together to produce music in synthesizers. This allows the computer to reproduce the sounds comprising the music through the instructions, rather than recording the actual waveform digitally, saving huge amounts of disk space. This is all very nice, and way over my head. But fortunately it is all irrelevant to this project.

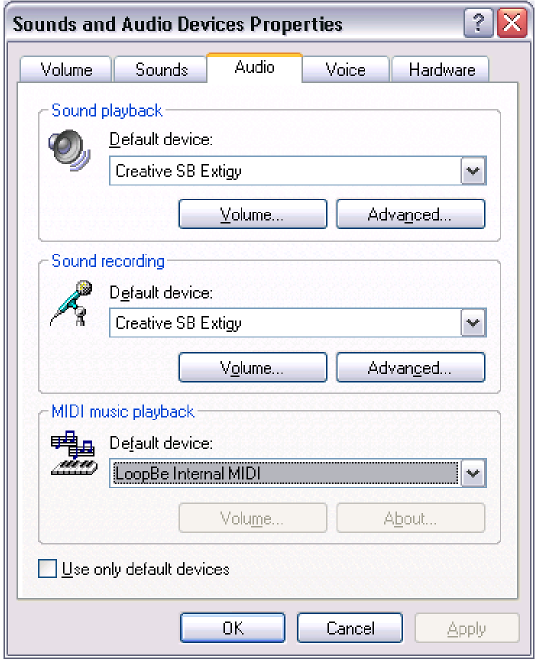
iii. **SysEx.** There is one other type of command included in the MIDI standard: system exclusive commands, or “SysEx.” A SysEx is used to initialize, configure, or setup a particular device. A SysEx command or event is a series of hexadecimal bytes, beginning with an F0 start byte, and an F7 as a terminator byte. MIDI files (.mid) are created and edited by software known as a “sequencer.” I found it a bit of a challenge to find a MIDI sequencer that allowed direct control of the SysEx command contents. Some had no ability at all; others only import predefined SysEx commands for specific manufacturer’s devices.

One program that did what I wanted I found at http://www.greatfreeware.com/Multimedia_and_Graphics/MIDI_Players_and_Editors/701.html. Their website at www.winjammer.com does not even seem to refer to the software, so I don't know if is even for sale anymore. But the demo version is capable of creating the MIDI files with the needed SysEx commands, so I haven't looked further. The actual contents of the SysEx commands do not really matter to Girder, so I simply created a bunch of MIDI files, each with a different SysEx command – a single hexadecimal byte – from 00 through 4F. The raw file I simply named "event xx.mid" where "xx" is the hex byte. Then, as I assigned them to the various Girder commands, I renamed them to represent the function they perform, so it is simple to insert the appropriate command in Media Shout. I have included a folder with the raw MIDI files which you can use as a starting library.

iv. **MIDI playback device.** In order to play a MIDI file, it must be sent to some sort of device. Normally this is the internal sound card. However, that won't get the command into Girder. Girder itself is not a Windows MIDI driver, so we need an intermediate interface to translate it: a loopback program.

v. **MIDI loopback program LoopBe.** LoopBe takes a MIDI input from one program, in this case Windows, and sends it to another program, in this case the Girder plugin.

1. Download LoopBe from one of many websites. The first one that was listed in my Google search was at http://www.topshareware.com/LoopBe1-download-17853.htm.
2. Install it.
3. Launch it.
4. Open the Sounds and Audio Devices Properties dialog box from your Windows Control Panel, and set the MIDI music playback default device to LoopBe. (This is from Windows XP. Other versions may be different.)



Sounds and Audio Devices Properties

Sound playback

Default device: Creative SB Extigy

Sound recording

Default device: Creative SB Extigy

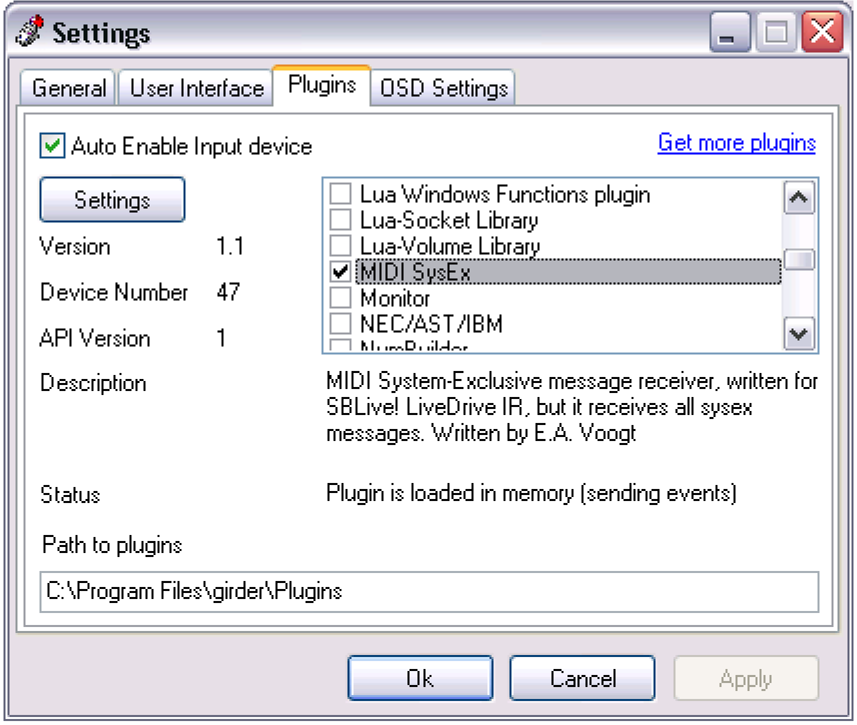
MIDI music playback

Default device: LoopBe Internal MIDI

□ Use only default devices

vi. **MIDI plugin for Girder.** The MIDI SysEx plugin is actually part of the Creative Remotes SBLive LiveDrive plugin (sblive.exe) which you can download from the Promixis Girder Plugin download page at http://www.promixis.com/downloads.php?mode=list&prodName=Girder&lucCode=800.

1. Download, and install it.
2. Launch Girder (if already running, exit and restart Girder to recognize the new plugin).
3. Open the Girder Settings dialog box (File/Settings…)
4. Select the Plugins tab.



☑ Auto Enable Input device

☐ Lua Windows Functions plugin
☐ Lua-Socket Library
☐ Lua-Volume Library
☑ MIDI SysEx
☐ Monitor
☐ NEC/AST/IBM
☐ NumBuilder

Version 1.1

Device Number 47

API Version 1

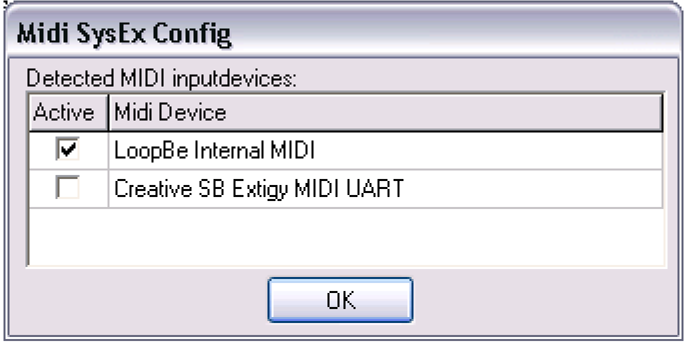
Description MIDI System-Exclusive message receiver, written for SBLive! LiveDrive IR, but it receives all sysex messages. Written by E.A. Voogt

Status Plugin is loaded in memory (sending events)

Path to plugins

C:\Program Files\girder\Plugins

5. Check the box selecting the MIDI SysEx plugin.
6. Click Apply.
7. Click on the MIDI SysEx plugin item again. The Settings button should now be enabled.
8. Click the Settings button.



