DMX512 Programmable Theater Lighting Controller

Final Proposal

Bradley University
EE 451 Senior Project

Advisor:
Dr. D. Schertz

by
Kris Kopel
Jeff Sand
December 8, 2000

Abstract

The project goal is to build a self-contained console for theater lighting control that will plug into the USB port on an x86 computer. It will allow both real-time and pre-programmed control of lighting instruments through the use of the industry standard DMX512 lighting control protocol. The system will consist of a PC connected through the Universal Serial Bus (USB) to a receiver chip and a microcontroller which will then interface to user input hardware and output circuitry. The microcontroller used will be a Motorola 68376. Also part of the project will be a user input interface for a set (or multiple sets) of faders (potentiometers) and buttons for real-time control of the lighting instruments connected to the console. The input blocks will be designed to be modular so the system can be configured with different amounts and types of user inputs. The system will output the dimmer control data in DMX512 format using a UART and an RS-485 receiver. The PC monitor will be used to display the status of the console and to let the user program queues into memory.

The Universal Serial Bus

The Universal Serial Bus is a newer PC peripheral interface that offers us several advantages over the ISA bus or the parallel port. First and foremost, it is a faster interface than the parallel port. This is critical for our application since data will have to move very fast for real-time control of lighting equipment. Second, the USB allows the system to be connected to the PC without needing to open the box or configure any internal hardware. Third, there are USB receiver chips on the market that offer a simple way to communicate with the USB bus with minimal external hardware. Finally, using the USB provides the opportunity to learn about a new PC communication method.

DMX512 Lighting Control

DMX512 is the industry standard digital communication protocol for theatrical, television, and special event lighting equipment. The protocol was designed in response to a number of competing proprietary systems developed in the early and mid 1980's. By 1990, DMX512 had virtually replaced all of these protocols and had become the de facto standard. It uses a digital bitstream operating at 250kbps over a RS-485 line. It is a fairly simple yet extremely useful asynchronous communication scheme. DMX512 has also been used, or some might say hacked, to operate fog machines and various other special effects equipment in addition to lights. There are some packages on the market for turning a PC into a lighting controller; however, these usually cost $1000 or more, so one goal of this project is to design one at significantly lower cost.

System Inputs and Outputs

The overall system block diagram is shown in Figure 1. The inputs to the system will be a PC with keyboard and mouse and a control console (also housing the microcontroller) with a series of faders and buttons. The microcontroller and the analog inputs will all be contained in one console, which will sit in front of the user next to the PC. The PC will be used as a graphical user interface with programming capabilities. It will be connected to the controller via the USB port so no internal connections will be required on the computer. The faders and buttons will be mounted on the console and will be used for real-time analog control of lights during a show and for programming queues. The user will be able to control the system using the analog controls, pre-programmed queues on the computer, or a combination of both. As shown in the block diagram in Figure 2, the faders and buttons will be in blocks of 8 of each (this number may change as the design progresses), and these blocks will be designed such that they will be modular, which will allow the console to be built with however many fader blocks the customer wishes to order.

