Microcontroller Based Multichannel Light Dimmer

University of Waterloo
Department of
Electrical and Computer Engineering

E&CE 499 Fourth Year Project
Final Report
for Professor P. Dasiewicz

Apr 7, 2001
Ed Maste
ID #96121900

Summary

The project described in this report is a microprocessor-controlled lighting dimmer that could be used in theatrical or other applications. The project consists of a physical hardware prototype and associated firmware to execute on the project's processor.

The report begins with a description of the customer requirements for the project. These were developed into functional specifications. The report then describes the design process for the project hardware and firmware. The testing methodology and results are also included.

It was concluded that the project works as expected and as required in all material aspects. One of the modes of operation could not be fully tested due to the lack of available test equipment. It is recommended that further testing be completed and some minor design issues be rectified if another version of the project was to be built.

Table of Contents

List of Figures

List of Tables

1. Design Abstract

The Microcontroller based Multichannel Light Dimmer (MMLD) is a microprocessor-controlled lighting dimmer that could be used in theatrical or other applications. The project will consist of a physical hardware prototype and associated firmware to execute on the project's processor.

The device will accept input from a wide range of interfaces: RS232, DMX512 (a theatrical lighting standard) or RS485, and the Musical Instrument Digital Interface (MIDI). The device will control a number of lamps by using some form of AC power control. The firmware in the microcontroller would handle all functions, from decoding the protocol on either of the input interfaces through to timing the firing of power triacs for the output.

2. Project Plan and Milestones

2.1. Design Challenges

The dimmer hardware will present some design challenges due to the large currents it will be required to handle (up to 15A). This will require careful attention to component selection, circuit design, printed circuit layout, and assembly technique. The dimmer also operates with AC mains potential (120V) so careful attention to safety will be required.

The dimmer will have a number of control interfaces, and each of these must be compliant with relevant standards to ensure interoperability with other equipment.

The dimmer's firmware has two main aspects: responding to data via its control interfaces and turning on the output switches at the correct time relative to the AC line voltage. It is anticipated both of these aspects will present hard real time problems. Failing to process the control data as it arrives will result in the dimmer having incorrect intensity values on one or more channels, and failing to turn on the output drivers at the correct time will result in erratic or incorrect output voltages. It is anticipated that these real-time requirements can be met by careful code design and the use of microcontroller resources (such as timers and interrupts) for dedicated tasks.

2.2. Milestones

Table 1 details each of the anticipated major milestones for the project, including the target completion date and the actual completion date if the milestone has been accomplished.

Table 1: Project Milestones

Milestone	Target Date	Status or Completion Date
Customer Requirements	January 15, 2001	January 17, 2001
Functional Specifications	January 22, 2001	January 22, 2001
High Level Design	January 29, 2001	February 4, 2001
Component Selection	February 5, 2001	February 18, 2001
Schematic Design	February 12, 2001	March 1, 2001
PCB Layout	February 19, 2001	March 11, 2001
PCBs Fabrication	February 26, 2001	March 16, 2001
Prototype Assembly	March 5, 2001	March 26, 2001
Initial Firmware Complete	March 12, 2001	April 1, 2001
Prototype Testing and Verification	March 19, 2001	April 20, 2001

2.3. Expected and Known Requirements

2.3.1. School Resources

The project designer has access to work space outside of school so the requirement for school lab space should be minimal. The labs may be used during the testing and verification stage for access to equipment such as oscilloscopes.

Should the school's circuit board milling machine be operational, it will likely be employed for the fabrication of PCBs for the project.

2.3.2. Industry Resources

It is anticipated that surface mount devices, including fine pitch devices, will be employed in the design and construction of this project. The hardware lab at Cisco in Waterloo will be used to solder these parts to the project's circuit board.

2.3.3. Software

The project will be developed using Open Source or freely available tools, so commercial software will not have to be purchased. Firmware written for the project's microcontroller will be developed in C and complied with GCC, if GCC has support for

the chosen microcontroller. Otherwise, a microcontroller will be chosen that has free assemblers or compilers available.

3. Customer Requirements

The project shall be an intelligent lighting controller that will provide automatic control of the brightness of lamps attached to it, for amateur theatrical or home applications. At least four independent channels of lighting control shall be provided.

It shall provide standard electrical sockets that these lamps can be plugged into, and will itself be plugged into a wall socket for power. The project shall operate on (at least) the AC power supply available in North America.

For household applications, each dimmer channel shall have a power handling capacity to allow two common table lamps, or one large floor-standing lamp, to be connected. To allow for theatrical use, each channel shall have a capacity to allow one large or several smaller theatrical lights to be connected. The overall capacity of the dimmer may be limited to the power available from one standard wall socket.

The project shall be compatible with standard dimmer control interfaces and shall also allow easy connection to a standard personal computer.

4. Functional Specifications

Based on customer requirements, the functional specifications indicate the scope of the design and include relevant documentation for all interfaces provided by the project.

4.1. AC Power Input

The nominal power supply from a standard AC outlet in Canada and the USA is 120V alternating current (AC) with a frequency of 60Hz (60 cycles per second). To allow for variations due to load and other factors, a tolerance of 10% shall be allowed on the voltage and 5% on the frequency. Thus, the dimmer should operate on 108V to 132V AC and 57 to 63 Hz.

A standard wall receptacle circuit provides up to 15A of current. Since this dimmer is intended for home or amateur theatrical use, this is the maximum current the dimmer may draw. The dimmer's input shall be fused with a 15A fast-blow fuse. This provides for a maximum connected load of 1800 watts across all dimmer channels, excluding power required by the dimmer itself.

4.2. Dimmer Channel Outputs

The dimmer shall provide one standard power receptacle per dimmer channel. The receptacle type shall be in accordance with the National Electric Manufacturers Association type 5-15R [1].

Several residential lamps were examined to determine the power handling requirements of a single dimmer channel. Table lamps mostly used 60W bulbs, with the maximum being 150W in one fixture. The highest power bulb, at 500W, was found in a floor standing halogen lamp.

The dimmer should also be useable for amateur theatrical productions. Table 2 provides a summary of the lights in use at The Centre in the Square in Kitchener [2]. This is a professional theatre but gives a good indication of the types of lights used for theatrical lighting. As indicated, almost all of the lights are 1000W or less.

Table 2: Light Quantity by Wattage at The Centre in the Square

Light Wattage	Quantity of Lights	Percentage of lights at Wattage or below
500W	6	1.1%
575W	194	37.4%
1000W	321	97.4%
1500W	8	98.9%
2000W	6	100%

Based on the power requirements of these theatrical and home lamps, each dimmer channel shall be capable of handling 1000W with some margin (which will be determined by the availability of components). At 120V this represents a current of 8.33A. The dimmer also should not have a minimum load requirement per channel. The smallest reasonable 120V bulb is a 5W Christmas-tree lamp so this may be taken as a practical minimum load if desired. Each of the dimmer channels must be individually fused at a current less than the capability of that channel.

The dimmer must be able to vary the brightness of lamps connected to each channel by changing the average or RMS power delivered to the lamp. The controllable range should be as wide as possible, but should be at least 10% to 90%. Phase control [3] will be used to vary the applied RMS voltage.

The dimmer should incorporate electrical interference filtering components so that it does not adversely affect other electronic equipment.

4.3. Electrical Safety

Because of the high voltages (AC mains potential) used in the dimmer and for the channel outputs, the dimmer must adhere to electrical safety guidelines. Galvanic isolation must be provided between any component connected to the AC line and the rest of the circuitry and interfaces that are not. Any devices used to provide such isolation shall have an isolation rating of at least 2000 volts. Creepage and clearance distances on the PCB shall be at least 0.125 inches to ensure adequate isolation.

4.4. Control Interfaces

4.4.1. RS-232

In order to provide easy interfacing with a standard personal computer, as indicated in the customer requirements, an RS-232 port shall be provided on the dimmer. This is the type of port commonly referred to as a “serial port” on a PC. RS-232 is more correctly called TIA/EIA-232-F and actually specifies only the physical layer electrical interface [4], but in common usage it also indicates that asynchronous serial data is being transmitted and received, at one of many standard baud rates.

No requirement for the protocol used by the RS-232 interface has been set; therefore a simple menu based system will be used to allow system configuration and setting dimmer channel values. The baud rate will be 38 400 bits per second which should allow menus to be displayed at a reasonable speed. (A menu with 10 lines of 40 characters each would take 40 * 10 / (38 400 / 10) seconds, or roughly 100 milliseconds.)

