MidiTron™

The MIDI to Real World Interface from Eroktronix

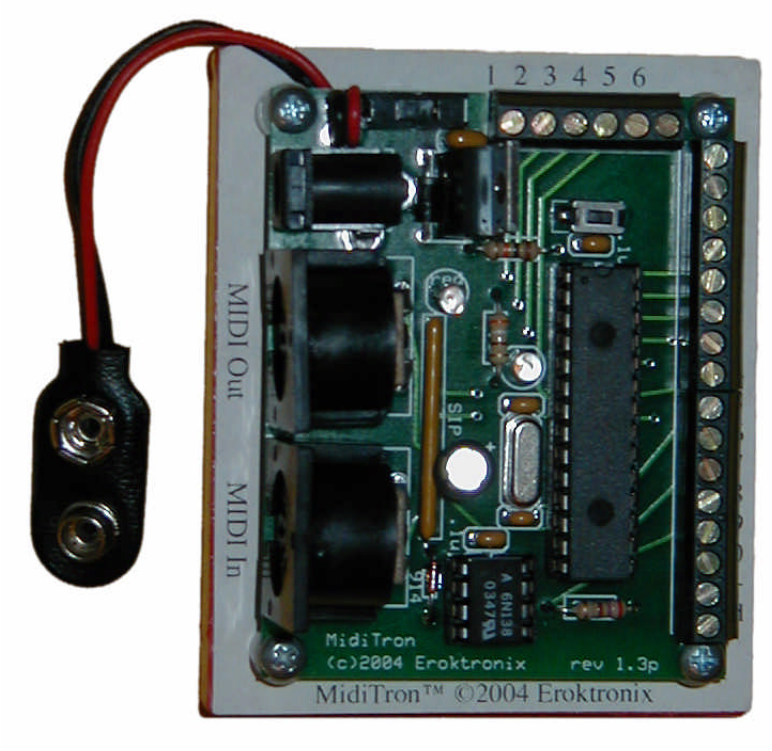
Manual version: 1.3

Introduction

MidiTron™ is a MIDI to real world interface. Its purpose is to provide MIDI interfacing to sensors, switches and voltage-controllable devices such as LEDs, motors, solenoids and relays.

Several design features distinguish MidiTron™ from other i/o solutions currently available:

• It is highly and easily user-configurable.
• Its small size allows it easily to be embedded in MIDI projects such as sensor-based instruments and robotic and lighting control applications.
• It takes advantage of MIDI's availability, standardization and reliability for interfacing and configuration. No special interface hardware (such as USB or serial) or drivers required.
• It is configurable via MIDI sysex. No special software or Max objects are required.



MidiTron™

Hardware

MidiTron™ has one MIDI In Port, one MIDI Out Port, twenty user-configurable i/o terminals and terminals providing power (5v) and ground.

Each i/o terminal can be configured as a digital input, analog input (ADC), digital output or analog output (PWM). Up to ten channels of analog input and up to twenty of the other types of i/o are available, in any combination.

A status LED in the center of the board blinks each time a MIDI byte is received by or sent from the board.

MidiTron™ can be powered by a 9v wall transformer or equally well from a 9v battery for embedded applications.

Powering MidiTron™

MidiTron™ can be powered from a 9v battery or a 9v wall transformer or "wall wart" (not included). Suitable wall transformers include Radio Shack part # 273-1767 or 273-1770 with Adaptaplug M (part # 273-1716), or any 9v adapter (300 mA or greater recommended) with a 2.1 mm ID/5.5 mm OD/positive center coaxial plug from digikey.com (for example, part # T405-P5P-ND), jameco.com (for example, part #100853) or elsewhere.

Configuration

MidiTron™ provides twenty terminals of configurable i/o. Configuration is accomplished via sysex commands sent to the MIDI In Port. MidiTron™ can be configured using the supplied Max patch (running within Max or MaxPlay, or as a standalone application) or from any device that can generate sysex commands in MidiTron™'s format (documented below).

MidiTron™ stores all configuration parameters in persistent ("flash") memory, retaining its configuration even when powered off and on. Thus, after configuration, it may be disconnected from a host computer and used henceforth with any MIDI equipment.

Each terminal can be in one of the following modes: Off, Digital Input, Analog Input, Digital Output, Analog Output or Status (terminal 20 only). Analog inputs are available on terminals 1-10 and must be consecutive, starting from 1.

Each mode has a set of configuration parameters specific to the context of the mode. Modes and configuration parameters are described in detail below.

Using MidiTron™

Here are the simple steps for setting up and using MidiTron™:

1) Connect MIDI cables between your computer or device and MidiTron™. Power the unit from either a 9v battery or a 9v DC adapter (center positive, 2.1 mm ID, 5.5 mm OD).

2) Use the supplied Max patch to set up the terminals for any valid combination of digital and analog inputs and outputs and to select which MIDI messages will be sent and received. The Max patch can be run in Max or Max Play on Mac OS9, Mac OSX or Windows. Once MidiTron™ is set up, it remembers its settings even if you power it off.

You can also set up MidiTron™ using any device or program that can send sysex. See the sysex spec in Appendix A for details.

3) Connect sensors and/or output devices to MidiTron™. See the connection guide in Figure 1 for terminal numbering and power/ground connections.

4) MidiTron™ will now send MIDI messages whenever sensor values change. When it receives MIDI messages corresponding to particular output terminals, it will change the outputs accordingly.

Safe hookup practices

When hooking up devices to MidiTron™, observe the following practices to minimize the possibility of damaging the unit or your devices.

