

Information

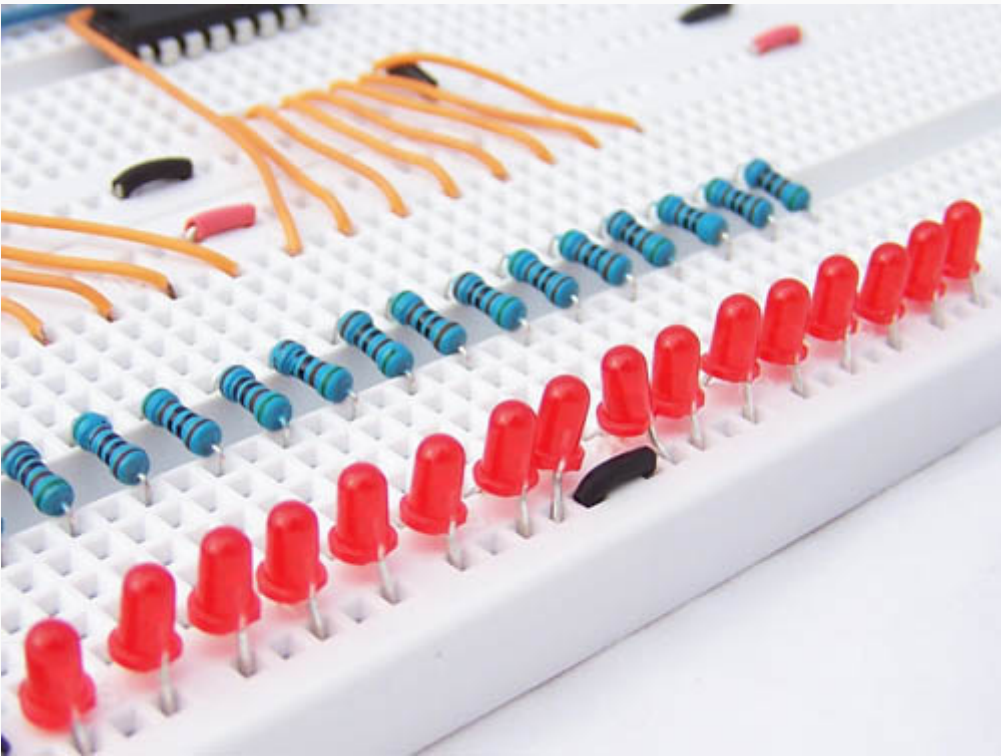
- [About Us](#)
- [Online Reviews](#)
- [Shipping & Returns](#)
- [Contact Us](#)



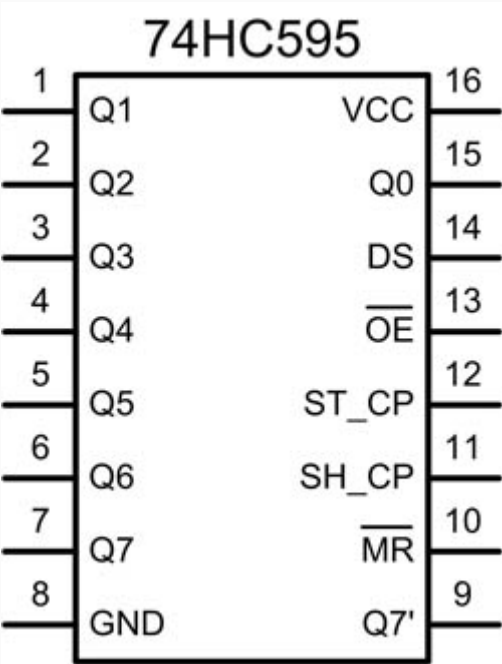
Introduction to 74HC595 shift register - Controlling 16 LEDs

[Read comments](#) [Leave a comment](#) 28-May-2010 9:23am

This tutorial shows you how to control 16 LEDs with just 3 control lines. We do this by daisy chaining [74HC595 shift registers](#)



The [74HC595 shift register](#) has an 8 bit storage register and an 8 bit shift register. Data is written to the shift register serially, then latched onto the storage register. The storage register then controls 8 output lines. The figure below shows the [74HC595](#) pinout.

