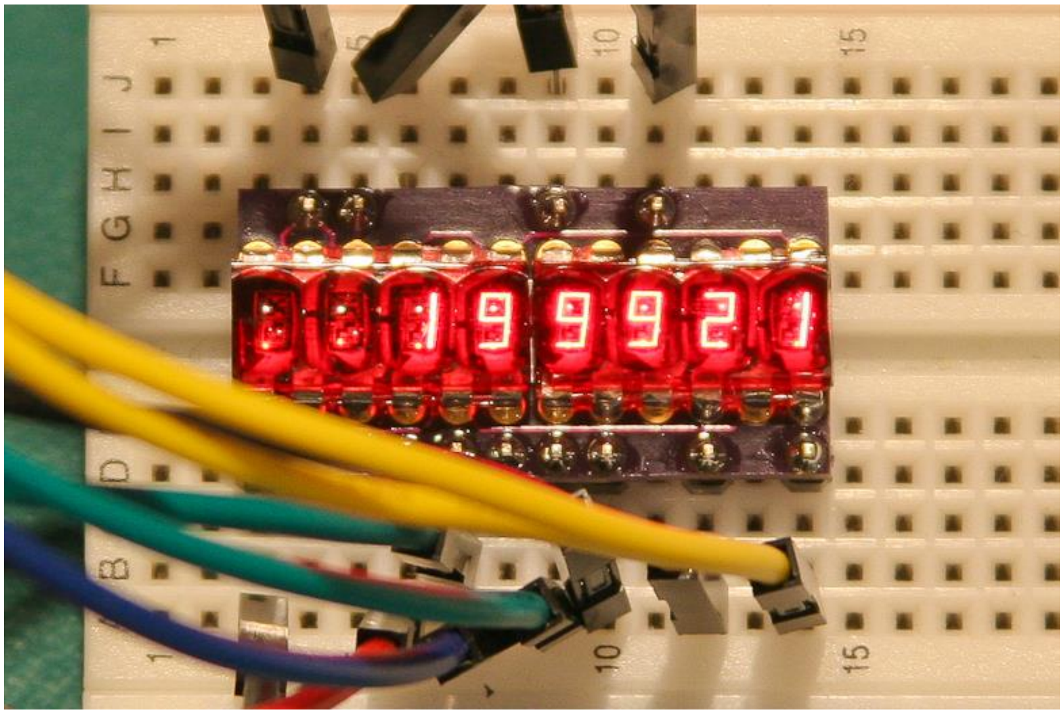
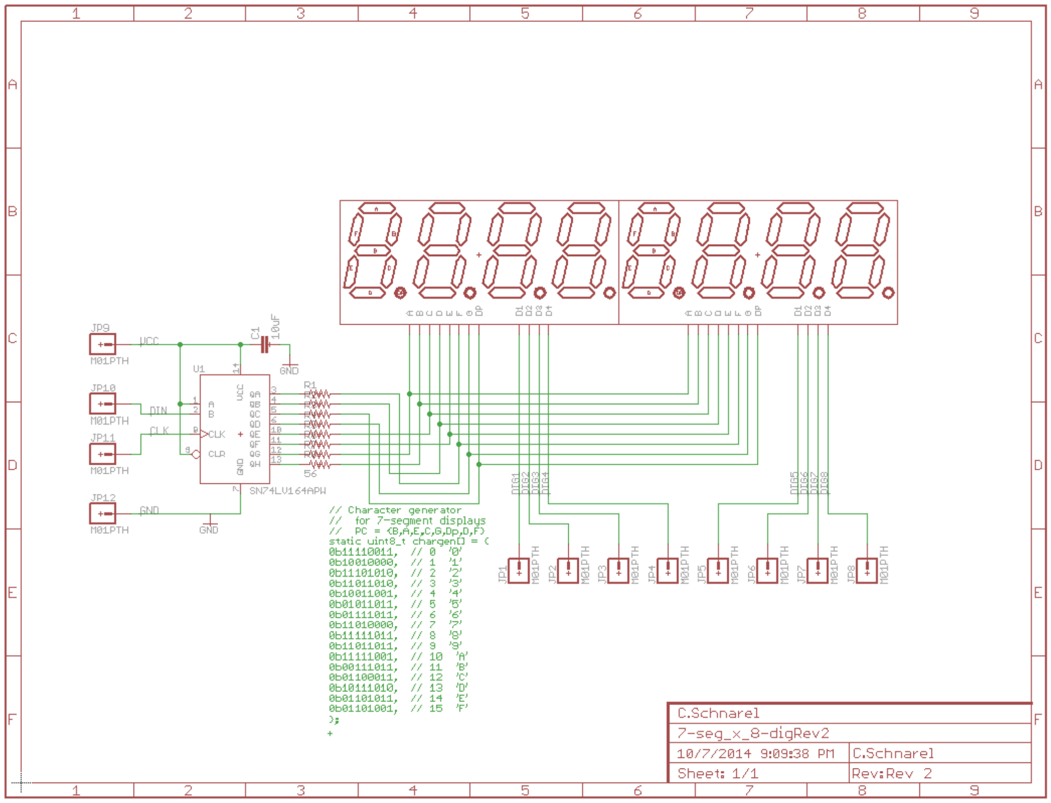
Hello!

