

# Binary Representations, Serial Communication and the Atmel 2560

# Administration...

- Top Hat or Zyante problems?

# Questions?

# Atmel Mega2560 Microcontroller

# Atmel Mega2560

U1IO			
90	PF7(ADC7/TDI)	PA7(AD7)	71
91	PF6(ADC6/TDO)	PA6(AD6)	72
92	PF5(ADC5/TMS)	PA5(AD5)	73
93	PF4(ADC4/TCK)	PA4(AD4)	74
94	PF3(ADC3)	PA3(AD3)	75
95	PF2(ADC2)	PA2(AD2)	76
96	PF1(ADC1)	PA1(AD1)	77
97	PF0(ADC0)	PA0(AD0)	78
1	PG5(OC0B)	PB7(OC0A/OC1C)	26
29	PG4(TOSC1)	PB6(OC1B)	25
28	PG3(TOSC2)	PB5(OC1A)	24
70	PG2(ALE)	PB4(OC2A)	23
52	PG1(RD)	PB3(MISO)	22
51	PG0(WR)	PB2(MOSI)	21
		PB1(SCK)	20
27	PH7(T4)	PB0(SS)	19
18	PH6(OC2B)		
17	PH5(OC4C)	PC7(A15)	60
16	PH4(OC4B)	PC6(A14)	59
15	PH3(OC4A)	PC5(A13)	58
14	PH2(XCK2)	PC4(A12)	57
13	PH1(TXD2)	PC3(A11)	56
12	PH0(RXD2)	PC2(A10)	55
		PC1(A9)	54
79	PJ7	PC0(A8)	53
69	PJ6(PCINT15)		
68	PJ5(PCINT14)	PD7(T0)	50
67	PJ4(PCINT13)	PD6(T1)	49
66	PJ3(PCINT12)	PD5(XCK1)	48
65	PJ2(XCK3)	PD4(ICP1)	47
64	PJ1(TXD3)	PD3(TXD1/INT3)	46
63	PJ0(RXD3)	PD2(RXD1/INT2)	45
		PD1(SDA/INT1)	44
82	PK7(ADC15)	PD0(SCL/INT0)	43
83	PK6(ADC14)		
84	PK5(ADC13)	PE7(ICP3/INT7)	9
85	PK4(ADC12)	PE6(T3/INT6)	8
86	PK3(ADC11)	PE5(OC3C/INT5)	7
87	PK2(ADC10)	PE4(OC3B/INT4)	6
88	PK1(ADC9)	PE3(OC3A/AIN1)	5
89	PK0(ADC8)	PE2(XCK0/AIN0)	4
		PE1(TXD0)	3
42	PL7	PE0(RXD0)	2
41	PL6		
40	PL5(OC5C)		
39	PL4(OC5B)		
38	PL3(OC5A)		
37	PL2(T5)		
36	PL1(ICP5)		
35	PL0(ICP4)		

# Atmel Mega2560

Pins are organized  
into 8-bit “Ports”:

- A, B, C ... L
  - But no “I”

U1IO			
90	PF7(ADC7/TDI)	PA7(AD7)	71
91	PF6(ADC6/TDO)	PA6(AD6)	72
92	PF5(ADC5/TMS)	PA5(AD5)	73
93	PF4(ADC4/TCK)	PA4(AD4)	74
94	PF3(ADC3)	PA3(AD3)	75
95	PF2(ADC2)	PA2(AD2)	76
96	PF1(ADC1)	PA1(AD1)	77
97	PF0(ADC0)	PA0(AD0)	78
1	PG5(OC0B)	PB7(OC0A/OC1C)	26
29	PG4(TOSC1)	PB6(OC1B)	25
28	PG3(TOSC2)	PB5(OC1A)	24
70	PG2(ALE)	PB4(OC2A)	23
52	PG1(RD)	PB3(MISO)	22
51	PG0(WR)	PB2(MOSI)	21
		PB1(SCK)	20
		PB0(SS)	19
27	PH7(T4)		
18	PH6(OC2B)		
17	PH5(OC4C)	PC7(A15)	60
16	PH4(OC4B)	PC6(A14)	59
15	PH3(OC4A)	PC5(A13)	58
14	PH2(XCK2)	PC4(A12)	57
13	PH1(TXD2)	PC3(A11)	56
12	PH0(RXD2)	PC2(A10)	55
		PC1(A9)	54
		PC0(A8)	53
79	PJ7		
69	PJ6(PCINT15)		
68	PJ5(PCINT14)	PD7(T0)	50
67	PJ4(PCINT13)	PD6(T1)	49
66	PJ3(PCINT12)	PD5(XCK1)	48
65	PJ2(XCK3)	PD4(ICP1)	47
64	PJ1(TXD3)	PD3(TXD1/INT3)	46
63	PJ0(RXD3)	PD2(RXD1/INT2)	45
		PD1(SDA/INT1)	44
		PD0(SCL/INT0)	43
82	PK7(ADC15)		
83	PK6(ADC14)		
84	PK5(ADC13)	PE7(ICP3/INT7)	9
85	PK4(ADC12)	PE6(T3/INT6)	8
86	PK3(ADC11)	PE5(OC3C/INT5)	7
87	PK2(ADC10)	PE4(OC3B/INT4)	6
88	PK1(ADC9)	PE3(OC3A/AIN1)	5
89	PK0(ADC8)	PE2(XCK0/AIN0)	4
		PE1(TXD0)	3
		PE0(RXD0)	2
42	PL7		
41	PL6		
40	PL5(OC5C)		
39	PL4(OC5B)		
38	PL3(OC5A)		
37	PL2(T5)		
36	PL1(ICP5)		
35	PL0(ICP4)		

