Renard Controllers

with a brief discussion of dimming protocols and methodology

Compiled by Brian Ullmark (budude)

With many thanks to Phil Short for his contributions to our hobby and this presentation.

August 17th, 2012

Topics Discussed

- Renard controller defined
- Types of Renard Controllers
- Functional Blocks
- Renard Protocol
- DMX Protocol
- Data Input/Output Circuitry
- AC Dimming w/Zero Cross with/without PWM
- Zero Cross/TRIAC Circuitry
- DC Dimming w/PWM
- DC Output Circuitry

What's a "Renard" anyway?

- The original Renard controller was designed by Phil Short in 2006 on the ComputerChristmas forum. There had been discussions of using a microcontroller to offload the work from the PC and Phil came up with the basic Renard controller. It used the same principle of the older '595' based controllers where data was sent serially to one or multiple controllers in the same path.
- The initial design was an 8-channel version that drove a pair of external 4-channel SSRs. For the most part, this design has not changed all that much from that time.
- The board used the PIC 16F688 microcontroller which at the time represented one of the best combinations of cost, memory size and features such as the UART to provide the serial interface to the PC and other controllers.

Renard Controllers Types

- With On-Board AC outputs/drivers:
  - Renard SS8, Renard SS16, Renard SS24
  - Renard24HC (high current)
  - Simple Renard 16LLC (low current)
- With On-Board DC outputs/drivers:
  - Renard48LSD, Ren4Flood
  - Ren24LV
  - Simple Renard RGB+W
- Those that require external SSRs (AC or DC)
  - Renard 64XC
  - Simple Renard 8, 16, 24, 32

Renard Controller Types – cont'd

- Note there are several others not shown here but they are not as commonly used. DIYC member dirknerkle has an entire array of small controllers using variations of the above but focused more on using wireless.
- In general, the non-Simple series use the 16F688 PIC microcontroller configuration for their core logic control and all share the same basic code.
- The "Simple" Renard series were designed to be easier to build and safer for new folks since they do not contain any high voltage components on the boards. They have fewer components and use different PIC microcontrollers but use a common code base as the other Renard controllers do. Most of the Simple series do not have on-board drivers which makes them flexible to drive any external AC or DC driver/SSR boards.

Renard Controller Functional Blocks

- Any Renard Controller can be broken into four major blocks:
  - Power Supply
  - Input/Output serial interface circuitry
    - RS-232 (input only)
    - RS-485
    - PIC to PIC communication (on/off board)
  - Timing circuitry
    - Oscillator
    - Zero Cross Detection
  - Output driver
    - Optoisolator
    - TRIAC (AC)
    - Transistor/MOSFET (DC)

Renard Protocol

- Besides the HW, Renard also defines the data protocol from the PC to the controller(s).
- Using Vixen as a typical sequencer, the plug-in collects the channel data at each interval timing (e.g. 50 mS) and packages the data into a Renard packet.
- A Renard packet consists of the following:
  - Byte 0 = 0x7E (sync byte) – indicates start of packet
  - Byte 1 = Command/Address Byte – usually 0x80
    - < 0x80 – not normal Renard protocol – retransmit entire packet as-is with sync byte (not common)
    - > 0x80 – for downstream controller – decrement by 1 - retransmit entire packet as-is with sync byte
    - = 0x80 – pull out next 8 bytes as dimming data – **retransmit remaining data** in packet format with 0x80 as Command/Address Byte again
  - Byte 2-n = Data representing channel stream collected by plug-in above
- This method of pulling the data out of the stream and passing on only the remaining may seem odd at first but it makes the boards auto-addressable and easy to set up!
- Because the protocol depends on sync bytes versus specific timing marks such as those used in DMX it's a bit more flexible with different interface speeds.

Renard Protocol – cont'd

- Special characters and other oddities
  - As mentioned, 0x7E represents the start of a new Renard packet. Obviously dimming data from the sequencing could contain this value! To get around this, a special escape sequence consisting of 0x7F-0x30 is sent instead by the plug-in.
  - 0x7D is sent periodically (~100 bytes) as a pad byte to allow for re-synchronization between the PC and controller transmitter/receiver. This can happen if the serial clock timing is not exactly the same between the two. This command could also be part of the data stream so an escape sequence of 0x7F-0X2F is sent by the plug-in.
  - The escape character itself 0x7F, can be represented in the data stream as well so an escape sequence of 0x7F-0x31 is sent instead.

Renard Code Variants

- Renard/DMX
  - Accepts standard DMX data stream consisting of Start Code (preceded by Break/MAB to sync frame) and up to 512 bytes of dimming data
  - Renard/DMX firmware matches the address byte to index within the data stream and pulls off 8 bytes and then **retransmits the remaining data** (not the original DMX stream – very important) with new DMX stream.
- Renard/Start Address Firmware
  - Similar to standard Renard/Serial firmware but uses an x8 offset to index the data stream. This is typically used with wireless broadcast methods used by the Ren-W boards. See the Wiki for an excellent explanation.
- Renard Diagnostic Code
  - Used to diagnose common issues with the board – again – visit the Wiki for an explanation of its use.

