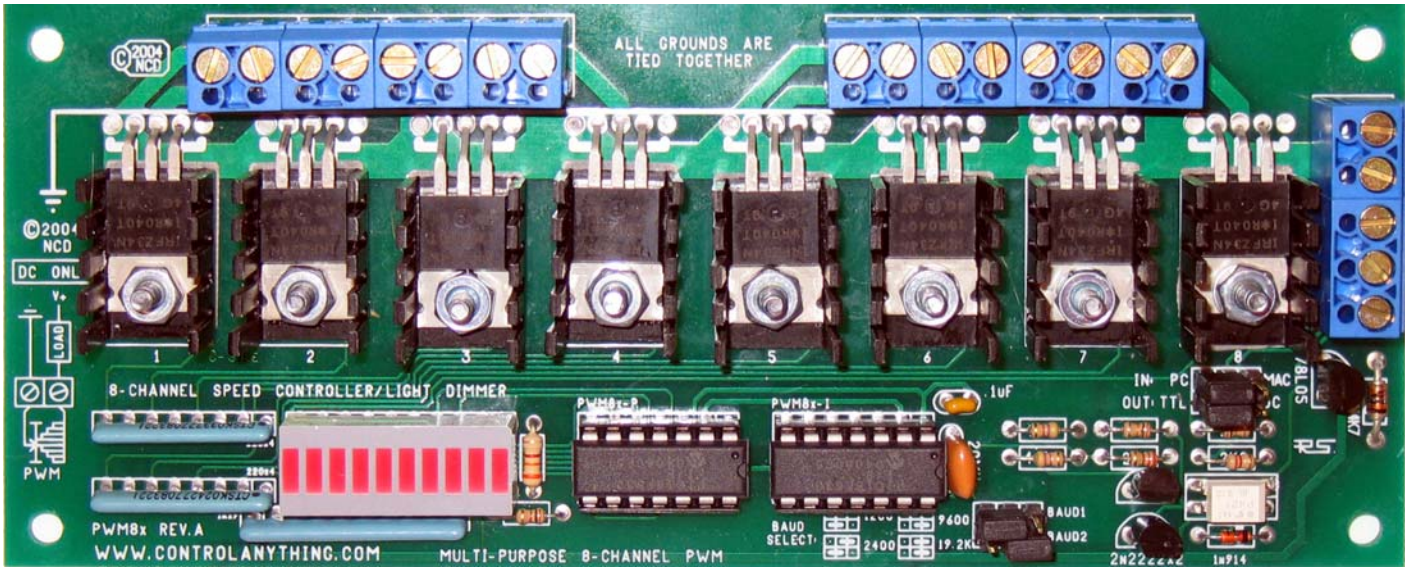
PWM8x

Programmable 8-Channel Pulse Width Modulation Controller



PWM85 Shown Above is an Ideal Solution for many computer controlled motor speed control and light dimming applications.

There are some computer control applications that are a little more demanding than simple on/off control signals. If you have ever had an application where you need to safely control the speed of DC motors or the brightness of lights, then our PWM8x series controllers may be exactly what you are looking for. The PWM8x series controllers are built around a custom pulse width modulation processor. This new processor is dedicated to providing 256 levels of pulse width modulation for 8 individual channels. Coupled with powerful fets, the PWM8x is capable of controlling 256 levels of brightness for 8 individual light bulbs under computer control. Rated for use with resistive and inductive loads, the PWM8x is also a very suitable solution for DC motor speed control applications.

256 Levels of Speed Control for 8 DC Motors
256 Levels of Brightness Control for 8 DC Lights
Programmable Level for Each Channel Upon Power-Up
8 On-board Hexfets with Heat Sinks (PWM83/PWM85 only)
Dedicated Serial Interface Processor
Dedicated 8-Channel PWM Processor
1200, 2400, 9600, 19.2K Baud Rate
Optoisolated RS-232 Input
Fast On Power-up Mode
Slow Start Power-Up Mode
Program Each Channel Separately
Program All Channels Simultaneously (to same level)
7-15VDC Operation
150 ma Maximum Power Consumption

WARNING: RISK OF ELECTICAL SHOCK WHILE DRIVING INDUCTIVE LOADS. HEAT SINKS MAY BECOME VERY HOT UNDER NORMAL OPERATION. SERIOUS BURN RISK!