1) Always turn MidiTron™ off when connecting and disconnecting devices.

2) Double-check your connections before switching MidiTron™ back on.

3) After making new connections and switching MidiTron™ back on, carefully touch the voltage regulator (the device with the metal tab sticking up, just below the power switch and to the right of the power jack). If it is getting hot (not just warm, but hot), immediately switch MidiTron™ off, and recheck your connections. (Be careful when doing this, as the regulator can get very hot in the event of a short.)

4) When switching on, it is a good idea to have a volt meter connected to read the voltage between ground and +5v (MidiTron™'s bottom two terminals). This should always read between 4.9 and 5.1 volts. If it drops significantly below 5 volts, you are probably creating a short from power to ground, or directly grounding a terminal which is outputting high.

Status LED

The status LED is the green in the center of the board. It blinks twice when the board is powered on. Henceforth, it blinks each time a MIDI byte is received by or sent from the board.

The status LED is shared with terminal 20 and only blinks when the Status function is enabled (see Note on Status LED and Terminal 20 below).

MidiTron™ programmer software

The MidiTron™ programming software is implemented in Cycling 74's Max program and is supplied both as a stand-alone program for Mac and PC, as well as a set of Max patches,

When you open the programmer, the first thing you must do is to set its MIDI In and MIDI Out port menus. This tells the programmer the ports to which MidiTron™ is connected.

The main body of the programmer screen consists of 20 configuration lines, corresponding to MidiTron™'s 20 terminals. Use the menus on the left side to set the mode for any terminal you wish to connect. For unconnected terminals, leave their mode set to Off (except terminal 20, which should normally be set to Status). See below for a detailed discussion of MidiTron™'s modes.

When you change a terminal's mode, the line to the right of the menu will be filled in with the option setting boxes and menus pertaining to the mode. Each time you change a terminal's mode or settings, its configuration is sent immediately to MidiTron™ via sysex.

For each enabled terminal, a control will appear to the right of the red line at the end of the configuration line. For input modes, this will be a Receive control; for output modes, it will be a Send control. These controls are wired to send data to or receive data from MidiTron™ based on the terminal configuration.

For Digital in mode, the control will be a check box; for Analog in mode, it will be a number box. The control will automatically be wired to receive data from the MidiTron™ for the corresponding terminal. Thus, it will show a digital input level as a checked or unchecked box, and an analog input level as a number from 0-127.

For Digital out and Analog out mode, the control will be a number box. Changing this box will send data to the board to control the corresponding terminal.

Along the top of the window to the right of the port settings are buttons to load, save, dump and capture parameter settings, and a button to open the Easy Controls configuration patch. Along the bottom of the window are settings for PWM resolution, PWM period and Slowdown factor, and the Factory reset button. All of these features are described below.

Important: Note that when you open the programmer, all menus are set to their default settings (terminals 1-19 set to Off, terminal 20 set to Status, etc.), not to the settings currently stored in MidiTron™. The board will receive settings only for the terminals which you change (unless you dump the current configuration to the board - see below). Therefore, what you see in the programmer window may not reflect the state of the device.

For example, if you set terminals 1 and 2 to Analog in, then close and reload the programmer, it will show terminals 1 and 2 as being in Off mode, even though the board will still be holding your previous configuration. (The programmer is not able to read the configuration from the board.) If you change terminal 2 to Digital in, terminal 1 will still remain as configured.

You can alleviate this confusion by saving your configuration to a file, and reloading it when you open the programmer patch. See the following discussion for details.

Loading, saving and dumping configurations

Using the controls at the top of the programmer window, you can store and recall configuration sets. The ‘save’ button saves all configuration settings to a text file, while ‘load’ reloads all settings from a previously saved file. Saved settings include all terminal configuration menus and controls plus settings for PWM resolution and period, and Slowdown factor. When you reload a configuration file, it fills in all menus with your saved settings and dumps the settings to MidiTron™.

Clicking the ‘dump’ button resends all configuration settings to MidiTron™. This is useful if you had set up a configuration with MidiTron™ off or disconnected and wish to send all configuration settings to the board.

The box labeled ‘capture’ contains a listing of the sysex messages from the last ‘dump’. If you wish to copy this listing (perhaps for use in another program), set up your desired configuration, click ‘dump’, double-click ‘capture’ to open its text window and copy the data from this window.

Easy Controls window

Clicking ‘open Easy Controls’ will open a programming subwindow. This window allows you to test each of the MidiTron™ modes by loading an example configuration to the board and providing a corresponding set of controls.

For example, clicking the button next to ‘Digital inputs’ configures the menus (and thus the MidiTron™) for Digital in mode for terminals 1-19. It provides a set of check boxes corresponding to this configuration, to enable you to view the digital input state of each terminal.

Similar sections are provided for Analog inputs, Digital outputs and Analog outputs.

MidiTron™ modes

Each MidiTron™ terminal can be in one of five modes: Off, Digital input, Analog input, Digital output, Analog output. Analog inputs are available on terminals 1-10 only and must be consecutive, starting from 1.

Each mode is described in detail below.

Digital input mode

The terminal will accept 0v to switch off and 5v to switch on. When the terminal is switched on, it will send its “On” MIDI command; when switched off, it will send its “Off” MIDI command. Available commands for this mode are Note Off, Note On, Poly Pressure, Control Change, Program Change, Channel Pressure and Pitch Bend.