DMX Protocol

- **This is just a basic description of how DMX works – read the various specs for all the gory details!**
- DMX-512 was originally developed in 1986 by the USITT (United States Institute for Theatre Technology) Engineering Commission and updated again in 1990. It is now administered by the ESTA (Entertainment Technologies Services Association). DMX512 is also known as ANSI E1.11-2008. It is used extensively in stage shows all over the world.
- A DMX packet starts with a Break condition - a logic low - for a certain amount of time followed by a Mark condition (aka as the Mark After Break or MAB) – a logic high – for a certain amount of time. This is in contrast to the special codes used by the Renard protocol.
- After the Break/MAB is detected, the controller knows to start receiving the actual data stream. The stream consists of a Start Code and then up to 512 bytes of data which is typically lamp dimming data but could be used for other purposes. Typically the Start Code is 0x00 but can be other values for special DMX gear.

DMX Protocol cont'd

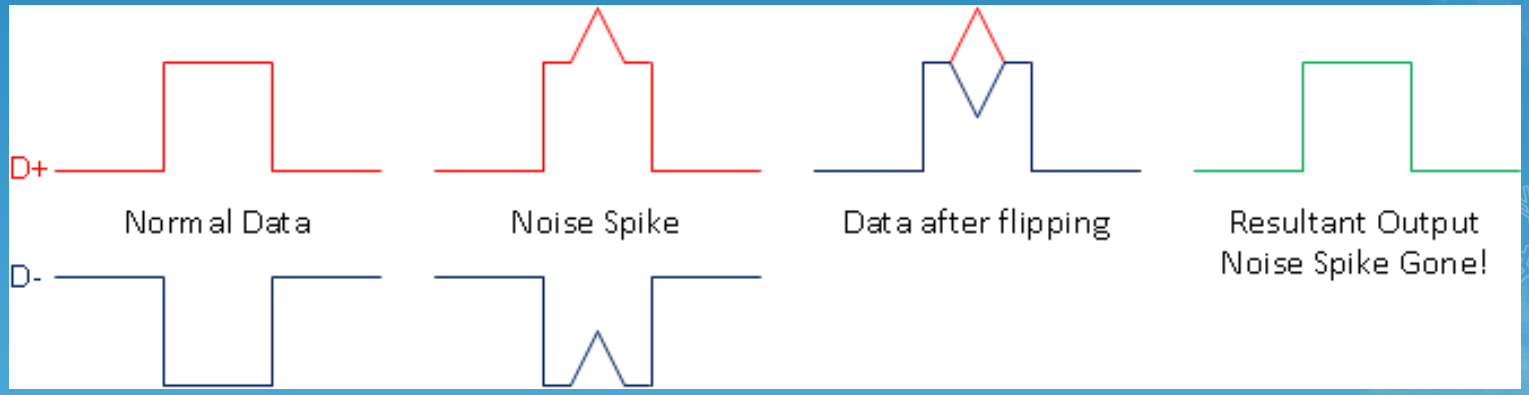
- DMX512 always uses RS-485 for it's transport and runs at 250Kbps. Note there are things such as "Hyper-DMX" and other turbocharging methods but they are not standards.
- Devices can be programmed with a starting address and they are responsible for determining the correct byte(s) of data from the packet to use. Normally all data is passed from the IN to OUT connectors (typically they are not even labeled as IN or OUT for this reason).
- DMX512/RS-485 lines must be terminated at both ends of the line – typically on transmitter (e.g. a USB-DMX dongle) and on the last device in the chain. This is to reduce reflections that can occur and cause noise up and down the transmission line.
- ACN (Architecture for Control Networks or ANSI E1.17), sACN (Streaming ACN or ANSI E1.31) and Art-Net are methods of transporting multiple DMX streams over Ethernet instead of using lots of RS-485 transmission lines. We'll leave this one for another day…

Data Flow - Input

- So – we now have data streaming into the board from the PC via Vixen with the Renard Plug-In (as an example). Let's see how it gets to the PIC!
- The data comes in from an RS-232 or RS-485 interface into the RS-485 chip.
  - RS-232 is a single-ended transmission method with data received on a single wire. Typically good for about 50 feet.
  - RS-485 is a differential transmission method with data received on two wires but the data is opposite polarity on each wire which is later compared with the same polarity at the receiver. This makes it very immune to noise and allows very long distances between devices (1000+ feet)
- The RS-485 chip translates these analog levels into a clean 5v logic signal that goes into pin 5 of the first (or only) PIC on the board.

RS-485/DMX Advantages

- RS-485:
  - As mentioned before, RS-485 uses differential data transmission – but what does this mean?
  - An example is shown below where a noise spike is introduced onto the line and how differential data transmission helps with this.