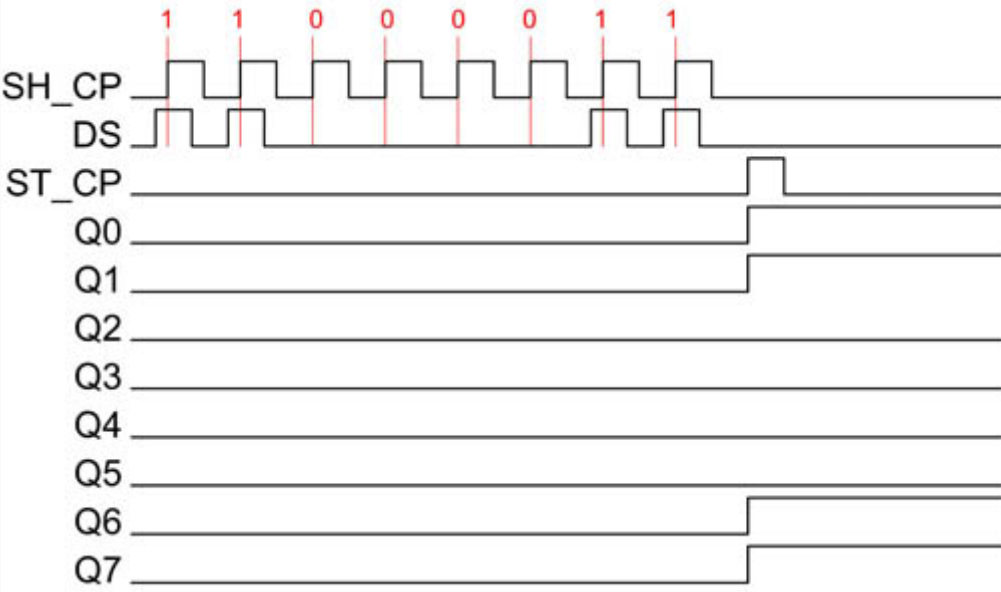


Pin 14 (DS) is the Data pin. On some datasheets it is referred to as "SER".

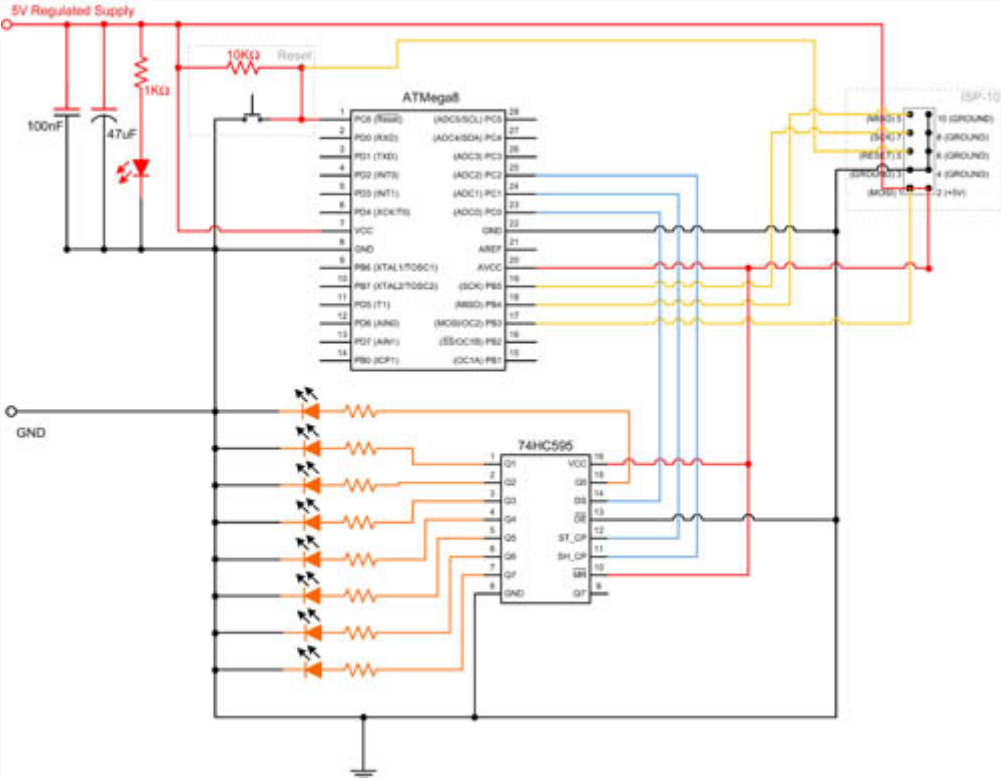
When pin 11 (SH_CP or SRCLK on some datasheets) goes from Low to High the value of DS is stored into the shift register and the existing values of the register are shifted to make room for the new bit.