Analog input (AD) mode

The terminal will accept a range of 0 to 5 volts and convert this voltage to a corresponding value using A/D conversion. It will send a MIDI message each time the converted value changes. Available commands for this mode are Poly Pressure, Control Change, Channel Pressure and Pitch Bend.

When using Poly Pressure, Control Change or Channel Pressure messages, the converted output value can range from 0-127. When using Pitch Bend, it can range from 0-16383.

Analog inputs are available on terminals 1-10. Analog inputs must be consecutive, starting at terminal 1. That is, if you want 3 analog inputs, you must use terminals 1-3. MidiTron™ will automatically disable any terminals not configured in this way. For example, if you set terminals 1, 2 and 4 to be analog inputs, but not 3, terminal 4 will be disabled.

Analog input mode details

When setting up Analog input mode, you can control its input and output ranges to get the best results for the sensor you are using.

The input min and max are used to compensate for the fact that some sensor configurations do not output 0v at their minimum or 5v at their maximum. To adjust the input range properly, do the following:

1) To begin with, set the range values as follows:
    input min = 0
    input max = 127
    output min = 0
    output max = 127

2) Cause your sensor to output its minimum voltage. Adjust input min until the MIDI message being output for this terminal has a data value of zero. You can adjust it so the sensor just hits zero at its minimum, or a little higher to provide a “dead zone” at the bottom of your sensor’s range.

3) Cause your sensor to output its maximum voltage. (Note: make sure this voltage never goes above 5v!) Adjust the input max until the MIDI message being output for this terminal has a data value of 127, or 16383 if using Pitch Bend. Similarly to step 2, you can adjust it so the sensor just hits 127 (or 16383) at its maximum, or a little lower to provide a “dead zone” at the top of your sensor’s range.

4) Optional: You can also choose to limit the output range of the sensor and/or reverse its data direction. To limit the output range, adjust the output min and max. To reverse data direction, set output min to be higher than output max (for example, output min = 127, output max = 0). MidiTron™ maps the sensor’s minimum value to output min and its maximum value to output max, and scales intermediate values accordingly.

For pitch bend, although output min and max are set using a range of 0-127, the output is mapped to 0-16383. To find the approximate high-resolution mapping values for pitch bend, multiply the output min and max by 128.

MidiTron™’s A/D conversion is done using 10-bit resolution. It then converts this value to the appropriate output value based on your range settings. When using pitch bend, the 10-bit A/D value is converted to a 14-bit pitch bend value. Thus, you will see stepping in the pitch bend value (i.e. the lowest four bit are insignificant), but the resolution is still higher than 7-bit (low-resolution) pitch bend.

Digital output mode

The terminal will output 0v when switched off or 5v when switched on. The output can be switched using one of the following MIDI input messages: Note On, Poly Pressure, Control Change, Program Change, Channel Pressure or Pitch Bend.

The output switches on when the MIDI message’s data value becomes greater than or equal to the high threshold value and off when it becomes less than the low threshold value.

The polarity controls the output voltage mapped to on and off states. With positive polarity, the output switches high (i.e. 5v) when the MIDI data value goes above the high threshold, and low (i.e. 0v) when it goes below the low threshold. With negative polarity, the output switches low (i.e. 0v) when the MIDI data value goes above the high threshold, and high (i.e. 5v) when it goes below the low threshold.

Negative polarity is useful when driving LEDs in “active low” mode. See Circuits below for more details.

Digital output mode: note on/off example

Let’s say we want terminal 1 to switch on when we hit middle C on a keyboard (transmitting on MIDI channel 1) and off when we release it; that is, turn on with pitch 60, velocity >= 1 and off with pitch 60, velocity = 0. To do this, set up terminal 1 as follows:

Mode = Digital out, Channel = 1, Command = Note on, Pitch = 60, Velocity-low = 1, Velocity-high = 1, Polarity = Positive.

This tells MidiTron™ to switch on when it receives a note on command (pitch 60) with velocity >= 1 (the Velocity-high setting) and off with velocity < 1 (the Velocity-low setting). Velocity < 1 can only be 0, which is equivalent to a note off command. Thus, on with note on and off with note off.

Digital output mode: controller example

Let's say we want terminal 1 to switch on and off using the mod wheel of a keyboard controller (control number 1, channel 1 for our example). We'd like it to switch on when we move the wheel above the halfway point and off below this point. We'd also like a "buffer zone," so the terminal doesn't flicker on and off if we hover around the halfway point. To do this, set up terminal 1 as follows:
Mode = Digital out, Channel = 1, Command = Control change, Ctrl # = 1, Ctrl value-low = 62, Ctrl value-high = 66, Polarity = Positive.

The halfway point in MIDI is 64. We have told MidiTron™ to switch on when the mod wheel goes slightly above this point (>= 66) and off when it goes slightly below (< 62). No switching will occur from 62 and 65, thus giving a buffer zone around the midpoint.

Note on Status LED and Terminal 20

Terminal 20 is shared with the status LED, which is connected using a standard "active low" circuit (see Circuits below) with a 390Ω resistor. The status function of this LED can be enabled/disabled and changed to other regular MidiTron™ functions using the programming patch.
Furthermore, terminal 20 is a special type of output known as "open drain". Instead of switching from 0v to 5v, it switches from 0v to a disconnected state. In order to make it switch high, it needs a pull-up resistor connected to it (usually a 1k to10k resistor connected between the terminal and 5v). The status LED connection can function as the pull-up in some cases.

If you use this terminal for your own functions, make sure you factor in the above considerations when designing your external circuitry.