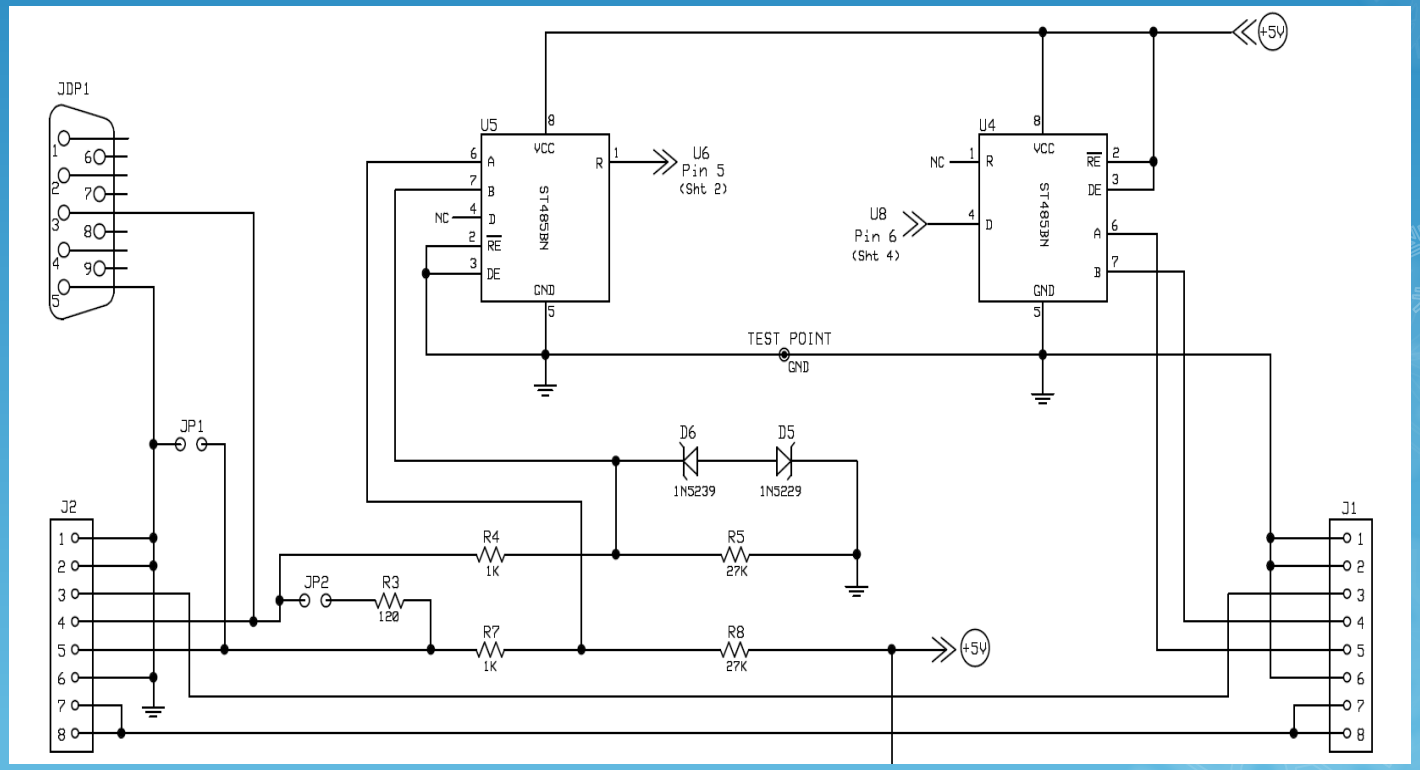
- DMX
  - Because of the much higher data rate than standard Renard/Serial (typically 57.6 Kbps), DMX can support a full set of 512 channels at 25mS interval timing without any issues. Renard/Serial is limited to only 142 at 57.6Kbps and 25mS interval timing. The smaller interval allows you to create smoother fade/ramps in your sequencing.

Data Flow - Output

- Recall how the Renard protocol works. There may or may not be data for another PIC after the first PIC retains its 8 bytes. If there is, it is repackaged into a complete Renard packet and sent out pin 6 of the PIC.
- If there are other PICs on the board, then this output from pin 6 goes into pin 5 (input) of the next PIC – and so on for each PIC on the board.
- If there is more data than contained within the board, the data is sent from pin 6 on the last PIC to the input of the RS-485 output chip. The conversion process translates the 5v logic level to RS-485 (differential) output – always – no RS-232 from this point on.
- Logically, from the sequencer/plug-in standpoint, you should view all of the individual boards as a single large controller. For instance three Ren24SS boards look to the sequencer as a single 72 channel controller.