Pin 12 (ST_CP or RCLK on some datasheets) is held low whilst data is being written to the shift register. When it goes High the values of the shift register are latched to the storage register which are then outputted to pins Q0-Q7.

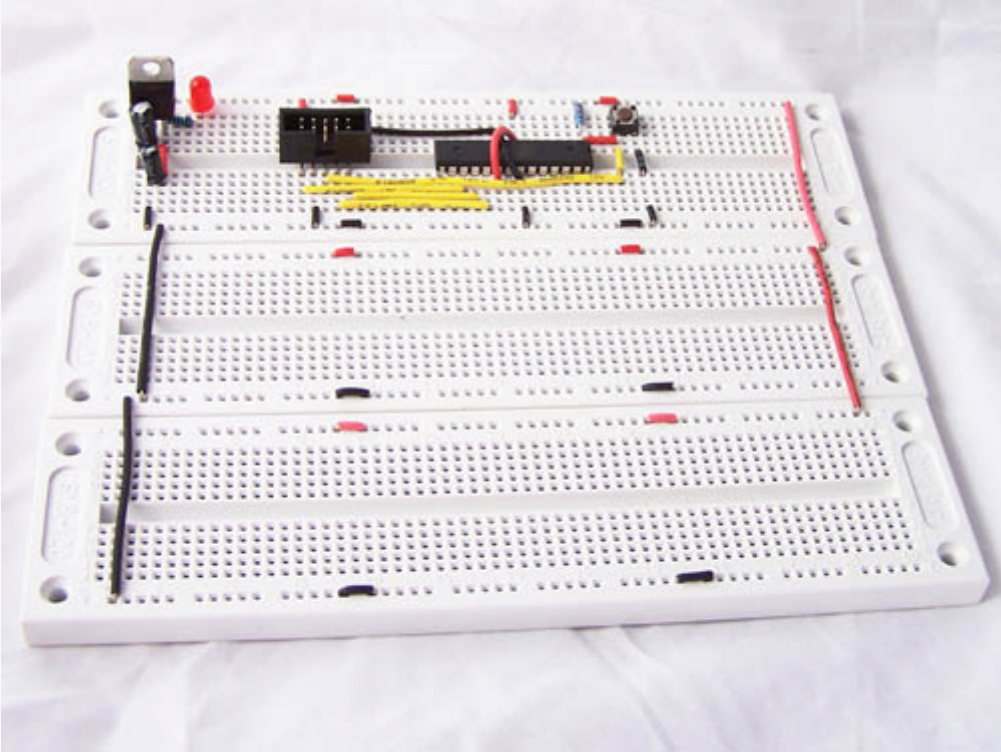
The timing diagram below demonstrates how you would set the Q0-Q7 output pins to 11000011, assuming starting values of 00000000.