If you wish to remove the status LED connection from the terminal, you can do so by carefully removing the 390Ω resistor (the one with orange-white-brown-gold rings) by clipping it out with a pair of wire nippers. If you need to restore a status LED after this, you will need to connect your own one externally to terminal 20 with a 390Ω pull-up resistor.

Analog output (PWM) mode

The terminal will output an effective range of 0-5v using pulse width modulation (PWM). The pulse width, and thus the effective output voltage, can be controlled using one of the following MIDI input messages: Poly Pressure, Control Change, Channel Pressure, Pitch Bend.

The data value of the MIDI message, from 0-127, is translated into a pulse width of 0-100%. (For pitch bend, the low-resolution 0-127 value is used.) For example, a value of 64 is a pulse width of 50%, or an effective output voltage of 2.5v.

The PWM frequency and resolution can be controlled from the MidiTron™ Max patch, or using sysex messages.

PWM details

MidiTron™, like most microprocessor-based devices, uses PWM to generate analog output. It generates a PWM output by rapidly cycling an output on and off, varying the time that each cycle spends high and low. PWM can be used to control the speed of motors and the level of LEDs and for many other analog functions.

The pulse width is defined as the amount of time that an output is high divided by time per output cycle (the period). Using the definitions in Figure 1, pulse width = T_{hi} / T. Expressed as a percent, it is (T_{hi} / T) * 100.

The output in Figure 1 has a pulse width of .75 or 75%. The PWM frequency is the inverse of the period, or 1 / T.

MidiTron™ allows you to control the global frequency and resolution of PWM. These can be set from the MidiTron™ Max patch (or by using sysex).

Resolution is set as a value from 2 to 7 bits. The number of possible pulse width steps per cycle is 2 to the power of the number of bits: 2 bits = 4 steps, 3 bits = 8 steps, 4 bits = 16 steps, 5 = 32 steps, 6 = 64 steps, 7 = 128 steps. MidiTron™'s maximum PWM resolution is 7 bits (or 128 steps). This means that it has a maximum of 128 different pulse width values, in steps of 1/128.

Setting a lower resolution has the effect of increasing the frequency. Each decrease in 1 bit of resolution doubles the output frequency. For example, changing the resolution from 7 to 6 bits cuts the resolution in half, to 64 steps per cycle, the frequency will double, but the step size will be 1/64. The output pulse width will change for every two steps of change in input value.

Frequency is set by setting the period in units of 1/10 microseconds per step. For example, a value of 1000 will result in a period of 1000/10 microseconds per step, or 100 microseconds per step. If, for example, the resolution is set to 6 bits, which equals 64 steps per cycle, this will result in a PWM period of 100 microseconds * 64 steps = 6400 microseconds per cycle. This corresponds to a PWM frequency of 1 / 6400 microseconds, or about 156 Hz. (Note: the actual period will be slightly longer than the set value, due to some additional processing that takes place.)

MidiTron™ enforces a minimum period (or maximum frequency) which is dependent on the number of terminals set to PWM (analog output). See Table 1 for details.

In general, a high frequency is desirable for most applications. However, the trade-off is that too high a frequency can slow down MidiTron™'s other functions (receiving and sending MIDI, processing other types of terminals). Also, more terminals set to PWM means less time for other functions.

If all this math bothers you, simply hook up an LED and experiment. Set up a PWM output and send it a MIDI message with a value of 64 (i.e. a pulse with of 64 / 128 = 0.5 = 50%. If you increase the period substantially (which decreases the frequency), the LED will blink on and off instead of appearing dim. Then, decrease the period, increase the step and change the MIDI value. With a higher step, you will see less gradual changes in the LED's brightness as you vary the MIDI value.

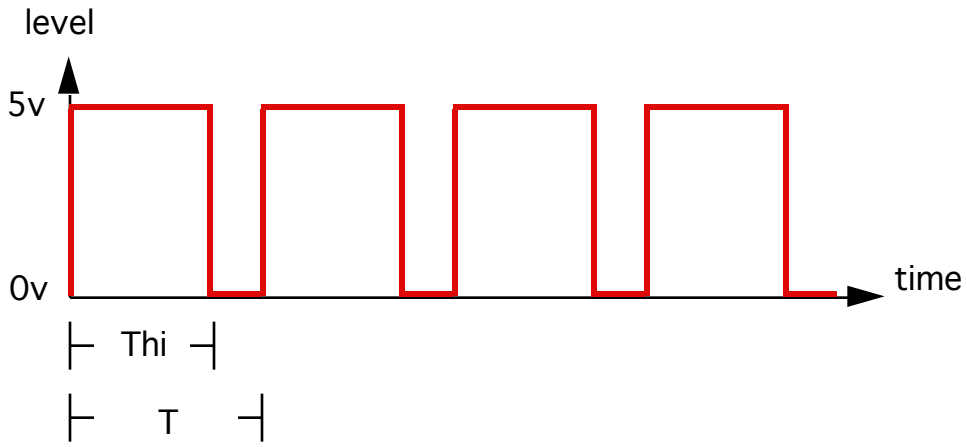


Figure 1: PWM definitions

# of PWM terminals	Minimum period	Frequency @ 2 bits	Frequency @ 3 bits	Frequency @ 4 bits	Frequency @ 5 bits	Frequency @ 6 bits	Frequency @ 7 bits
1	799	3,130	1,565	783	391	196	98
2	937	2,667	1,334	667	333	167	83
3	1076	2,323	1,162	581	290	145	73
4	1215	2,058	1,029	515	257	129	64
5	1353	1,847	924	462	231	115	58
6	1492	1,676	838	419	209	105	52
7	1631	1,533	767	383	192	96	48
8	1769	1,413	706	353	177	88	44
9	1908	1,310	655	328	164	82	41
10	2047	1,221	611	305	153	76	38
11	2185	1,144	572	286	143	71	36
12	2324	1,076	538	269	134	67	34
13	2463	1,015	508	254	127	63	32
14	2601	961	481	240	120	60	30
15	2740	912	456	228	114	57	29
16	2879	868	434	217	109	54	27
17	3017	829	414	207	104	52	26
18	3156	792	396	198	99	50	25
19	3295	759	379	190	95	47	24
20	3433	728	364	182	91	46	23