Renard Data Stream Circuitry

- Data is received on pin 4 with RS-232, pins 4/5 with RS-485 on the IN jack. Data is transmitted on pins 4/5 (always RS-485 remember). The Zener diodes and 1K/27K resistors are for over-voltage and short circuit protection.



Pinouts, Pinouts, Pinouts…

- Standard DMX gear, LOR and Renard use different pinouts for their connections – yet all use RS-485.
- DMX standards have officially defined the following as the standard for an RJ45 interface (either IN or OUT):
  - Pin 1 – Orange/White – Data+ #1
  - Pin 2 – Orange – Data- #1
  - Pin 3 – Green/White – Data+ #2
  - Pin 4 – Blue – N/C
  - Pin 5 – Blue/White – N/C
  - Pin 6 – Green – Data- #2
  - Pin 7 – Brown/White – Data Link Common for Data #1
  - Pin 8 – Brown – Data Link Common for Data #2
- The DMX standard supports two DMX streams on an RJ45 interface - however it's not as commonly used. It comes down to using pins 1, 2, 7 and 8 for a single data stream.

Pinouts, Pinouts, Pinouts… cont'd

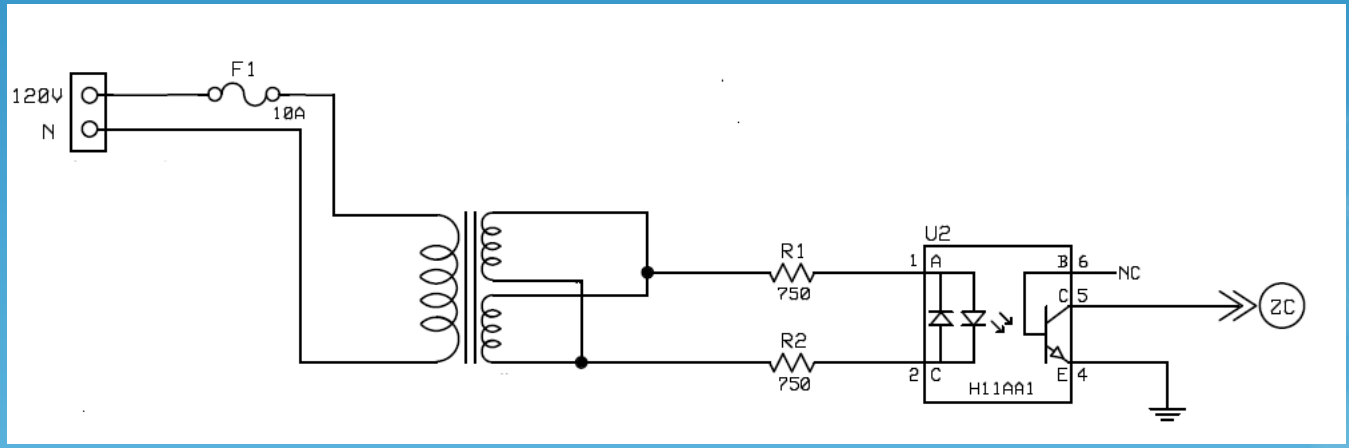
- LOR Pinout (either IN or OUT):
  - Pin 1,2,7,8 – N/C
  - Pin 3 – Green/White – +10Vdc (not always present)
  - Pin 4 – Blue – A/Data+
  - Pin 5 – Blue/White – B/Data-
  - Pin 6 – Green – Data Ground
- Renard Pinout (IN):
  - Pin 1 – Ground
  - Pin 2 – Ground
  - Pin 3 – Passes to OUT Pin 3 (originally used for ZC passage)
  - Pin 4 – Blue – B/Data- OR RS-232 Rx (OUT is Data- only)
  - Pin 5 – Blue/White – A/Data+ OR Ground (OUT is Data+ only)
  - Pin 6 – Ground
  - Pin 7 – Tied to Pin 8 and Passes to OUT Pin 7/8
  - Pin 8 – Tied to Pin 7 and Passes to OUT Pin 7/8

AC Dimming

- OK – so now the PIC has 8-bytes of channel data stored away that it pulled from the data stream – now what?
- For AC Dimming, the PIC needs to synchronize itself with the AC input line frequency. To do this, the controllers contain an opto-isolated zero-crossing detection chip (H11AA1). Essentially, it determines when the AC line crosses the 0 voltage level and converts it to a logic level signal to the PIC. This results in a signal being sent every 1/120 (remember – it hits zero twice!) or 8.33mS.
- The signal to the PIC on pin 4 interrupts the code flow and triggers the start of the dimming operation for all eight channels.