Detected MIDI inputdevices:

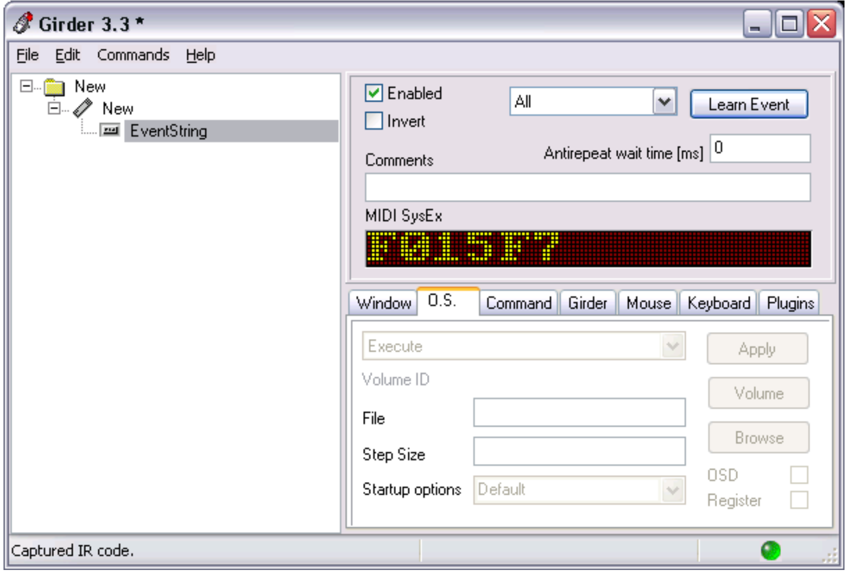
Active	Midi Device
☑	LoopBe Internal MIDI
☐	Creative SB Extigy MIDI UART

9. Check the box activating the LoopBe internal MIDI device.
10. Click OK.

11. Click OK on Settings.

vii. Learn a MIDI event in Girder

1. Create a command in Girder. Any event can trigger any command. To make this a simple and clear example, just make the command open a Notepad window:
    - a. Click Edit/Add Command
    - b. Click the OS tab
    - c. Select the Execute command
    - d. Type Notepad in the File textbox
    - e. Click Apply
    - f. Test it to be sure it is working so far
        - i. Right-click the command
        - ii. Select Test command
        - iii. A new Notepad window should open
2. Insert one of the MIDI SysEx files into a Media Shout script.
3. In Girder, with the test command highlighted, click the "Learn Event" button (be sure "All" is selected in the dropdown box adjacent to this button).
4. Fire the MIDI cue in Media Shout.
5. Girder should show the MIDI SysEx command in the LED display in Girder as shown below. In this case the sample was "event 15.mid"



Girder 3.3 *

☑ Enabled

☐ Invert

All

Antirepeat wait time [ms] 0

MIDI SysEx

F015F7

Execute

Startup options Default

☐ OSD

☐ Register

Captured IR code.

6. The next time the MIDI cue in Media Shout is fired, a Notepad window should open.
7. You now have Media Shout talking to the outside world!

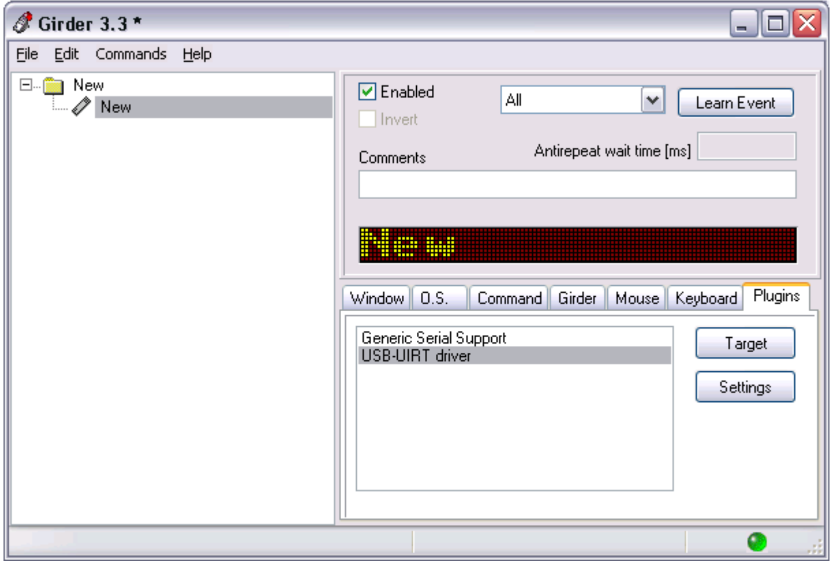
d. Infrared (IR) devices

i. How they work. Wireless remote control devices most commonly use some form of infrared (IR) emitter and detector. They transmit a series of pulses in a specific pattern. Different brands use different coding schemes and frequencies. To facilitate a system like we need here, we are basically building the ultimate "universal remote."

ii. **Devices.** First, you need an IR transceiver. There are a few on the market. The one I like is the USB-UIRT from http://www.usbuirt.com. It is reliable, cost effective, good compatibility, easy to use, and has good support. This plugs into the USB port on a computer. It has a driver that must be installed, and of course you need the Girder plugin.

iii. **Making it work in Girder.** The documentation that comes with the device details the precise method for the particular device. But they all share some basics:

1. Enable the Girder plugin by checking its box in the Plugins tab in the Settings dialog box and clicking Apply.
2. Create a command.
3. Select the Plugins tab and select the IR device plugin.