Table 1: PWM maximum frequencies

Running Status, and why it may screw you up

The following discussion applies if you are using MidiTron™ outputs (i.e. sending MIDI messages to MidiTron™ to control terminals configured as outputs).

In the MIDI standard, "running status" is a method which lets systems send MIDI messages with fewer than the full number of bytes. It is a way of sending shorter MIDI messages by getting rid of repeated "status" bytes. At some point in your project, this method is almost sure to cause you problems. Here's why.

In MIDI, the most common messages that are used are called Channel Voice Messages. These include note on, note off, control change, program change, pitch bend, channel pressure (aftertouch) and polyphonic pressure. All of these messages consist of a status byte, followed by one or two data bytes. The status byte contains both the MIDI command and channel.

The MIDI standard allows the sender to drop the status byte if it is the same as the status byte from the last message. For example, let's say the sender wants to send three "note on" messages on channel 1. The MIDI message would look like this:

<note on, ch=1> <pitch 1> <velocity 1>
<note on, ch=1> <pitch 2> <velocity 2>
<note on, ch=1> <pitch 3> <velocity 3>

Since the status byte doesn't change for notes 2 and 3, the sender can drop them, so the message looks like this:

<note on, ch=1> <pitch 1> <velocity 1>
<pitch 2> <velocity 2>
<pitch 3> <velocity 3>

If the receiver finds a "missing" status byte, it knows to re-insert the last one it received.

OK, so far so good. But, what if you start your sender up, and have it send a bunch of pitch commands on channel 1, but then switch on or connect your receiver after sending has started:

<note on, ch=1> <pitch 1> <velocity 1>
(turn on receiver here!)
<pitch 2> <velocity 2>
<pitch 3> <velocity 3>
...

?

The receiver will, of course, miss the first pitch in this situation. But the worse problem is that, having missed the first (and only) status byte in this stream, it will not know what MIDI command has been sent. It has no choice but to ignore data and not respond until it receives a new status byte.

There are two main things that cause the sender to send a new status byte: a change in command, or a change in channel. If, for example, you are sending alternating pitches on two different channels, a status byte will be sent each time:

<note on, ch=1> <pitch 1> <velocity 1>
<note on, ch=2> <pitch 2> <velocity 2>
...

Running status is implemented automatically (and can't be switched off) on most MIDI systems. You will almost certainly run into a running status problem when using MidiTron™ outputs if you don't take precautions. Here are a two suggestions on what to do:

1) Configure your terminals to respond to different types of MIDI messages or different channels. For example, instead of using control change messages on channel 1 with different controller numbers, use the same controller number, but different channels. Thus, every time your sender sends a control to a different terminal, it will send a new status byte.

2) Make your sender send an unused MIDI message every so often. For example, if you are not using aftertouch messages, make your system send an arbitrary aftertouch message once a second. (In Max, you can implement this with a metro.)

Note that when sending messages, MidiTron™ always sends a full MIDI message - that is, it does not implement running status.

Slowdown Factor

You can use MidiTron™'s slowdown factor to slow down the speed at which MIDI output messages are sent. A value of 0 means no slowdown. Each increase of 1 adds a minimum of 20 microseconds of delay between MIDI messages. This is a minimum due to the fact that other events which take time may intervene, such as processing MIDI input.

Factory Reset

The factor reset command (provided in the programming patch or available through sysex) resets MidiTron™'s terminal configurations. It sets all terminals to Off, except terminal 20, which is set to Status.

Schematics

This collection of schematics covers the basics of hooking up most types of sensors and output devices.

Digital input: Switches

Switches provide a digital input and can be hooked up in the configurations shown in Figures 1 and 2. Figure 1, the active low configuration, is the preferred method (due to electrical level requirements for many microprocessors). If you are using a normally open (N.O.) pushbutton switch, when the switch is depressed, it will send a low level (0v) to the terminal (thus the term “active low”). If the terminal is configured as a digital input, MidiTron™ will send the terminal’s Off Command. This may seem counterintuitive, but you should be able to get used to it after trying it out.

