Blog

LED Video Panel at Maker Faire 2014: Concept and Development

Posted by Paul Stoffregen in Embedded on May 24, 2014 12:34:00 AM

Freescale's booth at Maker Faire in San Mateo featured a large 4320 LED video panel, controlled by a single Teensy 3.1 board powered by Freescale's Kinetis K20 microcontroller (Cortex-M4).

Flash Required

Flash is required to view this media. Download Here.

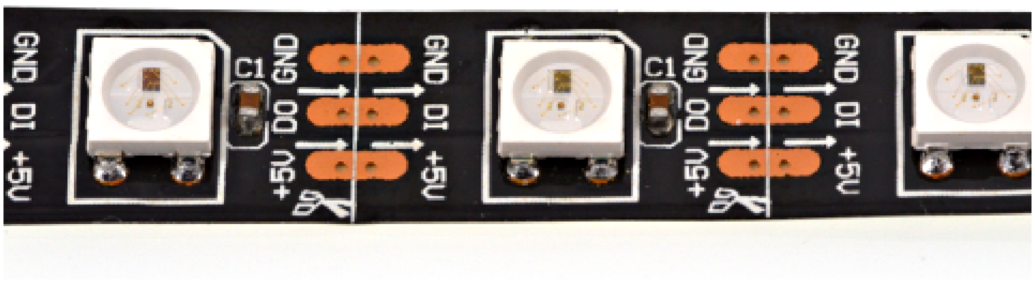
Erin Murphy, Paul Stoffregen and Robin Coon at PJRC assembled this LED panel over a period of 3 weeks leading up to Maker Faire. In this article, I'll show how we built it and how it works.

LED Strips

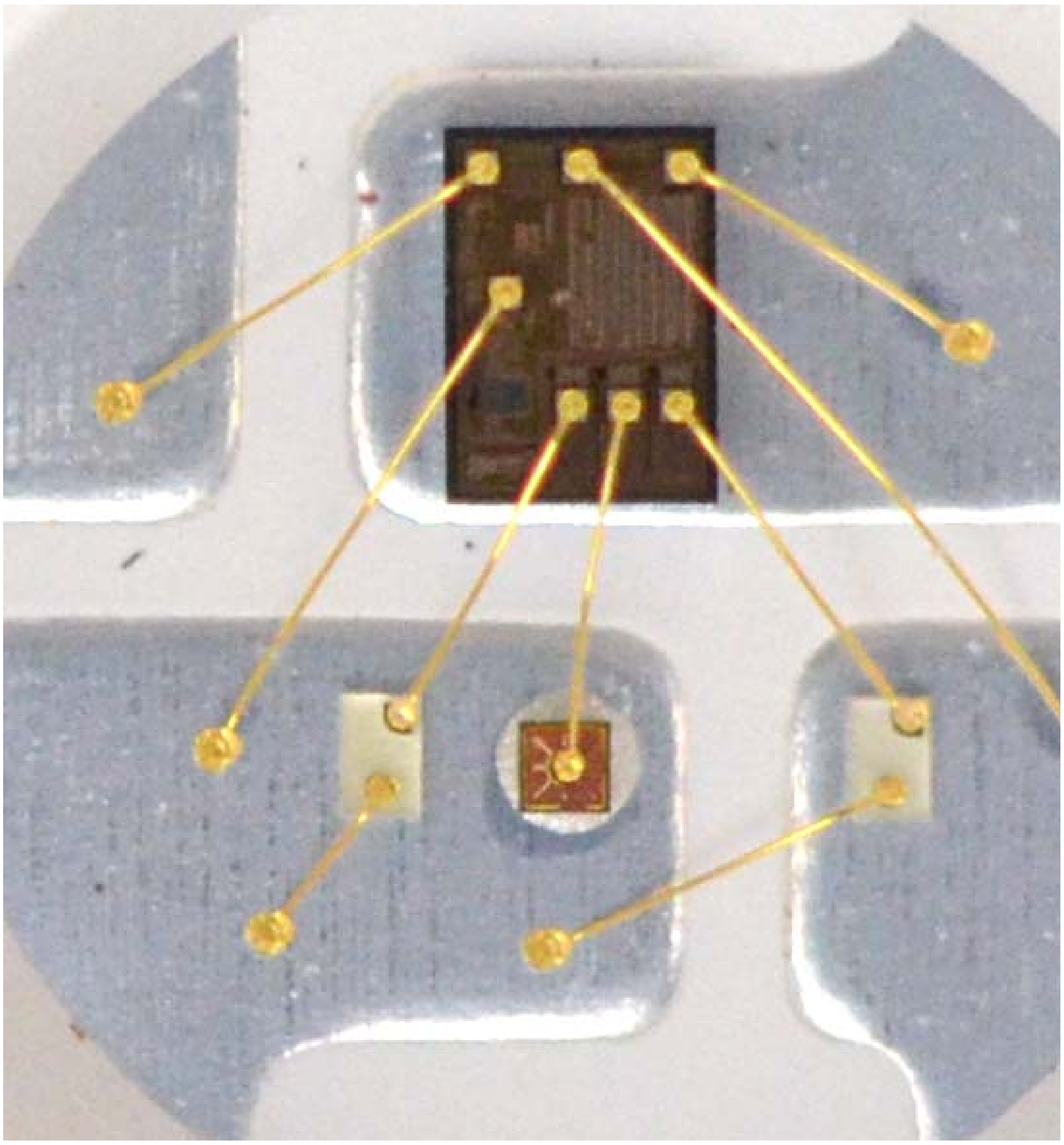
The LEDs are sold as strips 4 meter strips with 60 LEDs per meter.