Important Note from the Designer:

We have never developed a product quite like the PWM8x series controllers. This product serves as a test market product to determine its usefulness in computer control applications for our small niche of customers. If this product proves it's usefulness, we will expand the line to include more features, higher resolution, and expanded versatility...the good new is, most of these upgrades can be done in firmware, so your PWM8x will never become obsolete.

We are hoping to see a lot of response from our customers on this product. Particularly, we are seeking feedback on features that may be needed, failures that may occur, and stories of stretched limits. Like any product we manufacture, we intend to be very responsive with any electrical failures that may occur. We have been very fortunate to work with a number of very interesting customers who are using our products for some very extraordinary applications. We like to maintain the attitude that our customers are people we work with to make our products better. And about 99.9% of the time, our customers take the very same attitude towards us. This mutual respect results in better products that become more refined, more versatile, and more powerful as the years pass.

So please, tell us anything good, anything bad, or anything you want about your experience with this product. We need to hear from you on this product more than any other, as it may significantly define our future line of computer control products. All comments are welcome: Ryan Sheldon, ryan@controlanything.com.

FREE REPLACEMENT PARTS OFFER TILL JULY, 2006

We found it difficult to provide electrical ratings for the PWM8x series controllers, mostly because of the differences between inductive and non-inductive loads. Building a controller that is capable of universally working in every application is difficult, if not impossible. One of the biggest obstacles to overcome is the simple fact that we have no idea how our customers will end up using this product. While it's possibilities are endless, it's limits are theoretical numbers that may or may not apply to your application.

With a totally new product comes a totally new approach to supporting this design: We encourage our customers to use this product, use it hard, and damage it if you need to. Test it to it's fullest limits. Tell us what you did and what went wrong, we will log it into our records for future design purposes, please provide as much detail as possible. We will send you any replacement component(s) that reside on the PWM8x series controllers that fails under any circumstances (with the only exception being you may not exceed the rated voltages), which may be limited to 5 replacements per customer, until July, 2006, provided you have provided us with useful feedback indicating the electrical circumstances that caused component failure. Our customers will be authorized to conduct repairs and modifications without breech of warranty until July, 2006. Please report any problems to ryan@controlanything.com.

Theory of Operation

The PWM8x has two dedicated processors: The Interface processor is dedicated to handling serial commands while the PWM processor is dedicated to keeping the pulse stream as steady as possible. When a command is received by the Interface processor, the PWM chip is flagged. The new command is copied from the Interface processor to the PWM chip and the pulse stream starts back up again. The update occurs in just a few microseconds, and is undetectable to most real world control applications. The 8 signals generated by the PWM chip are directly connected to the gates of the 8 N-Channel Hexfets. The Hexfets connect your external device to ground, then disconnects at very high speed. You control the speed of the connection/disconnection, which in turn controls the speed of a motor or the brightness of an incandescent lamp.

Application and Usage Notes

The PWM8x can be used to control many types of devices with different voltage requirements at one time. The only rules being, all devices must operate from a DC voltage and all devices must be able to share a common ground. For instance, a 5VDC motor can be connected to channel 1, a 12VDC motor on channel 2, a 24VDC incandescent lamp on channel 3, a 12VDC incandescent lamp on channel 4, etc.

Dangerous Voltages

Devices such as motors, solenoids, or just about anything that contains a coils is called an inductive load. Inductive loads have very different electrical characteristics than resistive loads. Inductive loads generate a huge spike of electricity every time power is removed from the coil. This spike of electricity is carried all the way back to the heat sinks. Touching the heat sinks of the PWM8x while driving these kinds of loads can result in a surprisingly painful shock. In addition, inductive loads tend to produce more heat than resistive loads, posing a very serious burn risk. As a general rule, never touch the heat sinks while in operation and always be cautions of potential burns.

Heat

Hexfets tend to work more efficiently when they are on or off, but turning them on and off at a high rate of speed can produce a lot of heat, especially at lower speed settings using inductive loads. The heat generated by a Hexfet can be unbearably hot (too hot to safely touch). We have seen Hexfets become so hot, it melted enough solder off the connections, causing it to disconnect itself. We have also seen Hexfets get so hot, our board started to smoke. In most cases, the Hexfet was fine and nothing was damaged.

Device Failure

When a Hexfet fails, the Drain and the Source leads on the Hexfet typically become fused together. In this case, your motor would appear to be on all the time, or an incandescent light bulb will remain on, despite your PWM settings. The output of the PWM controller chip is short circuit protected at minimum. We consider your computer to be the most important thing to protect, so we use an Optoisolator to electrically separate the computer connection from the rest of the circuit. You can usually tell when a Hexfet is about to fail because the Hexfets will start to smoke. Smoke does not mean they have failed however, we have seen a lot of them smoke, and once cooled down, they work fine again.