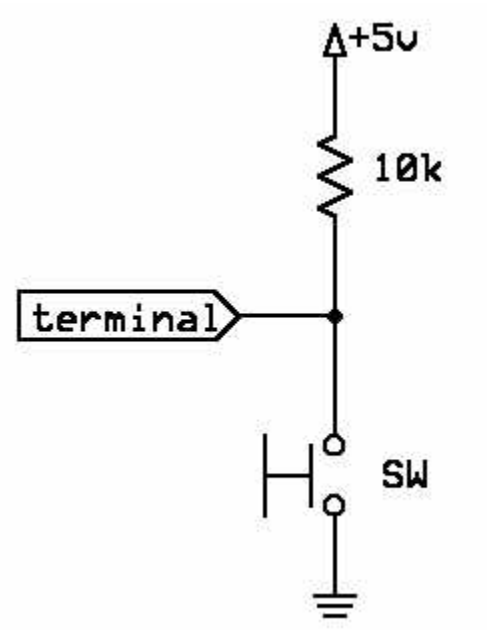


Figure 1: Switch, active low

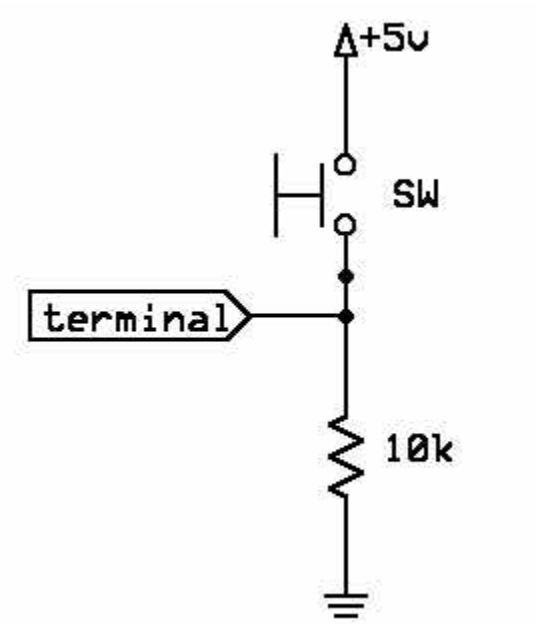


Figure 2: Switch, active high

Analog input: Potentiometers

A potentiometer (or “pot”) can be used to provide an analog input to a MidiTron™ terminal. Using the configuration in Figure 3, a pot will vary the input voltage to the terminal from 0-5v - in other words, the full input range of the terminal. 10k is a typical value for use in this configuration, but anything from 1k to 1M should work fine.

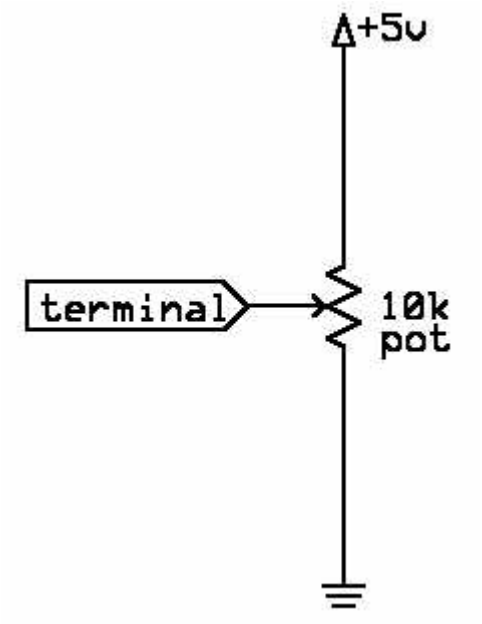


Figure 3: Potentiometer

Analog input: Resistive sensors

Resistive sensors, such as photoresistors, force sensing resistors (FSRs) or flex sensors, can be wired in either configuration shown in Figures 4 and 5. Using the Figure 4 configuration, the terminal voltage will decrease with increasing resistance. Using the Figure 5 configuration, the terminal voltage will increase with increasing resistance. Use whichever configuration is most convenient for your application.

You will also need to select an appropriate value for the fixed resistor R based on the resistance range of your sensor. In general, it is best to select a value for R that gives the largest voltage range for the sensor. The optimal value is R = sqrt(R_{min} * R_{max}); that is, the square root of the sensor's minimum resistance times the sensor's maximum resistance.

Once you compute the optimal value, select a standard resistance value that is closest to it. This will give the maximum voltage range for your sensor and thus the best resolution. Use the input min and max adjustments for the terminal to compensate for voltage offsets, as described above in the section **Analog input mode details**.

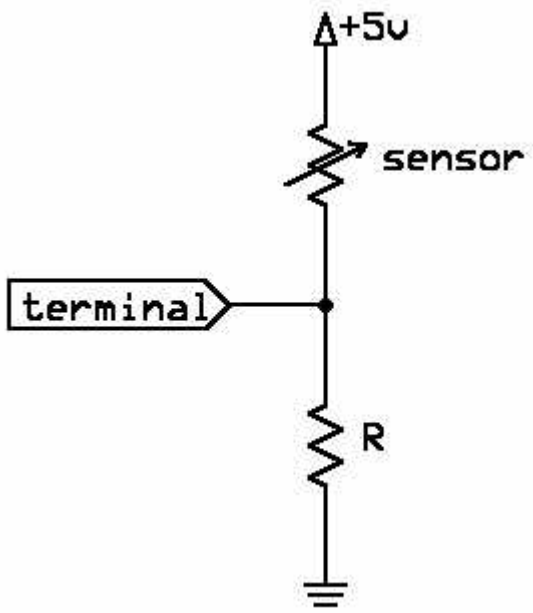


Figure 4: Resistive sensor (one way)