Girder 3.3 *

☑ Enabled

□ Invert

All

Comments

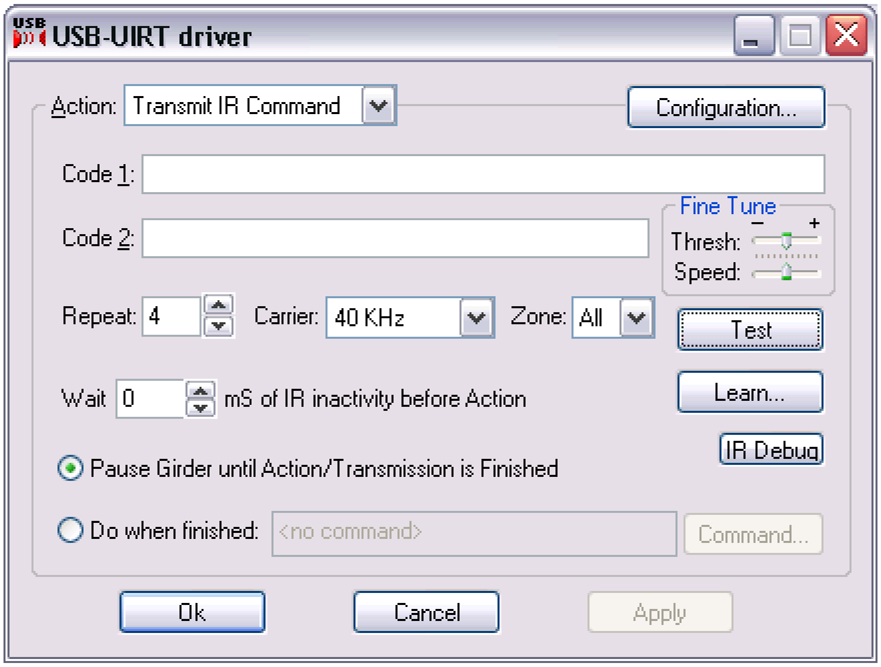
Antirepeat wait time [ms]

Generic Serial Support

USB-UIRT driver

4. Click the Settings button and learn the code. The dialog box for each brand of IR device will be different. The USB-UIRT dialog is shown here. However, each of them should have something similar to the following

functions:



Action: Transmit IR Command

Code 1:

Code 2:

Repeat: 4 Carrier: 40 KHz Zone: All

Wait 0 mS of IR inactivity before Action

● Pause Girder until Action/Transmission is Finished

○ Do when finished: <no command>

   a. Click Learn
   b. Place the remote control that came with the device you are trying to control (DVD player remote control for example) in close proximity (2 to 4 inches) to the IR transceiver device (USB-UIRT in this example), aiming it at the device sensor. I found that putting it at about a 60 degree angle increased reliability for this process.
   c. Press the button on the remote control for the function you are trying to learn, like “Play” or “Power” for example. If you have the units aimed properly, you should see a series of numbers displayed in the dialog box. This can take a bit of practice at first. Try holding down the button and moving the remotes around till you get a reliable stream of numbers. The device should automatically detect and adjust to the proper frequency
   d. Test the code. Most units will have a button on the dialog box enabling you to test the learned code. Aim the IR device at the unit to be controlled (DVD player, for example), and click the “Test” button. If it was learned correctly, the device should perform the desired action. If it doesn’t, repeat (a) through (d) until it works.
   e. Close the dialog box, saving the code – usually by clicking “OK”.

5. Now, test the Girder command.
   a. Select the Girder command.
   b. Aim the IR device at the unit to be controlled (DVD player, for example).
   c. Select “Command/Test Command” from the menu. If it was learned correctly, the device should perform the desired action. If it doesn’t, repeat step 4 above.

6. You now have a working Girder command. All you need is an event attached to it to trigger it. You can simply attach the MIDI event learned in part c.vii above, and/or any other event you desire.

7. Repeat this sequence for each command you wish to learn for each device you wish to control.

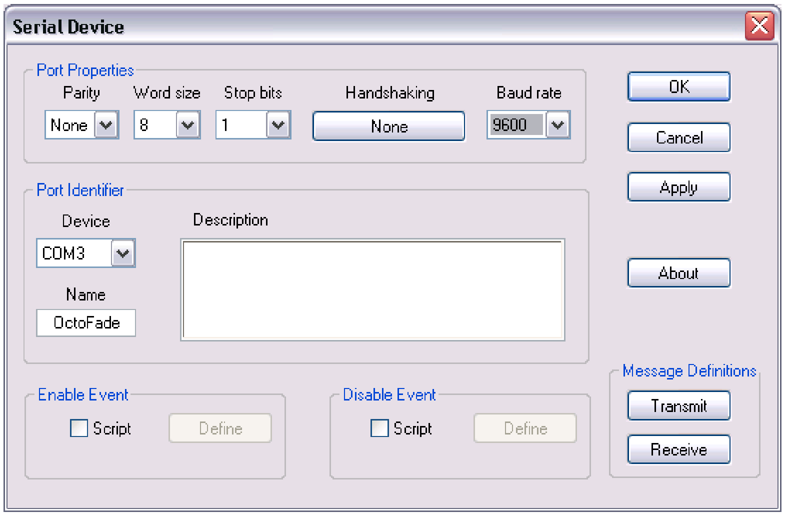
iv. Distance considerations. The distance limitations of USB devices prevent placing the IR device a great distance (more than 10 to 15 feet) from the computer. If you want to control a device a greater distance away, like, for example, the front of the church, try using an external IR emitter (the USB-UIRT and Tira both have external emitter jacks). Online discussions (http://166.70.183.44/phpBB2/viewtopic.php?t=96) indicate that these may be extended a considerable distance, although exactly how far isn't really clear. Distribution hubs are also available (http://www.smarthome.com/8191A.html). (This site also has IR emitters and lots of other devices that may work with Girder.) In the coming weeks, I will be experimenting with some of these, and will update this document at that time.

e. Serial plugin for Girder. The generic Serial Port device driver is available for download from the Girder Plugins download page at http://promixis.com/downloads.php?mode=list&prodName=Girder&lucCode=800. Install it per the instructions.

i. Setting up.

1. Enable the "Generic Serial Support" plugin via the Settings dialog box, plugins tab, and click Apply.
2. Select "Generic Serial Support" again, and click the Settings button.
3. You will need to setup each serial device you wish to control.

a. Click New.



Serial Device

Port Properties

Parity: None

Word size: 8

Stop bits: 1

Handshaking: None

Baud rate: 9600

Port Identifier

Device: COM3

Description:

Name: OctoFade

Enable Event: □ Script

Disable Event: □ Script

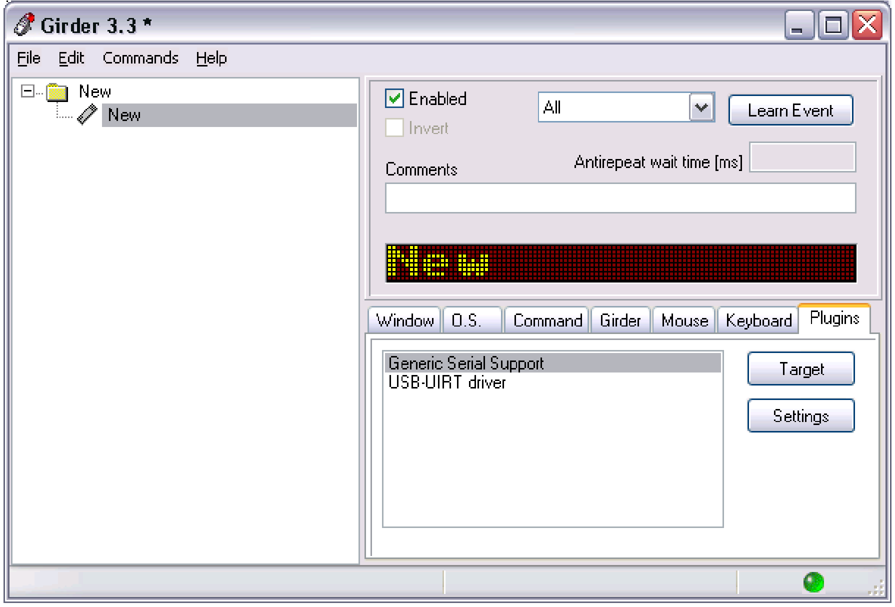
Message Definitions

b. Choose a name carefully – things break if you change the name.