Figure 1 - Overall System, Major Blocks

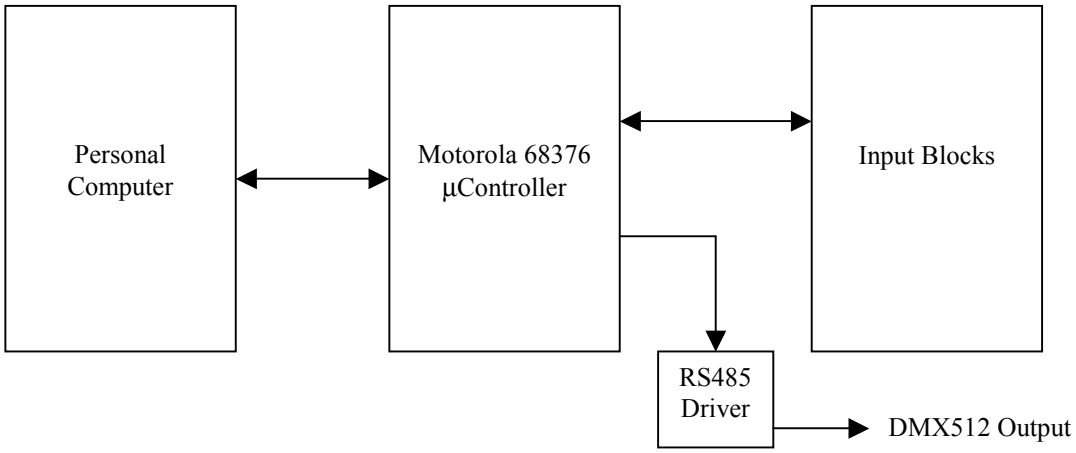
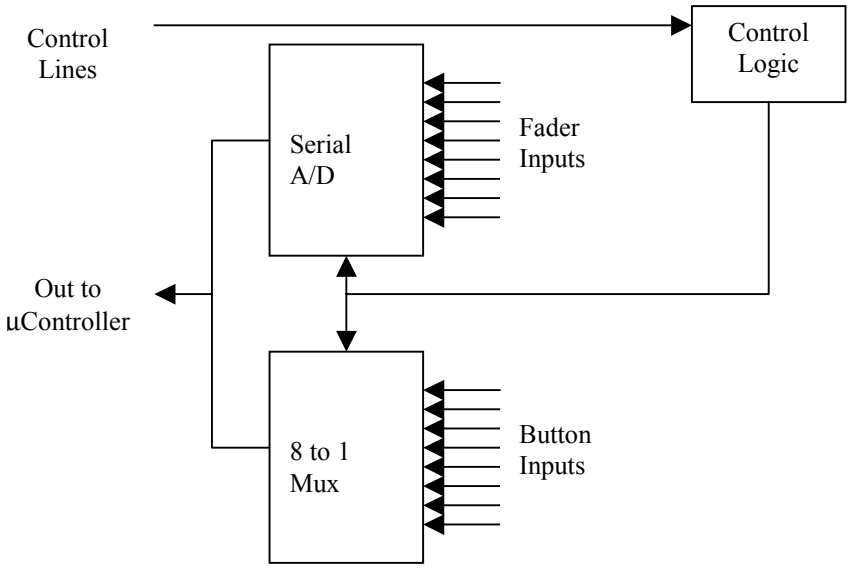


Figure 2 - Input Blocks



The outputs of the system, shown in Figures 3 and 4, are the PC monitor and a DMX512 output port. DMX512 is the industry standard lighting control protocol, and is a simple RS-485 based asynchronous communication scheme. The PC will also be used as an output in the sense that the monitor will show the status of various parts of the system, such as what lights are turned on and at what dimmer levels.

Figure 3 ñ Personal Computer

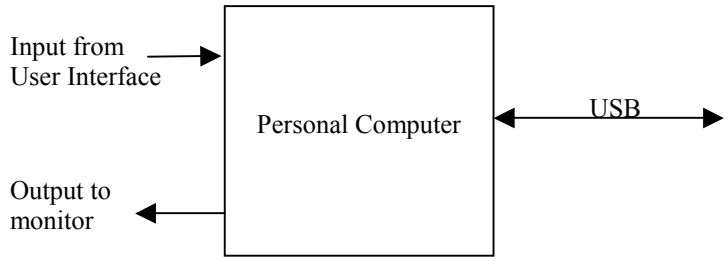
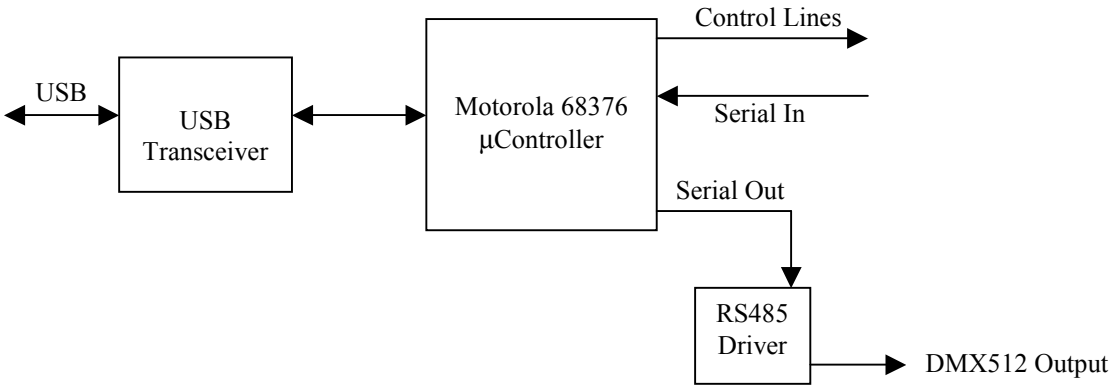


Figure 4 ñ Motorola 68376 µController



Thus there will be three main functional blocks in the system. The first is the PC (Figure 3) to handle all programmable aspects of the controller. It will have a GUI for programming and displaying the status of the system. It will be connected to the microcontroller block using a USB connection.