The RS-232 port shall also be used for debugging and initial system testing. In order to facilitate debugging of the other control interfaces, the RS-232 port must operate independently of, and concurrently with, the other interfaces.

The RS-232 connection shall use a female DB-9 connector, allowing a connection to a personal computer via a "straight-through" serial cable.

4.4.2. MIDI

The Musical Instrument Digital Interface (MIDI) is a specification describing a protocol and electrical interface, and is typically used by electronic musical instruments. The dimmer shall be controllable by MIDI data in order to allow the dimmer to be used in performances by electronic musicians.

The dimmer should respond to MIDI standard "note on" and "note off" messages. MIDI "note on" messages contain a velocity (volume) level; this will be interpreted as a dimmer channel brightness setting. The dimmer should be able to be configured to receive one MIDI channel and have a different note number assigned to each of the four channels.

MIDI specifies that asynchronous serial communication is used at a baud rate of 31 250 bits per second [5], and the data is transmitted via a current loop. Current flowing in a MIDI connection corresponds to the active, or space, state. An absence of current indicates an idle, or mark, state. The MIDI interface will use standard female five pin DIN connectors and provide optical isolation on the receive channel [6]. MIDI data shall be received on a MIDI In port, and this shall be passed through to a MIDI Thru port to allow daisy chaining with other devices. A MIDI Out port may also be included although at this time it is not anticipated that the dimmer will generate its own outgoing MIDI data.

Use of the MIDI interface may be mutually exclusive with use of the other interfaces (with the exception of the RS-232 interface).

4.4.3. DMX512

A very common interface and protocol in theatrical lighting is DMX512 [7], which specifies how data is carried from lighting controllers to dimmers and light fixtures. It allows up to 512 individual dimmer channels to be controlled by one control cable.

DMX512 specifies asynchronous serial data transmitted at a rate of 250 000 bits per second using eight bits per byte, two stop bits and no parity. The data is transmitted using a balanced scheme, electrically equivalent to RS-485. Male five pin XLR connectors are used for input and female five pin XLR connectors are used for output.

DMX512 also specifies the low level protocol used on the interface. A serial “break” is transmitted for at least 88 uS followed by at least 8 uS of idle (mark). A start code of 0x00 is then transmitted, followed by up to 512 eight-bit dimmer intensity values.

Use of the DMX512 interface may be mutually exclusive with use of the other interfaces, except for RS-232.

4.4.4. RS-485

Since DMX512 is electrically equivalent to RS-485, an RS-485 connection may easily be provided on the dimmer via screw terminals. RS-485 specifies the use of a balanced interface with two data signals that are identical except for polarity [8]. This allows for a high degree of noise immunity when compared to more common serial interfaces such as RS-232. The data rate and protocol used on the vanilla RS-485 interface does not need to be defined and the interface may be provided for future capability only. Use of the RS-485 interface may be mutually exclusive with use of the other interfaces except for RS-232.

4.5. Firmware Functionality

The firmware has three functional tasks: system configuration, receiving protocol data, and controlling the output power drivers.

The system configuration system operates over the RS-232 interface and must allow the user to choose which of the control interfaces the dimmer will respond to. As well, any parameters for the specific control interface must be user settable. The MIDI interface

requires that the MIDI channel and note numbers be settable, while the DMX512 interface requires that each channel can be assigned to a dimmer number.

The firmware must receive and process protocol data on the selected interface, and interpret that data within the time taken to transmit the following control message.

The firmware must monitor the AC line zero crossing to determine when to turn on the output drivers and then activate them at the appropriate time.

5. High Level Design

5.1. Phase Control

The dimmer will use the technique of AC phase angle control, where the output loads are turned on for a portion of the total AC line sinewave period, as shown in Figure 1. If the loads are turned on at the beginning of the AC cycle, they will receive approximately the full power available from the AC line. If they are not turned on during an AC cycle, they receive no power. If the loads are powered for some portion of an AC line cycle, they will receive a corresponding portion of the total available power.

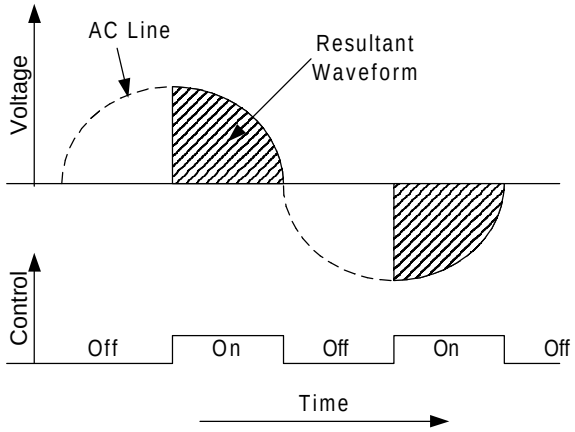

Figure 1: AC Phase Control

We define α as the phase delay between the AC line zero-crossing and the instant the output channel is turned on. The range on α is then zero to 180 degrees, with 0 degrees corresponding to full power an 180 degrees corresponding to zero power. The average and RMS voltage applied to the load, as a function of delay angle, appears in Figure 2 [9].

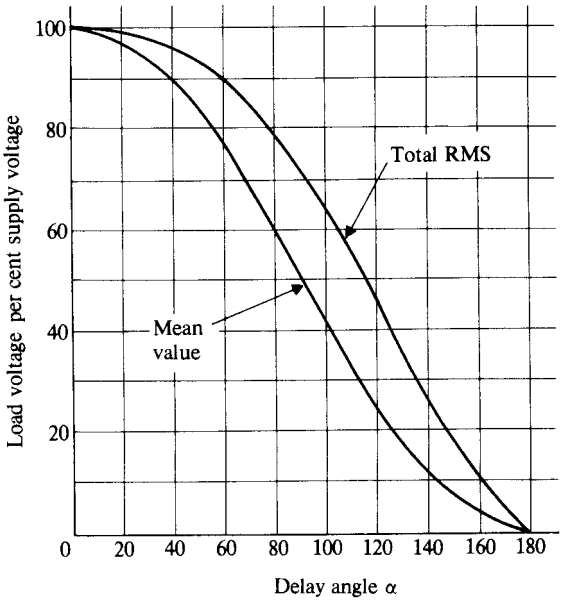

Figure 2: Load Voltage vs. Delay Angle

Clearly, the applied voltage to the dimmer's loads can be varied from essentially zero to 100% of the supply voltage by changing the delay angle.

5.2. Hardware Block Diagram

The block diagram for the dimmer appears in Figure 3. The critical component in the dimmer is a microcontroller. A microcontroller provides a low-cost method to easily implement the required functionality of the dimmer. It can simultaneously process input from any of the required control inputs, and control the power electronics for the individual channels. The microcontroller uses the input from an optically isolated zero crossing detector to determine how much time to delay before turning on individual channels.