I'm working on a new project that will use the *hp* Bubble display that is sold by SparkFun. I thought you might find this part of the project useful. The project is Arduino based and I have almost enough pins to drive two of the Bubble displays, maybe. Two displays would take 16 control lines. I don't have that many available, but I think I can devote 10 pins to the display. So I still have to reduce the pin count somehow. I could use the MAX7219CNG or the HT16K33 driver parts but those each cost as much as adding a second Arduino. Or I could use the Arduino I2C design from my earlier posting, but that takes up a lot of PCB space.

Instead, what I did was build a smaller display module that has a shift register on the same PCB as the displays. The shift register, an SN74LV164APW, costs less than $0.50 and doesn't take up much space. You could also use a SN74LV595APWR which costs a few pennies more and takes up just a bit more space.

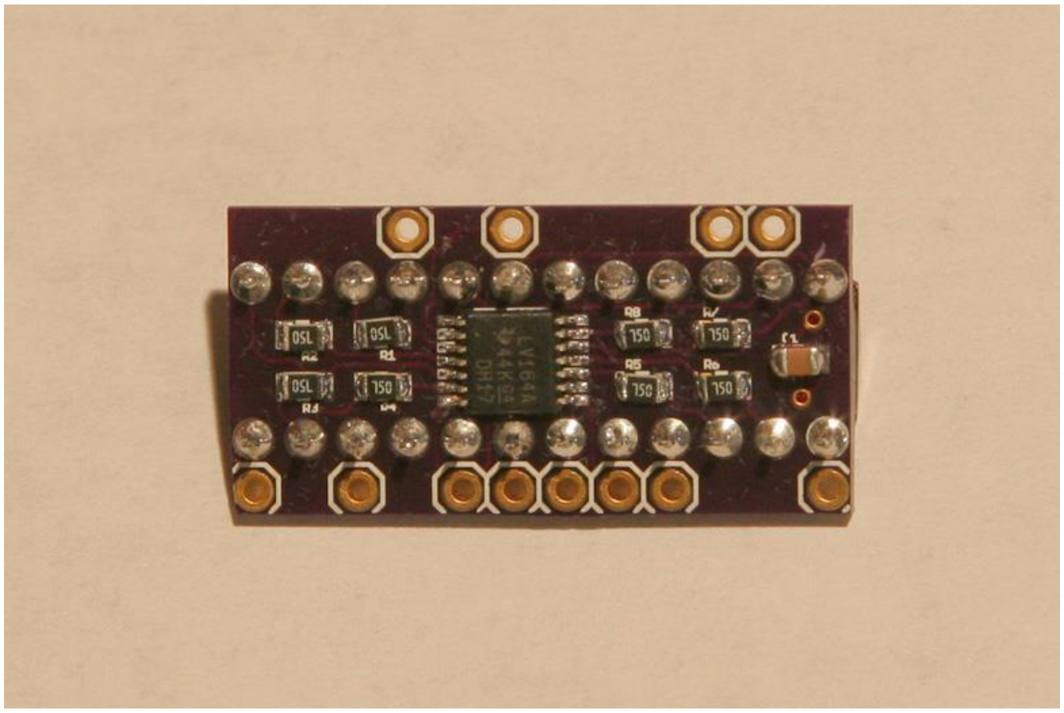


With the shift register, the eight (7 segments + decimal point) values are sent out serially using just 2 pins. This is not an I2C or SPI implementation. It's just a simple bit-bang serial drive directly to the shift register inputs. The other eight control lines, the digit selects, are driven directly from the Arduino.