Partial Failure

When a Hexfet fails partially, it should be replaced. A partial failure is caused by overheating, usually because too much current was pulled through the drain lead of the Hexfet. A Hexfet that has overheated is likely to be de-rated, meaning it no longer functions properly across the entire rated temperature range. It is easy to detect a partially failed Hexfet: They work fine for a while, then once they get up to temperature, they fuse the Drain and Source together. Once cooled down, they begin operating properly again.

Operational Limits

The ratings we have assigned to the Hexfets far under-rate the manufactures specifications. Hexfet manufacturers tend to assume you can dissipate the full TO-220 package rating of up to 50 Watts….which is simply not possible on a low-cost controller such as the PWM8x. The heat sinks on the PWM8x controllers are rated to dissipate only 2 Watts, the TO-220 package itself is capable of dissipating 1 Watt. Because of the conservative wattage ratting of the heat sinks, it is important to use Hexfets with a low on resistance. These Hexfets generate less heat to begin with, so a 2 Watt heat sink goes much further in terms of the number of amps that can be pulled through the drain.

The PWM81 uses the IRLD110 Hexfet with a Wattage Rating of 1.3 Watts and a Maximum Voltage Rating of 100 VDC, and a Maximum current of 1 Amp at 5VDC at 25º C. The PWM81 is ideal for automotive lighting applications (except headlights), controlling external relays, small solenoids, and very small DC motors.

The PWM83 uses the IRFZ34N Hexfet with a Wattage Rating of 68 Watts and a Maximum Voltage Rating of 55 VDC, and a Maximum current of 29 Amps at 10VDC at 25º C. The PWM82 is ideal for automotive lighting applications (except headlights), controlling external relays, solenoids, and DC motors used in robotic applications.

The PWM85 uses the IRL640 Hexfet with a Wattage Rating of 68 Watts and a Maximum Voltage Rating of 200 VDC, and a Maximum current of 17 Amps at 5VDC at 25º C.

The PWM85 uses the IRL640 Hexfet with a Wattage Rating of 68 Watts and a Maximum Voltage Rating of 200 VDC, and a Maximum current of 17 Amps at 5VDC at 25º C. The PWM85 is ideal for most automotive lighting applications, solenoids, contactors, and DC motors, including DC motor used in drive systems of small robots.

Sending Serial Commands

The PWM8x is capable of sending and receiving data via RS-232 serial communications. The PWM8x is compatible with just about any computer or microcontroller ever produced, including the Macintosh, Amiga, Basic Stamp, and of course, Windows & DOS based machines.

Regardless of the system you are using, you will need access to a programming language that supports program control of the serial port on your system.

A terminal program is not suitable for controlling the PWM8x. Commands should be sent using ASCII character codes 0-255 rather than ASCII characters (A, B, C etc.). See "ASCII Codes vs. Characters" on this page.

Most systems require you to open the appropriate serial port (COM port) prior to sending or receiving data.

Because there are so many different ways to send and receive data from various languages on various platforms, we will provide generic instructions that can be easily converted to your favorite language.

For example, if this manual says "Send ASCII 254", the user will need to translate this instruction into a command that is capable of sending ASCII character code 254.

To Send ASCII 254 from Visual Basic, you will use the following line:

MSComm1.Output = Chr$(254)

In Qbasic, you can send ASCII 254 using the following line of code:

Print #1, Chr$(254);

Note that sending ASCII character code 254 is NOT the same as sending ASCII characters 2, 5, and 4 from a terminal program. Typing 2, 5, and 4 on the keyboard will transmit three ASCII character codes.

In your program, you may want to ask the PWM8x for data such as the E3C device number. If so, your programming language will support commands for reading data from the serial port.

For your convenience, we have provided several programming examples in Visual Basic 6 for controlling the PWM8x. These examples should greatly speed development time. You may want to visit **www.controleverything.com** for the latest software and programming examples.

Programming examples for the PWM8x are much more extensive for Visual Basic 6 users than for any other programming language. If you are not a VB programmer, you may consider looking at the VB6 source code, as it is easily translated into other popular languages.

Regardless of your programming background, the provided Visual Basic 6 source code is very easy to understand and will likely resolve any communication questions you may have. VB6 programming examples may be viewed in any text editor.

