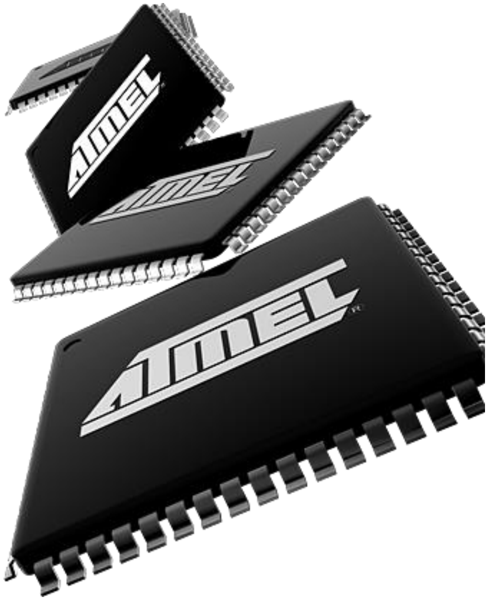
Principles and Applications of Microcontrollers

Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

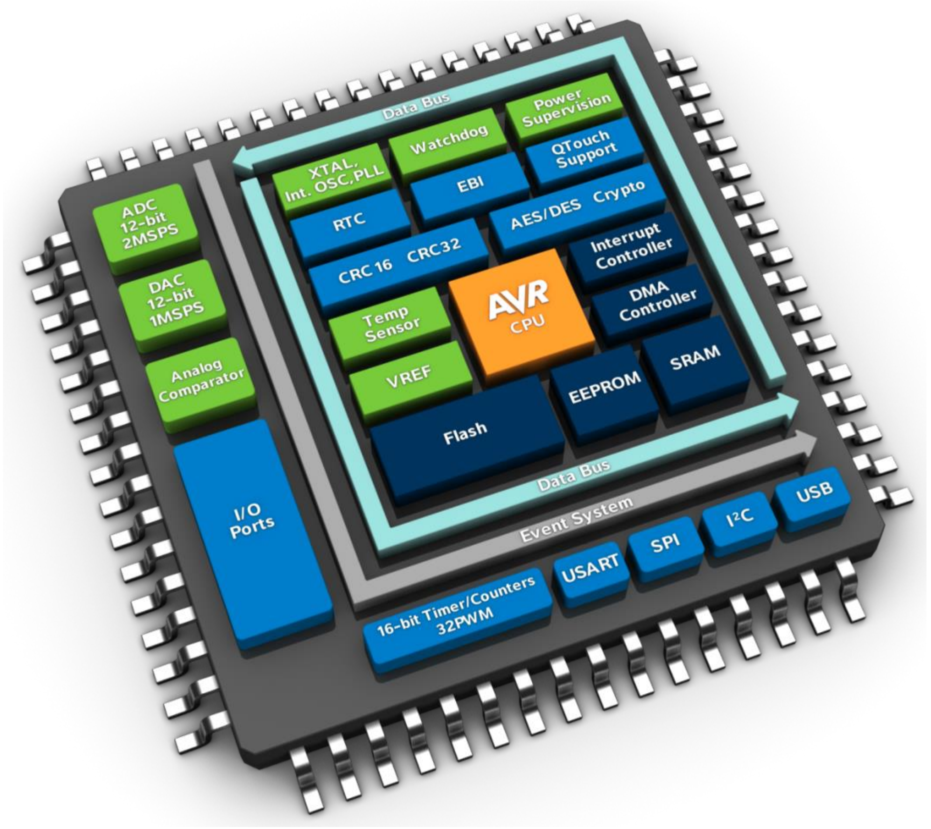
Today:

- AVR microcontrollers
- Input/Output (I/O)

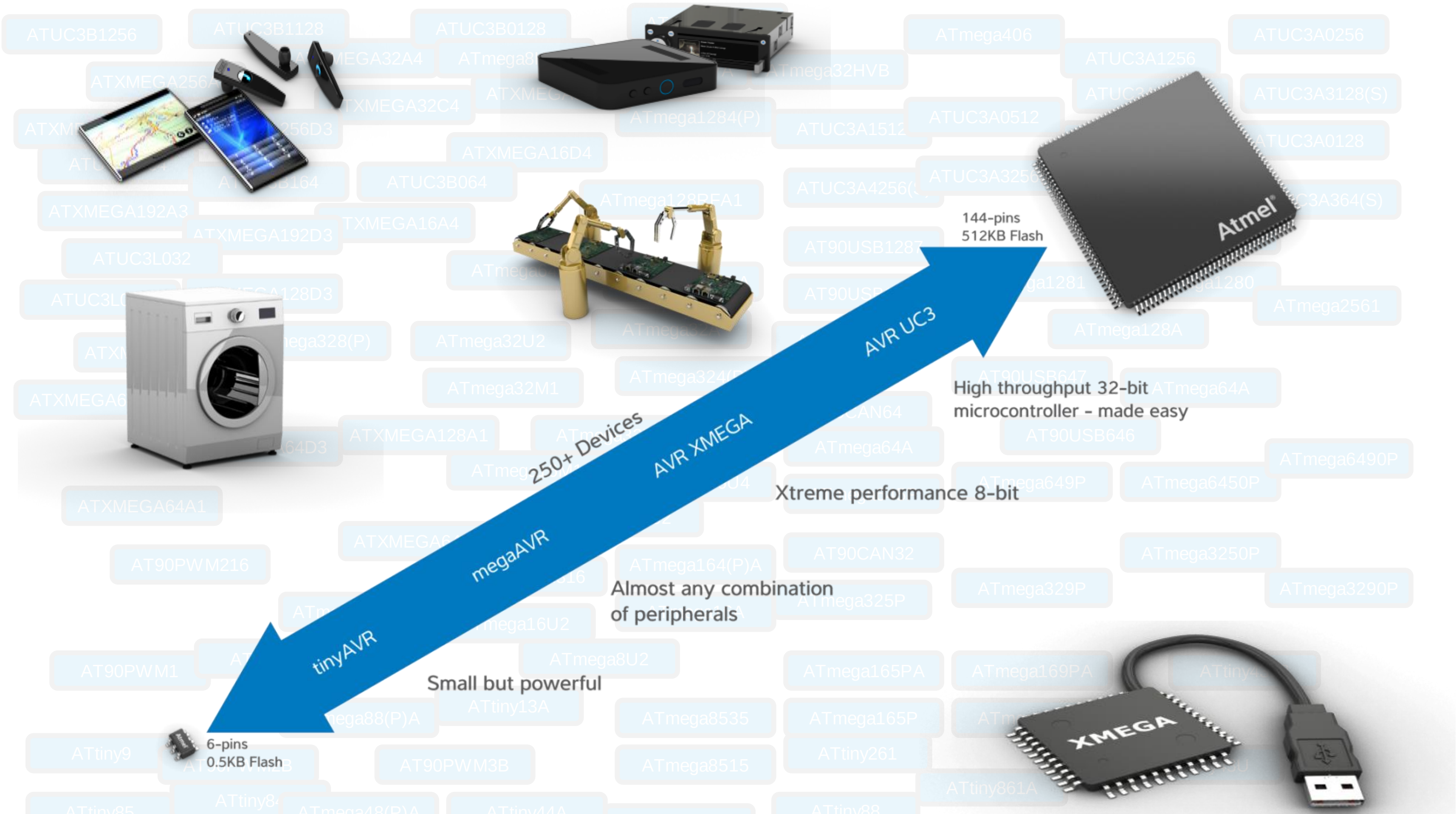


Outline

- AVR microcontrollers
  - Family
  - ATmega328P
  - Features
- AVR I/O
  - Pinout of ATmega328P
  - I/O registers
  - I/O programming
  - Program download
- Getting started