# Digital Input/Output

- Each port has three special-purpose registers that control its behavior.
  - Each pin can either be an input or output!
- For port B, they are:
  - DDRB: data direction register B
  - PORTB: port output register B
  - PINB: port input B

# Data Direction Register: DDRx

- 8-bit wide register
  - Controls one pin with each bit
- 0 -> this is an input pin
- 1 -> this is an output pin
- Note: only configure pins as an output if you really mean it!



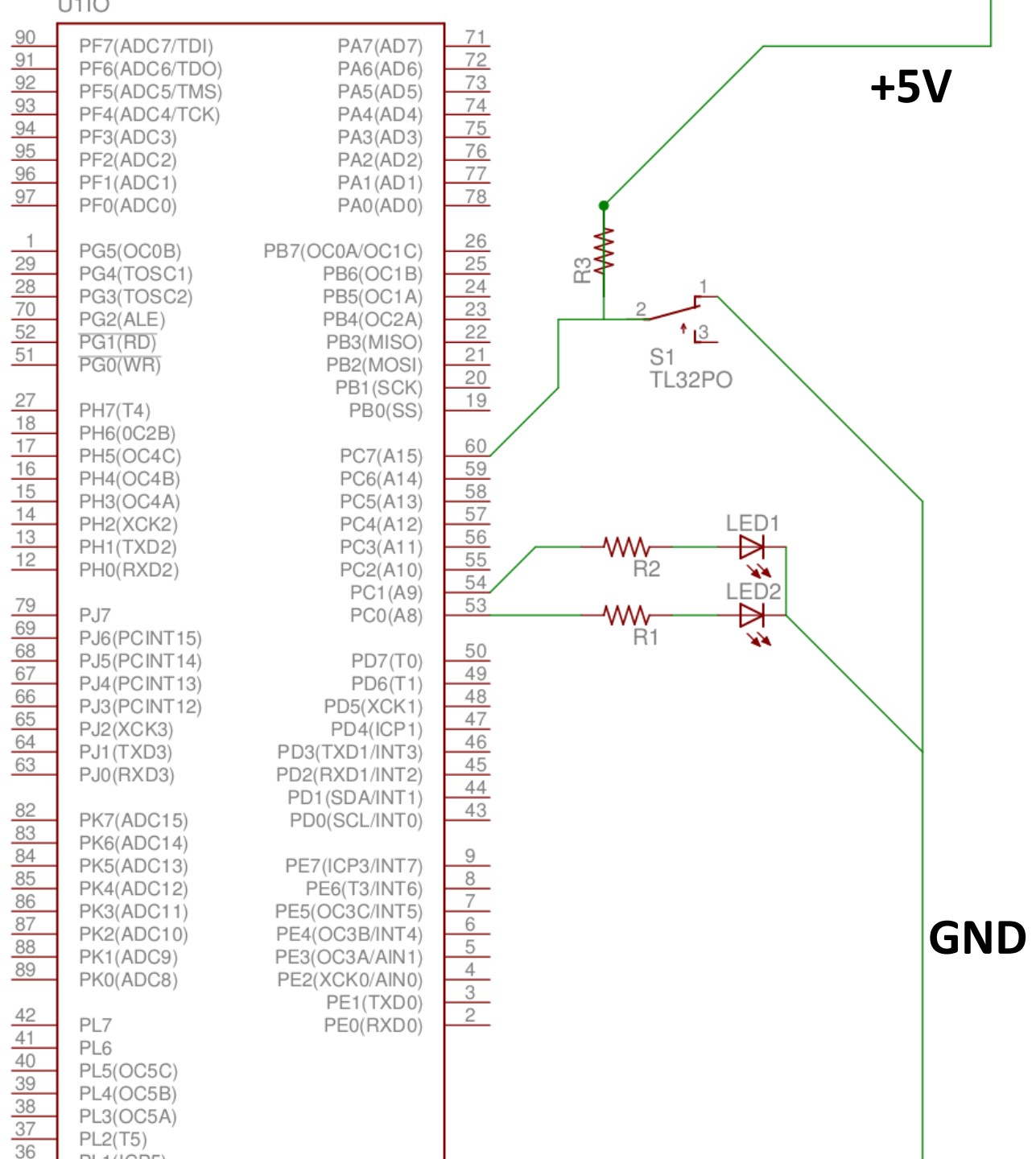
# Port Output Register: PORTx

- Also one pin per bit
- If configured as an output:
  - 0 -> the pin is held at 0 V
  - 1 -> the pin is held at +5 V

# Port INput register: PINx

- One pin per bit
- Reading from the register:
  - 0 -> the voltage of the pin is near 0 V
  - 1 -> the voltage of the pin is near +5 V
- If nothing is connected to the pin, then the pin will appear to be in a random state