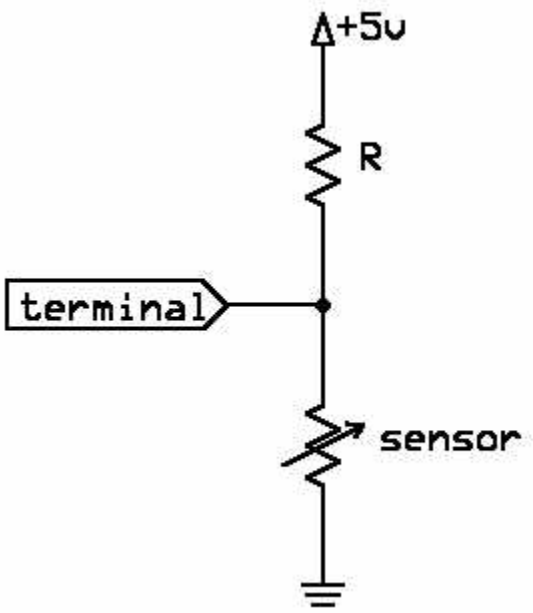


Figure 5: Resistive sensor (another way)

Digital and analog output: LEDs

LEDs can be driven by terminals in either digital or analog output modes. Digital output provides on/off, whereas analog output provides for dimming.

For digital output, using the configuration in Figure 6, the LED will turn on when the terminal goes high (5v) and off when it goes low; the opposite is true for Figure 7. For analog output, the LED will get brighter with increasing analog values; again, the opposite is true for Figure 7.

For digital output, you can use to polarity setting to compensate for either wiring configuration, so that received MIDI commands will have the effect you desire. In general, you will probably want to use positive polarity for active high configurations and negative polarity for active low.

Historically, active low was the preferred configuration due to microprocessor electrical capabilities. However, MidiTron™ outputs will work equally well in either configuration.

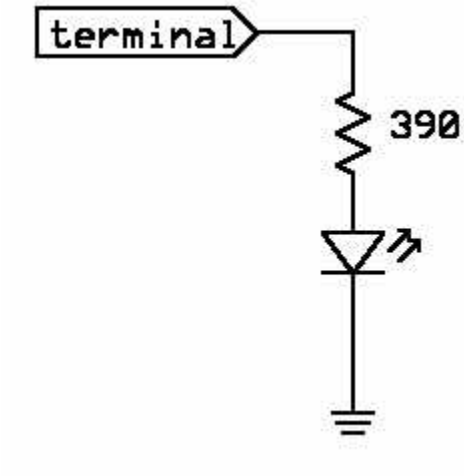


Figure 6: LED, active high

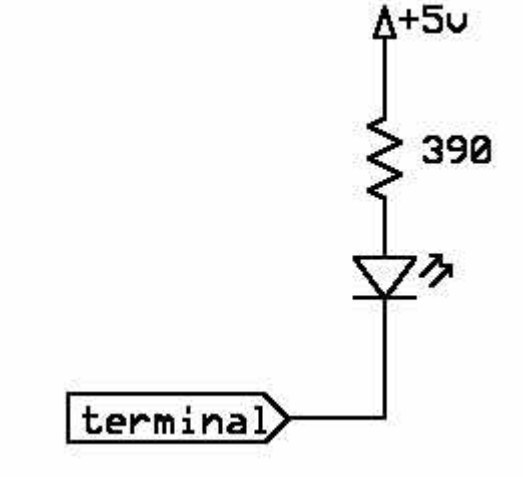


Figure 7: LED, active low

Digital output: Relays, solenoids and motors

MidiTron™ digital outputs can be used to switch DC electromechanical devices such as relays, solenoids and motors using the circuit in Figure 8. The device is represented as a coil in the circuit. When the terminal outputs a high level (5v), it will switch on the NPN transistor, which will allow current to flow through the coil and turn on the device.

Vcoil is dependent on the device's voltage requirements (i.e. it does not need to be 5v). The transistor type is dependent on the device's voltage and current requirements. For many devices, an NPN Darlington transistor such as a TIP122 will work well.

The diode is required because devices with coils will kick back current when they switch off. A diode wired in this way will protect the transistor and power supply from damage. A 1N4001 will work well for many applications.

It is best to wire the diode as close to the device's terminals as possible (rather than close to the MidiTron™ connections). Be careful to wire the diode in the correct direction, or your circuit will not function.

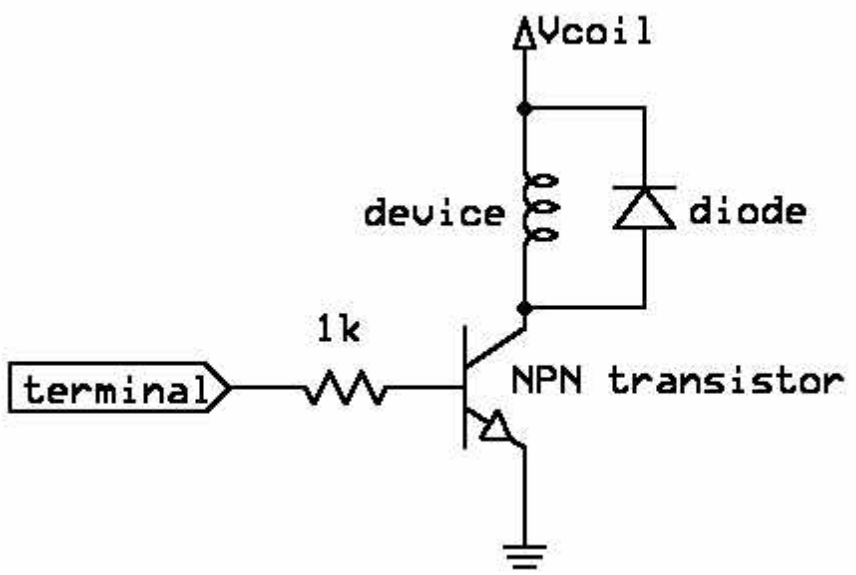


Figure 8: Motor, solenoid or relay

Analog output: DC motors with speed control

The circuit in Figure 8 can also be used for speed control of DC motors. The wiring remains the same, but the MidiTron™ terminal is used in analog output mode. Varying the output (PWM) value from 0-127 will cause the effective voltage seen by the motor to range from 0v to Vcoil.