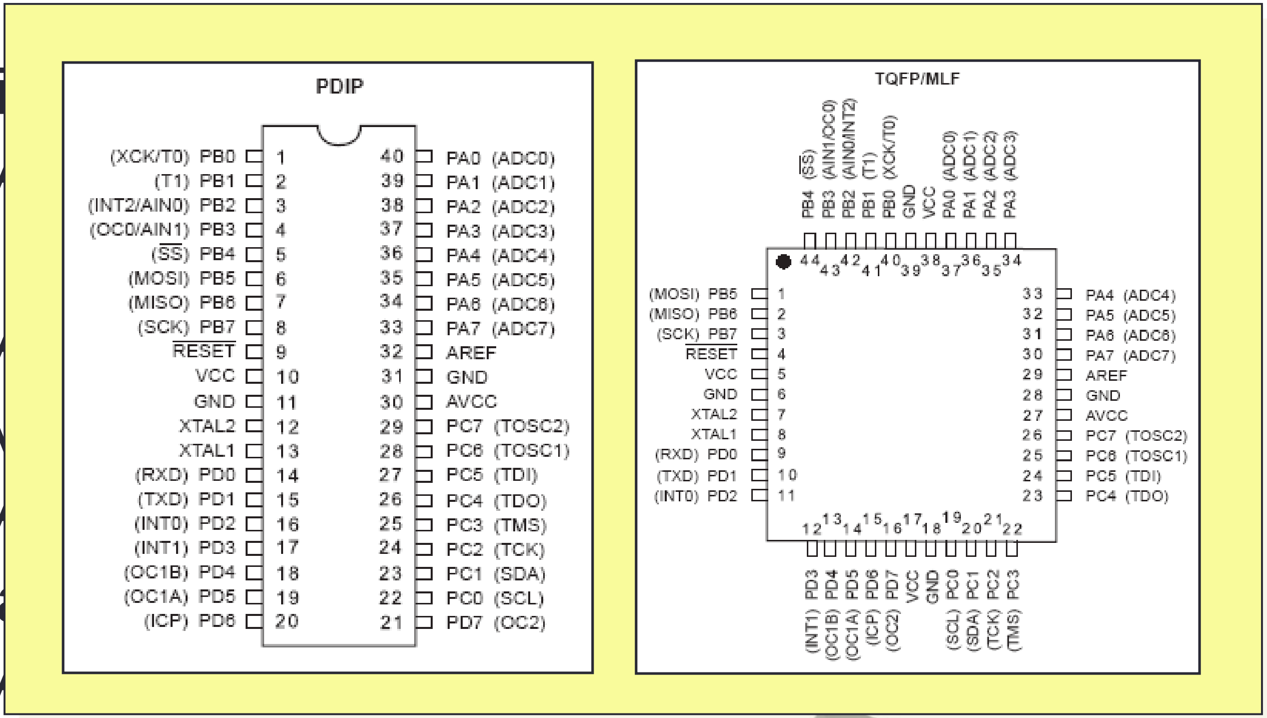
Why AVR?

