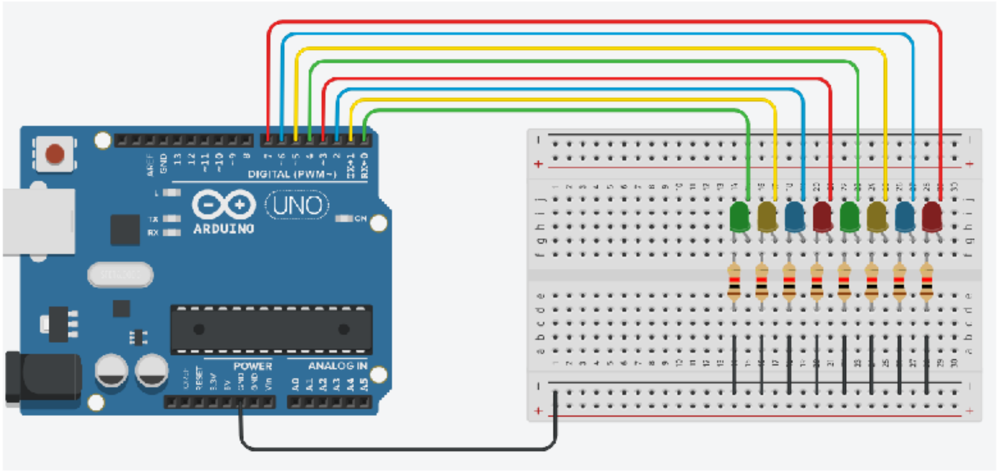
Embedded System

Week 11
Information display (2)

Outline

- Direct access to I/O registers
- Eight LED with 74HC595

Control of Multiple LEDs



Serial Control of LEDs with digitalWrite

```
// The setup() method runs once, when the sketch starts
void setup()
{
        for (int i = 0;i<8;i++)
        {
                // initialize the digital pin as an output
                pinMode(i, OUTPUT);
        }
}
// the loop() method runs over and over again.
void loop()
{
        for (int i = 0;i<8;i++)
        {
                digitalWrite(i, HIGH); // turn the LED on
                delay(1000); // wait a second
                digitalWrite(i, LOW); // turn the LED off
                delay(1000); // wait a second
        }
}
```

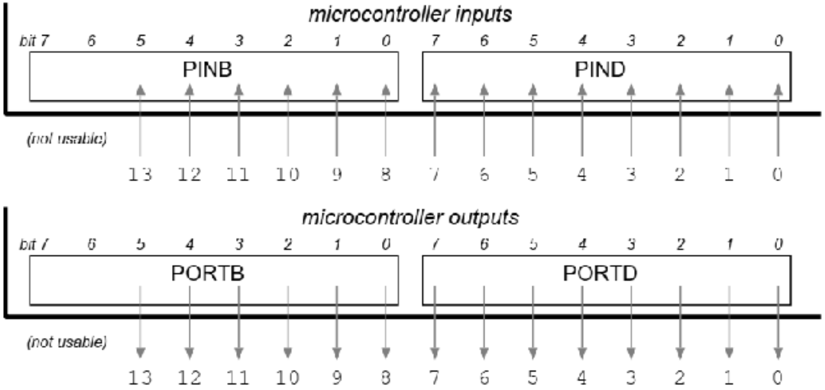
Pseudo-parallel blinking of LEDs with digitalWrite

```
// The setup() method runs once, when the sketch starts
void setup()
{
        for (int i = 0;i<8;i++)
        {
                // initialize the digital pin as an output
                pinMode(i, OUTPUT);
        }
}
// the loop() method runs over and over again,
void loop()
{
        for (int i = 0;i<8;i++)
        {
                digitalWrite(i, HIGH); // turn the LED on
        }
        delay(1000); // wait a second
        for (int i = 0;i<8;i++)
        {
                digitalWrite(i, LOW); // turn the LED off
        }
        delay(1000); // wait a second
}
```

Direct access to I/O registers

I/O pins are connected to *pin* and *port* registers

- each register has 8 bits
- PIN registers is for input, PORT registers are for output
- digital pins 0 to 7 are connected to PIND/PORTD
- digital pins 8 to 13 are connected to PINB/PORTB