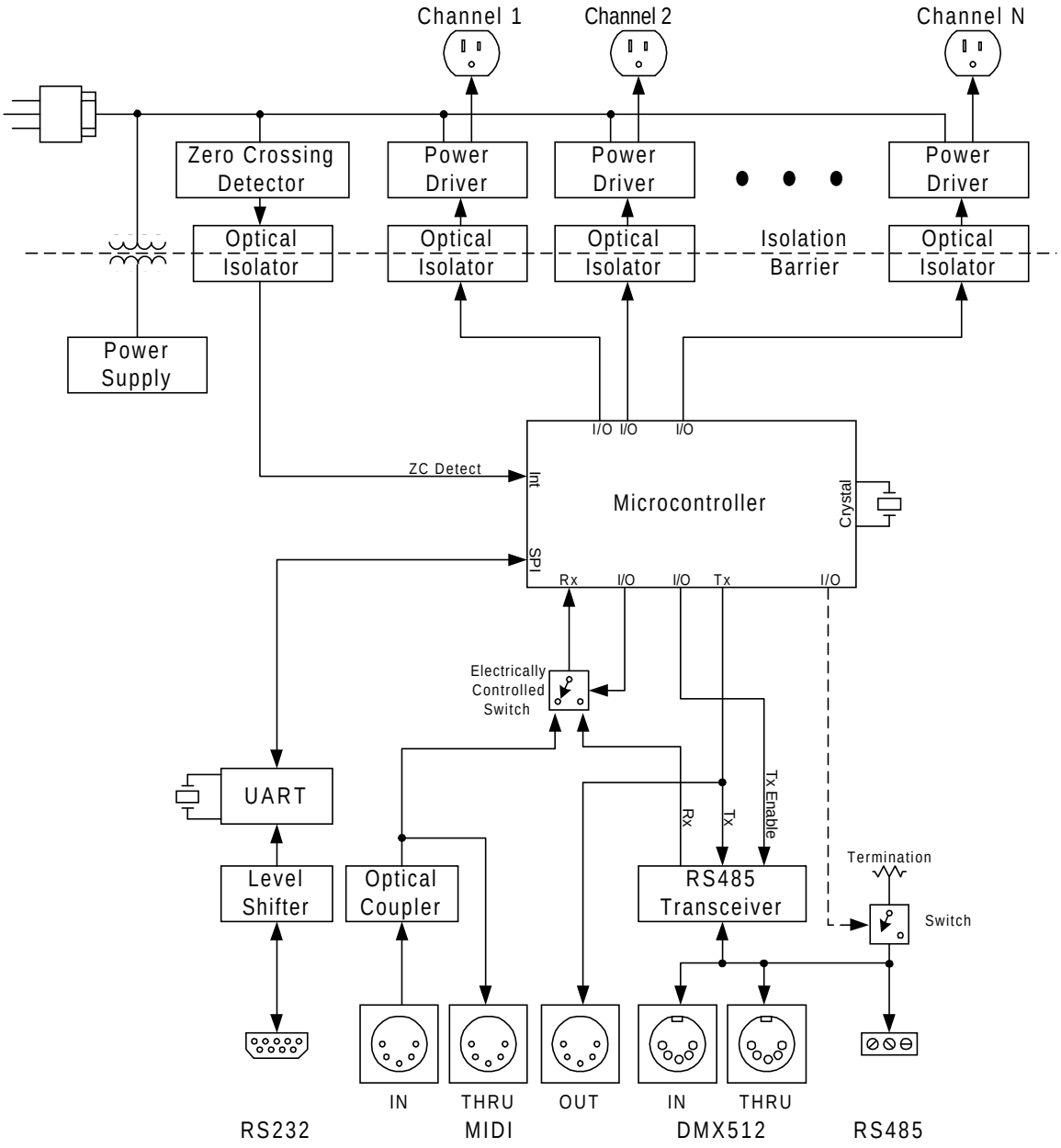

Figure 3: Dimmer block diagram

5.3. Firmware

The firmware has three main tasks. The primary task is the real-time phase angle control of the output circuits relative to the AC line zero crossing. In addition, it must process intensity messages on the various control inputs. Finally, it has to handle system

configuration and management tasks (such as selecting which input is active). It is intended that each of these functions be implemented as autonomous modules and be primarily interrupt-driven to avoid the downsides of polling schemes.

5.3.1. Phase Angle Controller

As mentioned, the dimmer will be based on a microcontroller. Most microcontrollers include timers and external interrupt capabilities. Many microcontroller timers provide compare and capture functionality, often providing a mode where a specified action can occur once the timer increments to a certain value. The phase angle control system will make use of these interrupt-driven resources to provide real time TRIAC firing without burdening the processor.

5.3.2. Debug / Supervisory and Control Inputs

All communications between the dimmer and external devices will be via one of two serial ports. One serial port (UART) is dedicated to debug and supervisory tasks and will be implemented as an RS232 port, while normal MIDI and DMX/RS485 communications can share another UART. A software-settable control signal should be provided to facilitate the switching between the two uses of the one UART. Both of these UARTs should be completely interrupt driven to reduce the burden on the processor.

5.3.3. Supervisory Tasks

The dimmer requires some supervisory control, which will be implemented via a menu on the debug and supervisory RS232 port. The supervisory tasks include the selection of the active control mechanism (MIDI, DMX512 etc.) and system configuration (such as enabling or disabling line termination). In addition, intensity control data will be accepted on the supervisory and debug port to allow for system testing and personal

computer (PC) control. The supervisory and debug port should provide some acknowledgement after receiving a request so that any associated control software can determine that the dimmer is still connected and operational.

6. Low Level Hardware Design

The hardware design began with the selection of components for the power supply, zero crossing detector and output drivers. The component selection and design process will be described in the following sections. Component designators (i.e. U3 or R7) refer to the schematics in Appendix A.

6.1. Component Selection and Schematic Design

6.1.1. Power Supply

The dimmer is to be a self-contained unit and thus requires a power supply to provide low voltage DC for its electronics from the AC power line. Because the current requirements of the dimmer's circuitry will be quite low, a standard linear topology was chosen. The components used in the dimmer will be standard, conventional parts that will use a supply voltage of 5V. These specifications dictate the use of a low-cost, ubiquitous linear regulator -- the National Semiconductor LM7805 [10] as U3. The LM7805 requires an input voltage of at least 7.5V in order to guarantee regulation, so the unregulated power supply should supply at least this voltage under worst-case current consumption, assumed to be about 200mA.

Because a full-wave bridge rectifier will be used for efficiency (diodes D1-D4), we can assume that about 1.4V will be lost across the bridge (0.7V per conducting diode). We therefore need a peak voltage of at least 9V from the power transformer. Because the RMS voltage of a sinusoidal AC signal is 0.707 times the peak, a power transformer with a 6.3V RMS secondary is required. A Tamura Corporation 3FS-312 transformer was selected as T1, which has a 6.3V RMS secondary at 400 mA [11].

6.1.2. Zero Crossing Detector

In order for the dimmer to determine at which point on the AC cycle to turn on individual channels, it needs a reference for the AC zero crossing. For safety reasons this zero crossing detector must be isolated from the low voltage electronics. This isolation can be achieved at low cost with an optoisolator, a component which uses light to provide galvanic isolation. Typically, a light emitting diode (LED) will be used as the light source and a light sensitive transistor as the receptor.

U5 is the optoisolator used as the zero crossing detector and is a 6N138 from Fairchild Semiconductor. It uses a darlington (dual transistor) output stage for high gain, providing a current transfer ratio (CTR) of typically 2000%. That is, the output current changes by 200 times the change in input current. The part is therefore useful for directly driving logic level inputs. The typical input parameters are: forward voltage V_F=1.30V, forward current I_F = 1.6mA. The maximum forward current is 20mA.

The zero crossing detector works by first full-wave rectifying the AC line using diodes D5, D7, D8 and D16. Full-wave rectifying before the zero crossing detector guarantees that the output waveform at the loads will be symmetric in the positive and negative half cycles, which is important for powering some types of loads such as transformers. This produces positive-going half-sinewave pulses at 120Hz. The peak voltage is $120V \times \sqrt{2} = 170V$. By using 20K of series resistance, basic Ohm's law determines the maximum current through the optoisolator to be $I = V/R = 170V - 1.4V/20000\Omega = 8.5mA$. The AC line voltage at which the optoisolator is guaranteed to be on is

$V = IR = 1.6mA \times 20k\Omega = 32V$. This point on the sinewave corresponds to $Sin^{-1}\left(32V/170V\right) = 11°$. In practice, the optoisolator will probably turn on much earlier.

6.1.3. Output Power Drivers

Triacs Q1-Q4 are used as the output drivers due to their low cost, small size and ruggedness. The BTA216X-600B device from Philips Electronics was chosen, due to some excellent characteristics. It comes in a fully-isolated package, which makes heatsinking easier and safer. It is also a so-called "snubberless" triac. Most triac circuits require a resistor-capacitor (RC) snubber to prevent false triggering when driving certain types of loads. The Philips snubberless triac is very resistant to this false triggering due to its design. (The schematic has provisions for such a snubber as a contingency if the Philips parts could not be obtained in time.) The BTA216X-600B is rated for 16A which easily allows for the 1000W per channel power requirement.

In order to provide isolation and ease triggering of the triacs, MOC3011 optoisolators (Fairchild Semiconductor) are used. These are designed to directly drive a triac's gate input, and the output circuits are basically copies of Fairchild's application note [ref!].

6.1.4. Microcontroller

The heart of the dimmer is U7, an Atmel AT90S8515 microcontroller. This microcontroller was chosen for several reasons. The project designer has extensive experience with the AVR series, as well as development tools for the parts. The GNU Compiler Collection (GCC) targets the AVR as a cross-compiler, allowing the project to be developed in C. The part also has eight kilobytes of flash memory for program storage, allowing for reasonably sized programs to be loaded. The AVR series is also in-

circuit programmable which eases the development cycle and allows for incremental firmware design. U7 runs at 8MHz, a frequency determined by crystal Y1.

The ‘8515 has some additional features beyond other members of the AVR series. It provides two external interrupt sources, one 16 bit and one 8 bit timer, a built-in UART, and a synchronous peripheral interface (SPI) serial port. One of the interrupts is triggered by the zero crossing detector. The 16 bit timer has an interrupt on compare match mode, and can be used for controlling the output turn-on point. The built-in UART is dedicated to receiving and transmitting on the control interfaces (MIDI or RS485/DMX512). Finally, the SPI port is connected to a separate UART, to provide a serial port for debugging.