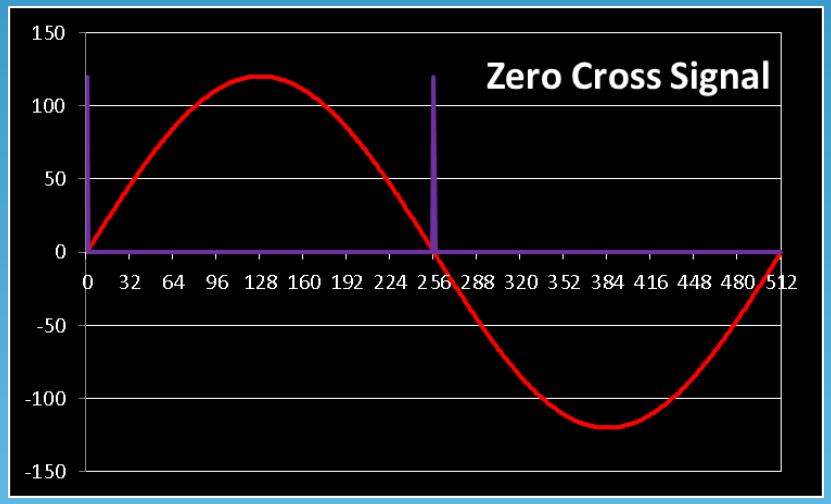
Zero Cross circuitry (Ren24SS)

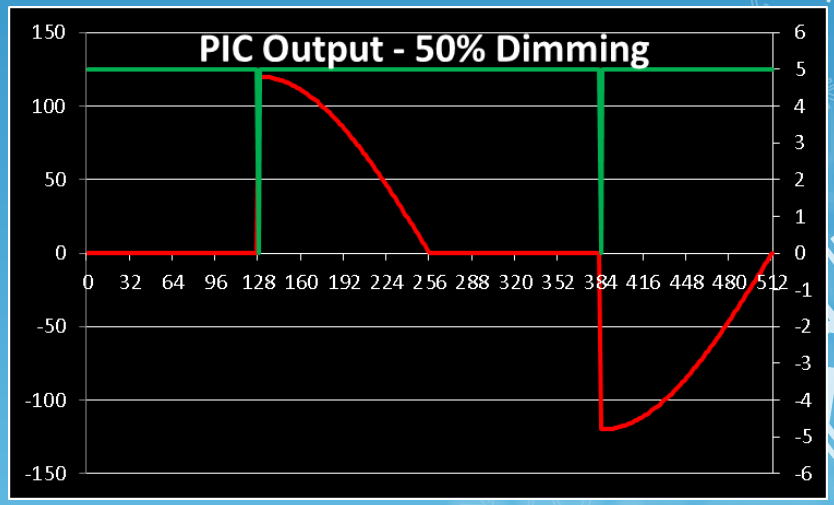
- Here you can see the zero cross detection circuitry from the Ren24SS. The two 750 ohm resistors limit the current to the internal LEDs within the chip. These LEDs turn on the transistor within and provide the trigger to interrupt the PIC to start the dimming cycle again.



AC Dimming – cont'd

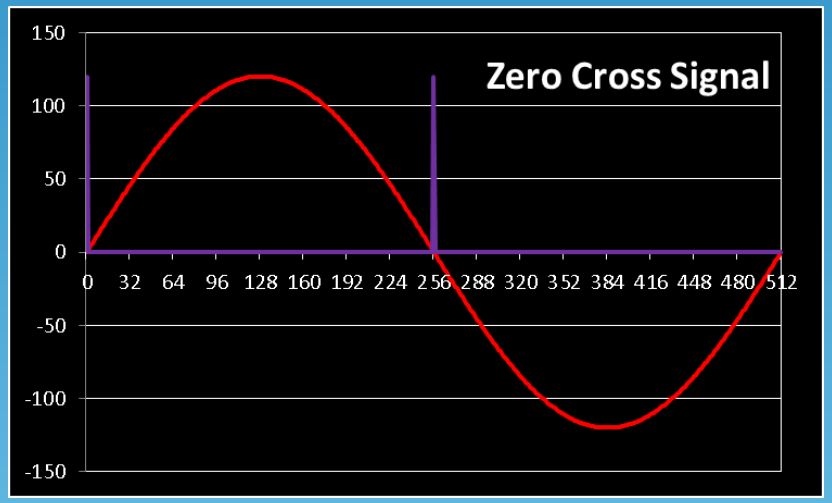
- Dimming works by turning on the output at a particular point within the AC half-cycle. The higher the dimmer value, the sooner the output is turned on, the lower the value, the later the output is turned on. Since we know the half-cycles occur every 8.33 mS, the controller needs to divide up that time period into 256 steps.
- The dimming value is counted down and when it reaches zero, the PIC turns on the output which in turn triggers the TRIAC. The output is shut off but the TRIAC stays on until the voltage reaches zero at which point the process starts over again. It does this for all eight channels on each output.