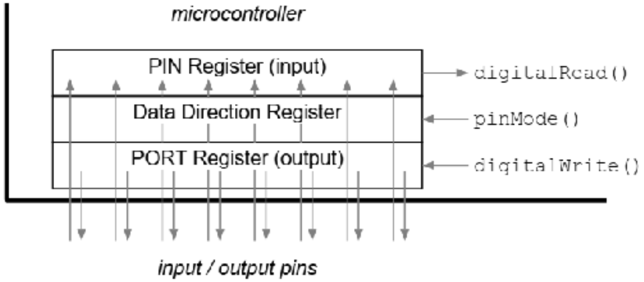


Direct access to I/O registers

Each physical pin is either an input or an output, but *not both* at the same time

pin configuration (input or output) is done using the *data direction register* (DDR)

- each DDR is 8 bits wide: each bit configures one digital pin
  - 0 for input: corresponding PIN bit can be read as 1 or 0
  - 1 for output: corresponding PORT bit can be set to 1 or 0
- DDRD configures digital pins 0–7; DDRB configures digital pins 8–13



Direct access to I/O registers

- the registers can be read/written *directly* by programs using symbolic names

			registers	
pins	*port*	*direction*	*output*	*input*
0 – 7	PORTD	`DDRD`	`PORTD`	`PIND`
8 – 13	PORTB	`DDRB`	`PORTB`	`PINB`

reading a PORT register returns the last value you wrote there

binary constants can be written, e.g: B00100000 (bit 5 set)
use 'bit-wise' operators (&, |, ^, and ~) to modify only relevant pins

Important!

`~x`	inverts each bit in `x`		`~B00100000 ==  B11011111`
`x &= y`	clears bits in `x` where `y` has `0`s		`x &=  B11011111`
		or	`x &= ~B00100000`
`x |= y`	sets bits in `x` where `y` has `1`s		`x |=  B00100000`
`x ^= y`	inverts bits in `x` where `y` has `1`s		`x ^=  B00100000`

Example of Direct access to I/O registers

Suppose that two LEDs are connected to pins 12 and 13, the following program can be used for controlling the two LEDs

```
void setup()
{
  DDRB |= B00110000; // set pins 13, 12 as outputs
}

void loop()
{
  PORTB &= ~B00100000; // set pin 13 LOW, LED is off
  delay(100);
  PORTB |= B00100000; // set pin 13 HIGH, LED is on
  delay(100);

  PORTB ^= B00010000; // toggle pin 12
}
```

Exercises 1

- Assemble the circuit by referring to the code in page 3

- Realize the serial control and parallel blinking (refer the program on page 4 and 5) of 8 LEDs using the I/O registers

Outline

- Direct access to I/O registers
- Eight LED with 74HC595

Limitation of UNO

- In this lesson, you will learn how to use eight LEDs with an UNO R3 without needing to give up 8 output pins!
- Although you could wire up eight LEDs each with a resistor to an UNO pin you would rapidly start to run out of pins on your UNO.
- But when you have many sensors to connect with UNO, the pins maybe not enough.

74HC595

- So, instead of doing that, we are going to use a chip called the 74HC595 Serial to Parallel Converter.
- This chip has eight outputs and three inputs that you use to feed data into it a bit at a time.

How 74HC595 Shift Register works?

- The 595 has two registers (which can be thought of as “memory containers”), each with just 8 bits of data. The first one is called the Shift Register. The Shift Register lies deep within the IC circuits, quietly accepting input.
- Whenever we apply a clock pulse to a 595, two things happen:
  - The bits in the Shift Register move one step to the left. For example, Bit 7 accepts the value that was previously in bit 6, bit 6 gets the value of bit 5 etc.
  - Bit 0 in the Shift Register accepts the current value on DATA pin. At the rising edge of the pulse, if the data pin is high, then a 1 gets pushed into the shift register. Otherwise, it is a 0.

How 74HC595 Shift Register works?

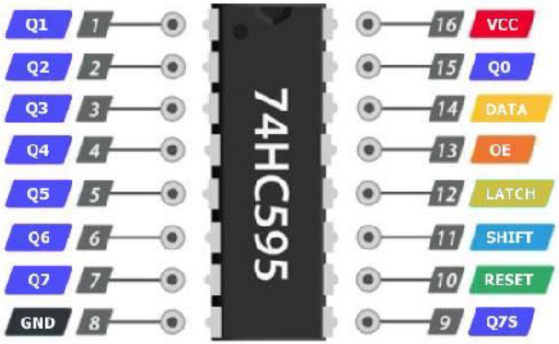
- On enabling the Latch pin (Low→High), the contents of Shift Register are copied into the second register, called the Storage/Latch Register.
- Each bit of the Storage Register is connected to one of the output pins Q_0–Q_7 of the IC, so in general, when the value in the Storage Register changes, so do the outputs.