c. Set the appropriate baud rate, parity, port, etc. per your equipment. If you don't know the proper settings for this, you may need to refer to the manufacturer's documentation or contact the manufacturer.

ii. **Outputs.** Output to a serial device is relatively easy. The main problem I encountered with output is that the device frequently returned a bunch of information. I did not need most of this information, but I had to insert a bunch of delays between my output commands to allow the unit time to respond before sending the next command.

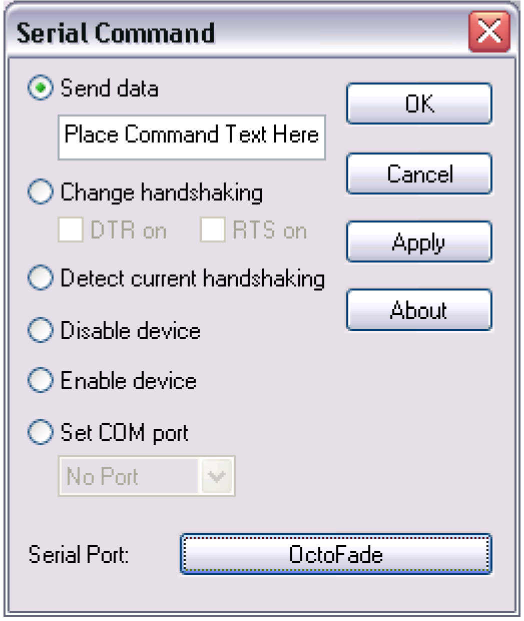
1. Create a Girder command.
2. Select the Plugins tab.
3. Select the Generic Serial Support plugin.



☑ Enabled
□ Invert
All
Antirepeat wait time [ms]
Comments

Generic Serial Support
USB-UIRT driver

4. Select Settings. The following dialog box appears:



● Send data
Place Command Text Here
○ Change handshaking
□ DTR on □ RTS on
○ Detect current handshaking
○ Disable device
○ Enable device
○ Set COM port
No Port
Serial Port: OctoFade

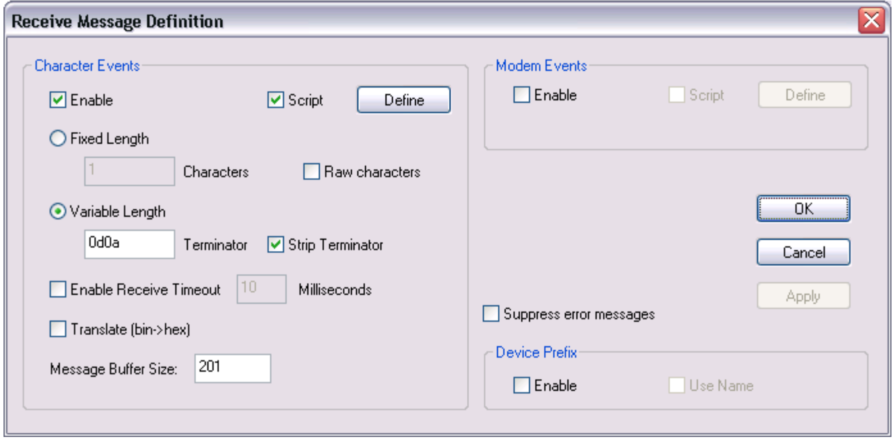
5. The button at the bottom, with “Serial Port:” beside it enables you to select the device you setup in step i.3 above.
6. Replace “Place Command Text Here” with the character string you wish to send to your device. For example, with the Octo Fade, to switch to input

number one, simply send a “1c”. This was considerably more complicated for the Smart Fade, which requires separate commands to select the input, and to perform the switch. For example, to fade to input #1, “1i” is the first output, then a 100ms delay, then “8192y” is the second output.

7. Note that you may need to adjust the Transmit Message Definitions in the Girder Settings dialog box (see i.3 above) as needed for your device application. Especially consider the Terminator and Hex->Bin conversion settings.

iii. **Input.** Serial device input is a bit more complicated. A lot depends on the format and predictability of the serial devices response structure. There are two primary ways to handle input:

1. Events. This would be similar to learning a MIDI or keyboard event – click Learn Event, and send the event string. This requires that the event string being sent from your device be short, simple, predictable and controllable. The strings I received from the Octo Fade and Smart Fade had none of these characteristics, so I gave up on this method for my application.
2. Receive script. This is set in the Receive Message Definitions dialog box in Girder Settings (see i.3 above).



Receive Message Definition

Character Events

☑ Enable ☑ Script

○ Fixed Length

1 Characters □ Raw characters

● Variable Length

0d0a Terminator ☑ Strip Terminator

□ Enable Receive Timeout 10 Milliseconds

□ Translate (bin->hex)

Message Buffer Size: 201

Modem Events

□ Enable □ Script

□ Suppress error messages

Device Prefix

□ Enable □ Use Name

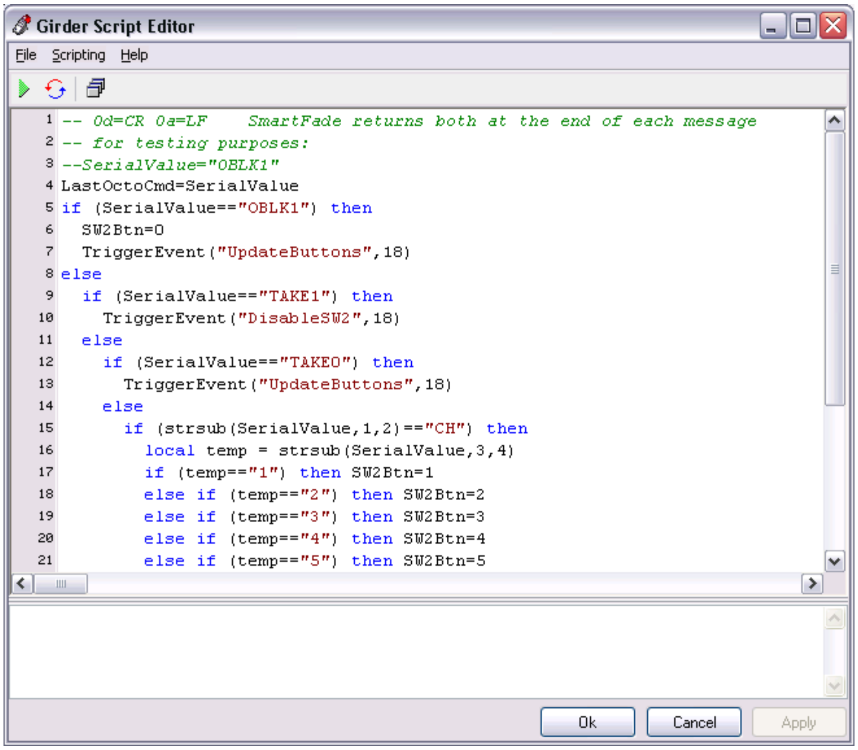
a. Enable reception of character events.