6.1.5. Debugging UART and TIA/EIA-232 interface

The dimmer design includes an external UART in order to ease the bring-up (initial) phase and allow for debugging of the dimmer, even when the internal UART is being used for the control interfaces. The UART is U8, a MAX3100 from Maxim Integrated Products. Crystal Y2 provides a 3.6864MHz reference for U8. This frequency is chosen as it is an integer multiple of the baud clock required for standard asynchronous communications (e.g. 9600 baud or 57600 baud).

U10 is another Maxim part, a MAX202. This part converts transistor-transistor logic (TTL) signal levels to the bipolar voltage swings required by the TIA/EIA-232 interface. It has a built-in switched capacitor voltage convertor to generate the necessary +/- 10V supplies.

6.1.6. MIDI Interface

The dimmer can receive control data via a MIDI interface, to allow it to be used by electronic musicians and controlled by existing musical gear. The MIDI interface consists of a 6N138 optoisolator (U13), and a 74VHC125 quad buffer (U9). MIDI specifies an isolated current loop interface, which is most easily accomplished with an optoisolator. The logic output of U13 is buffered by U9A, where it is made available to U11, a 74VHC157 quad two-to-one multiplexor. This multiplexor allows the microcontroller's internal UART to be connected to either the MIDI interface or the RS485/DMX512 interface, depending on the state of the SERSEL signal.

The MIDI input is passed to buffer U9B to provide a "MIDI Thru" connection, to allow daisy-chaining of additional gear. The microcontroller's UART transmit signal is buffered by U9C to provide a "MIDI Out" connection. The dimmer will not make use of this connection at this time; it is provided only to allow for future expansion.

6.1.7. DMX512 and RS485 interface

DMX512 is simply an RS485 interface at the electrical level so these two control interfaces will share the same hardware. They will differ only in the connector used and protocol details. These interfaces will be collectively referred to as the "RS485 interface."

The RS485 interface is based on a Maxim MAX487 transceiver. The MAX487 converts between TTL level serial signalling and balanced RS485 levels. The MAX487 can both convert TTL to RS485 and convert RS485 to TTL, depending on the state of two control signals, DE and RE. DE enables the transmit portion (TTL to RS485) of the component,

and is connected to the microcontroller. In the current project this signal will not be used because the RS485/DMX512 port is a receiver only. The RE signal enables the receiver portion of the component, and is connected to ground; the receiver is permanently enabled.

Because RS485 and DMX512 specify a balanced, controlled impedance cable, termination is required at the endpoints of the network. To simplify installation and preclude the necessity of separate terminator plugs, the dimmer incorporates software controllable termination. 120 Ohm resistor R32 matches the characteristic impedance of the cabling specified for RS485. Relay K1 connects this resistor across the RS485 bus when the TERMEN signal is high, indicating that termination is desired.

6.2. PCB Layout

Due to safety concerns and the high voltages and currents involved in the dimmer project, a printed circuit board (PCB) will be required. Any sort of prototype construction would likely be too fragile (allowing a user to be exposed to high voltages) or not robust enough to withstand the high currents.

The circuit board places all of the high voltage components along one side of the board and the low voltage components over the rest of the board area. The only components allowed to bridge the two areas are the isolation components (optoisolators and transformer). A red line 0.125 inches wide is drawn across the board indicating the gap between the high voltage and low voltage sections, and to indicate the required clearance distance for safety regulations.

A standard copper thickness on PCBs is 1.5oz (0.5oz of copper and 1.0oz of plating). Using these thicknesses, and allowing a 20 degree Celsius temperature rise, an online trace width calculator [12] suggested a minimum trace width of 235 Mils (0.235 inches). For this reason all of the high current traces on the PCB design are approximately one quarter inch wide (250 Mils) or more.

The PCB layouts appear in Appendix B of this report.

7. Low Level Firmware Design

7.1. Real-Time Phase Angle Control

The primary task of the firmware is the real-time phase angle control. This firmware module must perform three tasks: calculating the delay between zero-crossing and TRIAC turn-on, turning on the triac at the appropriate point in the AC cycle, and turning off all triacs at the AC line zero-crossing.

The microcontroller's 16-bit "Timer/Counter1" is used for timing the turn-on points for each of the four output triacs. The timer's clock source can be set to one of many inputs, including the processor's clock or external inputs. The timer is incremented on each clock edge. The microcontroller's clock will be used as the timer's clock source, so that the timer count will increment at a constant rate. In addition, the zero-crossing edge interrupt will reset the count to zero, resulting in a count value that starts at zero on each AC line zero crossing and increments throughout the AC half-cycle. This is shown in Figure 4.

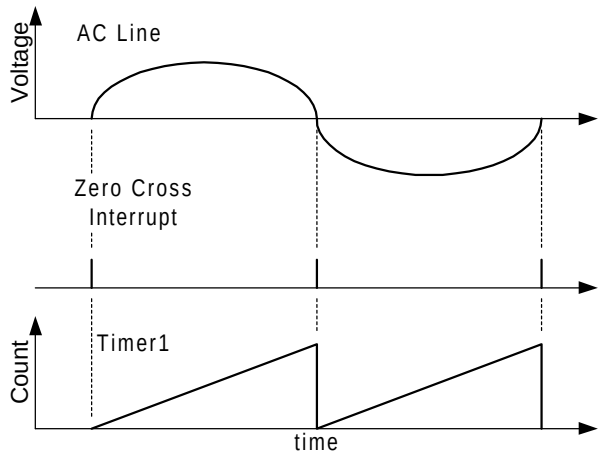

Figure 4: Phase Angle usage of Timer/Counter1

The microcontroller's clock can be prescaled (divided) by certain powers of two between one and 1024 before being routed to Timer/Counter1. Since the dimmer's specifications require it to work with a frequency of 60Hz ± 5%, the maximum period is $T_P = \frac{1}{60Hz - 5\%} = 17.5mS$. Thus, one AC line half-cycle has a duration of 8.77 mS. The 8 MHz clock of the microcontroller has a period of 125 nS, so there are at most 70175 clock cycles per AC half cycle. Timer/Counter1 is a 16 bit timer, so it has a maximum value of 65535. If the microcontroller's clock was not prescaled before being routed to Timer/Counter1, the count value would overflow within the AC half-cycle, which is not acceptable. The next-smallest division ratio is by eight, so that value will be used. The terminal count of Timer/Counter1 (at 60 Hz – 5%) is then $Count_{Term} = 8.77mS \div \frac{1}{8000000MHz/8} = 8772$, and at the nominal 60 Hz the terminal count is 8333.

The timer has two compare registers associated with it, known as Timer/Counter1 Output Compare Register A and B. The microcontroller has two 16-bit comparators, which continually compare the Output Compare Registers with the current value of Timer/Counter1. Several actions can be performed by the hardware on a match, including the generation of an interrupt, or changing the state of a port pin.

The phase angle control module uses the interrupt generation capability. The compare match interrupt service routine (ISR) is responsible for taking any action required at the instant of a match. Initially, the comparison value is set to the count at which the triac

should turn on. An interrupt is generated when the Timer/Counter1 count equals this value, which occurs at time (a) in Figure 5. The Compare ISR would then set a port pin to turn on the triac, and reset the compare value so that the next interrupt occurs just before the end of the half-cycle. At time (b) in Figure 5 the Compare ISR again executes, and clears the port pin to turn off the triac. The compare register is reloaded with the turn-on value, and the cycle repeats.