74HC595 Shift Register Pinout

GND should be connected to the ground of Arduino.

VCC is the power supply for 74HC595 shift register which we connect the 5V pin on the Arduino.

DATA (Serial Input) pin is used to feed data into the shift register a bit at a time.



74HC595 Shift Register Pinout

- SHIFT (Shift Register Clock) is the clock for the shift register. The 595 is clock-driven on the rising edge. This means that in order to shift bits into the shift register, the clock must be HIGH. And bits are transferred in on the rising edge of the clock.
- LATCH (Register Clock / Latch) is a very important pin. When driven HIGH, the contents of Shift Register are copied into the Storage/Latch Register; which ultimately shows up at the output. So the latch pin can be seen as like the final step in the process to seeing our results at the output, which in this case are LEDs.
- RESET (Shift Register Clear) pin allows us to reset the entire Shift Register, making all its bits 0, at once. This is a negative logic pin, so to perform this reset; we need to set the RESET pin LOW. When no reset is required, this pin should be HIGH.

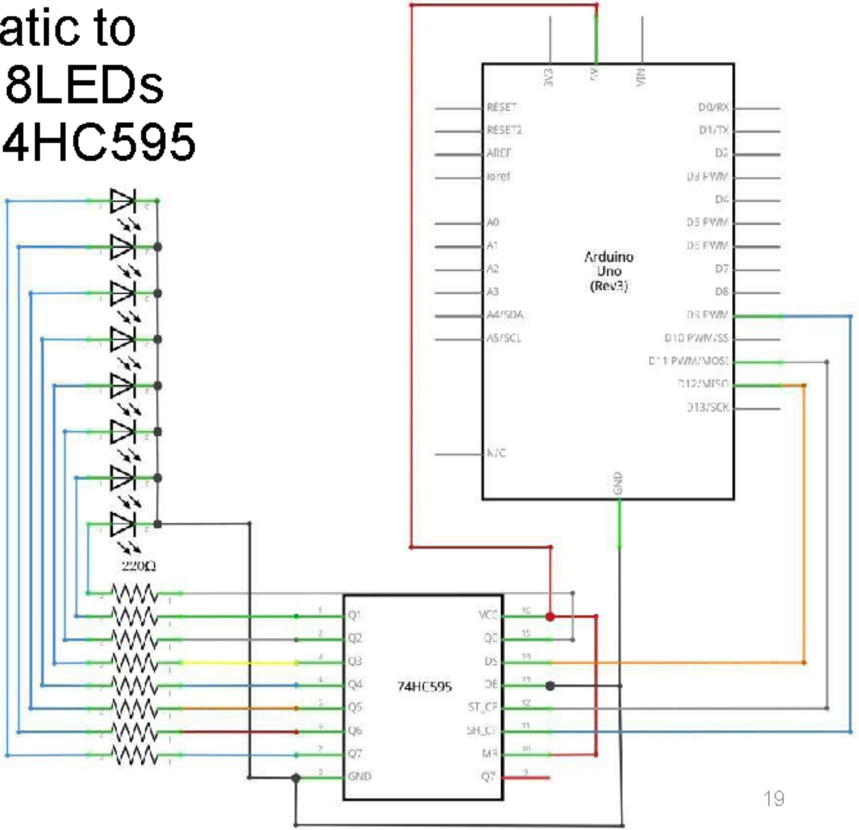
74HC595 Shift Register Pinout

- OE (Output Enable) is negative logic too: When the voltage on it is HIGH, the output pins are disabled/set to high impedance state and don’t allow current to flow. When OE gets low voltage, the output pins work normally.

  Q0–Q7 (Output) are the output pins and should be connected to some type of output like LEDs, 7 Segments etc.

  Q7s Pin outputs bit 7 of the ShiftRegister. It is there so that we may daisychain 595s: if you connect this Q7s to the DATA pin of another 595, and give both ICs the same clock signal, they will behave like a single IC with 16 outputs. Of course, this technique is not limited to two ICs – you can daisychain as many as you like, if you have enough power for all of them.