ASCII Codes vs. Characters

The differences between ASCII codes and ASCII characters tend to generate a lot of confusion among first-time RS-232 programmers. It is important to understand that a computer only works with numbers. With regard to RS-232 data, the computer is only capable of sending and receiving numbers from 0 to 255.

What confuses people is the simple idea that the numbers 0 to 255 are assigned letters. For instance, the number 65 represents the letter A. The number 66 represents the letter B. Every character (including numbers and punctuation) is assigned a numeric value. This standard of assignments is called ASCII, and is a universal standard adopted by all computers with an RS-232 serial port.

ASCII characters codes can be clearly defined as numbers from 0 to 255.

ASCII characters however are best defined as letters, A, B, C, D, as well as punctuation, !@#$%, and even the numbers 0-9.

Virtually all programming languages permit you to send ASCII in the form of letters or numbers. If you wanted to send the word "Hello" out the serial port, it is much easier to send the letters H, e, l, l, and o than it is to send the ASCII character codes that represent each letter.

For the purposes of controlling NCD devices however, it is much easier to build a numeric command set. Especially when communicating to devices where you want to speak to lots of outputs (which are numbered), inputs (which are also numbered), or control specific devices using their device number (from 0 to 255).

Put simply, it is easier to control NCD devices using ASCII character codes 0 to 255 than it is to use ASCII characters A, B, C, D, etc.

Because terminal programs are ASCII character based, it may be difficult to generate the proper series of keystrokes that would be necessary to activate a particular function. Therefore, they are not suitable for controlling NCD devices. In a real world control application, a terminal program would not likely be used to control NCD devices anyway. Therefore, a programming language that supports the transmission and reception of ASCII character codes 0 to 255 is highly recommended.

The E3C Command Set: Software Control of Multiple NCD Devices

The E3C command set allows you to control up to 256 NCD devices from a single serial port. It is OK to mix different types of devices, as long as the devices are E3C compliant. The PWM8x supports the full set of E3C commands, plus a set of extended commands for storing and recalling the device number.

How does E3C Work?
First of all, each device must be assigned a device number from 0 to 255. The PWM8x must be programmed with a device number, which is accomplished using the "Store Device Number" command shown below.

E3C stands for Enabled 3-Wire Communication. Put simply, when you first power up your computer and all the devices attached to the serial port, all devices will respond to your commands.

Using the E3C command set, you can specify which devices will listen and which devices will ignore your commands. Note that E3C commands are never ignored by any device, regardless of the commands you send to the controller.

The number to the left of each command indicates the ASCII character code that must be sent to issue the command. All commands must be preceded with ASCII character code 254 to place the device in command mode. See examples at right.

The E3C Command Set

248 Enable All Devices:
Tells all devices to respond to your commands.

249 Disable All Devices:
Tells all devices to ignore your commands.

250 Enable a Selected Device:
Tells a specific device to listen to your commands.

251 Disable Selected Device:
Tells a specific device to ignore your commands.

252 Enable Selected Device Only:
Tells a specific device to listen to your commands, all other devices will ignore your commands.

253 Disable a Selected Device Only:
Tells a specific device to ignore your commands, all others will listen.

Extended E3C Commands

The PWM8x supports two additional E3C commands which should only be used when a single device is attached to your serial port. Extended commands will report back to the computer.

255 Store Device Number:
Stores the device number into the controller. The device number takes effect immediately. The enabled/disabled status of the device is unchanged.

247 Recall Device Number:
Allows you to read the stored device number from the controller.

E3C Visual Basic Programming Examples

The E3C command set is easily used from any programming language that supports serial communication. The following Visual Basic 6 Example source code demonstrates subroutines that can be used to control which devices will listen and which devices will ignore your commands.