These are "addressable" WS2812B LEDs, meaning the color (intensity of red, green and blue) of each LED can be individually controlled.

The LED strips use 3 wires, 2 for power and a single data line referenced to the power ground. Each LED has a data input (DI) and data output (DO) pin, allowing data to flow down the entire strip.



Inside each WS2812B LED is a WS2811 controller chip and the red, green and blue LED chips.



These WS2812B LEDs are the least expensive addressable LEDs on the market. They can be purchased from numerous Chinese merchants on AliExpress and DHgate .

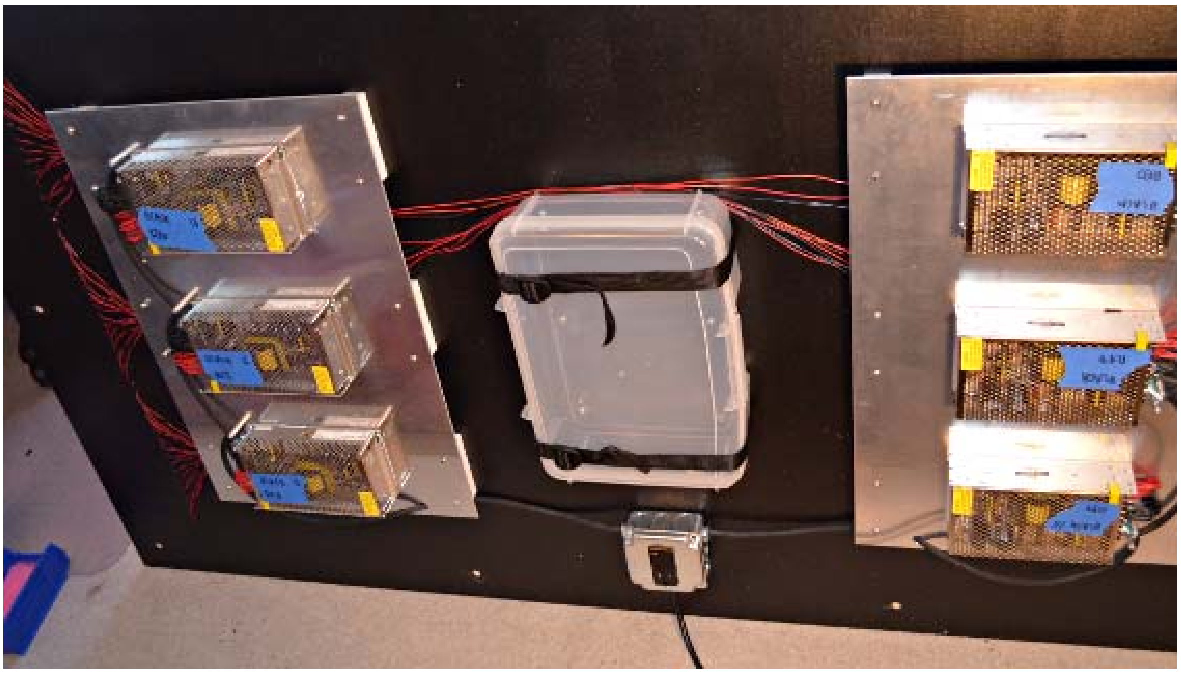
The special signal format used by the single wire data line creates some special technical challenges when driving a large number of LEDs, which are discussed below.