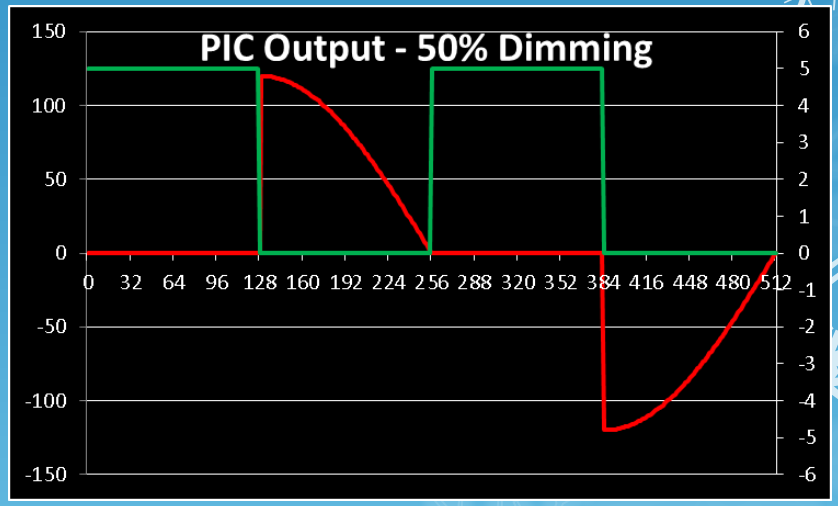




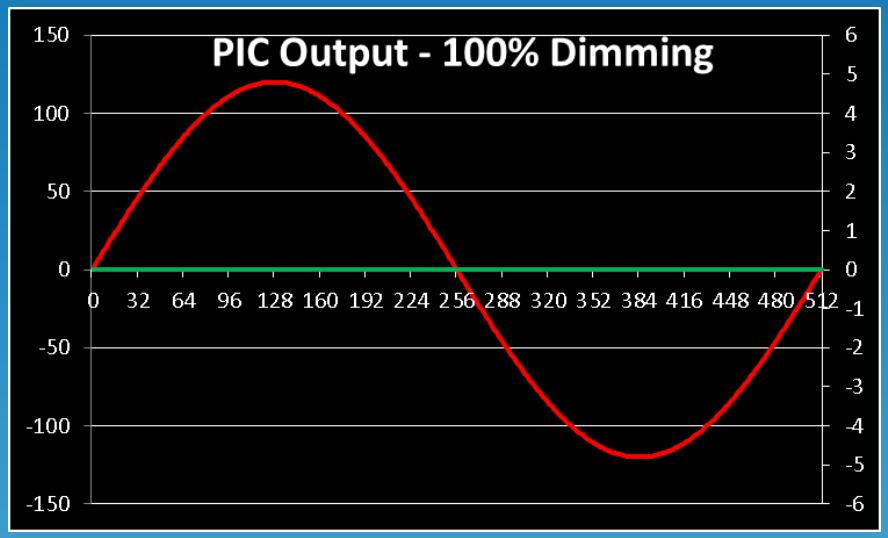
AC Dimming – cont'd

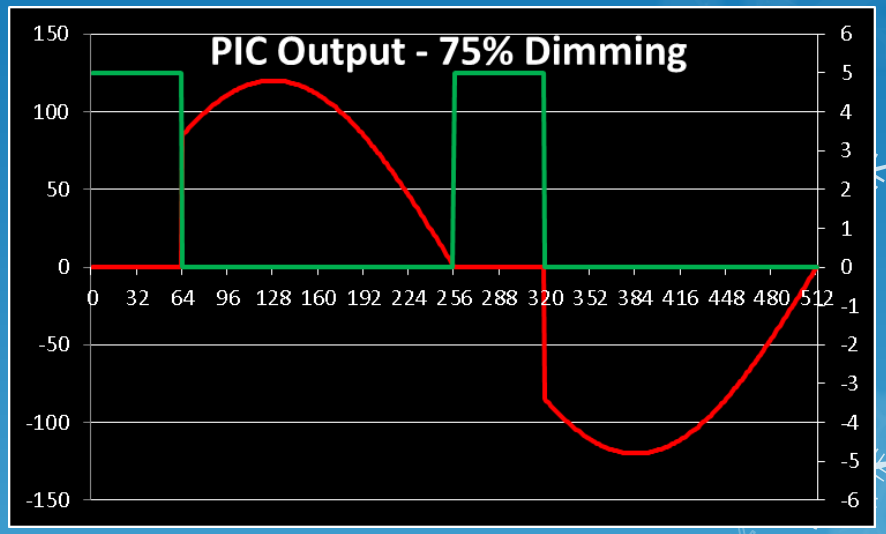
- Because of variances in the line voltage, differing load, and the TRIACs themselves, the TRIAC may turn off a bit sooner or later than it should which can sometimes cause LED lights to flicker a bit more than incandescent lights.
- To get around this, the PIC code has a feature to enable Pulse Width Modulation or PWM on the outputs. This turns on the output for the entire duration of the dimming cycle versus a single pulse which helps ensure the TRIAC stays on the entire time. PWM code is pretty much the standard version used on all Renard controllers for this reason.