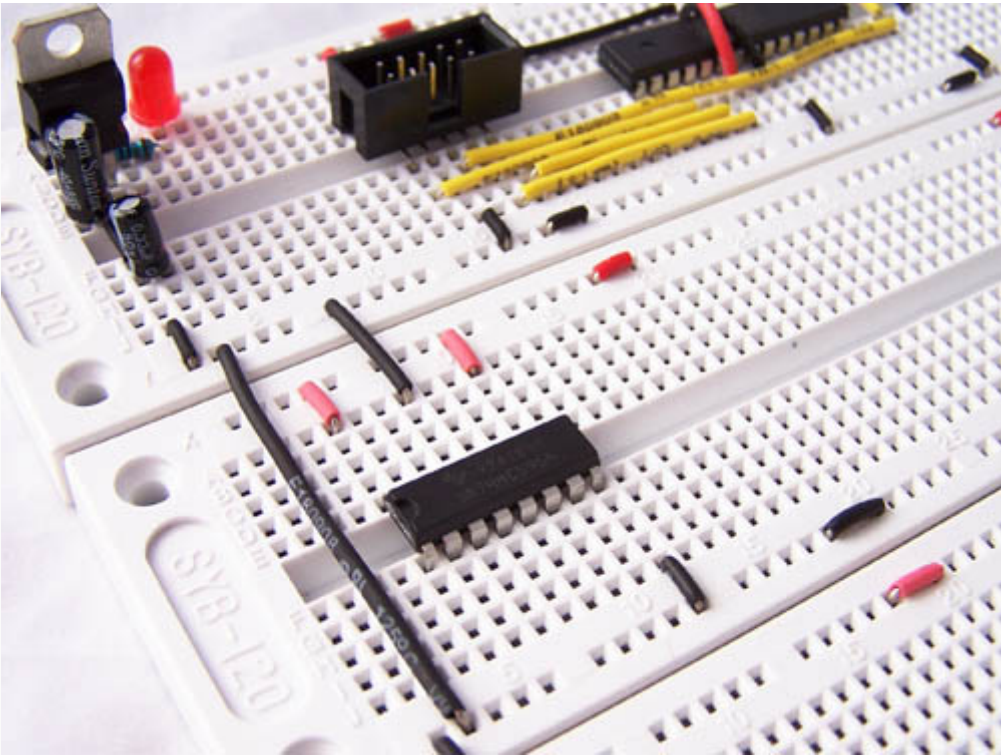
The circuit we are building is showed below, followed by the build steps



We will start with an [Atmega8 breadboard circuit](#) which we have used for many of our other tutorials. We add 2 extra breadboards and route power to these.



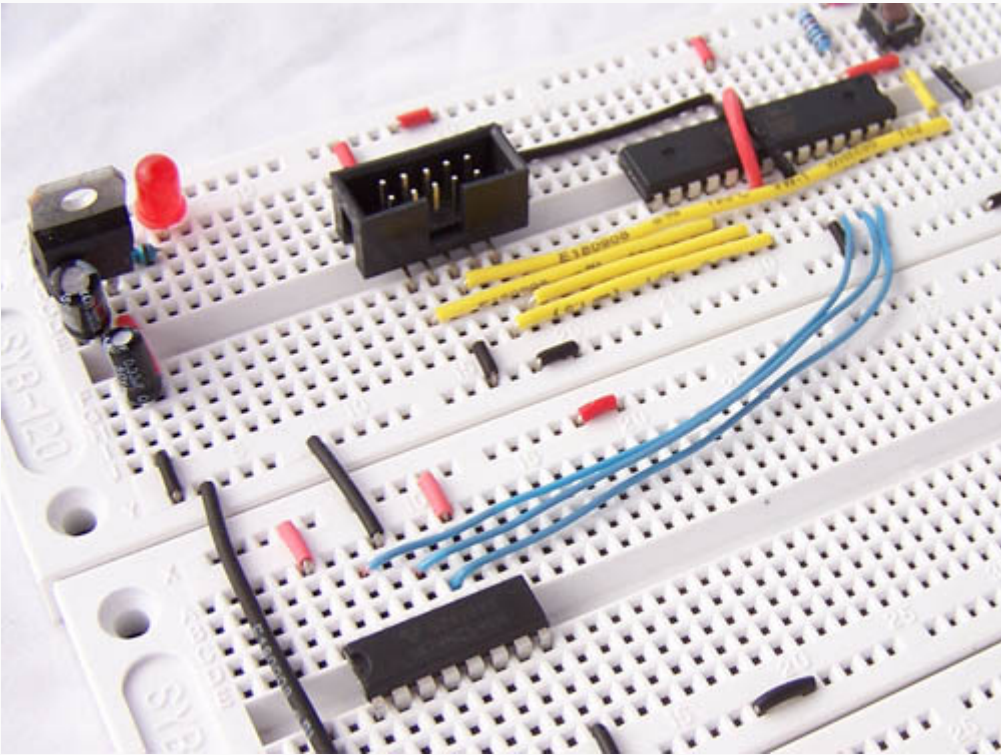
We add the [Shift Register](#) and connect it to +5V and Ground.