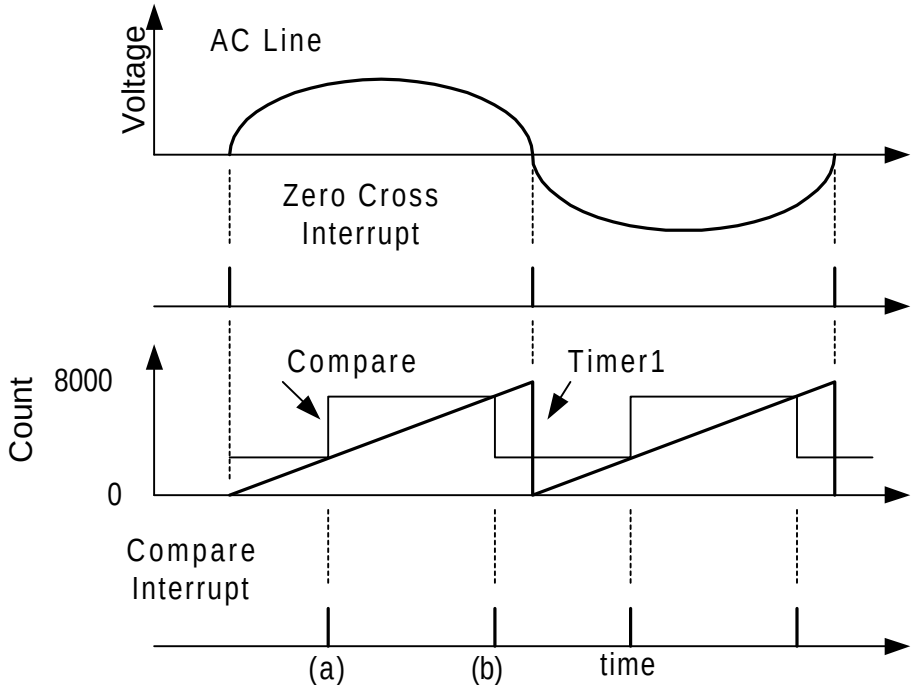

Figure 5: Phase Control Comparator Operation

This procedure can be extended for additional channels. By sorting the turn-on comparison counts, the ISR simply has to switch on the appropriate triac in the sequence and reload the comparison register with the next value in order.

The intensity control structures will be held in global variables. In order to avoid a race condition, all of the ordered compare values must be updated at the beginning of the

cycle. This leads to the zero-cross ISR pseudocode and global variables shown in Figure 6.

```
struct
{
    integer        time
    integer        port_setting
} intensity_data;

intensity_data    new_intensity_data[5]
intensity_data    cur_intensity_data[5]
boolean           g_update_intensity_data = false

ZeroCrossISR()
{
    TimerCounter1_value = 0;
    if (g_update_intensity_data)
    {
        copy (new_intensity_data, cur_intensity_data)
    }
    CompareValue = cur_intensity_data[0].time
}
```

Figure 6: Zero-Cross Interrupt Pseudocode

The compare interrupt then becomes quite simple. On each compare match, the next `port_setting` is written to the triac output port, and the `CompareValue` is set to the next `time` value. If the `port_setting` corresponds to turning off all channels, the `current_step` is set back to 0 in preparation for the next half-cycle. This results in the ISR pseudocode and one additional global variable as shown in Figure 7.

```
integer           current_step = 0
CompareISR()
{
    TriacOutputPort = cur_intensity_data[current_step].port_setting
    if (port_setting == all_channels_off)
    {
        current_step = 0
    } else {
        current_step = current_step + 1
    }
    CompareValue = cur_intensity_data[current_step].time
}
```

Figure 7: Compare Interrupt Pseudocode

The only remaining task for the dimmer firmware module is to compute the `intensity_data` structures based on the desired channel intensities. The minimum intensity occurs when the channel is not turned on at all, while the maximum intensity occurs when the channel is turned on right after the AC zero-crossing. Thus, lower compare values correspond to greater intensities. Given an 8-bit intensity value and a Timer/Counter1 terminal count of 8000, the delay before turning on a given channel is calculated as $TurnOnTime = 8000\left(1 - \frac{intensity}{256}\right)$. The dimmer firmware module provides a `set_levels` function to perform this calculation. Once the delay is calculated for each channel, the delays must be sorted from earliest to latest. In addition, to simplify the comparison ISR, channels having the same intensity (and thus the same delay) should be coalesced into one. The pseudocode appears in Figure 8.

```
integer           intensity[4];

set_levels()
{
    for (i = 0 to 3)
    {
        new_intensity_data[i].time = 8000 – (8000 * intensity[i] / 256)
        new_intensity_data[i].port_setting = set_bit(i)
    }
    sort_data (new_intensity_data[])
    coalesce_data (new_intensity_data[])
    last_element(new_intensity_data[]).time = 8000
    last_element(new_intensity_data[]).port_setting = all_channels_off
}
```

Figure 8: Level Setting Pseudocode

7.2. Debug / Supervisory UART

As indicated the hardware description, an external UART is provided for debug and supervisory tasks. This UART communicates with the microcontroller via the serial peripheral interface (SPI) bus, a synchronous serial bi-directional bus. The Atmel AVR

microcontroller has built-in hardware support for the SPI bus, simplifying the task of communicating with the UART. The SPI hardware generates an interrupt after transmitting and receiving eight bits. The driver for the external UART uses two circular buffers for receiving and transmitting data on the debug / supervisory port. The SPI interrupt initiates a dummy write if no data is available to be transmitted, or sends the appropriate character if any is. As well, if a character was read during the transaction it is enqueued on the receive buffer.

Simplified pseudocode for the UART driver appears in Figure 9. The actual driver is somewhat more complex because the UART uses 16 bit words, requiring two SPI transfers per character.

```
circular_buffer   rs232_tx
circular_buffer   rs232_rx

SPI_ISR()
{
    word   spi_in, spi_out
    spi_in = read (spi_data_register)
    if (rx_flag_is_set (spi_in) AND NOT (is_full (rs232_rx) ) )
    {
        rs232_rx.enqueue ( data_byte (spi_in) )
    }
    if (byte_available (rs232_tx) )
    {
        spi_out = transmit_byte (rs232_tx.dequeue() )
    } else {
        spi_out = dummy_write
    }
    write (spi_data_register, spi_out)
}
```

Figure 9: SPI ISR for UART Communication

The debug / supervisory ISR is not responsible for processing the incoming data; instead, it simply places received data in a buffer and takes data to transmit from a buffer.

Standard functions are provided to interact with these buffers. Their pseudocode appears in Figure 10.

```
// print out a character
rs232_putc(char c)
{
    while (rs232_tx.is_full())
    {
        do_nothing
    }
    rs232_tx.enqueue(c)
}

// read in a character
rs232_getc()
{
    while (rs232_rx.is_empty())
    {
        do_nothing
    }
    return (rs232_rx.dequeue())
}

// is a character available
rs232_hasc()
{
    return (NOT rs232_rx.is_empty())
}
```

Figure 10: Debug / supervisory high level interface pseudocode

7.3. Control UART

The interface to the control UART is rather simple, because the UART is built-in to the microcontroller. It provides an interrupt when a character is received, and when the UART is available for a character to be transmitted. Since both MIDI and DMX512 are unidirectional protocols as far as the current scope of the dimmer project is concerned, the transmit capabilities of the internal UART will not be used.

Unlike the debug / supervisory UART, the receive interrupt is responsible for processing the incoming data according to the protocol associated with the selected control input. The pseudocode for the UART receive interrupt appears in Figure 11, and the two modes of operation are described below.

7.3.1. MIDI mode

The MIDI mode requires the UART to be set to 31250 bits per second. MIDI consists of status bytes and data bytes, which are distinguished by having bit seven set or cleared respectively. The status byte indicates the MIDI channel and message type, while the data byte(s) indicate the data associated with that message. The dimmer responds to key down and key release messages. Key down messages are sent when a note is started, and consist of the note number (each pitch is assigned a unique number) and the velocity, or volume. The dimmer interprets the note number (modulo four) as the dimmer channel, and the velocity as the intensity. This allows standard music composition packages to control the dimmer. MIDI provides no error checking data.

7.3.2. DMX512 / RS485 mode

DMX512 is a unidirectional protocol using a serial interface at a speed of 250 000 bits pre second (or 0.25 Mbps). The RS485 protocol adopted by the dimmer will be identical to the DMX512 protocol, except that the baud rate will be 19 200 bits per second to make it compatible with standard terminal software. The protocol consists of a break signal (continuous active, or space, state on the serial line) followed by a mark (idle) state. A start code is then transmitted which indicates the format of the following data bytes. A start code of zero indicates that 8-bit intensity data follows. In this case, zero or more intensity values are transmitted as single bytes, corresponding to global dimmer channels

zero through n-1 where n is the number of bytes transmitted after the start code. Once the n bytes are transmitted, a new break signal can be transmitted to begin the next update. The DMX512 protocol includes no error checking data.

```
UART_RX_ISR()
{
    character c = UART_rx_byte()

    if (serial_mode == MIDI)
    {
        midi.process(c)
        if (midi.channel == DIMMER_MIDI_CHANNEL)
        {
            if (midi.message == KEY_DOWN)
            {
                intensity[midi.keynumber MOD 4] = midi.velocity
            }
            else if (midi.message == KEY_UP)
            {
                intensity[midi.keynumber MOD 4] = 0
            }
        }
    }
    else if (serial_mode == DMX512 OR serial_mode == RS485)
    {
        if (UART_rx_break_detected)
        {
            dmx_active = FALSE
            dmx_channel = -1
        }
        else
        {
            if (dmx_channel = -1)
            {
                if (c == 0)
                {
                    dmx_active = TRUE
                }
            }
        }
        dimmer_channel = dmx_channel – DMX_START_CHANNEL
        if (dmx_active AND 0 <= dimmer_channel <= 3)
        {
            intensity[dimmer_channel] = c
        }
        dmx_channel = dmx_channel + 1
    }
}
```

Figure 11: Control UART receive ISR pseudocode

7.4. Supervisory Control

The final aspect of the dimmer firmware is the supervisory control code. It is responsible for initializing the hardware, allowing the user to choose and configure the desired control input, configuring the hardware, and setting intensity values for manual testing or direct computer control.