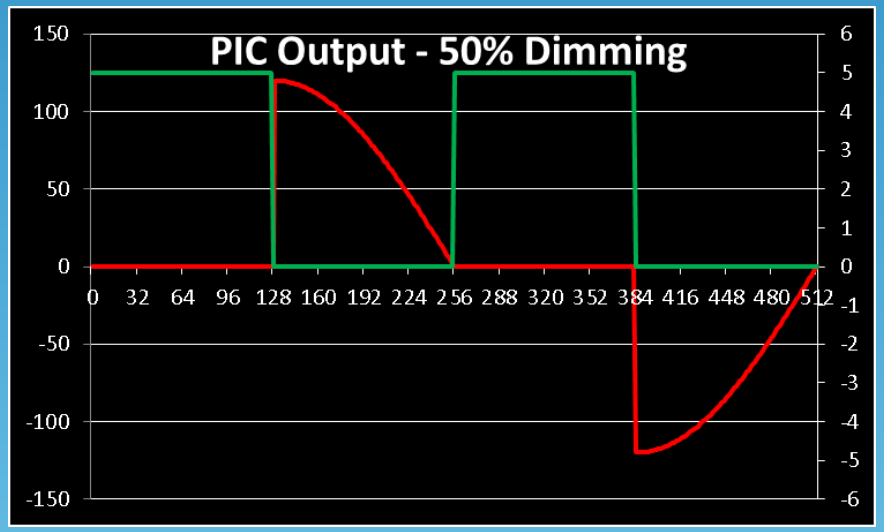


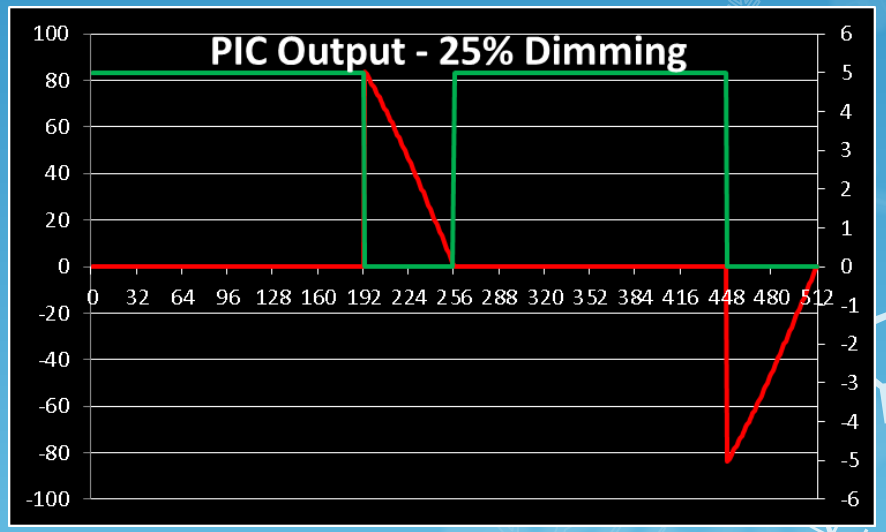


- Here are examples of the output being dimmed from 100%, 75%, 50% and 25%.



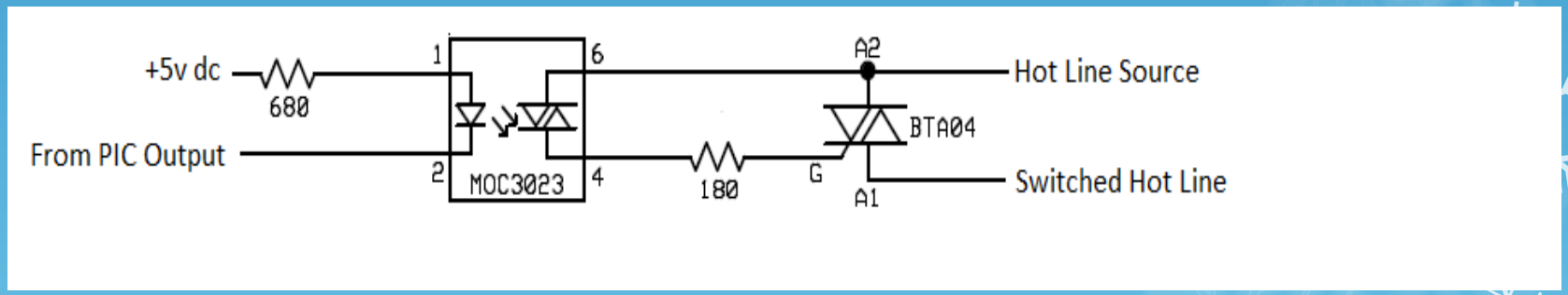






TRIAC Output circuitry

- Below is the TRIAC output from the Ren24SS. The MOC3023 provides line voltage isolation to prevent any possibility of 120v reaching the logic portions of the board as well as back to your PC and yourself!
- The 680 ohm resistor provides current limiting for the opto-isolator LED and the 180 ohm resistor limits current flow through both the opto-isolator output and the Gate of the TRIAC. Changing the value of the 180 ohm resistor allows the TRIAC to work with different AC voltages such as 24vac and 220/240vac.
- This is the same circuit used on the external SSR boards as well so the principal is the same. Note also that the PIC "sinks" the current from 5v source through the opto-isolator input. This means you have to drive the PIC output LOW to turn ON the AC line via the opto-isolator.