Sample Code: The E3C Command Set

```
Public Sub EnableAllDevices()
    'Enable All E3C Devices
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(248)    'E3C Enable All Device Command
End Sub

Public Sub DisableAllDevices()
    'Disable All E3C Devices
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(249)    'E3C Disable All Device Command
End Sub

Public Sub EnableSpecificDevice(Device)
    'Enable A Specific E3C Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(250)    'E3C Disable Specific Device Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Disabled
End Sub

Public Sub DisableSpecificDevice(Device)
    'Disable A Specific E3C Devices, Other Devices will be unchanged
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(251)    'E3C Disable Specific Device Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Disabled
End Sub

Public Sub DisableAllDevicesExcept(Device)
    'Disable All E3C Devices Except (Device)
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(252)    'E3C Disable All Device Except Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Active
End Sub

Public Sub EnableAllDevicesExcept(Device)
    'Enable All E3C Devices Except (Device)
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(253)    'E3C Enable All Device Except Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Inactive
End Sub


Public Sub StoreDeviceNumber(Device)
    'Store an E3C Device Number into the Controller
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(255)    'E3C Store Device Number Command
    MSComm1.Output = Chr$(Device) 'Device Number that will be Stored
    WaitForReply                  'Wait for R16 to Acknowledge Command
End Sub

Public Function GetDeviceNumber()
    'Read the E3C Device Number from the Controller
    MSComm1.Output = Chr$(254)    'Enter Command Mode
    MSComm1.Output = Chr$(247)    'E3C Get Device Number Command
    Do                            'Wait for Device to Reply
        DoEvents                  'Allow Windows to MultiTask
    Until MSComm1.InBufferCount > 0 'If the Device Replies
    GetDeviceNumber = Asc(MSComm1.Input)'Get Device Number from Buffer
End Sub
```

The PWM8x Command Set

The PWM8x supports an extensive command set, used to set the pulse width of all output channels, set operation modes, and store and recall data from the board. Most users will not use many of the functions built into this controller. The best way to familiarize yourself with the capabilities is to carefully read through the command set in this section. The "plain English" examples provide a quick, easy to understand definition of what each command does.

The number to the left of each command indicates the ASCII character code that must be sent to issue the command. All commands must be preceded with ASCII character code 254 to place the device in command mode. See examples at right.

Setting PWM Values

The term "Set PWM" means "Set Pulse Width Modulation". The PWM value may be any number from 0 to 255. A value of 0 turns off your externally connected devices, 255 turns it on all the way, 128 is 50% duty cycle (half speed on a motor, half brightness on a light). From this point forward, think of setting the PWM value as setting the amount of power you would like to ration to your external device. In reality, you are actually controlling how long the device is allowed to be connected to ground.

Controlling PWM Values on Output Channels

0 - Set All Output Channels to the Same PWM Value (0-255)
1 - Set Output Channel 1 PWM Value (0-255)
2 - Set Output Channel 2 PWM Value (0-255)
3 - Set Output Channel 3 PWM Value (0-255)
4 - Set Output Channel 4 PWM Value (0-255)
5 - Set Output Channel 5 PWM Value (0-255)
6 - Set Output Channel 6 PWM Value (0-255)
7 - Set Output Channel 7 PWM Value (0-255)
8 - Set Output Channel 8 PWM Value (0-255)
A parameter value of 0-255 is required for the above commands, used to set the speed of a motor or the brightness of a light.

Store Defualt Powerup PWM Values

9 - Store Startup PWM Values (0-255)
This command is used to store the currently selected PWM Values for all 8 channels as the default power-up values. This command can be used in a room with DC lighting, allowing different areas of the room to have different levels of brightness when the lights are turned on. A parameter of 0-255 is required for this command; but has no function.

Setting the Startup Mode

10 - Store Startup Mode (1, not 1)
The Store Startup Mode command is used to set the way the PWM8x powers up. A parameter value of 1 causes all 8 channels to begin cycling immediately, meaning all outputs become immediately active (based on the Stored Default Powerup PWM Values above). A parameter value of anything other than 1 causes each channel to rise slowly to its Stored Default Powerup PWM Value, beginning with output channel 1. This mode is ideal for room lighting applications. When the lights are turned on in the room, different areas of the room will raise in brightness level until all 8 levels have reached their maximum stored level.

Visual Basic Programming Examples

Many Visual Basic 6 programming examples are provided to assist in the development of software for controlling the PWM8x. Additional source code can be found on our web site at www.controleverything.com.

Sample Code: Reading Data

The following function may be used to read data from the board. At the time of writing, this function is only useful for reading the E3C device number. The PWM8x is not capable of returning other parameters to the user.

```
Public Function GetData()                'Read Data from PWM8x
    Do                                   'Wait for Device to Reply
      DoEvents                           'Allow Windows to Multitask
    Until MSComm1.InBufferCount > 0      'If the Device Replies
    GetRelayStatus = Asc(MSComm1.Input) 'Get Status from Serial Buffer
    Debug.Print GetData                  'Display in Immediate Window
End Sub
```