After initializing and displaying a menu, the control firmware waits in an infinite loop, polling for either a received character or a zero-crossing. If a zero-crossing is detected, the dimmer firmware's `set_levels` function is called to update and sort the intensity data.

If a character is detected, it is read in and processed according to the simple menu scheme. Sending a "0" through "3" indicates that the intensity value for that given channel will follow as a two-digit hex value. Other single characters are assigned configuration tasks as indicated in the pseudocode of Figure 12. Sending a "m" or a "d" followed by a two-digit hex value configures the MIDI channel or DMX starting channel respectively.

```
main ()
{
    initialize_hardware()
    display_menu_and_config()
    rs232_putc (‘>’)
    do_forever
    {
        if (rs232_hasc())
        {
            c = rs232_getc()
            if (c == ‘t’)
            {
                toggle_termination()
            }
            else if (c == ‘c’)
            {
                select_next_control_input()
            }
            else if (c == ‘0’ to ‘3’)
            {
                intensity[c] = get_hex_value()
            }
            else if (c == ‘d’)
            {
                DMX_START_CHANNEL = get_hex_value()
            }
            else if (c == ‘m’)
            {
                MIDI_CHANNEL = get_hex_value()
            }
            else if (c == ‘?’)
            {
                display_menu_and_config()
            }
            rs232_putc (‘>’)
        }
        if (zero_cross_occurred())
        {
            set_levels()
        }
    }
}
```

Figure 12: Supervisory Control Pseudocode

8. Assembly and Testing

8.1. Component Acquisition

The bill-of-materials for the dimmer circuit assembly along with the bill-of-materials for the complete project appear in appendix C. Each component indicates the source, and the price (estimated and actual) if it was not available as a sample. Most of the components required for the dimmer were obtained as samples from various manufacturers or their representatives. The project designer greatly appreciates the support of these companies, which are listed in Table 3 along with their contributions.

Table 3: Project sponsors

Company	Contribution
Atmel Clarsand Ltd.	Microcontroller samples
Cisco Systems (Waterloo)	Lab space and equipment for surface mount soldering
Fairchild Semiconductor Candian Source Corporation	Optocoupler samples
Keystone Electronics EMX Enterprises	Screw terminals
Kingbright LEDs	Surface mount LEDs
Maxim	MAX3100 UART MAX202 level shifter MAX487 RS485 transceiver
Mill-Max	Surface mount PLCC sockets
ON Semiconductor	Logic IC samples
Philips Semiconductors Tech-Trek Limited	Triac samples Logic IC samples
Samtec	Surface mount 0.1” headers

8.2. PCB Fabrication

It was originally intended that the printed circuit board (PCB) for the project would be fabricated by the Electrical and Computer Engineering department’s PCB milling machine. However, attempts to obtain access to the machine to produce the circuit board

proved unsuccessful and an external quick turnaround PCB fabrication company (Alberta Printed Circuits) was employed instead. This dramatically increased the cost of this portion of the project. However, it resulted in having a professionally produced circuit board in a short period of time. A high quality board is important, considering the high currents and voltages involved in this project.

8.3. Prototype Assembly

The prototype circuit board was hand assembled at Cisco Systems of Waterloo and at the author's home. The small surface-mount circuit components (such as some integrated circuits, resistors and capacitors) were soldered at Cisco using the advanced soldering equipment there. Larger components such as the triacs and other through-hole parts were assembled at home.

8.4. Hardware Errors

During assembly and subsequent initial testing, several hardware design errors became apparent. These were all schematic-entry issues, and not higher level design issues. Each of the errors is described below.

8.4.1. UART crystal ground connection

Due to the use of an incorrect ground symbol, the ground for the UART's crystal and associated capacitors (Y2, C13 and C14) was not connected to the global circuit ground. This was corrected by adding a jumper wire to the proper ground, as shown in Figure 13.

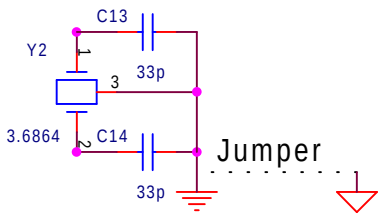

Figure 13: UART crystal ground problem

8.4.2. UART pinout

Initially a surface mount UART was to be used as the debug UART. However, this part could not be obtained before the circuit board was sent out for fabrication, so a through hole dual inline package (DIP) part was substituted. Unfortunately, the surface mount part includes two extra pins, which are not to be connected externally. When the part was changed to a DIP part, these pins remained and the actual UART had fewer pins than its socket. In addition, the RX and TX pins in the component library were reversed. These issues were resolved by cutting the incorrect copper PCB traces with a knife, and soldering jumper wires on to make the correct connections. The incorrect and correct pinouts are shown in Figure 14.

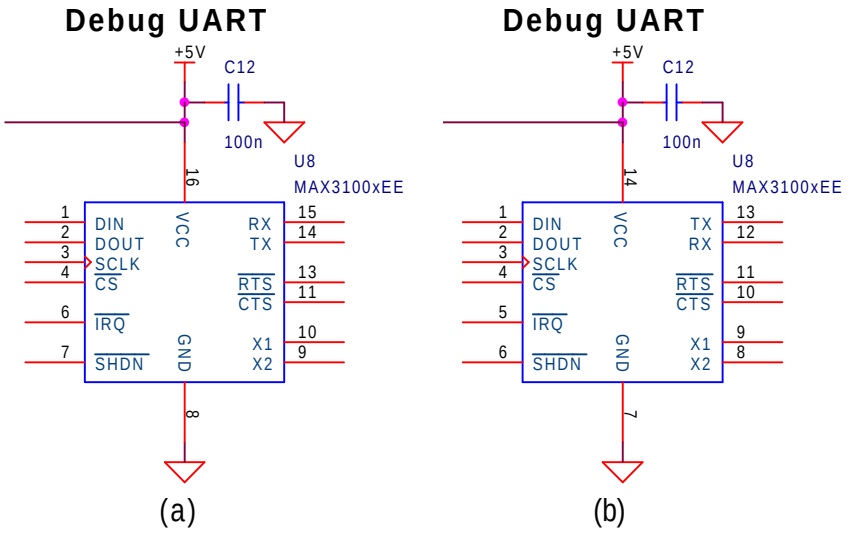

Figure 14: Debug UART pinout - (a) correct (b) incorrect

8.4.3. Triac pinout

Due to a misinterpretation of the triac's datasheet, their pinout was also incorrect, as pins 1 and 2 were swapped as shown in Figure 15.

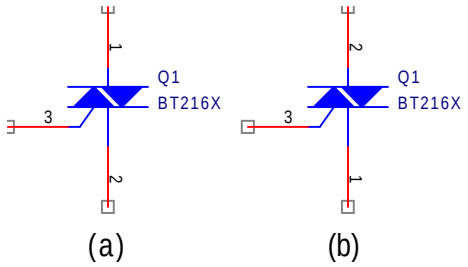

Figure 15: Triac pinout - (a) incorrect (b) correct

The resolution for this issue was to carefully bend the pins, such that the triac's actual pin 1 went into the pin 2 hole in the circuit board, and pin 2 went into the pin 1 hole, as indicated in Figure 16.

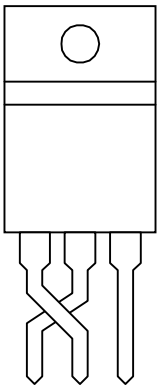

Figure 16: Triac modification to correct pinout

8.4.4. Logic gate U9C

Logic gate U9C in the MIDI output section of the schematic was placed backwards, with the input and output connections reversed. Figure 17 shows both the incorrect connections as implemented in the PCB and the desired connection. To fix this problem, pins 8 and 9 need to be lifted from their associated copper pads on the PCB, and small jumper wires soldered in to make the correct connection.