# A First Circuit

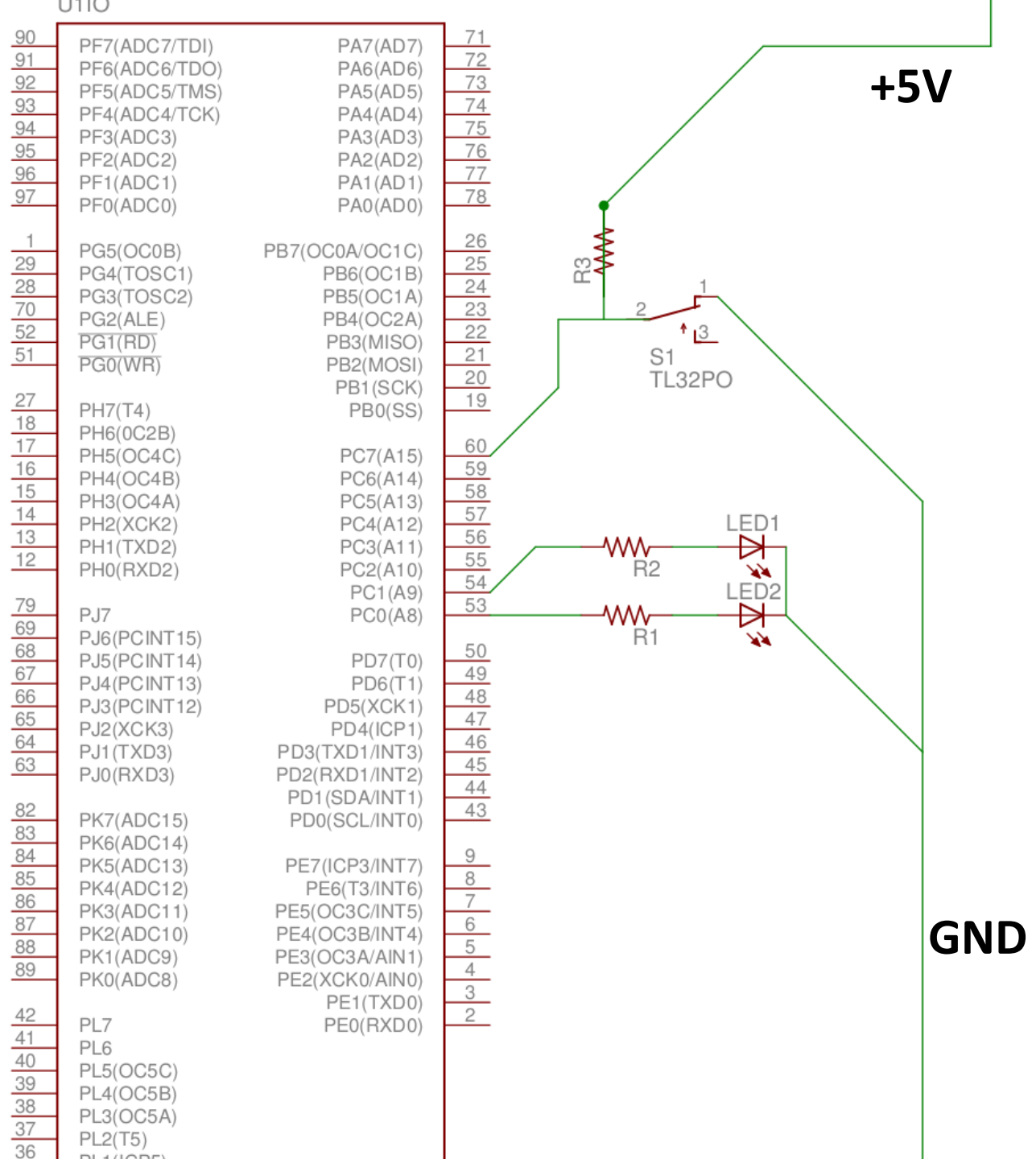


# A First Program

Flash the LEDs at a regular interval

- How do we do this?

A



# A First Program

```
main() {  
    DDRC = ???;  
  
    while(1)  
    {  
  
    }  
  
}
```

# A First Program

```
main() {  
    DDRC = 0x3;  
  
    while(1) {  
        PORTC = 0x1;           // sets PC0 to 1  
        delay_ms(100);  
        PORTC = 0x0;           // set PC0 to 0  
        delay_ms(100);  
    }  
}
```

# A Second Program

```
main() {  
    DDRC = 3;    // Set port C pins 0, and 1 as outputs  
  
    while(1) {  
        PORTC = 0x3;  
        delay_ms(250);  
        PORTC = 0x1;  
        delay_ms(250);  
        PORTC = 0x2;  
        delay_ms(250);  
        PORTC = 0x0;  
        delay_ms(250);  
    }  
}
```

**What does this program do?**

# A Second Program

```
main() {  
    DDRC = 3;    // Set port C pins 0, and 1 as outputs  
  
    while(1) {  
        PORTC = 0x3;  
        delay_ms(250);  
        PORTC = 0x1;  
        delay_ms(250);  
        PORTC = 0x2;  
        delay_ms(250);  
        PORTC = 0x0;  
        delay_ms(250);  
    }  
}
```

**Flashes LED on PC1 at 2 Hz  
on PC0: 1 Hz**

**Duty Cycle for each: 50%**



# Port-Related Registers

Some of the C-accessible registers for controlling digital I/O:

	Directional control	Writing	Reading
Port B	DDRB	PORTB	PINB
Port C	DDRC	PORTC	PINC
Port D	DDRD	PORTD	PIND