Sample Code: Controlling Output Levels

```
SetLevel 0,128      'Set All Channels to Half Brightness
SetLevel 0,64       'Set All Channels to Quarter Brightness
SetLevel 0,255      'Set All Channels to Full Brightness
SetLevel 1,160      'Set Channel 1 to 3/4 Brightness
SetLevel 8,128      'Set Channel 8 to Half Brightness

Public Sub SetLevel(Channel,PWM)        'Channel Parameter = 0 to 8
                                        'PWM Parameter = 0 or 255
       MSComm1.Output = Chr$(254)       'Enter Command Mode
       MSComm1.Output = Chr$(Channel)   'Send Channel Command
       MSComm1.Output = Chr$(PWM)       'Relay to Turn On
End Sub
```

Sample Code: Reading Status of Relays

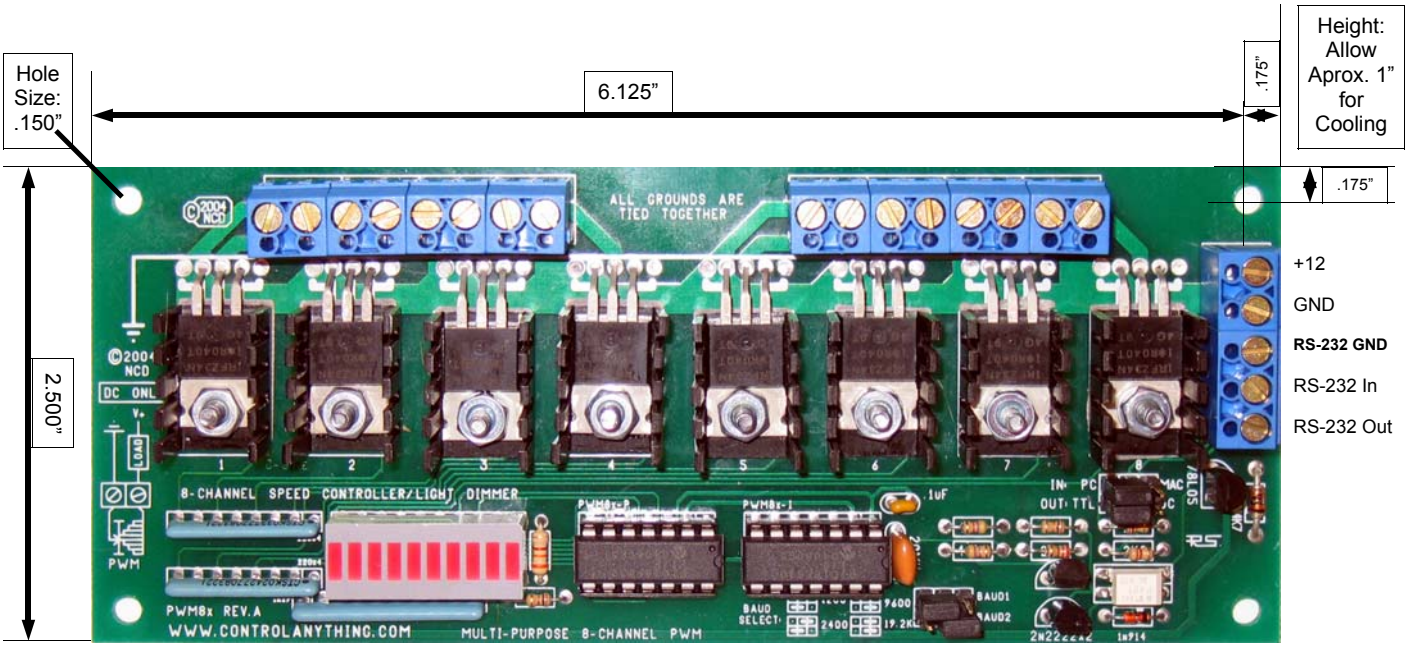
```
Public Sub StoreStartup()         'Store Startup Values
       MSComm1.Output = Chr$(254) 'Enter Command Mode
       MSComm1.Output = Chr$(9)   'Send Channel Command
       MSComm1.Output = Chr$(0)   'This Parameter Required & Ignored
End Sub
```

Sample Code: Power-Up Relay Pattern

```
StartupMode 0   'All Channels Instantly On
StartupMode 1   'Channels 1-8 Rise Slowly to Full Stored PWM Values

Public Sub StartupMode(Mode)       'Store Startup Values
       MSComm1.Output = Chr$(254)  'Enter Command Mode
       MSComm1.Output = Chr$(10)   'Send Channel Command
       MSComm1.Output = Chr$(Mode) 'This Parameter Required & Ignored
End Sub
```