To construct the 4320 LED panel, 18 of these 4 meter reels were used. The 4 meter lengths were cut into two sections of 1.5 meters, and the remaining 1 meter sections were and soldered together, until a total of 48 sections, each 1.5 meters (90 LEDs) were made.

Power Supplies

Each LED consumes approximately 1/4 watt when illuminated fully white (red, green & blue at full intensity). When all 4320 LEDs are on, the maximum power is approximately 1000 watts!

To supply so much power, six 200W power supplies were used. The power supplies are mounted to aluminum sheets, separated from the plywood base by 1.5 inches, to allow air flow for convention cooling.



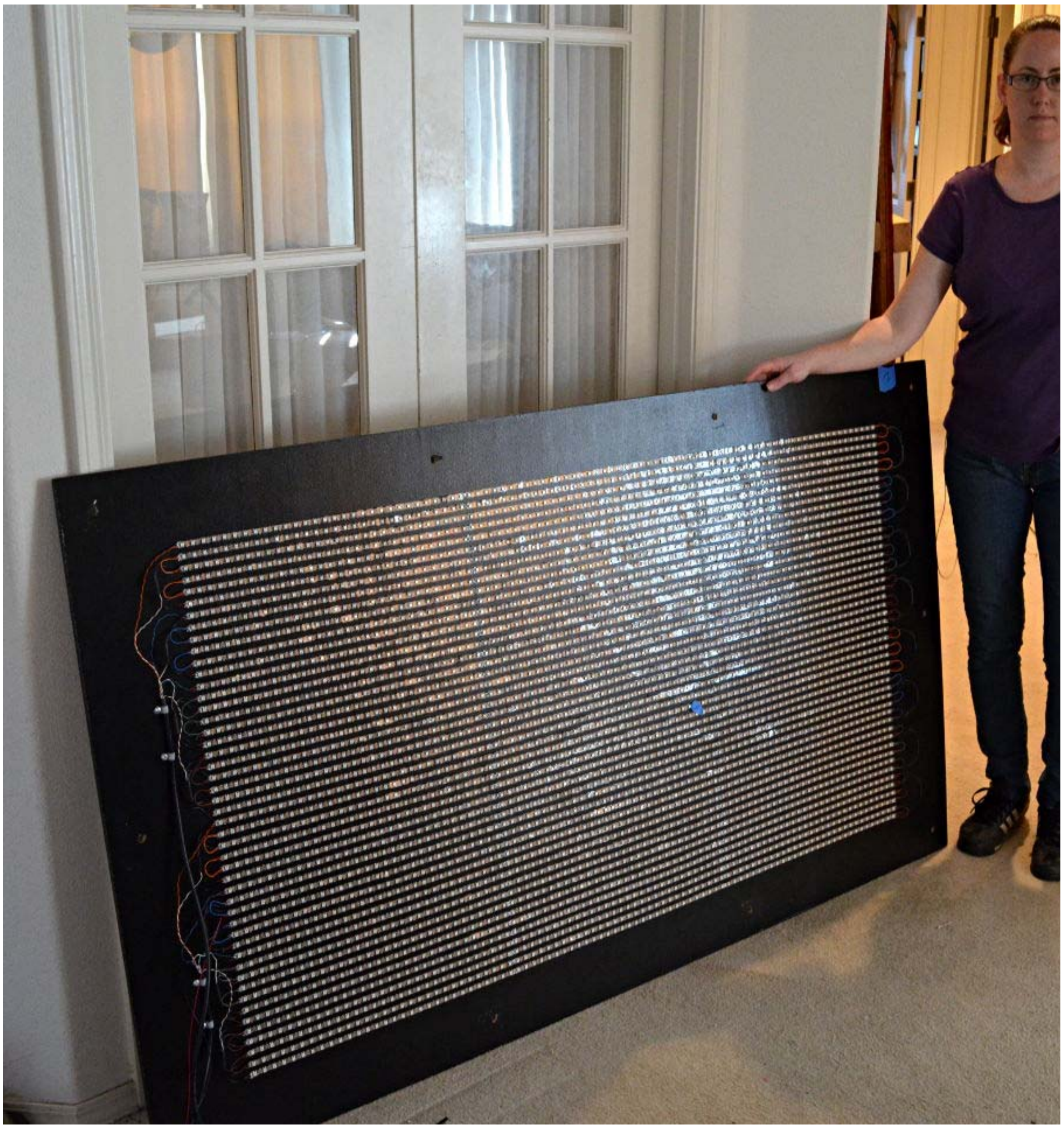
Each power supply can provide 40 amps of current at 5 volts. When all LEDs are turned on, the total current draw is approximately 200 amps. To allow such a large current flow, each LED strip is connected by a pair of 20 gauge wires on each end. Preparing and soldering 192 wires turned out to be quite a time consuming job (thanks Erin).