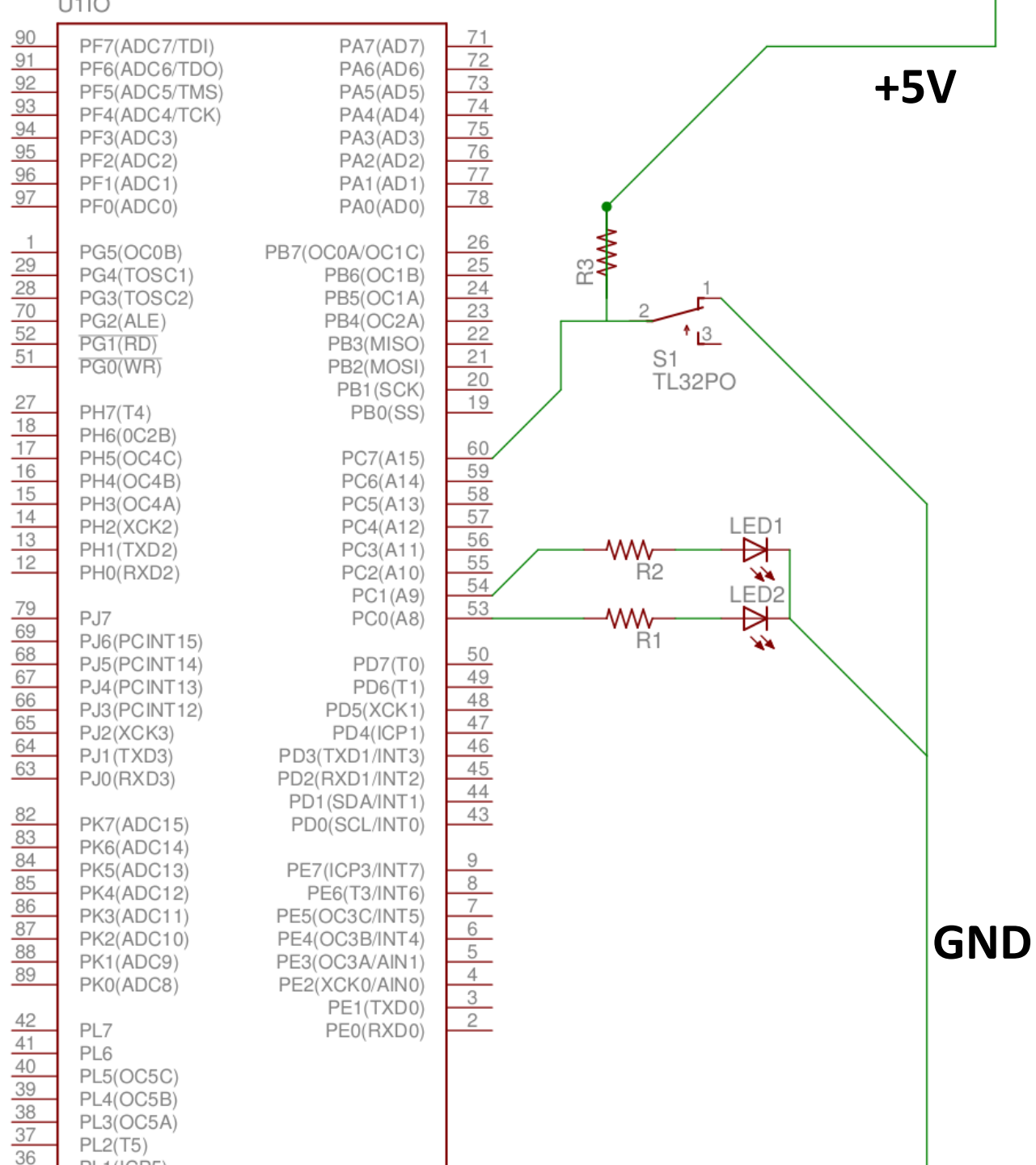
... go to Bit Manipulation

# A Third Program

If switch reads zero, turn PC0 on and PC1 off

Otherwise, turn PC0 off and PC1 on

A



# A Third Program

```
main() {  
    DDRC = 0x3;  
  
    while(1)  
    {  
  
    }  
}
```

# A Third Program

```
main() {  
    DDRC = 0x3;  
  
    while(1)  
    {  
        if(PINC & 0x80) {  
            PORTC = 0x2;  
        }else{  
            PORTC = 0x1;  
        }  
    }  
}
```

# Arduino Mega Board

(see schematic)

# Quiz





# Input/Output Systems

Processor needs to communicate with other devices:

- Receive signals from sensors
- Send commands to actuators
- Or both (e.g., disks, audio, video devices, other processors)

# I/O Systems

Communication can happen in a variety of ways:

- Binary parallel signal
- Analog
- Serial signals

# Serial Communication

- Communicate a set of bytes using a single signal line
- We do this by sending one bit at a time:
  - The value of the first bit determines the state of a signal line for a specified period of time
  - Then, the value of the 2<sup>nd</sup> bit is used
  - Etc.

# Serial Communication on the Mega2560

Our mega 2560 has FOUR Universal, Asynchronous serial Receiver/Transmitters (UARTs):

- Each handles all of the bit-level manipulation
  - Your software only worries about the byte level
- UART #1 is attached to the USB connection between the Arduino board and your laptop