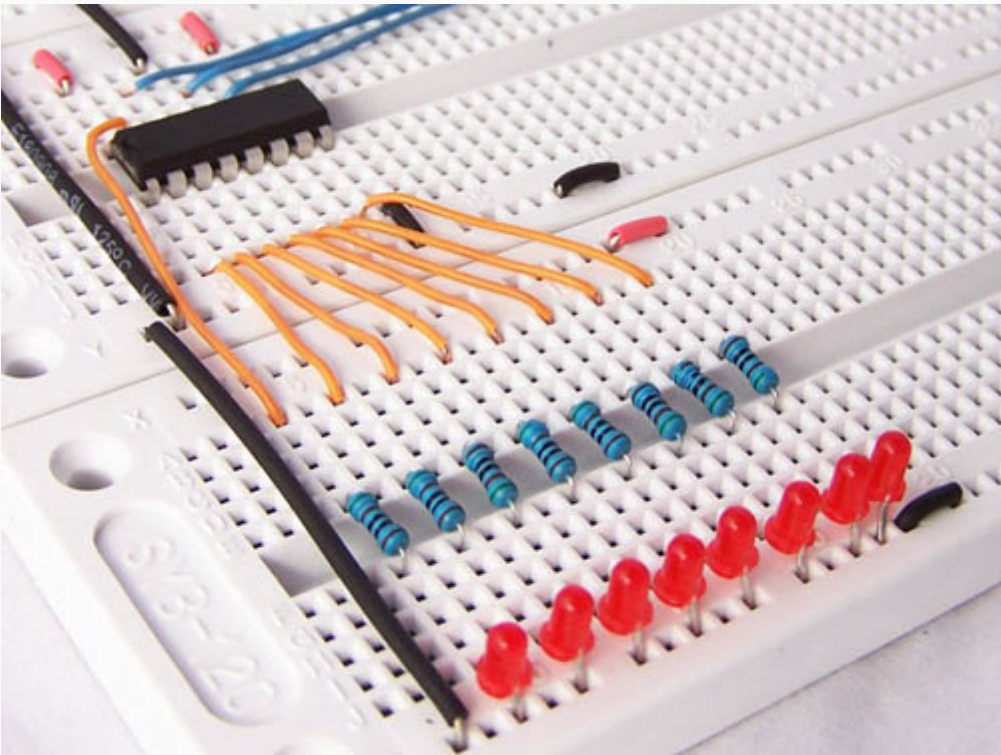
We now run the following control lines between the microcontroller and **Shift Register**

- PC0 to DS
- PC1 to ST_CP
- Pc2 to SH_CP

These are the blue wires shown below.