For effecting both speed and direction control of a DC motor, an H-bridge IC such as an LMD18200 is recommended. You will need to use more than one terminal of the MidiTron™ for control. Connect an analog output to the PWM input of the H-bridge, a digital output to the direction input, and if desired, a digital output to the brake input.

Sensors and sources

Web sites
http://stage.itp.nyu.edu/~tigoe/pcomp/index.shtml and *http://stage.itp.nyu.edu/danclass/physical*: two excellent references on Physical Computing - all things about hooking up electronics, sensors, microcontrollers, etc.
http://www.ircam.fr/equipes/analyse-synthese/wanderle/Gestes/Externe/index.html: "Trends in Gestural Control of Music"
http://www.pacificsites.com/~brooke/Sensors.shtml: fairly exotic list of sensors

Electronics (new)
http://digikey.com: one of the most extensive selections of electronics
http://jameco.com: less overwhelming than digikey.com and often has most of what you might need, good for bend sensors and motors
http://mouser.com: extensive selection, some hard-to-find items, good source for sliders (linear pots)

Electronics (surplus)
http://allelectronics.com and *http://goldmine-elec.com:* Great sources for inexpensive motors, solenoids, switches, relays, etc.

Electronics search engines
http://findchips.com: searches about 20 electronics companies at once for what you want
http://globalspec.com: "The Engineering Search Engine"

Force Sensing Resistors (FSRs)
http://interlinkelec.com: They sell a kit with a lot of FSRs. They will also sell them individually (with a $60 minimum order), but you have to call or email them, as this is not listed on the site.

Capacitive
http://qprox.cox: Similar to Theremin-type sensing, they offer both on-off and continuous sensing.

Video Game Parts
*http://happcontrols.com:*the source for all things related to video games (buttons, joysticks, etc.)

Some sensors and devices you might like to try
Most of these devices can be found from the electronics suppliers listed above.
switches: pushbutton, momentary, on-off, snap action, lever, video game, joysticks, reed (magnetic), Hall effect (magnetic), carpet, floor, window, door
potentiometers: standard, multiturn, linear (slider), joystick
variable resistors: flex/bend, force sensing (FSR), photocell, thermistor (temperature)
other sensors: wind/air pressure, accelerometer, infrared (proximity), ultrasonic (proximity)
lights and control: relays, LEDs, electroluminescent wire, low-voltage DC bulbs (with transistor drivers), AC switching (with relays)
robotics: DC motors, stepper motors, solenoids

Electrical Specifications

Power input requirement	9 VDC, 300 mA or greater
Regulated power output	5 VDC from “+5v” terminal; 500 mA max, or 1 A with heat sink added to voltage regulator
I/O Terminals	20; #1-10 general purpose including analog input; #11-20 general purpose excluding analog input; #20 shared with status LED, open drain type output
Terminal input voltage	low level = 0v - 0.8v; high level = 2.0v - 5.0v
Terminal output voltage	low level = 0v typical, 0.6v max; high level = 5.0v typical, 4.3v min
Terminal input current	1 µA
Terminal output current	25 mA per terminal (source or sink); 200 mA total for all terminals (source or sink)

System Exclusive Specification

Terminal setup	<Header><Terminal #><Mode><Mode data...><Trailer>
Special command	<Header><Special cmd><Special data...><Trailer>
Header	<0xF0><0x00><0x77><0x01><0x01>
Trailer	<0xF7>
Mode	<Off = 0x00 | Digital out = 0x01 | Analog out = 0x02 | Digital in = 0x03 | Analog in = 0x04>
Mode data (Off)	none
Mode data (Digital out)	<Cmd code + Channel><Data1><Off thresh><On thresh><Polarity>
Mode data (Analog out)	<Cmd code + Channel><Data1>
Mode data (Digital in)	<Cmd code + Channel(off)><Data1 (off)><Data2 (off)><Cmd code + Channel(on)>... ...<Data1 (on)><Data2 (on)>
Mode data (Analog in)	<Cmd code + Channel><Data1><In min><In max><Out min><Out max>
Cmd code	<MIDI command value & 0x70> = <Note off = 0x00 | Note on = 0x10 | Poly pressure = 0x20 | Control change = 0x30 | Program change = 0x40 | Channel pressure = 0x50 | Pitch bend = 0x60>
Channel	<MIDI channel (0x00-0x0F>
Data1, Data2	<MIDI data value (0x00-0x7F)>
Off thresh, On thresh	<Data threshold value (0x00-0x7F)
Polarity	<Negative = 0x00 | Positive = 0x01>
In min/max, Out min/max	<Scale value (0x00-0x7f)>
Special command	<Factory reset = 0x62 | Slow factor = 0x63 | PWM resolution = 0x64 | PWM period = 0x65>
Special data (Factory reset)	none
Special data (Slow factor)	<Factor (0x00-0x7f)>
Special data (PWM resolution)	<Resolution (0x02-0x07)>
Special data (PWM period)	<Period bits 15:14><Period bits 13:7><Period bits 6:0>