# Mega2560 UART C Interface

Lib C support (standard C):

`char fgetc(fp) : receive a character`

`fputc('a', fp) : put a character out to the port`

`fputs("foobar", fp) : put a string out to the port`

`fprintf(fp, "foobar %d %s", 45, "baz") :  
put a formatted string out to the port`

# Mega2560 UART C Interface

OUlib support:

```
fp = serial_init_buffered(1, 38400, 40, 40)
```

Initialize port one for a transmission rate of 38400 bits per second (input and output buffers are both 40 characters long)

Note: declare fp as a global variable:

```
FILE *fp;
```

```
serial_buffered_input_waiting(fp)
```

Is there a character in the buffer?

See the Atmel HOWTO: [examples\\_2560/serial](#)

# Reading a Character from the Serial Port

```
int c;
```

```
c=fgetc(fp) ;
```

Note: fgetc() “blocks” until a byte is available

- Will only return with a value once a character is available to be returned

# Processing Serial Input

```
int c;
while(1) {
    if(serial_buffered_input_waiting(fp)) {
        // A character is available for reading
        c = fgetc(fp);
        <do something with the character>
    }
    <do something else while waiting>
}
```

`serial_buffered_input_waiting(fp)` tells us whether a byte is ready to be read



# Mega2560 UART C Interface

Also available:

- `fscanf()` : formatted input

See the LibC documentation or the AVR C textbook

# Next Time

Project 0: compiling and downloading for the mega 2560