Next we connect up the LEDs and resistors. I used 510 Ohm resistors, but a range of other sizes are acceptable. ohmslawcalculator.com has a great **LED Resistor Calculator** that you can use.



To demonstrate the circuit, I wrote a small bit of code which produces a knight rider pattern on the 8 LEDs.

```
1.
2. #include <avr/io.h>
3. #include <util/delay.h>
4.
5. #define DS_PORT    PORTC
6. #define DS_PIN     0
7. #define ST_CP_PORT PORTC
8. #define ST_CP_PIN  1
```

```
9. #define SH_CP_PORT PORTC
10. #define SH_CP_PIN 2
11.
12. #define DS_low() DS_PORT&=~_BV(DS_PIN)
13. #define DS_high() DS_PORT|=_BV(DS_PIN)
14. #define ST_CP_low() ST_CP_PORT&=~_BV(ST_CP_PIN)
15. #define ST_CP_high() ST_CP_PORT|=_BV(ST_CP_PIN)
16. #define SH_CP_low() SH_CP_PORT&=~_BV(SH_CP_PIN)
17. #define SH_CP_high() SH_CP_PORT|=_BV(SH_CP_PIN)
18.
19. //Define functions
20. //=====
21. void ioinit(void);
22. void output_led_state(unsigned char __led_state);
23. //=====
24.
25. int main (void)
26. {
27.     ioinit(); //Setup IO pins and defaults
28.
29.     while(1)
30.     {
31.         /*
32.          * Output a knight rider pattern
33.          *
34.          * 10000000
35.          * 01000000
36.          * 00100000
37.          * 00010000
38.          * 00001000
39.          * 00000100
40.          * 00000010
41.          * 00000001
42.          * 00000010
43.          * 00000100
44.          * 00001000
45.          * 00010000
46.          * 00100000
47.          * 01000000
48.          */
49.
50.         for (int i=7;i>=0;i--)
51.         {
52.             output_led_state(_BV(i));
53.             _delay_ms(100);
54.         }
55.
56.         for (int i=1;i<7;i++)
57.         {
58.             output_led_state(_BV(i));
59.             _delay_ms(100);
60.         }
61.     }
62. }
63.
64.
65. void ioinit (void)
66. {
67.     DDRC = 0b00000111; //1 = output, 0 = input
68.     PORTC = 0b00000000;
69. }
70.
71. void output_led_state(unsigned char __led_state)
72. {
73.     SH_CP_low();
74.     ST_CP_low();
75.     for (int i=0;i<8;i++)
76.     {
77.         if (bit_is_set(__led_state, i))
78.             DS_high();
79.         else
80.             DS_low();
81.
82.
```