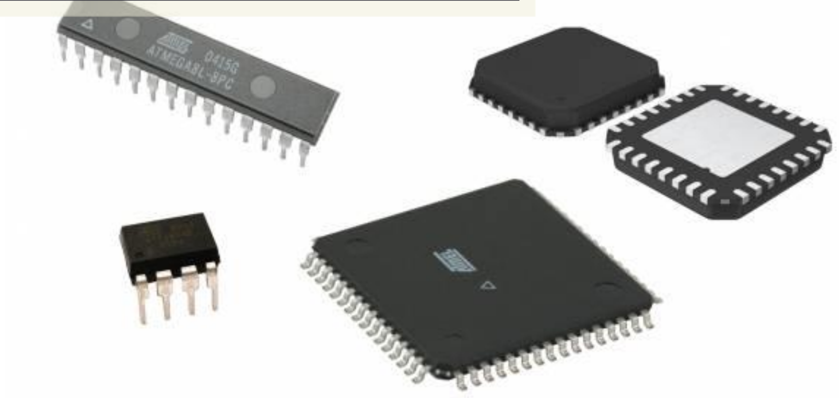


USB, Ethernet,
CAN, ADC, DAC

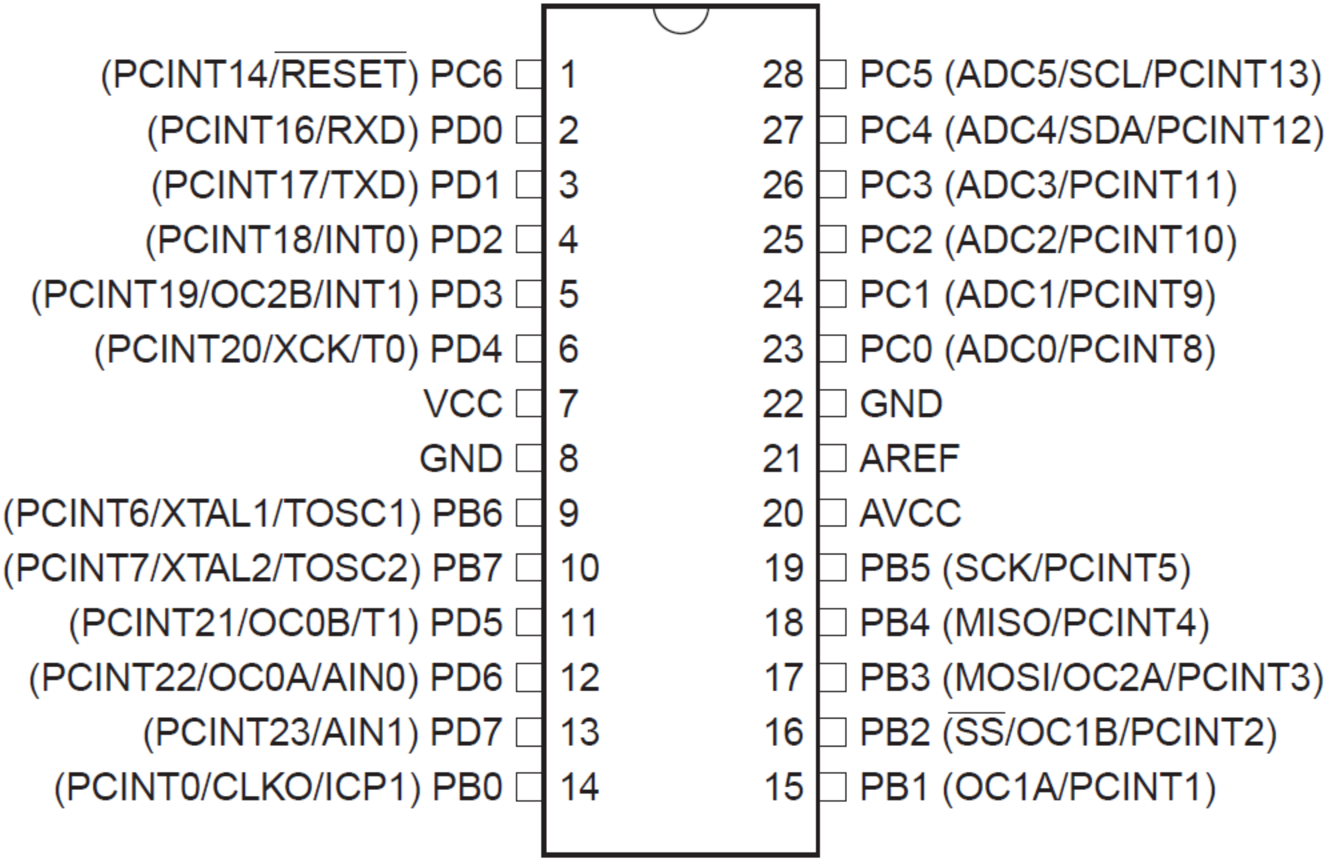
AVR Microcontroller Family

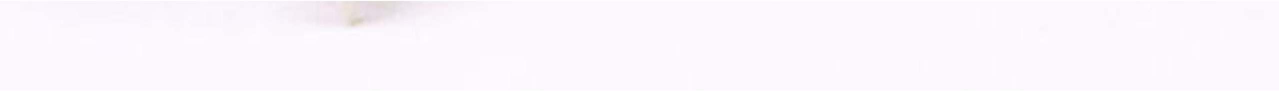
- Classi
  - e.g. A
- Mega
  - e.g. A
- Tiny A
  - e.g. A
- Speci
  - e.g. A
- Package
  - PDIP (plastic dual in-line package)
  - TQFP (thin quad flat pack)





ATmega328P

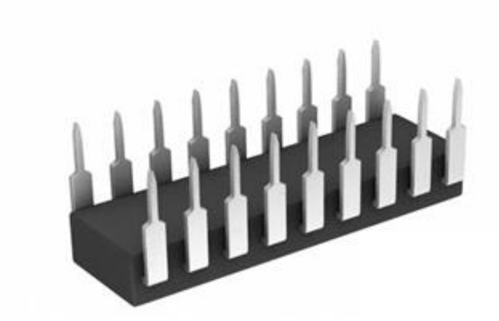

(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)



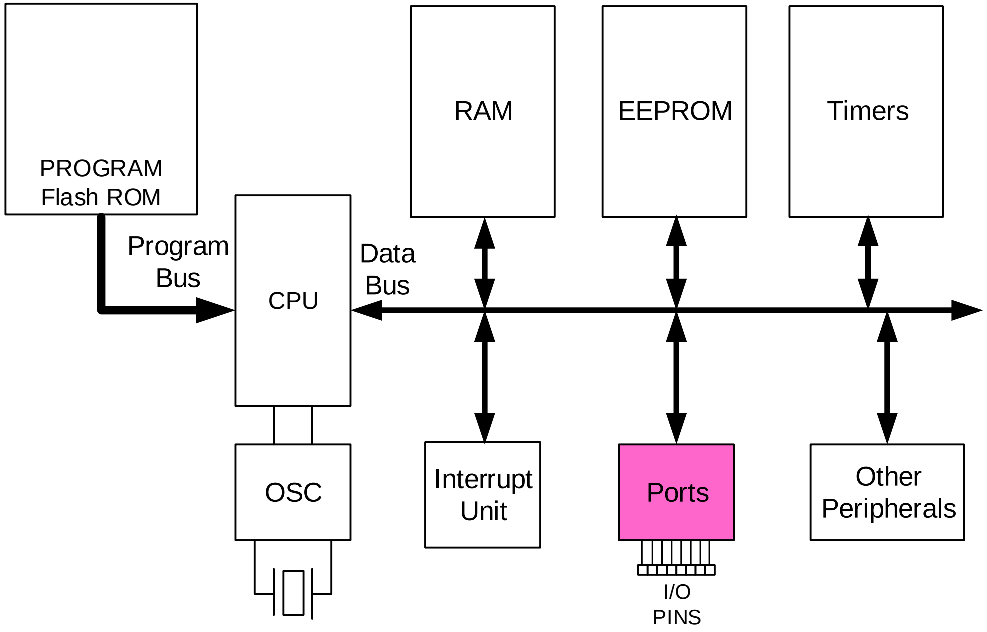
Peripherals of ATmega328p

- 23 general purpose I/O pins
- 6 PWM channels
- 6 ADC channels
- Two 8-bit & one 16-bit timer/counters
- Real time counter with separate oscillator
- Serial USART
- SPI & I^2C serial interfaces
- Analog comparator
- Programmable watchdog timer

Outline (Cont'd)



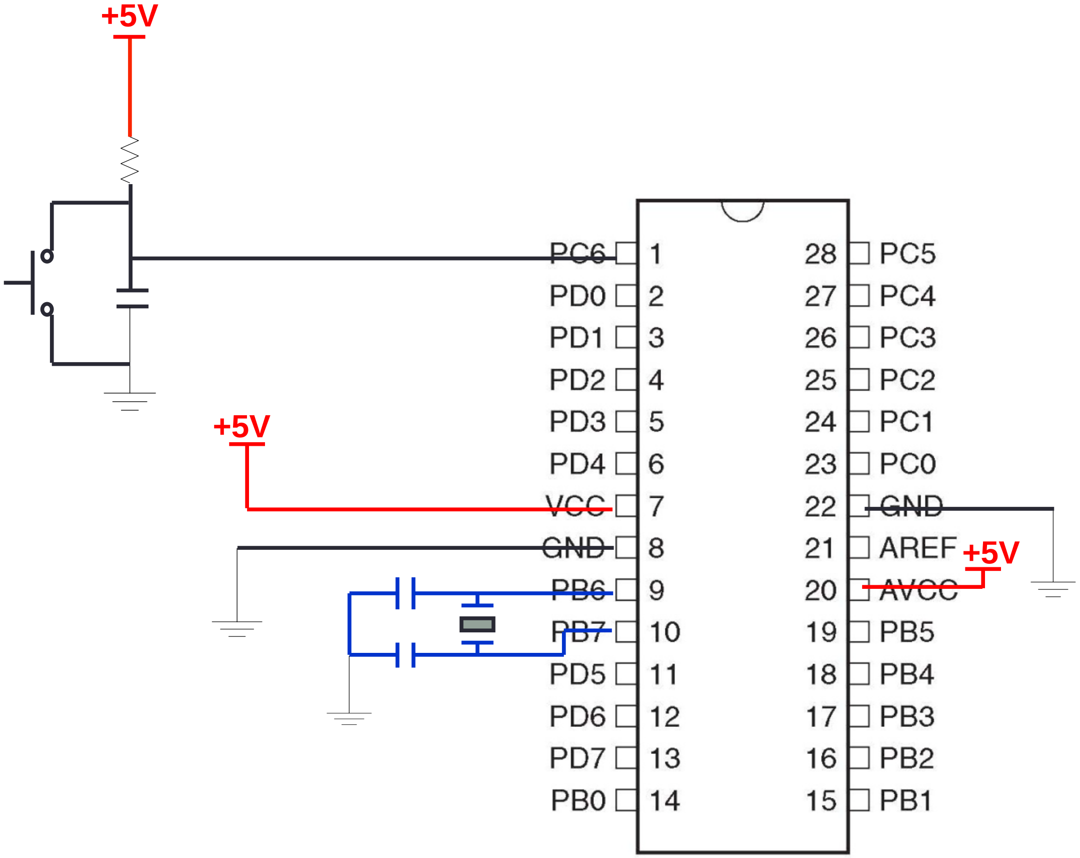
- AVR microcontrollers
  - Family
  - ATmega328P
  - Features
- AVR I/O
  - Pinout of ATmega328P
  - I/O registers
  - I/O programming
  - Program download
- Getting started