The second block is the microcontroller (Figure 4), for which we plan to use a Motorola 68376. It will be connected to a USB transceiver for communication with the PC and output to the DMX512 interface. It will have serial input from the input blocks. The microcontroller's on-board UART will handle the output in DMX512 format, which can support up to 512 dimmers (lighting circuits) in one chain. The output of the UART will be connected to an RS-485 transmitter. The DMX512 output takes the form of a 5-pin XLR connector.

The third block will be the input blocks (Figure 2). As this is a modular design, each input block will contain a set of faders and push buttons which will send a single serial data line to the microcontroller. This will be accomplished using the Queued Serial Peripheral Interface (QSPI) on the 68376. The logic block shown in Figure 2 represents the glue logic necessary for the A/D control lines and the button status output.

PC Modes of Operation

There will be three modes of operation. The two normal operating modes will be real-time control and programmed control. In real-time control, the user will operate the system using the analog controls on the console to control the lights for a scene in real time. In programmed control, the computer will control the lights based on pre-programmed queues that the user has defined. Note that these two modes can be used at the same time and operate the same way in hardware; they are simply different methods of software control. The computer will poll the console for the fader data at regular intervals (30-40 times per second) and store this information in memory. It will then, depending on how the user has the software configured, either:

- Use the fader levels to determine the updated output
- Bypass the faders and output only the pre-programmed queues
- Enable both of these modes of operation at the same time, setting the output levels to whichever of the two modes' output levels would be higher

This last "combined" mode may require an example to illustrate. Supposed the user has programmed a sequence that will flash several dimmers to full power, one at a time, and that this sequence will be used as an effect during a production. For testing purposes, however, the lighting designer may wish to test this sequence but at the same time keep the rest of the lights in the sequence on at 20% so he can see where they are aimed. To accomplish this, he could run the programmed queue, while at the same time leaving the appropriate faders on at 20%. This way, when a dimmer would be at 100% as part of the sequence, it would still turn on 100%, but when the sequencer would turn the dimmer off, the fader would take precedence and the light would remain on at 20%.

The third mode of operation will be where the PC acts as a status monitor only and will not control the lights. In this mode, the microcontroller will have control over the system rather than looking to the PC for the output. This will allow the user to control the dimmers using the faders directly, without the need for the PC. It could also be used for troubleshooting the system if a failure occurs.

Microcontroller Modes of Operation

The microcontroller will similarly have two modes of operation. In the normal PC-controlled mode, it will mainly serve the purpose of relaying data. The microcontroller will continuously sample the fader levels and pass this information through the USB link to the PC on command, and it will relay the output levels from the PC to the DMX512 output.

The second mode of control for the microcontroller will be a standalone mode to accompany the PC's status monitor mode. In this mode of operation, the microcontroller will not only sample the fader data but also bypass the PC and output this data to the dimmers directly. This will allow the console to operate independent of the PC for troubleshooting purposes or for the convenience of the user.

Project Status

The 68376 Development Board has been tested and its interface to the debugging environment has been setup. Chips have been selected for use in the input blocks, the USB connection and for driving the DMX512 channel. The user software and USB development station with both Windows 98SE and Linux has been set up. The website is online and has project deliverables downloadable at http://cegt201.bradley.edu/projects/proj2001/pcusbdmx/.

Schedule

Week 1:	Build header connectors for microcontroller bus and begin work on temporary PC/microcontroller interface
Week 2:	Develop temporary PC interface
Week 3:	Finish temporary interface, start DMX output block
Week 4:	Finish and test DMX output block
Week 5:	Begin USB interface, input blocks
Week 6:	Continue USB interface
Week 7:	Finish input blocks
Week 8:	Finish USB, begin testing
Week 9:	Test inputs blocks
Week 10:	Test/verify DMX output stream
Week 11:	Miscellaneous hardware testing
Week 12, 13:	Add extra features if time permits

Week 1:	Develop interface software for temporary PC/microcontroller interface
Week 2:	Begin USB drivers
Week 3:	Write initial DMX512 program for 376
Week 4:	Finish and test temporary x86/376 output software
Week 5, 6, 7:	Continue USB driver software and begin user software
Week 8:	Start testing USB software on x86 and 376
Week 9:	Test input block communication
Week 10, 11:	Continue and test user software
Week 12, 13:	Add additional programming and user interface features as time permits

Parts List

Motorola 68376 Development Board	(1)
Philips PDIUSBD12D USB transceiver	(1)
Maxim MAX148 A/D Converter	(1 per input block)
Maxim MAX349 Parallel to Serial 8 bit MUX	(1 per input block)
Maxim MAX483 RS485 driver	(1)
USB type B receptacle	(1)
2k Audio Slide Potentiometers	(8 per input block)
Push buttons	(8 per input block)