```
// Character generator
//  for 7-segment displays
//  PC = {B,A,E,C,G,Dp,D,F}
static uint8_t chargen[] = {
0b11110011,  // 0  '0'
0b10010000,  // 1  '1'
0b11101010,  // 2  '2'
0b11011010,  // 3  '3'
0b10011001,  // 4  '4'
0b01011011,  // 5  '5'
0b01111011,  // 6  '6'
0b11010000,  // 7  '7'
0b11111011,  // 8  '8'
0b11011011,  // 9  '9'
0b11111001,  // 10  'A'
0b00111011,  // 11  'B'
0b01100011,  // 12  'C'
0b10111010,  // 13  'D'
0b01101011,  // 14  'E'
0b01101001,  // 15  'F'
};
```

What? You don't see the shift register? Here is a picture of the back side of the PCB showing the shift register hidden on that side along with the current limit resistors. Note that I put the resistors on the segment lines, not the digit lines, so that brightness stays even no matter which character is being displayed.

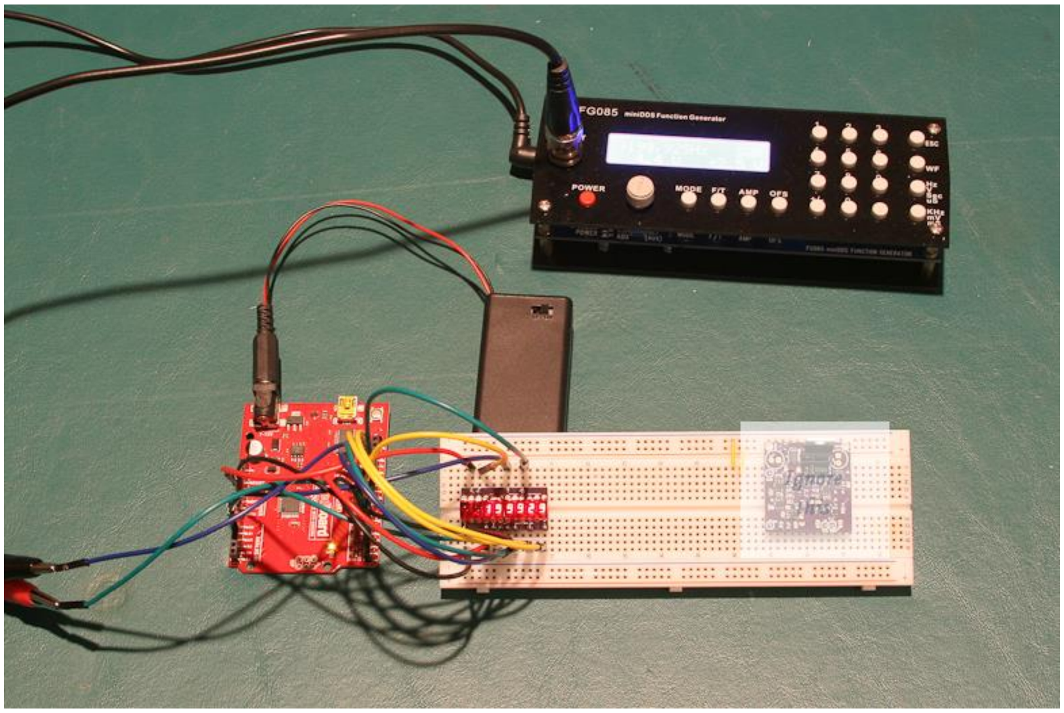


The shift register has nowhere near the drive capability needed to get maximum brightness from these displays. See my earlier posting for details on drive currents when multiplexing LEDs. In this case that is acceptable. It works well enough to be readable on my electronics bench and that's where this will be used.