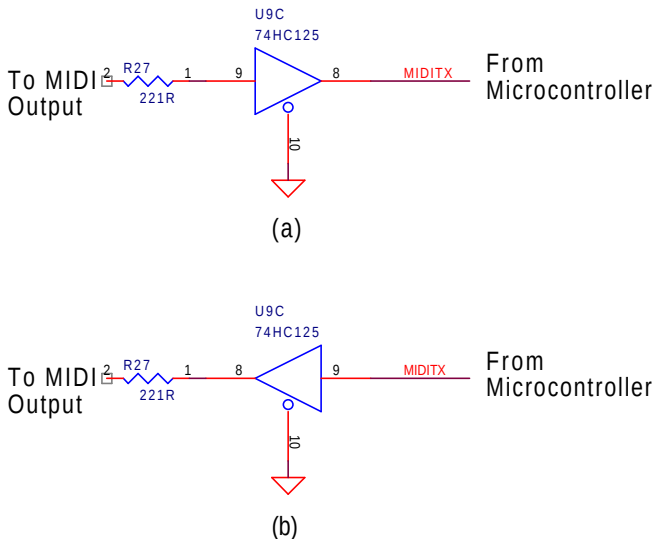

Figure 17: Logic gate U9C - (a) incorrect (b) correct

8.5. Debug UART Testing

After the hardware issues were corrected, the debug UART driver was implemented and tested. It was important to have the debug UART operational at an early stage in order to facilitate the completion of the rest of the design.

Initially the debug UART worked in the transmit direction but not the receive direction. The hardware connects the UART's IRQ output to one of the general purpose external interrupts on the microcontroller; the original intent was that the ISR for the external interrupt could initiate a receive or transmit sequence only when the UART required service. However, it was found that the UART would keep the IRQ asserted even after the byte was read from it. In order to get the debug UART working quickly and reliably, the external interrupt was disabled, and the SPI interrupt rewritten so that as soon as one transaction with the UART completes another one is started.

With this new driver, the UART seemed reliable. A short test program was written and linked with the UART driver that would echo back received characters after a variable delay. This worked properly indicating that the transmit and receive functionality worked and that the debug UART buffers were operational.

8.6. Dimmer Testing

The next dimmer module to be verified was the output drivers and control section. To test the dimmer functionality a small program was written to receive four sets of hex values over the debug UART and then set the intensity values for each of the four channels. A desk lamp was connected to each channel in turn and it was verified that the brightness monotonically increased with increasing values.

The next step in the testing involved connecting a 100W load to the first dimmer channel, and applying incrementing values to the dimmer. After each new value was set, the voltage at that channel's output was measured with a Fluke model 12 multimeter. In additionl, the voltage supplied to the dimmer, measured at the outlet it was plugged into, was 114V. The measured data is recorded in Table 4

Table 4: Dimmer voltage test results

Setting	Percent Full Scale	Measured Voltage	Percent Full Scale
0x00	0.0%	0	0.0%
0x10	6.3%	1.923	1.7%
0x20	12.5%	6.72	5.9%
0x30	18.8%	12.96	11.4%
0x40	25.1%	20.8	18.2%
0x50	31.4%	30.06	26.4%
0x60	37.6%	40.47	35.5%
0x70	43.9%	51.2	44.9%
0x80	50.2%	61.5	53.9%
0x90	56.5%	71.5	62.7%
0xA0	62.7%	81.1	71.1%
0xB0	69.0%	90.1	79.0%
0xC0	75.3%	98	86.0%
0xD0	81.6%	104.5	91.7%
0xE0	87.8%	109.2	95.8%
0xF0	94.1%	112.3	98.5%

The results are presented graphically in Figure 18. Note the similarity between the shape in the graph, and the theoretical results from Figure 2.

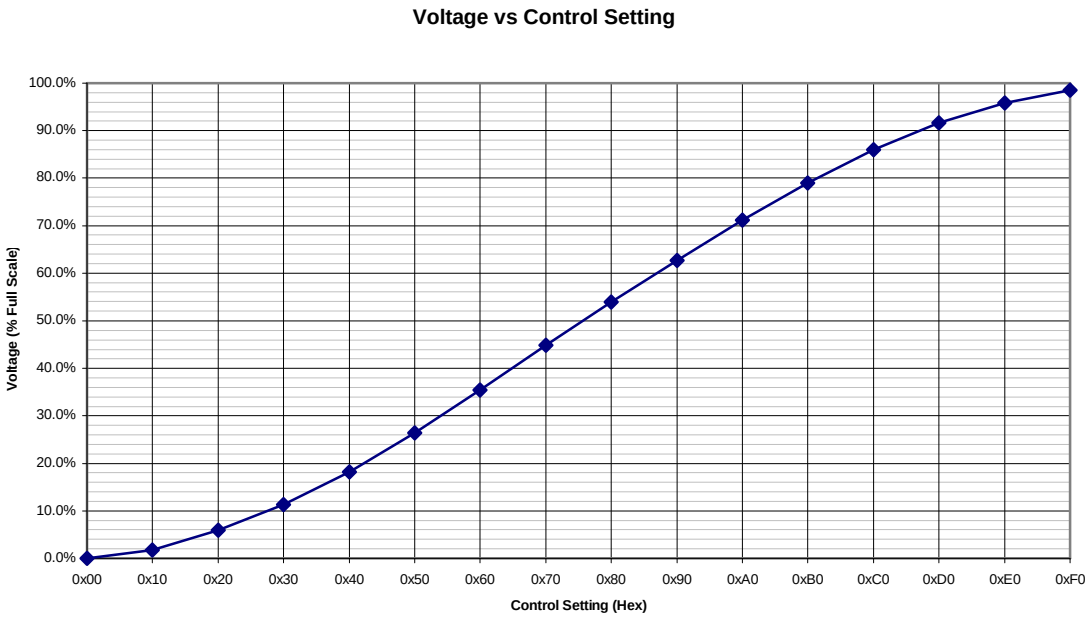

Figure 18: Output voltage vs. control input

Several loads were connected to the dimmer in order to verify its current handling capability. These loads included a 500W halogen lamp, two 60W desk lamps and approximately 200W of assorted reading lamps and Christmas lights, for a total of 820W. This load was attached to one channel and the dimmer was set to 0%, 50% and 100% of full brightness and operated at each setting for 20 minutes. No undue heating or other problems were observed.

Unfortunately fuses and noise-suppression inductors could not be sourced in time for the completion of the project. A case has been constructed for the project that includes a hole for a fuseholder to be added in the future. The dimmer did not cause undue noise in nearby audio equipment even when operating with 1000W loads. However, suitable inductors should be added and tested in the future to ensure the dimmer will cause no problems.

8.7. MIDI Input Testing

In order to test the MIDI input, the four notes depicted in Figure 19 were entered into Midisoft Studio for Windows, a music sequencing program. Windows MIDI mapper was configured to output on the computer's external MIDI port, which was connected to the dimmer via a standard interface cable. In MIDI, middle C is note number 64. Since there are 12 notes per octave, every C note number modulo four will equal zero. As expected, when the measure shown was "played," each of channels one to four turned on and then off in order.

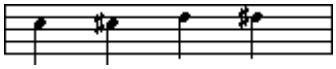

Figure 19: MIDI lighting test measure

8.8. RS485 and DMX512 Input Testing

The RS485 control input was tested by manually simulating the DMX512 format data in a Windows terminal emulator, set to 19 200 bits per second. Tera Term is a freely available terminal emulator that allows a break signal to be sent (by pressing Alt-B), in addition to arbitrary characters. In order to mimic the DMX512 protocol, the keystrokes as shown in Table 1 were entered in Tera Term.

Table 5: Keystrokes to simulate DMX512 protocol

Protocol Element	Value	Keystroke
Break		<Alt-B>
Null start code	0	<Alt-000>
Dimmer channel 1 intensity	65	A
Dimmer channel 2 intensity	97	a
Dimmer channel 3 intensity	90	Z
Dimmer channel 4 intensity	122	z

When this data was entered, the dimmer channels were set to voltages consistent with the expected values.

Unfortunately access to proper DMX512 transmitting equipment could not be obtained, so DMX512 protocol data at 250 000 bits per second could not be tested. However, the protocol appears to function properly at 19 200 bits per second.

9. Conclusions

It was concluded that each channel of the dimmer output can take on any voltage between zero and 98% of full scale, meeting the specifications of at least 5% to 95%.