Construction

Physically constructing such a large LED display requires many seemingly simple but very time consuming steps. Preparing the plywood panel, with 2 coats of white primer, over 100 carefully drilled holes, and 2 coats of black paint is no small task.

The aluminum heat dissipating sheets and plastic spacer blocks also required many accurate drilled holes.

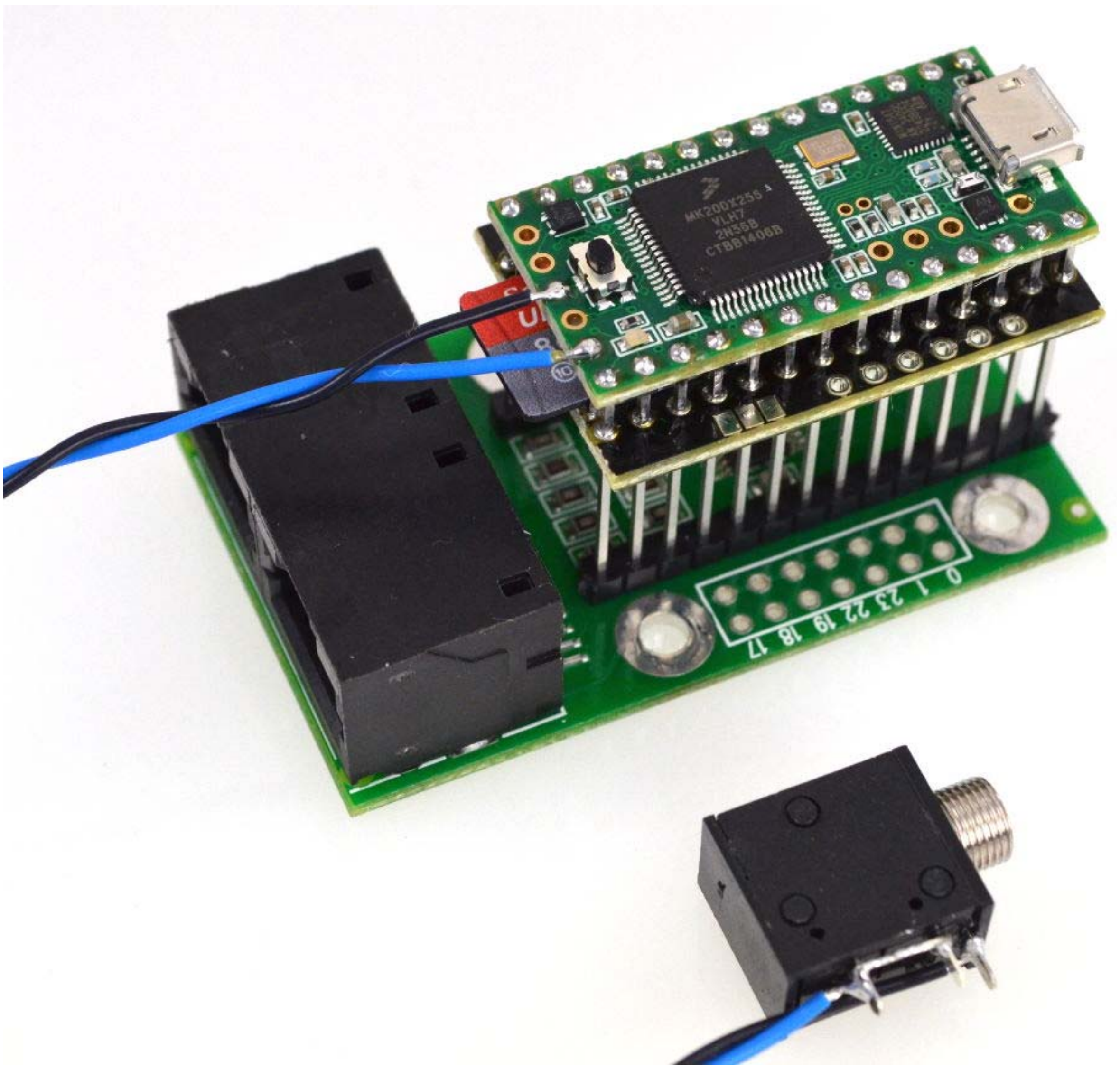
Soldering hundreds of power wires, routing and connecting them to the power supplies, and accurately positioning 48 strips onto the panel is a task that seems simple, but requires very careful attention to detail. Even small misalignment can skew the distance between LEDs, resulting in a visually poor result.