To simplify and compact the PCB routing each shift register output is routed to whichever segment is most convenient. The simplified routing scrambles the segment order but we really don't care since the scramble is hidden in the character generator. In this case the bit order of the segments is (MSB to LSB) {B,A,E,C,G,Dp,D,F}.

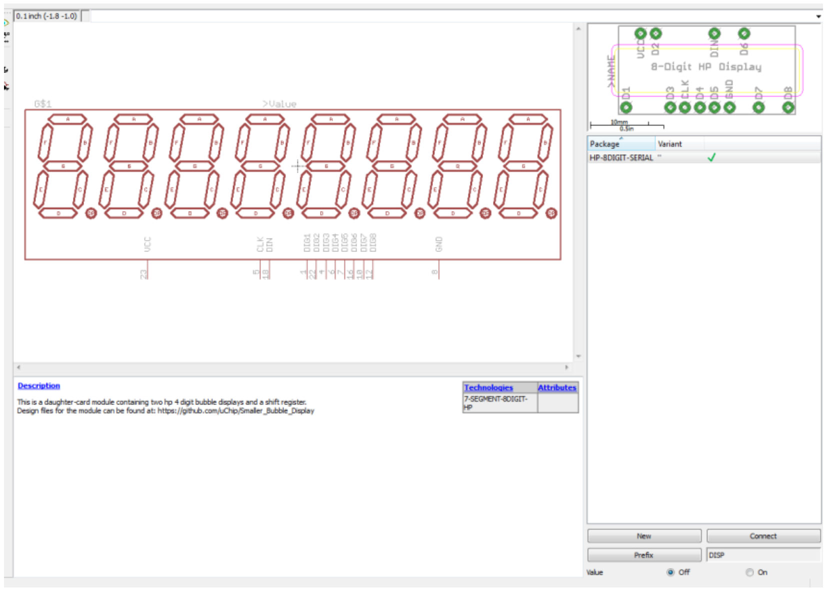
This is about the third time I've had to rewrite the character generator to scramble the bit order to match a PCB. I got tired of doing it by hand so I wrote a program to do it for me. It's a simple program. I actually wrote it as an Arduino sketch. The #define statements at the top of the sketch specify what bit order you want. The program then prints out compiler ready code. Just copy/paste the code from the Arduino runtime window into your sketch.

I needed some example code to test out and demonstrate the display module. I decided a good example would be to implement a frequency counter. A Google search will turn up several Arduino-based frequency counter implementations. I tried a few of them and like Martin Nawrath's the best. It is supposedly good to about 8 MHz and seemed pretty accurate and stable when I breadboarded it using my SparkFun RedBoard.



I modified Martin's code to output to the *hp* displays. I did not bother implementing the display code as an Arduino library since it's simple and often the display code will need to be interleaved with the application code to achieve the right timing.

Finally, since I want to be able to use this display module in multiple projects, I also made an Eagle component representing the module. This way I can just drop the module in wherever I need too.



Description

This is a daughter-card module containing two hp 4 digit bubble displays and a shift register.
Design files for the module can be found at: https://github.com/uChip/Smaller_Bubble_Display

Technologies	Attributes
7-SEGMENT-8DIGIT-HP	

Package	Variant	
HP-8DIGIT-SERIAL	''	✓

All the most recent design files and code for this project can be found in the following GitHub repositories.

- Smaller_Bubble_Display - small 8-digit, 7-segment, LED display module
  - Eagle files for display module & BOM
  - Eagle Library with Display module as a component
  - Display example code: Frequency counter based on Martin Nawrath's library
- Bit_Swizzler - Code to remap bits in character generator for 7-segment display

PCBs for the display module are available from OSHPark.com here.

Good luck in all your projects!

- Chip