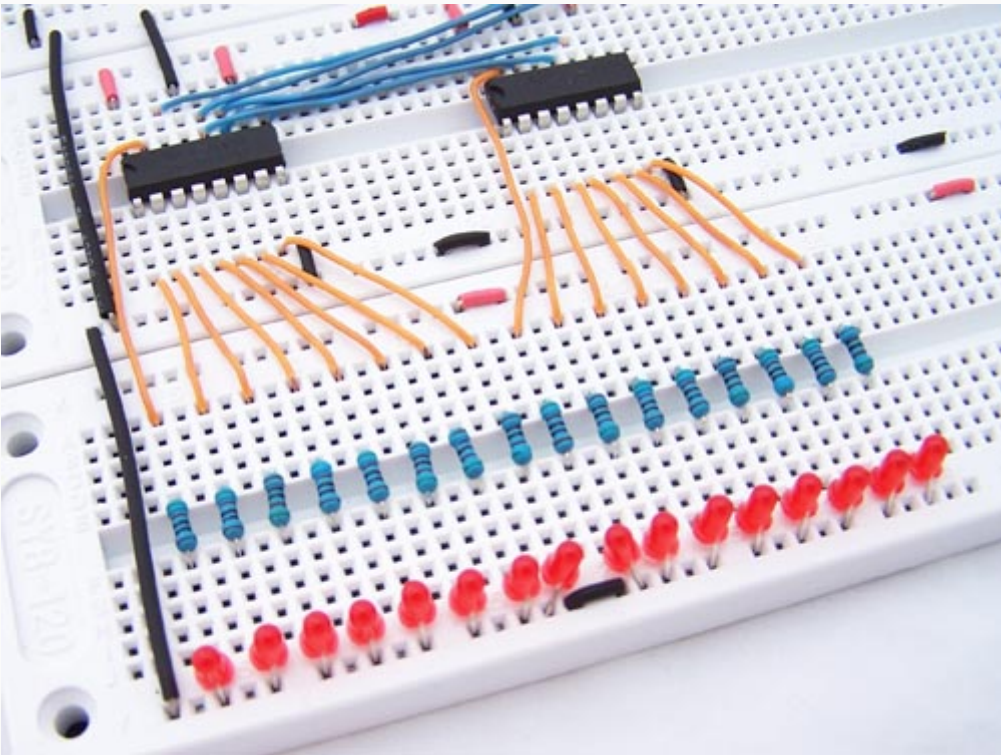
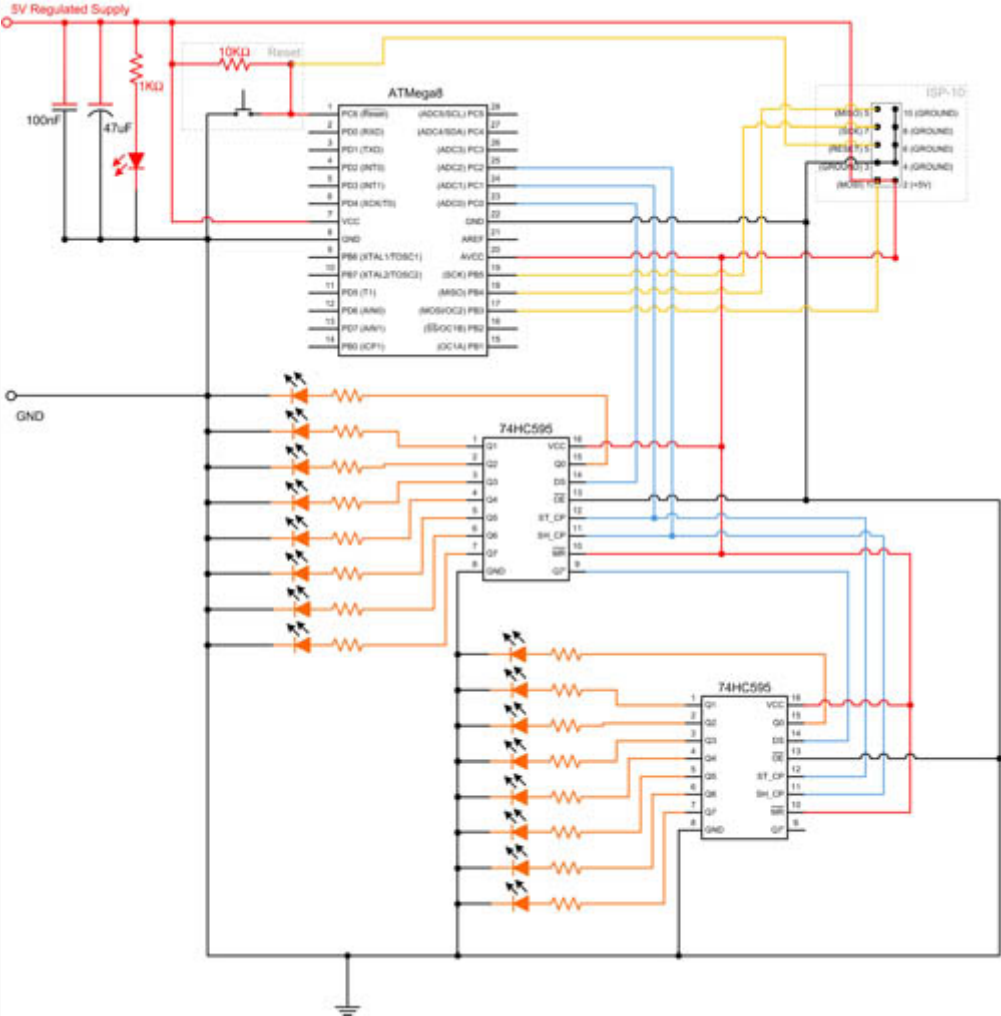


```
83.     SH_CP_high();
84.     SH_CP_low();
85. }
86.     ST_CP_high();
87. }
```