Erin Murphy spent many hours carefully aligning all these LED strips on the panel.

Electronics

All 4320 LEDs are controlled by a single Teensy 3.1 development board, powered by a Freescale Kinetis K20 (Cortex-M4) microcontroller.



Mounted underneath the Teensy 3.1 board are a WIZ820+SD Adaptor , for connecting a Micro SD card which stores the video data, and an OctoWS2811 Adaptor , which provides a buffer chip and impedance matching resistors to drive the data signals.

Audio output (the blue wire in the photo above) is taken directly from the 12 bit on-chip DAC. Ordinary PC computer speakers were used for the sound output.

Not visible in the photo above is special wiring for the Micro SD card CS signal. Pin 4 is cut between the Teensy and WIZ820+SD, and between the WIZ820+SD and Octo board, pins 3 and 4 are shorted together. This routes the SD card's CS signal to pin 3. Pin 4, the default wiring for the WIZ820+SD adaptor, can't be used because pin 4 it's reserved by OctoWS2811. (more info here)

A Sandisk Ultra Micro SD card was used to store the video data file.

Software

Teensy 3.1 is programmed using the Arduino IDE. Arduino is a simplified programming environment intended for novices. It's also very useful for rapid prototyping.

A beautiful aspect of Arduino is very impressive projects can often be made with relatively simple code which leverages advanced libraries to do all the "heavy lifting". In this project, the OctoWS2811 library handles updating the LEDs, the Audio library takes care of sound output, and the Arduino SD library (based on Bill Greiman's SdFat library) allows the Micro SD card to be read easily.

> **Update:** The source code for this project is now an example included with the OctoWS2811 library. Simply install Teensyduino into a copy of Arduino, selecting the optional libraries during install. Then, in Arduino, you can open this code from **File > Examples > OctoWS2811 > VideoSDcard**.

The data is stored uncompressed, with 5 byte headers before each block of video or audio data. The 5 byte header was chosen as a simple and arbitrary way to label each block of data within the file.

The video playing code is extremely simple. If a header begins with the "*" character, the following data is a video frame. The remaining header bytes indicate how larger it is, and how many microseconds are intended to elapse before displaying it.

```
if (sd_card_read(header, 5)) {
  if (header[0] == '*') {
    unsigned int size = (header[1] | (header[2] << 8)) * 3;
    unsigned int usec = header[3] | (header[4] << 8);
    if (sd_card_read(drawingMemory, size)) {
      while (elapsedSinceLastFrame < usec) ; // wait
      elapsedSinceLastFrame -= usec;
      leds.show();
```

In this code fragment, the while loop deserves some explanation. The variable "elapsedSinceLastFrame" is a special "elapsedMicro" variable type, which is actually a C++ object that implement an integer which appears to increment 1000000 times per second. Each read of elapsedSinceLastFrame actually checks the system time, which is tracked by the ARM SysTick interrupt and present value. This is typical of programming in Arduino, where high level abstractions provide simple access to hardware features.

Extra real-time USB printing code (not shown here) was used to verify timing. It turns out approximate 17 of each video frame's 33.3 millisecond are spent in that waiting loop. The Cortex-M4 and Freescale's eDMA engine (used by the OctoWS2811 and Audio libraries) are incredibly fast.