Pinout of ATmega328P

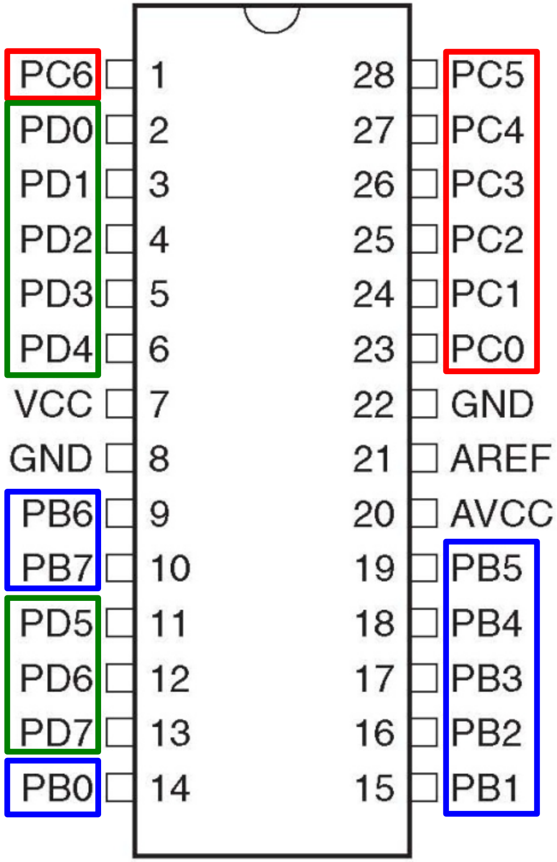
1. Vital Pins:
   - Power
     - VCC
     - Ground
   - Crystal
     - XTAL1
     - XTAL2
   - Reset
2. ADC pins
   - AVCC
   - GND
   - AREF
3. I/O pins

How many pins are available for I/O?



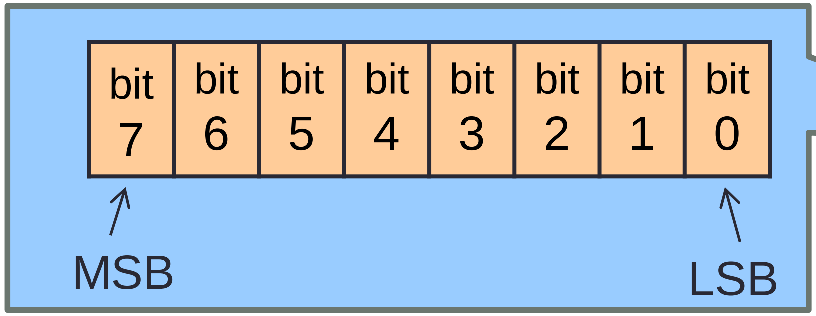
I/O Ports

- Connecting to the outside world
- Each pin can be used as either an input or an output at a time
- Grouped into three ports:
  - Port B: PB0 – PB7
  - Port C: PC0 – PC6
  - Port D: PD0 – PD7

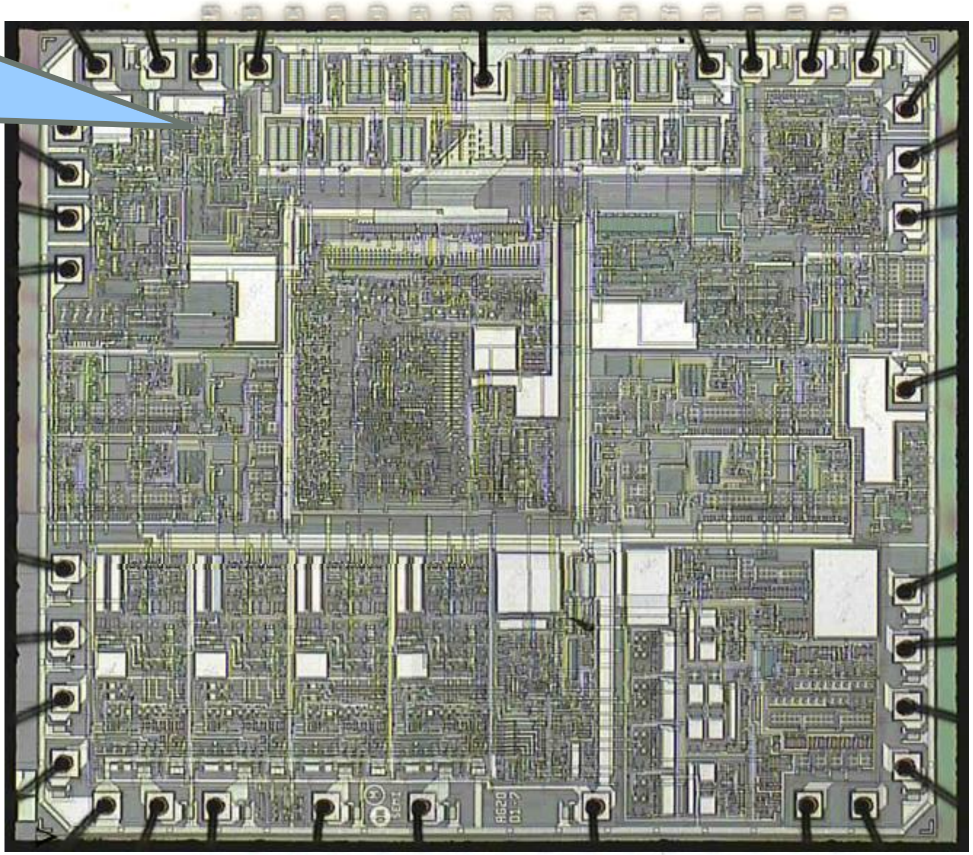


Registers

- A register is a device that holds a small set of data
- 8-bit registers on ATmega328P



bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

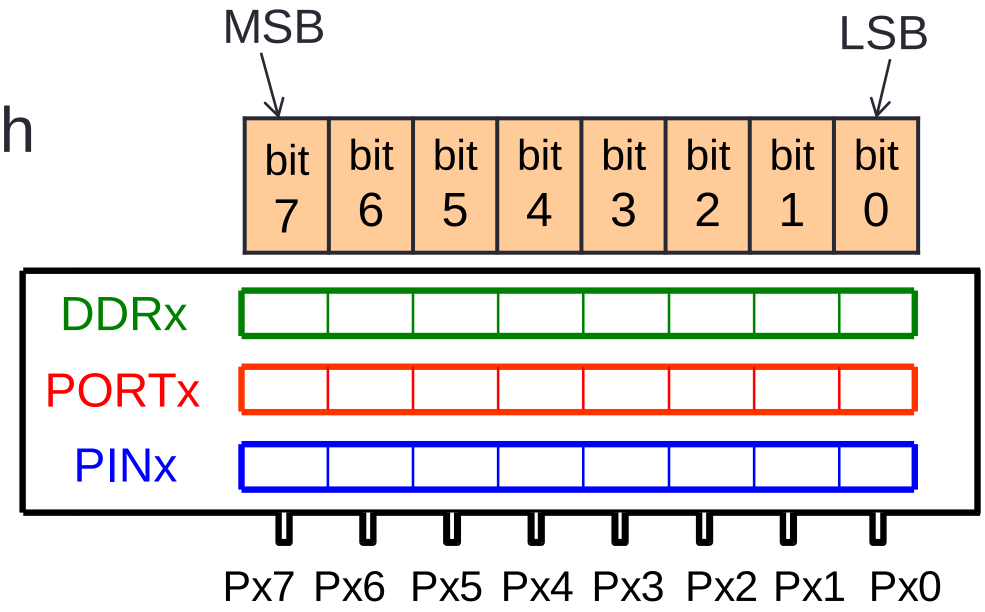


- Three types of registers
  - Data register
  - Control register
  - Status register