DC Dimming

- DC Dimming always uses the PWM feature to drive the outputs since the drivers (transistors/MOSFETs) do not latch on as the TRIACs do.
- DC Dimming by itself does not require any synchronization to the AC line and therefore no zero circuitry is required.
- However – if you have a controller that is using both AC and DC SSRs, the controller must be synced as with normal AC dimming. This is why boards like the Ren64XC and Ren Simple boards can drive both types of SSRs while a Ren48LSD (no ZC chip – in fact no AC at all!) cannot.
- Attempting to drive an ACSSR with a DC only board would result in the PIC output not being synced to the AC line and you would therefore get very erratic dimming (essentially useless).

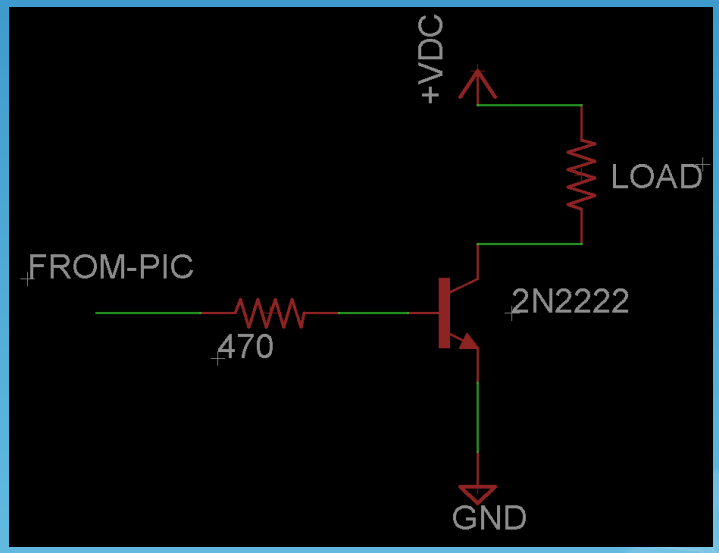
DC Output Circuitry

- There are several methods to drive DC level outputs. Typically the method used is decided by how much current is required to be switched by the drivers.
- The Ren24LV uses ULN2803 driver chips which have a Darlington pair output. This results in very little current from the PIC to turn on the output. The ULN2803 can drive up to 400 mA on a single output but not all at the same time. In that example it is limited to 125 mA per output.
- The Ren48LSD use simple NPN transistors that provide up to 400 mA per output. The PIC outputs drive the transistor base on through a current limiting resistor which enables current to flow from the Collector to the Emitter.
- The DCSSR uses heftier MOSFET (Metal-Oxide Semiconductor Field Effect Transistor) which work in a similar fashion to normal transistors where a gate input enables a channel (aka the “Field Effect”) from the Source to the Drain allowing current to flow. MOSFETs can provide anywhere from 1A or so up to tens of amps but the designs are usually limited by the board traces/connector/wiring

DC Output Circuitry – cont'd

- Note that regardless of which method is chosen, the PIC output typically needs to go HIGH to turn on the transistor gate/base for DC operation. This is opposite to the AC SSR method of sinking (PIC output LOW) current from the opto-isolator circuit. This must be taken into account when installing Renard code on the PIC. There are options in both the Renard/Serial and Renard/DMX code to either SINK or SOURCE power to/from the final outputs. The Renard64XC is an example where both AC and DC SSRs can be driven from the same controller providing the proper code is installed on the PIC.

- Transistor Output:



Inductive Loads

- Inductive vs Resistive loads
  - Inductive loads use magnetic fields. Examples - Motors, solenoids, and relays. If it moves, it's probably an inductive load.
  - Inductive loads can cause a surge of voltage created by the collapsing magnetic field in the inductor/coil known as a kickback voltage. This kickback voltage can be **much** higher than the voltage to the inductive load. Circuits should be protected from this by diodes or other forms of protection.
  - Resistive loads (lights/LEDs) convert current into other forms of energy such as heat so there is risk of kickback.
  - A diode placed across the coil/inductor leads can reduce this kickback voltage and protect your outputs. These are sometimes called snubber diodes. There are other options as well using Resistor/Capacitor (RC) circuits, back-to-back Zener diodes and Metal Oxide Varistors (MOVs) as well.

Questions?