This is all very impressive, but didn't I say we were controlling 16 LEDs? To do this we need to add another **74HC595 shift register**, more LEDs, more resistors and more orange and blue wires.

We use the Q7' pin to daisy chain the **shift registers** together.

The modified circuit is shown below.



To code to produce the 16 LED knight rider pattern is

```
1.
2. #include <avr/io.h>
3. #include <util/delay.h>
4.
5. #define DS_PORT    PORTC
6. #define DS_PIN     0
7. #define ST_CP_PORT PORTC
8. #define ST_CP_PIN  1
9. #define SH_CP_PORT PORTC
10. #define SH_CP_PIN  2
11.
12. #define DS_low()   DS_PORT&=~_BV(DS_PIN)
13. #define DS_high()  DS_PORT|=_BV(DS_PIN)
14. #define ST_CP_low() ST_CP_PORT&=~_BV(ST_CP_PIN)
15. #define ST_CP_high() ST_CP_PORT|=_BV(ST_CP_PIN)
16. #define SH_CP_low() SH_CP_PORT&=~_BV(SH_CP_PIN)
17. #define SH_CP_high() SH_CP_PORT|=_BV(SH_CP_PIN)
18.
```

```
19. //Define functions
20. //=====
21. void ioinit(void);
22. void output_led_state(unsigned int __led_state);
23. //=====
24.
25. int main (void)
26. {
27.     ioinit(); //Setup IO pins and defaults
28.
29.     while(1)
30.     {
31.         // Output a knight rider pattern
32.
33.         for (int i=15;i>=0;i--)
34.         {
35.             output_led_state(_BV(i));
36.             _delay_ms(50);
37.         }
38.         for (int i=1;i<15;i++)
39.         {
40.             output_led_state(_BV(i));
41.             _delay_ms(50);
42.         }
43.     }
44. }
45.
46.
47. void ioinit (void)
48. {
49.     DDRC = 0b00000111; //1 = output, 0 = input
50.     PORTC = 0b00000000;
51. }
52.
53. void output_led_state(unsigned int __led_state)
54. {
55.     SH_CP_low();
56.     ST_CP_low();
57.     for (int i=0;i<16;i++)
58.     {
59.         if ((_BV(i) & __led_state) == _BV(i)) //bit_is_set doesn't work on unsigned int so we do
this instead
60.             DS_high();
61.         else
62.             DS_low();
63.
64.
65.         SH_CP_high();
66.         SH_CP_low();
67.     }
68.     ST_CP_high();
69. }
```

We just stopped at 16 LEDs, but we can continue daisy chaining more shift registers. This technique is not just limited to LEDs of course and we can use it to multiply output ports to drive many other kinds of devices.

One word of warning regarding this technique. When you power on the circuit, the output lines are set to some arbitrary value. Now it takes less than a microsecond to set them to your desired values, but for some circuits this may cause problems. In that case you can use to MR and OE pins to reset the storage registers.

No related posts.

74HC595, ATmega8, Breadboard

46 Comments

Protostack

1 Login


Recommend 4

Share

Sort by Best



Join the discussion...

 Remco • 4 years ago

Hi,

A nice article about how this shift register works. Now I know how that works!