I/O Registers

- Each port is associated with three 8-bit registers:
  - DDRx: data direction
  - PORTx: output
  - PINx: input

  where x = B, C, or D



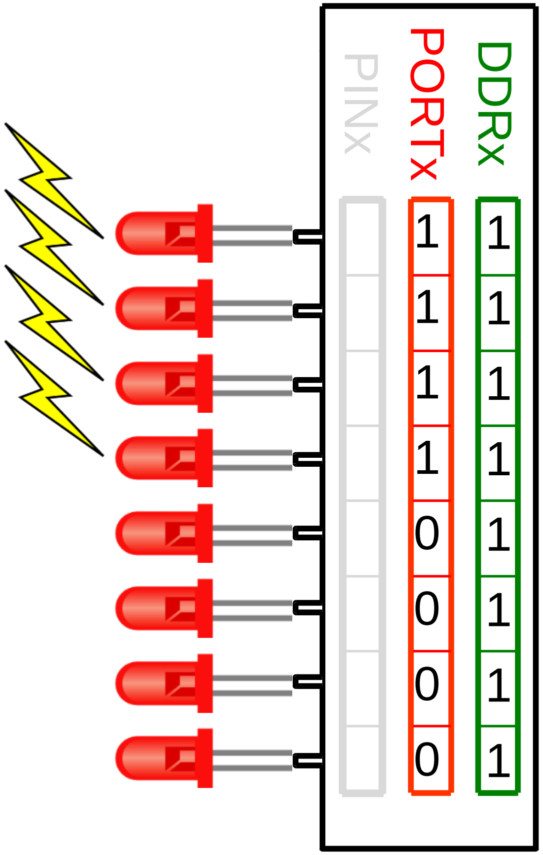
- Appropriate controls of the registers enable the input or output of the ports
- Bit to bit corresonding

Output Mode

- DDRx = 1
- PINx is NOT used
- PORTx contains the output voltages
  - 1: logic HIGH – 5V
  - 0: logic LOW – 0V

- For example:

```
DDRD=0b11111111;
PORTD=0b11110000;
```

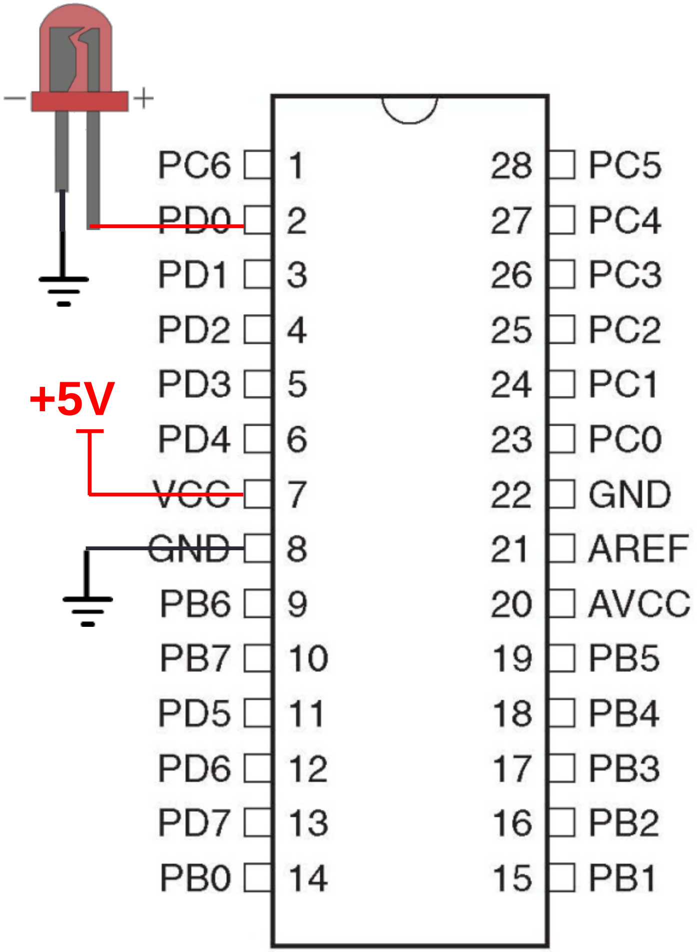


PINx	PORTx	DDRx
	1	1
	1	1
	1	1
	1	1
	0	1
	0	1
	0	1
	0	1

Example: Flashing An LED

- Connect an LED to PD0 and flash it at a frequency of 1 Hz
- Pseudo program code:

```
DDRD=0b11111111;
while(1){
   PORTD=0b00000001;
   delay(500);
   PORTD=0b00000000;
   delay(500);
}
```



Trilogy of AVR MCU Programming

1. Write source code in assembler or higher language
2. Compile or assemble the code to obtain the executable file (hex-file), which is usually called “firmware image” or “machine code”
3. Use a programmer and software to download firmware image to a microcontroller

Programming

Compiling

Hex-file Downloading

Programming and Compiling

- Integrated development environment (IDE)
  - Atmel AVR studio
- Language to program the microcontroller
  - C/C++
  - Assembly