b. Set it to Variable length commands.

c. For my application, both the Octo Fade and Smart Fade have a carriage return/line feed (CRLF) after each response. This translates to a hexadecimal 0d0a. I elected to have the plugin strip this, simplifying the parsing script.

d. Check the Script box.

e. Click Define.



```
-- 0d=CR 0a=LF    SmartFade returns both at the end of each message
-- for testing purposes:
--SerialValue="OBLK1"
LastOctoCmd=SerialValue
if (SerialValue=="OBLK1") then
  SW2Btn=0
  TriggerEvent("UpdateButtons",18)
else
  if (SerialValue=="TAKE1") then
    TriggerEvent("DisableSW2",18)
  else
    if (SerialValue=="TAKE0") then
      TriggerEvent("UpdateButtons",18)
    else
      if (strsub(SerialValue,1,2)=="CH") then
        local temp = strsub(SerialValue,3,4)
        if (temp=="1") then SW2Btn=1
        else if (temp=="2") then SW2Btn=2
        else if (temp=="3") then SW2Btn=3
        else if (temp=="4") then SW2Btn=4
        else if (temp=="5") then SW2Btn=5
```

f. The script is written in the Lua language. Links to the Lua language reference and Girder extensions to it can be found in the Girder Forums and in the Help menu. SerialValue is a system variable passed to this script by the plugin, and contains the string up to the terminator. It is simply a matter of parsing and testing this string and doing something with it. You can set variables to be tested and used in other commands, or trigger events immediately. My scripts use a combination of these techniques.

f. Video Control Panel - a C# application

i. **Purpose.** The original intent of this project was to create all the necessary commands in Media Shout to control IR devices, video switchers, etc. These could be in scripts or in the Box. However, I discovered a couple of factors that made this as the sole solution undesirable:

1. We need to be able to send commands to video switchers during songs, however, there is a bug in Media Shout 2.5.5 which causes video backgrounds to go blank (black) if executing an audio cue during a Lyric cue with a video background.
2. The lack of visual indication of the status of the video switchers.
3. What was really needed was:
    a. a simple “control panel” – a floating (always on top) toolbar that made the essential video switcher functions available at all times,
    b. without needing to switch software screens,
    c. provide immediate and direct access to both video switcher functions and Media Shout,

d. still allow scripting of Media Shout,

e. and visually display the video switcher status, even when that status was changed by some means other than the control panel software, like Media Shout or manually pressing switcher buttons.

Thus was born the Video Control Panel. It is written in C#: a new Visual Studio.NET language, which combines the simplicity of use of Visual Basic, and the power and syntax of C++.

ii. **Output from Video Control Panel.** Because Video Control Panel runs on the same machine as Girder, the most direct, simplest, and fastest means to send commands to Girder is via the Girder Component Object Model. Examples of the code to do this can be found in the source code files included. COM events received in Girder are "learned" just like MIDI or IR events, and attached to the desired Girder command(s). Fortunately, Girder allows as many events to trigger the same command or multi-command as you want, so the same command can be triggered by MIDI, serial port, or Video Control Panel.

iii. **Input to Video Control Panel.** Unfortunately, the Girder COM objects at the time of this writing do not provide sufficient feedback to report the video switcher status we need. Fortunately, Girder provides many other options. The Girder Internet Event Client and Server seemed like a good choice, however it has a complicated token/response security protocol that seemed like overkill for this application, since these applications run on the same machine. Using Lua, it is relatively simple to open a socket to Video Control Panel using the IP loopback address (127.0.0.1), and send a short message. Video Control Panel has a timer that checks the socket frequently for incoming messages, as indicated by the "♦" symbols, and interprets them to update the status indicators.

g. Putting it all together – odds & ends

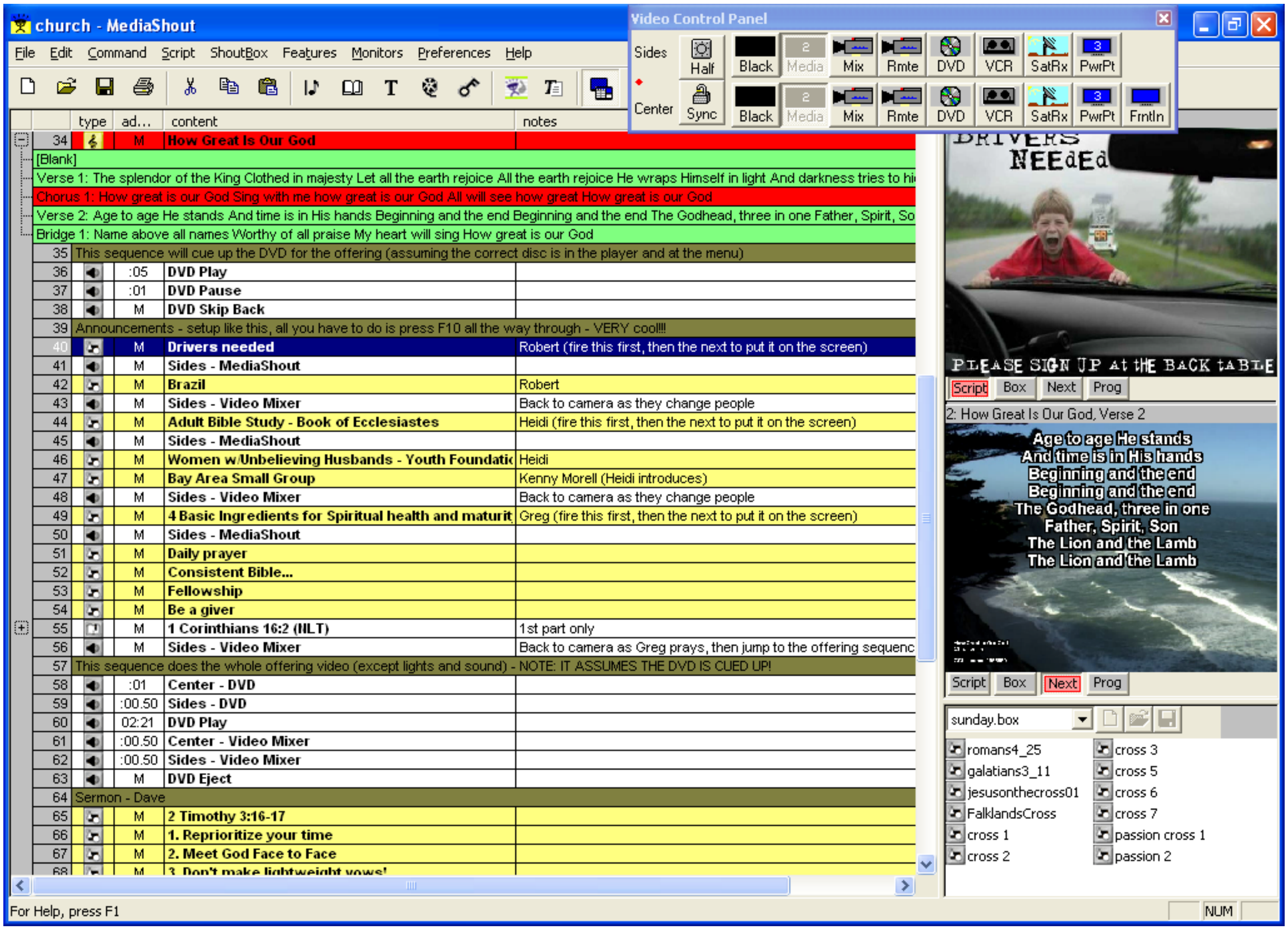
i. In this application, both video switchers require RS-232 connections. Unfortunately, the computer only had a single RS-232 comm port. It is becoming increasingly difficult to find a simple RS-232 port card for the PCI bus. This is old technology by today's standards. Fortunately there are other options. What I was looking for was an Ethernet IP to RS-232 port adapter or a USB to RS-232 adapter. I tried a USB to RS-232 adapter made by Belkin. This was a complete failure. When the drivers were loaded, and the first command sent to the RS-232 port, the USB keyboard and mouse ceased to function. I also tried an Ethernet to RS-232 adapter made by StarTech. This worked, but was not stable. I repeatedly experienced the infamous Blue Screen of Death (BSOD). A colleague alerted me to another brand of Ethernet to RS-232 adapter, called Moxa (www.moxa.com). This didn't turn up in the cursory Google search, but has turned out to be very reliable. We have been using the Moxa adapter for two or three months with zero BSOD's or failures of any kind.