The audio code works similarly. Any header beginning with "%" is assumed to be audio data. Every 256 bytes is giving to the audio library, which will automatically stream it to the Kinetis 12 bit DAC.

```
} else if (header[0] == '%') {
  unsigned int size = (header[1] | (header[2] << 8)) * 2;
  while (size > 0) {
    unsigned int len = size;
    if (len > 256) len = 256;
    int16_t *p = audio.getBuffer();
    if (!sd_card_read(p, len)) {
      error("unable to read audio frame data");
      return;
    }
    audio.playBuffer();
    size -= len;
  }
```

Video Data

This special uncompressed video format, with 5 byte headers and pixels formatted in OctoWS2811's transposed bit packing is utterly non-standard. A way to convert ordinary video to this format was created for this project.

The image conversion is done by a Processing based program, called movie2sdcard. The video is simply played at normal speed, with an event handler for each video frame. Processing provides image scaling, then simple Java code transposes the pixel data into OctoWS2811's format and writes the binary data to a file. This output file can be played, but without the sound.

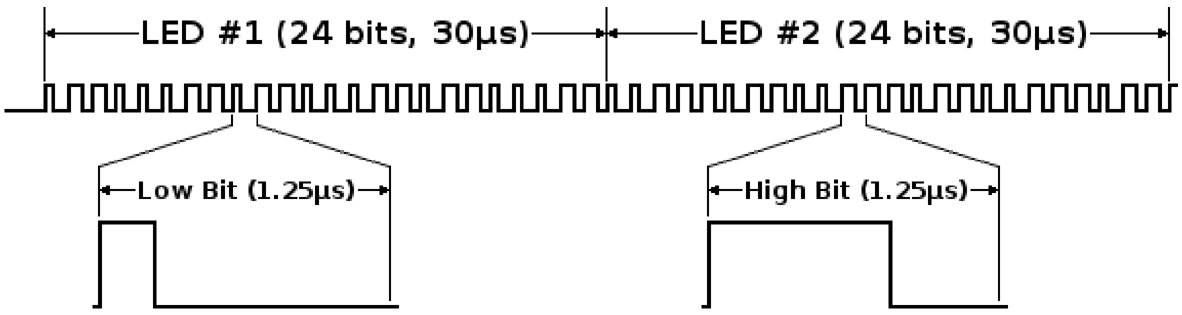
Adding the audio stream involves a few steps. First, ffmpeg is used to extract the audio data from the original video. The sox is used to convert the audio sample rate and store only the raw 16 bit data. Finally, a special "addaudio" program reads the raw data and the special video format, and writes a merged video and audio file with the 5 byte headers expected by the Arduino programming.

Complete source code, and additional details, are available at this forum post.

OctoWS2811 and Audio Library

Simple Arduino-style programming is possible because the difficult low-level hardware access is handled by sophisticated software libraries that interface with the Kinetis hardware.

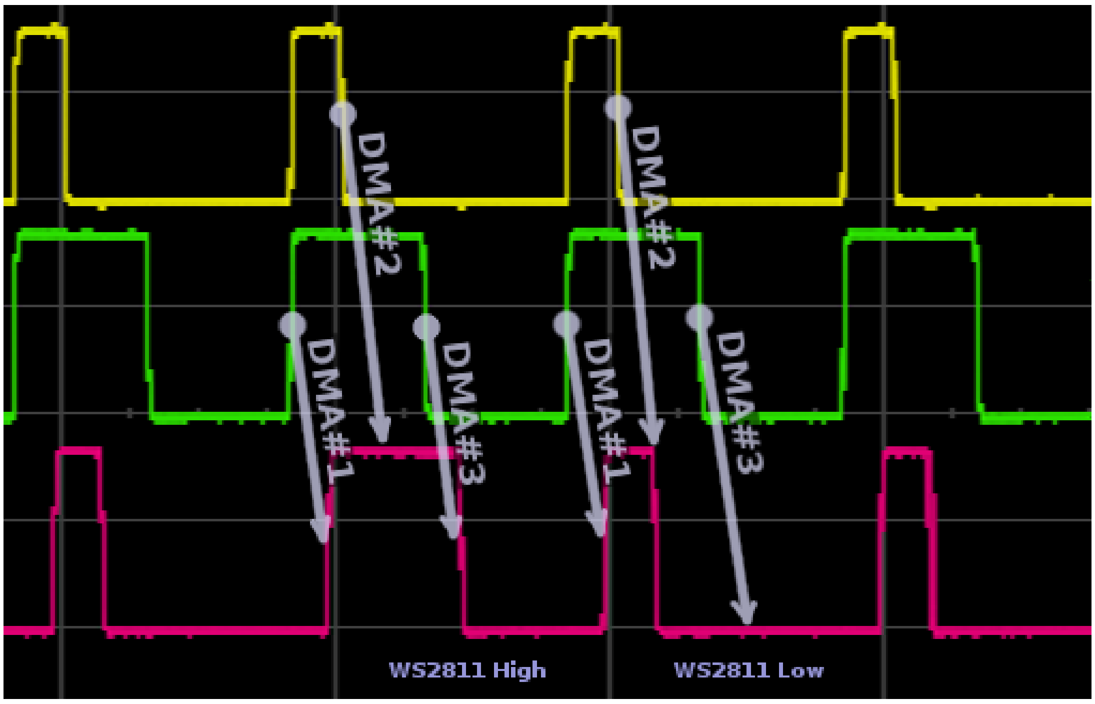
The WS2811 controller chips in each LED require a special 1-wire data format with precise timing. Data is always transmitted at 800 kbits/sec, where the width of each pulse indicates if it's a 0 or 1.