Example Program Code

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRD=0b11111111;
    while (1){
       PORTD=0b00000001;
       _delay_ms(500);
       PORTD=0b00000000;
       _delay_ms(500);
    }
}
```

- Check:

\util\delay.h

\avr\iom328p.h

\avr\io.h

in “\\Program Files (x86)\Atmel\Studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\”

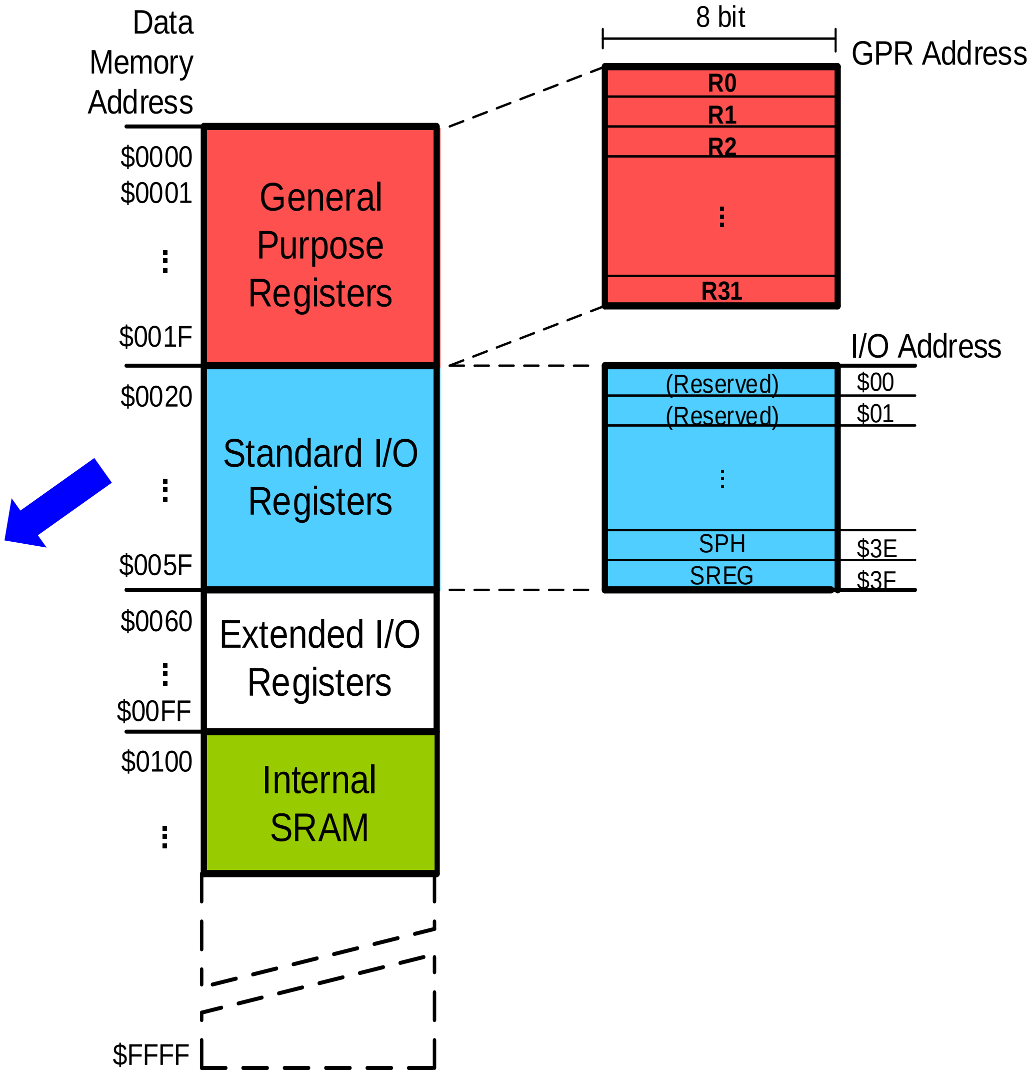
Atmel AVR Toolchain

- A collection of tools/libraries used to create applications for AVR MCU
- PDF library reference
- Online library reference
- Commonly used libraries:
  - <math.h>
  - <time.h>
  - <avr/interrupt.h>
  - <util/delay.h>
  - <stdio.h>
  - <stdlib.h>

Name and Address of I/O Registers

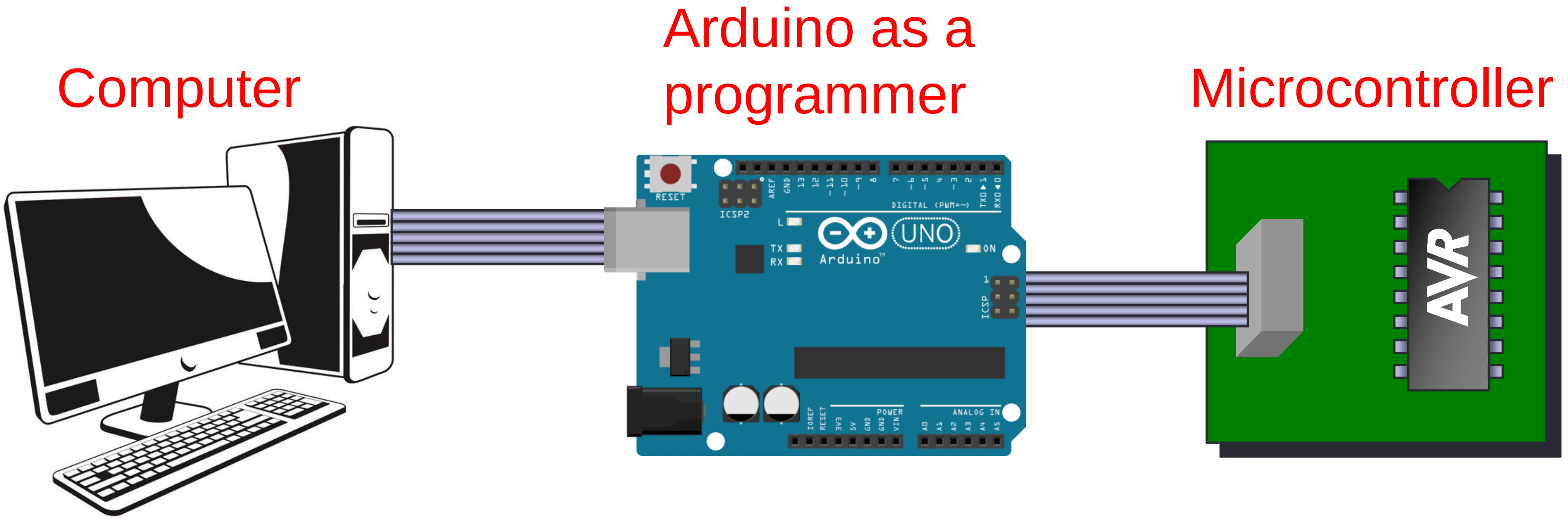
- I/O registers has
  - Memory address
  - I/O address
  - Name

Address		Name
I/O	Memory	
0x03	0x23	PINB
0x04	0x24	DDRB
0x05	0x25	PORTB
0x06	0x26	PINC
0x07	0x27	DDRC
0x08	0x28	PORTC
0x09	0x29	PIND
0x0A	0x2A	DDRD
0x0B	0x2B	PORTD



Hex-file Downloading

- Need appropriate tools for program downloading
- Program downloading using in-system programming (ISP) protocol
- Downloading through a programmer (e.g., Arduino)



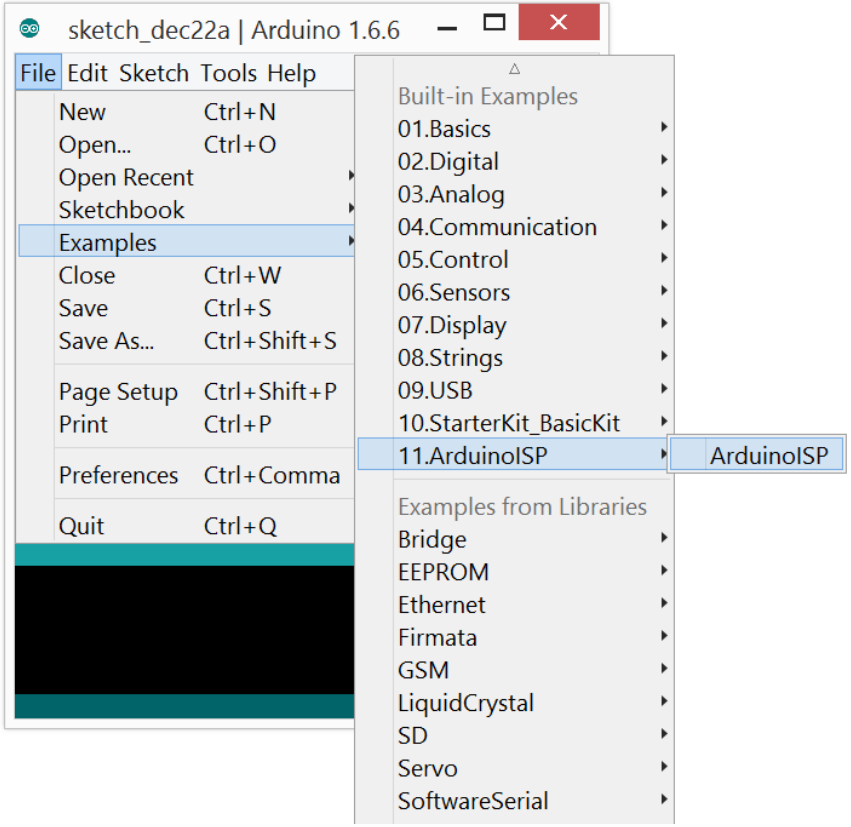