ii. The biggest complication in the development of this system was the complexity of the command protocol used by the Smart Fade video switcher. This was especially exacerbated by the desire to benefit from the manual fade ability to overlay camera image and Media Shout output (or half-fade). Resolution of this problem took about four weeks of “spare time” programming and testing – about 40 hours of work.

iii. In an effort to expand the flexibility of the system, including showing two different text messages on two different screens at once, I added a second dual-screen video card to the computer. This gives us up to four different video outputs. By doing this, we can use both Media Shout and PowerPoint at the same time, and another output for expansion or other custom software, each displayed on a different screen.

    1. The primary output, screen 1, is the control screen for Media Shout and Video Control Panel. Screen 2 is the Media Shout display screen, and is the primary input for both video switchers. Screen 3 is used primarily as the PowerPoint display screen and input 7 to the video switchers. Screen 4 is currently seldom used, but can be connected to input 8 of the Octo Fade video switcher on demand.
    2. This screen 4 output will be used for the first time to overlay the names of people being baptized over a camera image of the baptistery for the center screen, while worship lyrics from Media Shout will be displayed on the side screens. The software for the baptism names is custom software which communicates via IP sockets over a wireless Ethernet connection. A user will enter names of those being baptized in a laptop down front by the baptistery. Those names will be sent via IP to the computer running Media Shout and another piece of custom software which receives the names and displays them on monitor output 4. Monitor output 4 is put in title mode on the video switcher, overlaying it with another input, specifically, a video camera.

iv. As shown in the sample, various MIDI events for IR devices and serial devices can be combined with specified timings and the Video Control Panel operating simultaneously. “Sides…” in Video Control Panel and Media Shout cues refer to switching the side video screens and “Center…” refers to the center video screen. During announcements we typically keep a camera shot on the center screen and information slides on the side screens, except when prerecorded videos (usually on DVD) play while offering is being collected.



	type	ad...	content	notes
34	♪	M	How Great Is Our God	
			[Blank]	
			Verse 1: The splendor of the King Clothed in majesty Let all the earth rejoice All the earth rejoice He wraps Himself in light And darkness tries to hi	
			Chorus 1: How great is our God Sing with me how great is our God All will see how great How great is our God	
			Verse 2: Age to age He stands And time is in His hands Beginning and the end Beginning and the end The Godhead, three in one Father, Spirit, So	
			Bridge 1: Name above all names Worthy of all praise My heart will sing How great is our God	
35	This sequence will cue up the DVD for the offering (assuming the correct disc is in the player and at the menu)			
36	◀	:05	DVD Play	
37	◀	:01	DVD Pause	
38	◀	M	DVD Skip Back	
39	Announcements - setup like this, all you have to do is press F10 all the way through - VERY cool!!!			
40	▣	M	Drivers needed	Robert (fire this first, then the next to put it on the screen)
41	◀	M	Sides - MediaShout	
42	▣	M	Brazil	Robert
43	◀	M	Sides - Video Mixer	Back to camera as they change people
44	▣	M	Adult Bible Study - Book of Ecclesiastes	Heidi (fire this first, then the next to put it on the screen)
45	◀	M	Sides - MediaShout	
46	▣	M	Women w/Unbelieving Husbands - Youth Foundatic	Heidi
47	▣	M	Bay Area Small Group	Kenny Morell (Heidi introduces)
48	◀	M	Sides - Video Mixer	Back to camera as they change people
49	▣	M	4 Basic Ingredients for Spiritual health and maturit	Greg (fire this first, then the next to put it on the screen)
50	◀	M	Sides - MediaShout	
51	▣	M	Daily prayer	
52	▣	M	Consistent Bible...	
53	▣	M	Fellowship	
54	▣	M	Be a giver	
55	▢	M	1 Corinthians 16:2 (NLT)	1st part only
56	◀	M	Sides - Video Mixer	Back to camera as Greg prays, then jump to the offering sequenc
57	This sequence does the whole offering video (except lights and sound) - NOTE: IT ASSUMES THE DVD IS CUED UP!			
58	◀	:01	Center - DVD	
59	◀	:00.50	Sides - DVD	
60	◀	02:21	DVD Play	
61	◀	:00.50	Center - Video Mixer	
62	◀	:00.50	Sides - Video Mixer	
63	◀	M	DVD Eject	
64	Sermon - Dave			
65	▣	M	2 Timothy 3:16-17	
66	▣	M	1. Reprioritize your time	
67	▣	M	2. Meet God Face to Face	
68	▣	M	3. Don't make lightweight vows!	