Schematic to control 8LEDs using 74HC595



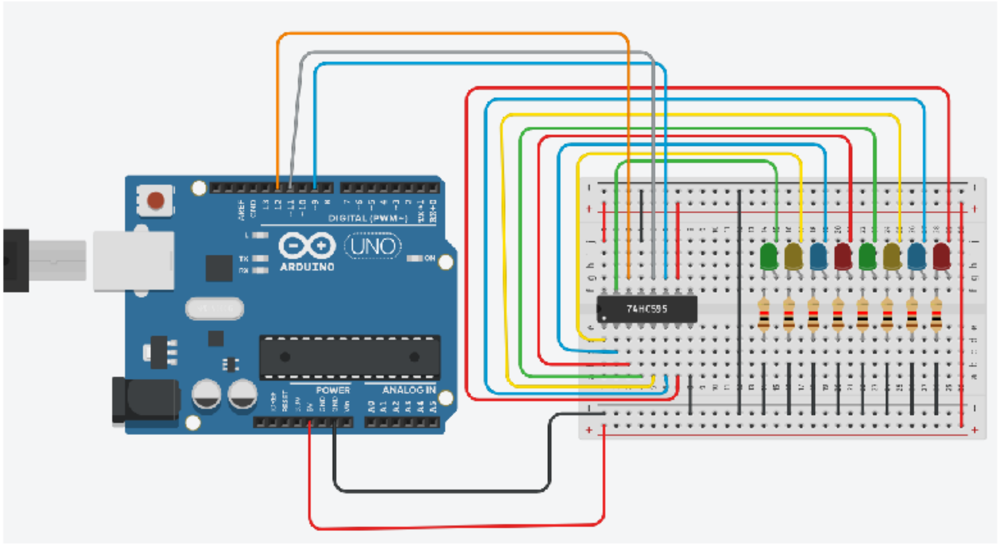
DS: DATA

ST_CP: LATCH

SH_CP: SHIFT

MR: RESET

Circuit to control 8LEDs using 74HC595



Program to control 8LEDs using 74HC595

```
int tDelay = 1000;
int latchPin = 11;      // (11) ST_CP [RCK] on 74HC595
int clockPin = 9;       // (9) SH_CP [SCK] on 74HC595
int dataPin = 12;      // (12) DS [S1] on 74HC595

byte leds = 0;

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
  leds = 0;
  updateShiftRegister();
  delay(tDelay);
  for (int i = 0; i < 8; i++)
  {
    bitSet(leds, i);
    updateShiftRegister();
    delay(tDelay);
  }
}
```

Explanation for program

- The first thing we do is define the three pins we are going to use.
- These are the UNO digital outputs that will be connected to the latch, clock and data pins of the 74HC595.

- Next, a variable called 'leds' is defined. This will be used to hold the pattern of which LEDs are currently turned on or off.
- Data of type 'byte' represents numbers using eight bits. Each bit can be either on or off, so this is perfect for keeping track of which of our eight LEDs are on or off.

Explanation for program

- The 'setup' function just sets the three pins we are using to be digital outputs.
- The 'loop' function initially turns all the LEDs off, by giving the variable 'leds' the value 0.
- It then calls 'updateShiftRegister' that will send the 'leds' pattern to the shift register so that all the LEDs turn off.
- We will explain how 'updateShiftRegister‘ works later.

Explanation for program

- The loop function pauses for half a second and then begins to count from 0 to 7 using the 'for' loop and the variable 'i'.
- Each time, it uses the Arduino function 'bitSet' [1] to set the bit that controls that LED in the variable 'leds'.
- It then also calls 'updateShiftRegister' so that the leds update to reflect what is in the variable 'leds'. There is then a half second delay before 'i' is incremented and the next LED becomes light.

[1] https://www.arduino.cc/reference/en/language/functions/bits-and-bytes/bitset/

Explanation for program

- The function 'updateShiftRegister', first of all sets the latchPin to low, then calls the UNO function 'shiftOut' before putting the 'latchPin' high again.
- This takes four parameters, the first two are the pins to use for Data and Clock respectively.
- The third parameter specifies which end of the data you want to start at. We are going to start with the right most bit, which is referred to as the 'Least Significant Bit' (LSB).
- The last parameter is the actual data to be shifted into the shift register, which in this case is 'leds’.
- When the latchPin goes from low to high the sent data gets moved from the shift registers memory register into the output pins, lighting the LEDs.

Exercises 2

- Assemble the circuit by referring to the page 20 to control LED using 74HC595
- Design a different dynamic pattern and extend the code to control LED based on the pattern (e.g. light one LED each time)

Summary

- Direct access to I/O registers
- Eight LED with 74HC595

Embedded System

Week 12
Information display (2)