Outline Dimensions PWM8x Series Controllers



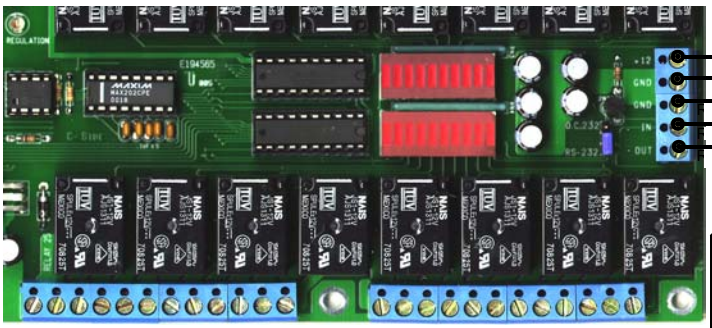
Power and Serial Connections	
+12	Connect to a Power Source Between +7 and +18VDC to Power the Logic on the Board
GND	Connect to the Ground of the Power Source
RS-232 Ground	Connects to the RS-232 Ground of your Computer
RS-232 In	Connects to the RS-232 Output of your Computer
RS-232 Out	Connects to the RS-232 Input of your Computer

Jumper Settings

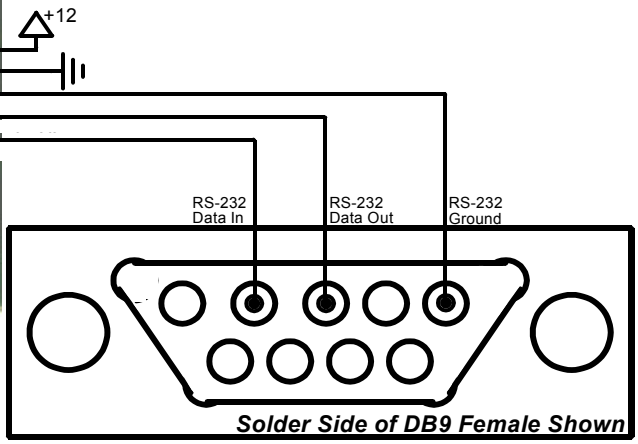
Baud Rate Jumpers settings are read by the PWM8x only when power is applied. For this reason, it is important to power cycle the board to read the new jumper settings. Serial Communications Jumpers may be changed at any time.

Important Note for Laptop Users

The RS-232 voltage levels on laptop computers are different than on desktop computers. For this reason, you MUST set the PC/MAC jumper to the MAC position for ALL LAPTOP COMPUTERS. You will not be able to communicate reliably to the board unless this jumper is set properly. You may change this jumper at any time without damaging the computer or the controller.