DRIVERS NEEdEd
PLEASE SIGN UP At tHE BACK tABLE

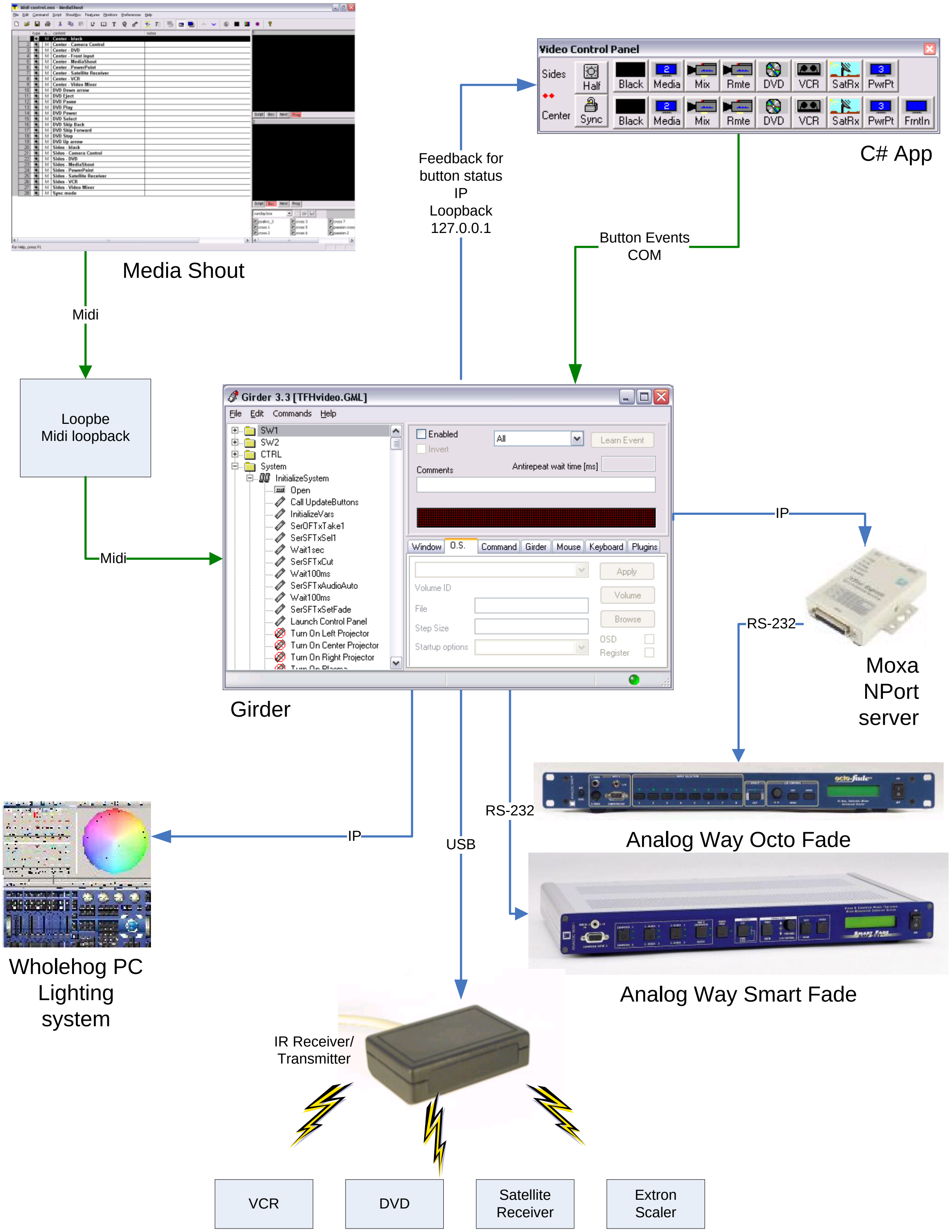
2: How Great Is Our God, Verse 2

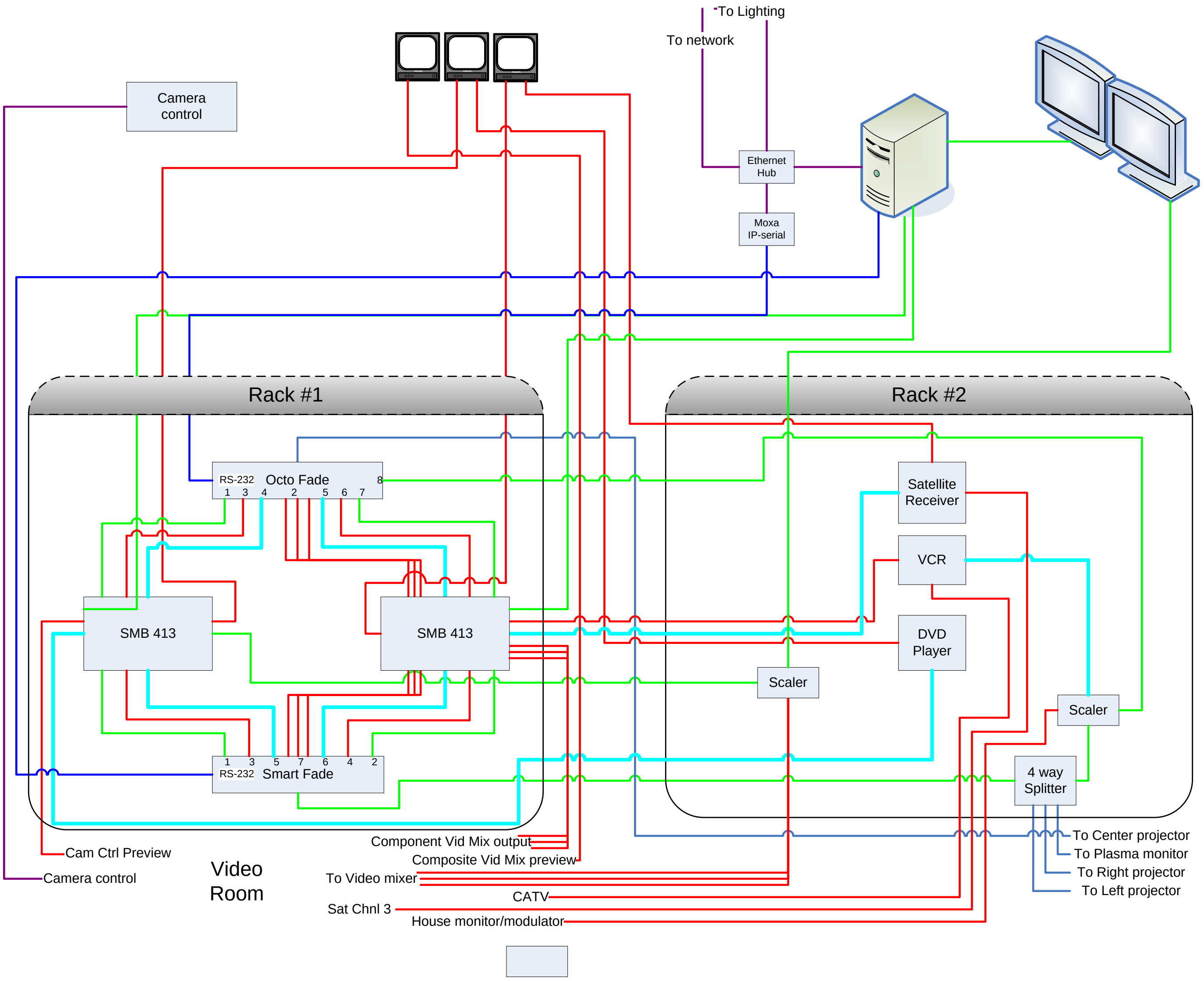
Age to age He stands
And time is in His hands
Beginning and the end
Beginning and the end
The Godhead, three in one
Father, Spirit, Son
The Lion and the Lamb
The Lion and the Lamb