It was concluded that each channel of the dimmer can independently handle a 1000W load as required by the specifications.

It was concluded that the debug / supervisory RS232 port on the dimmer works as expected, and can be used to configure the dimmer and set channel intensities.

It was concluded that the MIDI input works as expected and MIDI note data can set the channel intensities according to MIDI key number and MIDI velocity.

It was concluded that the RS485 port works at 19 200 bits per second and decodes DMX512 data into channel intensities, however real DMX512 data at 250 000 bits per second was not tested.

10. Recommendations

It is recommended that the DMX512 input be tested at the full 250 000 bit per second rate. This will probably require the assistance of a group possessing a DMX512 transmitter, such as the Drama department at the University of Waterloo.

It is recommended that the hardware issues and pinout problems addressed in section 8.4 be corrected in a new version of the printed circuit board.

It is recommended that fusing and output filters be added to each dimmer channel.

11. References

[1] Quail Electronics, *NEMA Plug and Receptacle Configurations*, http://www.quail.com/locator/nema.htm (Current Feb 2, 2001)

[2] The Centre in the Square, *Lighting Instrument Inventory*, http://www.centre-square.com/electrical.htm#Instruments (Current Feb 2, 2001)

[3] Teccor Inc., *Phase Control Using Thyristors*, Application Note AN1003, http://www.teccor.com/thyristor/an1003.pdf (Current Feb 6, 2001)

[4] National Semiconductor, *Summary of Well Known Interface Standards*, Application Note AN-216, http://www.national.com/an/AN/AN-216.pdf (Current Feb 6, 2001)

[5] Coesel, N., *Generic MIDI interface*, http://www.midiweb.com/hww/midi/uc_midi.txt (Current Feb 6, 2001)

[6] Coesel, N., *Serial Card MIDI Interface*, http://www.midiweb.com/hww/midi/ser_midi.gif (Current Feb 6, 2001)

[7] United States Institute for Theatre Technology Inc., *DMX512 Home Page*, http://www.usitt.org/DMX/DMX512.htm (Current Feb 6, 2001)

[8] National Semiconductor, *Transceivers and Repeaters Meeting the EIA RS-485 Interface Standard*, Application Note AN-409, http://www.national.com/an/AN/AN-409.pdf (Current Feb 6, 2001)

[9] Finney, D., The Power Thyristor and its Applications, p. 35, McGraw-Hill Book Company Limited, Toronto, 1980

[10] National Semiconductor, *LM340/LM78Mxx Series 3-Terminal Positive Regulators*, http://www.national.com/ds/LM/LM340.pdf (Current Mar 7, 2001)

[11] Tamura Corporation, *Tamura Home Page*, http://www.tamuracorp.com/ (Current Mar 7, 2001)

[12] Suppanz, B., *PCB Trace Width Calculator*, http://www.geocities.com/CapeCanaveral/Lab/9643/TraceWidth.htm (Current Mar 7, 2001)

Appendix A: Schematics

Pages:

1. Power supply, zero crossing detector and output drivers
2. Microcontroller and debug UART
3. Control input interface components and level shifters

Insert schematic page A-1

Insert schematic page A-2

Insert schematic page A-3

Appendix B: PC Board Layouts

Pages:

1. Copper layout, top layer
2. Copper layout, bottom layer
3. Drill drawing
4. Component placement guide

Insert PCB B-1

Insert PCB B-2

Insert PCB B-3

Insert PCB B-4

Appendix C: Annotated Bill of Materials

Pages:

1. Circuit board assembly
2. Circuit board assembly (cont'd)
3. Complete dimmer assembly

Qty	Reference Designators	Description	MFG & Part No	Supplier	Est.	Price
	Capacitors					
1	C1	1000u lytic		stock	$0.00	$0.00
18	C2,C3,C5,C6,C7,C8,C9,C12, C15,C17,C18,C19,C20,C21, C22,C23,C24,C25	0805 100n		stock	$0.00	$0.00
1	C4	47u lytic		stock	$0.00	$0.00
4	C10,C11,C13,C14	0805 33p		stock	$0.00	$0.00
1	C16	1u 10V lytic		stock	$0.00	$0.00
	Resistors					
4	R1,R4,R7,R13	150R 1/4W TH Axial		Sayal	$0.00	$0.20
4	R2,R5,R8,R14	330R		stock	$0.00	$0.00
4	R3,R6,R9,R15	47R 1/2W TH Axial		Sayal	$0.00	$0.40
2	R10,R12	10K 1/2W TH Axial		Sayal	$0.00	$0.10
3	R11,R24,R33	2k21		stock	$0.00	$0.00
9	R16,R17,R18,R19,R20,R28, R29,R30,R31	500R		stock	$0.00	$0.00
1	R21	100K		stock	$0.00	$0.00
5	R22,R23,R25,R26,R27	221R		stock	$0.00	$0.00
1	R32	120R		stock	$0.00	$0.00
1	R34	0R Jumper		stock	$0.00	$0.00
	Diodes & Discretes					
8	D1,D2,D3,D4,D5,D7,D8,D16	Power Diode	1N4005	stock	$0.00	$0.00
4	D6,D14,D15,D21	Signal Diode	1N4148	stock	$0.00	$0.00
9	D9,D10,D11,D12,D13,D17, D18,D19,D20	Surface Mount LED		kingbright	$0.00	$0.00
4	Q1,Q2,Q3,Q4	Triac	BT216X	Tech-Trek	$0.00	$0.00
1	Q5	Small signal NPN	2N3904	stock	$0.00	$0.00
	Connectors & Misc					
6	J1,J2,J3,J4,J5,J6	Screw Terminal	Keystone 7693	EMX	$0.00	$0.00
4	J7,J8,J9,J10	Surface Mount Header	Samtec TSM-105-01-S-DV	Samtec	$0.00	$0.00
1	K1	Relay DPDT 5V coil DIP	unknown	Supremetronic	$2.50	$0.00
1	T1	Power Transformer	Tamura 3FS-312	Longman Sales	$0.00	$0.00
1	Y1	8MHz crystal	8MHz	Sayal	$0.00	$1.00
1	Y2	3.6864MHz crystal	3.6864MHZ	Sayal	$0.00	$1.00

Qty	Reference Designators	Description	MFG & Part No	Supplier	Est.	Price
	Integrated Circuits					
4	U1,U2,U4,U6	Optocoupler	Fairchild MOC3011	CSC	$0.00	$0.00
1	U3	Regulator	LM7805CT	stock	$0.00	$0.00
2	U5,U13	Optocoupler	Fairchild 6N138	CSC	$0.00	$0.00
1	U7	Microcontroller	Atmel AT90S8515	Clarsand	$0.00	$0.00
1	U8	SPI UART	MAX3100xEE	Maxim	$0.00	$0.00
1	U9	Quad buffer	ON Semi 74VHC125D	ON Semi	$0.00	$0.00
1	U10	RS232 Level Shifter	MAX202	Maxim	$0.00	$0.00
1	U11	Dual 4-1 Mux	ON Semi 74VHC157D	ON Semi	$0.00	$0.00
1	U12	RS485 Level Shifter	MAX487	Maxim	$0.00	$0.00
	Misc					
1	X1	PLCC Socket	Mill-Max 540-99-044-17-400000	Mill-Max	$0.00	$0.00
				Total Component Expense:	$2.50	$2.70

NOTE: Transformer 3FS-312 from Longman Sales did not arrive in time for the project to be completed; a "wall adaptor" power supply was substituted instead

Qty	Description	MFG / Part No	Supplier	Est.	Price
	Board				
1	PC Board	Custom	Alberta Printed Circuits	$0.00	$125.00
1	PC Board components	Per BOM pg 1		$2.50	$2.70
	Assembly				
1	Power cord		Sayal	$1.00	$1.00
5	1m length of wire (5 colours)	T-90 14 gauge	Home Depot	$2.50	$2.00
12	Crimp-on screw lugs		Home Depot	$2.40	$1.20
2	Standard receptacles		Home Depot	$2.00	$1.00
1	Electrical box		Home Depot	$2.00	$4.00
1	Electrical faceplate		Home Depot	$0.50	$1.50
	Cabinet				
1	Mounting screws	4-40 by 1.5"	Student Machine Shop	$0.00	$0.00
1	Aluminum, bottom of case	16 ga 8" x 20"	Main Shop	$0.00	$0.00
1	Aluminum, top of case	18 ga 8" x 14"	Main Shop	$0.00	$0.00
			Total Component Expense:	$12.90	$138.40
			Total Component Expense (Excluding PCB):	$12.90	$13.40