Each LED consumes the first 24 bits it receives, then begins passing all remaining data to the following LED. The first 24 pulse control the first LED, then the next 24 control the 2nd LED, and so on. A silent time of 50 us or greater marks the beginning of a new data frame.

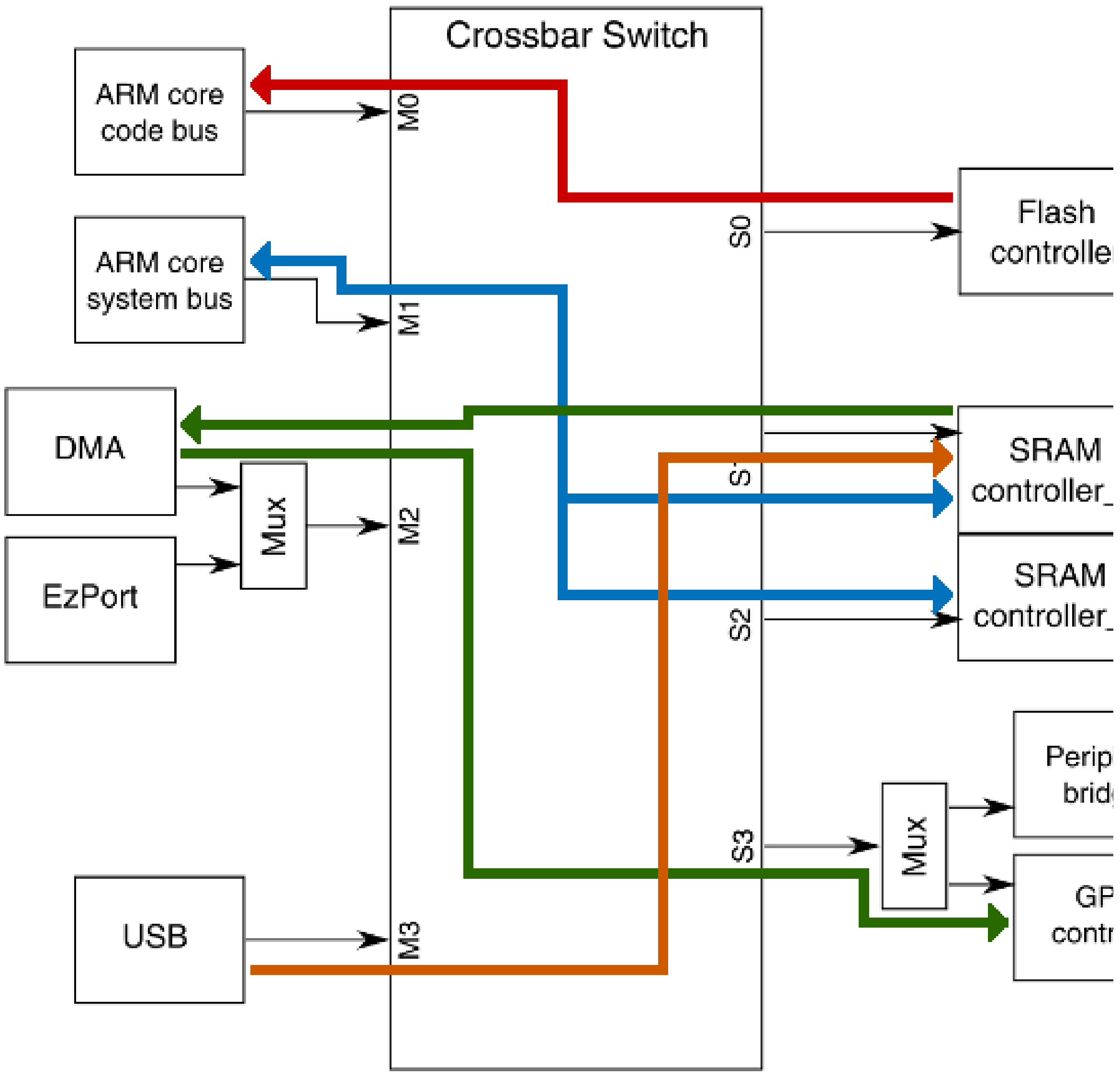
OctoWS2811 is named "Octo" because it transmits 8 streams of these pulses simultaneously. Because each LED requires 30 microseconds, eight parallel outputs allows a very large LED array to update 8 times faster. In this 4320 LED project, each of the 8 outputs controls 540 LEDs, which requires 16.2 ms, or approximately half of a 30 Hz video frame time.