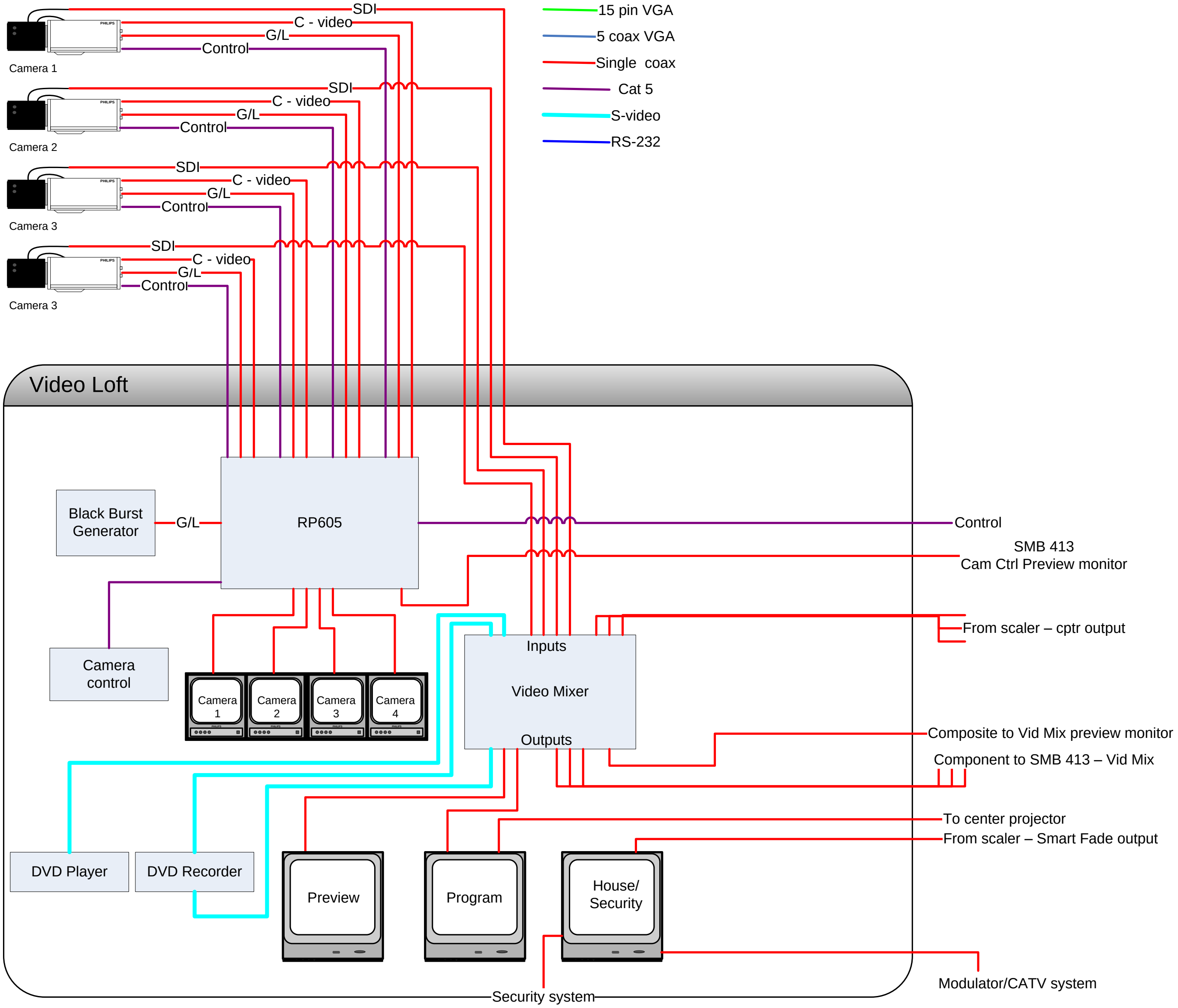
3. **Next steps. Several things are planned to expand this system:**
   a. **Lighting system.** TFH uses WholeHog PC 2 software, by Flying Pig Systems, to control the lighting system. It supports keyboard macros to perform complex tasks including multiple lights and timings. Theoretically, by running another copy of Girder on the lighting computer, making an IP socket connection from the copy of Girder on the Media Shout computer, and using Girder to emulate keystrokes to trigger lighting system macros, Media Shout should be able to trigger complex lighting changes with minimal effort and impact on the lighting system operation.
   b. **Projector control.** By having Girder turn on and off the projectors (and plasma screen), I want to eliminate the problem where some novice operators forget to turn off projectors, thus leaving them on for days on end. This will involve IR distribution to multiple outputs over distances of up to 200 feet, and power control modules.
   c. **Video camera system control.** Theoretically, Girder could control both the remote control camera system (Panasonic RP-605) and video mixer (MX-70). This is not practical for large multi-operator services, but could be very useful in smaller services with limited available operators.

4. **Summary.** With enough ingenuity, time, and patience, a very effective system can be automated, improving presentation quality, reducing errors, in a very cost-effective manner, all with Girder as the center.

TFH Video Control System
Basic Block Diagram




To Lighting
To network
Camera control
Ethernet Hub
Moxa IP-serial
Rack #1
Rack #2
RS-232 Octo Fade
1 3 4 2 5 6 7 8
SMB 413
SMB 413
Satellite Receiver
VCR
DVD Player
Scaler
Scaler
4 way Splitter
1 3 5 7 6 4 2
RS-232 Smart Fade
Cam Ctrl Preview
Camera control
Video Room
Component Vid Mix output
Composite Vid Mix preview
To Video mixer
CATV
Sat Chnl 3
House monitor/modulator
To Center projector
To Plasma monitor
To Right projector
To Left projector


15 pin VGA
5 coax VGA
Single coax
Cat 5
S-video
RS-232
Camera 1
SDI
C - video
G/L
Control
Camera 2
SDI
C - video
G/L
Control
Camera 3
SDI
C - video
G/L
Control
Camera 3
SDI
C - video
G/L
Control
Video Loft
Black Burst Generator
G/L
RP605
Control
SMB 413
Cam Ctrl Preview monitor
Camera control
Camera 1
Camera 2
Camera 3
Camera 4
Inputs
Video Mixer
Outputs
From scaler – cptr output
Composite to Vid Mix preview monitor
Component to SMB 413 – Vid Mix
To center projector
From scaler – Smart Fade output
DVD Player
DVD Recorder
Preview
Program
House/
Security
Modulator/CATV system
Security system