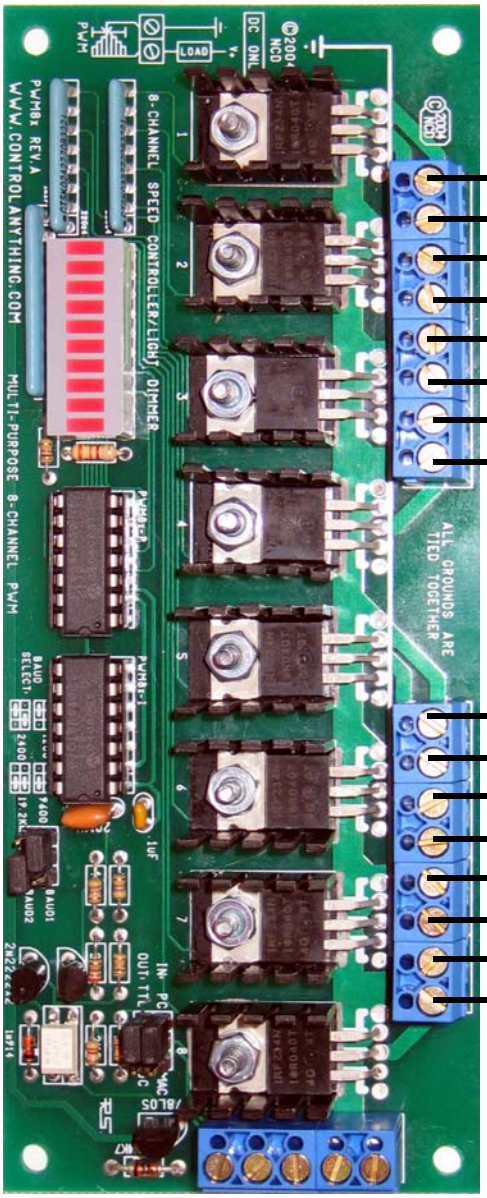


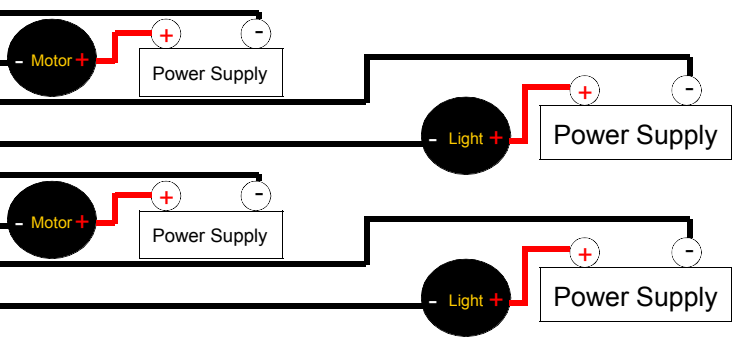
Nearly all NCD devices have the same 5-Position Terminal Block shown above for connecting power and data to the controller.



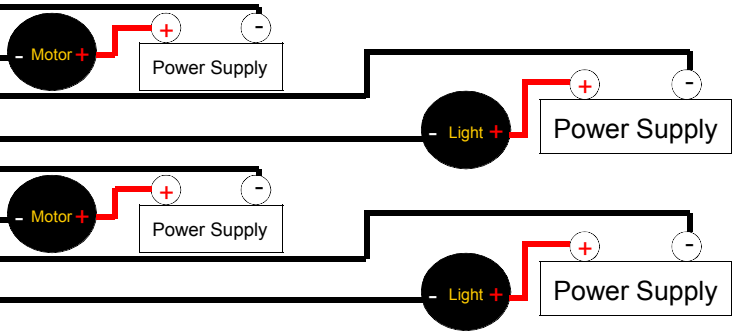
PWM8x Example Connections

The PWM8x can control 256 levels of pulse width modulation for eight independent channels. As the PWM values are changed in software, the LEDs will change in brightness. The bottom LED shown at left is typically ON. When a command is received, this LED turns off and the 2nd from the bottom LED will stay lit until your command has executed. Once completed, the second LED will turn off and the 1st LED will come back on.





The PWM8x is capable of controlling lights and motor of different voltages using a dedicated power supply for each device. You can also use a shared power supply if all external devices operate at the same voltage. You should NOT use the power supply included with the Quick Start Kit to power anything other than the PWM8x controller or damage will result.



The PWM8x works by controlling the length of time an external load, such as a light or motor, is connected to ground. The more time the device is allowed to be connected to ground, the more power the external load will appear to have. The PWM8x provides 256 levels of speed control for motors and 256 levels of brightness control for light bulbs.

Setting an inductive load such as a DC motor to a low speed for an extended period of time tends to generate more heat than allowing the motor to operate at it's full on state. Inductive loads (any externally connected device with a coil) generates flyback voltages that can result in dangerous high voltages and an excessive amount of heat. Never touch the Hexfets, Heat Sinks, or Connectors while in operation.

5-Year Repair or Replace Warranty

Warranty
NCD Warrants its products against defects in materials and workmanship for a period of 5 years. If you discover a defect, NCD will, at its option, repair, replace, or refund the purchase price. Simply return the product with a description of the problem and a copy of your invoice (if you do not have your invoice, please include your name and telephone number). We will return your product, or its replacement, via UPS Ground Service in the US and Canada Only. Additional shipping charges will apply to international customers.

This warranty does not apply if the product has been modified or damaged by accident, abuse, or misuse.

30-Day Money-Back Guarantee
If, within 30 days of having received your product, you find that it does not suit your needs, you may return it for a refund. NCD will refund the purchase price of the product, excluding shipping/handling costs. This guarantee does not apply if the product has been altered or damaged.

Copyrights and Trademarks

Disclaimer of Liability
NCD is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, and any costs or recovering, reprogramming, or reproducing any data stored in or used with NCD products.

Technical Assistance
Technical questions should be e-mailed to Ryan Sheldon at ryan@controlanything.com. Technical questions submitted via e-mail are answered several times daily. Technical support is also available by calling (417) 646-5644 from 9:00 A.M. to 4:00 P.M. Central Standard Time.

NCD Contact Information

Mailing Address:
National Control Devices
P.O. Box 455
Osceola, MO 64776

Telephone:
(417) 646-5644

FAX:
(417) 646-8302

Internet:
ryan@controlanything.com
www.controlanything.com
www.controleverything.com