To achieve 8 outputs efficiently, OctoWS2811 uses the Kinetis eDMA engine. Two PWM waveforms are created, corresponding to the 0 and 1 pulse shapes. The waveform edges trigger DMA events, which copy data directly from RAM to the Kinetis GPIO register than controls 8 pins.



DMA#1 always copies an all 1s byte (0xFF), causing all 8 bits to high to start a new output pulse. DMA#2 copies a byte from OctoWS2811's display buffer, causing the LEDs to receive a different length pulse. DMA#3 copies an all 0s byte (0x00), causing the waveform to always go low, before the process repeats. This DMA-based data copying allows the Cortex-M4 CPU to remain almost completely unused, except for extra bus utilization, while the LEDs update.

The Kinetis microcontrollers have a switched bus matrix and dual-bank RAM, which minimizes bus contention.



While the DMA controller is moving data from RAM to the GPIO register (the green path), the Cortex-M4 is able to simultaneously fetch code (the red path) and stack-based variables (the blue path).

The audio library also uses DMA to move audio samples from buffers to the 12 bit DAC. Audio buffers are managed using nested priority interrupts (a feature of the Cortex-M NVIC), where a high priority DMA completion interrupt causes a lower priority buffer management interrupt to run automatically. This interrupt priority nesting allows the Audio library to perform transparently manage all sound data transfer, for an extremely simple Arduino-style interface that doesn't burden novice programmers with rapidly responding to real-time events.

Teensy 3.1 aims to make these incredibly powerful hardware features, a fast 32 bit CPU core, complex DMA engine, nested prioritized interrupts, and switched bus matrix accessible to novices with simplified Arduino style programming through these libraries, so they can create fun projects never before possible on traditional microcontrollers!

13665 Views Categories: Embedded Software, Embedded Hardware

Tags: cortex-m, kinetis, development_board, how_to, led, maker_faire, teensy, led_video_panel

3 Comments

Login / Register to comment.

Lori Kate Smith May 27, 2014 9:13 PM

Thanks for taking the time to provide all the detail. It was a very popular demo. I didn't have an inkling of how much time it would take to build the board to hold the lights.

I hope that it finds other places to be displayed.

Actions Like (1)

jensbauer Jun 17, 2014 3:05 AM

Great article. I've had a look at WS2811/WS2812 LED strips, and it seems you can get various types.
If you are going to build a display like the one here, you will need LED strips that have 60 LEDs per meter.
But you can archieve a slightly higher resolution, if you buy strips that have 74 LEDs per

meter (5m reels here).
I just came across some 100 LEDs per meter strips, which would give you an even higher resolution, they're only 1 meter long, but if you ask the seller, he says that they can make them up to 2 meter in length.
I recommend *not* going for the 144 LEDs per meter, because they you cannot space them evenly; eg. same distance horizontally and vertically. Remember to measure from the center of the LEDs when spacing them out.

Actions Like (0)

Paul Stoffregen Oct 29, 2014 1:57 PM

More followup questions answered here...

http://forum.pjrc.com/threads/26943-LED-Video-Panel-from-Maker-Faire-questions

Actions Like (2)