Computer

Arduino as a programmer

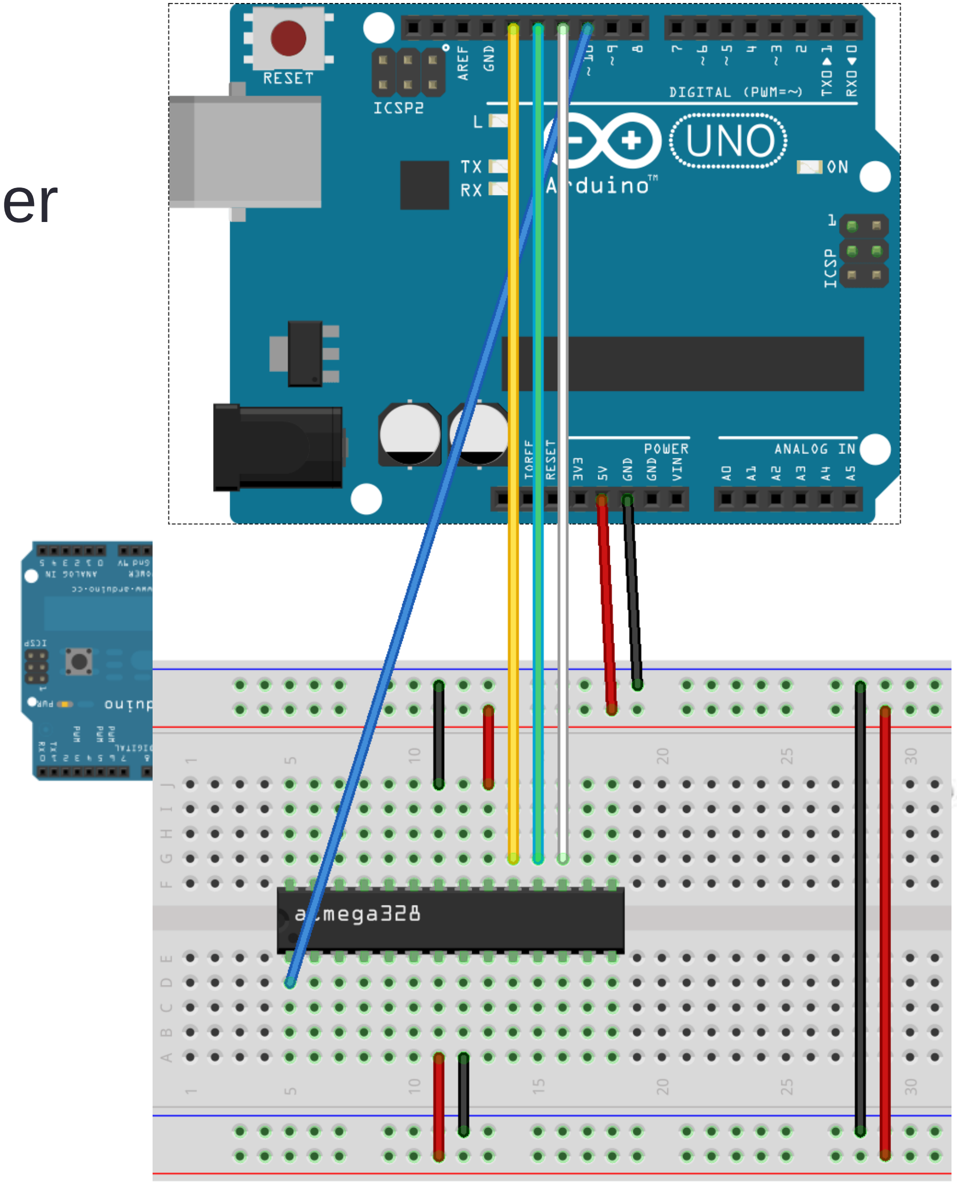
Microcontroller

ArduinoISP

- Turn Arduino into a programmer (i.e., program downloader)



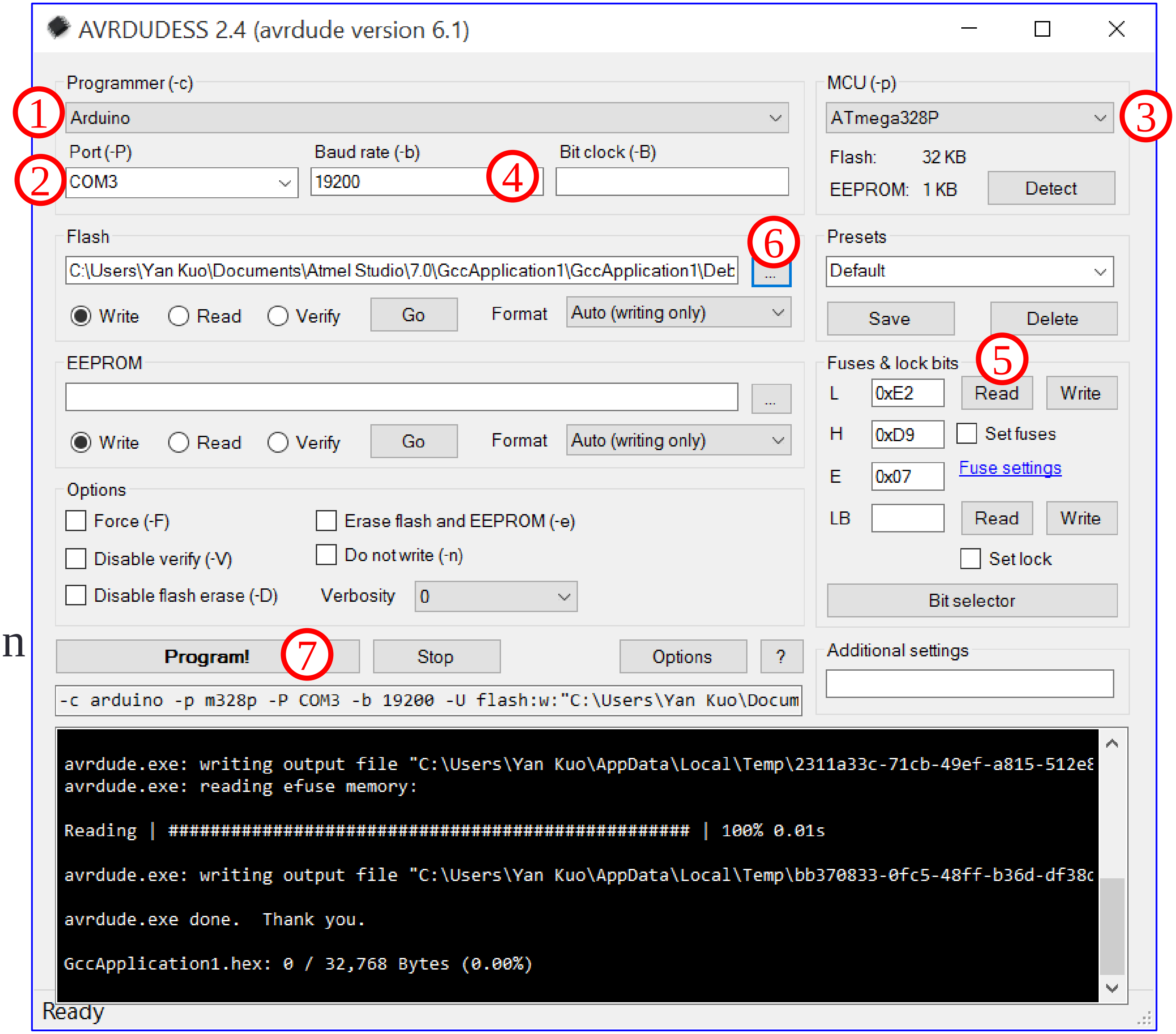
- Wire Arduino to ATmega328P



AVRDUDESS

- An utility to download programs to AVR MCU

1. Choose **Arduino**
2. Choose proper **Port**
3. Choose **Atmega328P**
4. Type “**19200**”
5. Confirm the connection by clicking **Read**
   Default fuse setting:
   L **E2**, H **D9**, E **07**
6. Choose the .hex file
7. Click **Program!**


AVRDUDESS 2.4 (avrdude version 6.1)

Programmer (-c): Arduino

Port (-P): COM3

Baud rate (-b): 19200

Bit clock (-B):

MCU (-p): ATmega328P

Flash: 32 KB

EEPROM: 1 KB

Flash: C:\Users\Yan Kuo\Documents\Atmel Studio\7.0\GccApplication1\GccApplication1\Deb

● Write ○ Read ○ Verify

Format: Auto (writing only)

EEPROM:

● Write ○ Read ○ Verify

Format: Auto (writing only)

Presets: Default

Fuses & lock bits

L: 0xE2

H: 0xD9

E: 0x07

LB:

□ Set fuses

□ Set lock

Options

□ Force (-F)

□ Erase flash and EEPROM (-e)

□ Disable verify (-V)

□ Do not write (-n)

□ Disable flash erase (-D)

Verbosity: 0

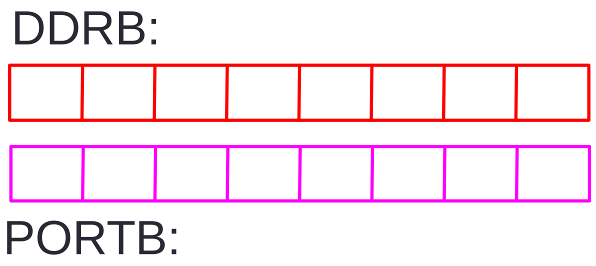
Additional settings:

```
-c arduino -p m328p -P COM3 -b 19200 -U flash:w:"C:\Users\Yan Kuo\Docum
```

```
avrdude.exe: writing output file "C:\Users\Yan Kuo\AppData\Local\Temp\2311a33c-71cb-49ef-a815-512e8
avrdude.exe: reading efuse memory:

Reading | ################################################## | 100% 0.01s

avrdude.exe: writing output file "C:\Users\Yan Kuo\AppData\Local\Temp\bb370833-0fc5-48ff-b36d-df38c

avrdude.exe done.  Thank you.

GccApplication1.hex: 0 / 32,768 Bytes (0.00%)
```

Ready

Example: Toggling Output


DDRB:

PORTB:

- Make all the 8 pins of Port B outputs
- Initialize the odd pins logic HIGH and the even pins logic LOW
- Toggle the pins forever between “on” and “off” states with a time delay of 500 ms

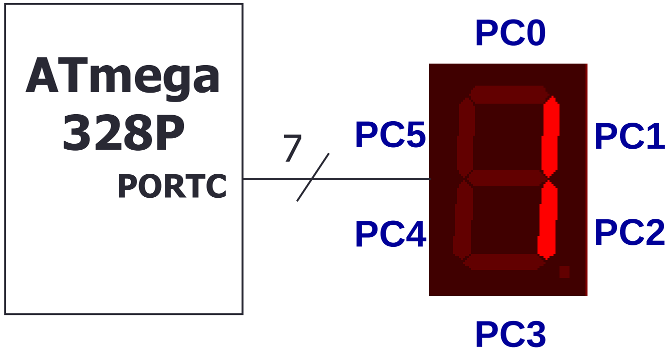
```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB=0xFF;       //0xFF=0b11111111
    while (1){
       PORTB=0x55;   //0x55=0b01010101
       _delay_ms(500);
       PORTB=0xAA;   //0xAA=0b10101010
       _delay_ms(500);
    }
}
```

Example: 7-segment LED

- A common cathode 7-segment is connected to Port C
- Display “1” on the 7-segment:

DDRC:	1	1	1	1	1	1	1	1
PORTC:	0	0	0	0	0	1	1	0

```
#include <avr/io.h>

int main(void)
{
   DDRC=0xFF;
   PORTC=0b00000110;
}
```

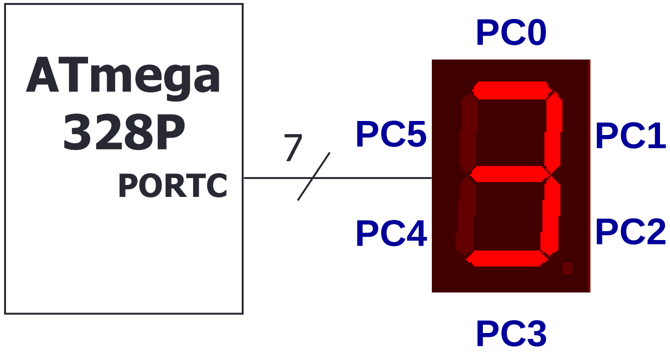


Example: 7-segment LED

- A common cathode 7-segment is connected to Port C
- Display “3” on the 7-segment:

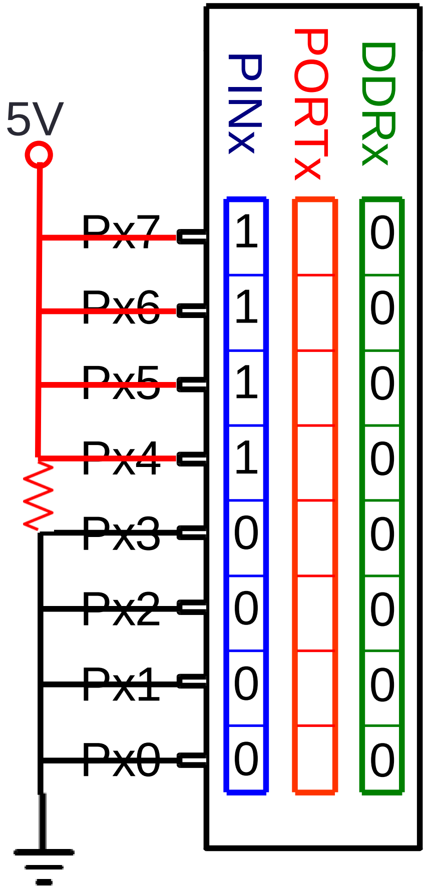
DDRC:	1	1	1	1	1	1	1	1
PORTC:	0	1	0	0	1	1	1	1

```
#include <avr/io.h>

int main(void)
{
   DDRC=0xFF;
   PORTC=0b01001111;
}
```



Input Mode (without Pull-up)

- DDRx = 0
- PINx contains the input logic signal
  - 1: logic HIGH – 5V
  - 0: logic LOW – 0V
- PORTx is NOT used (if not considering pull-up)



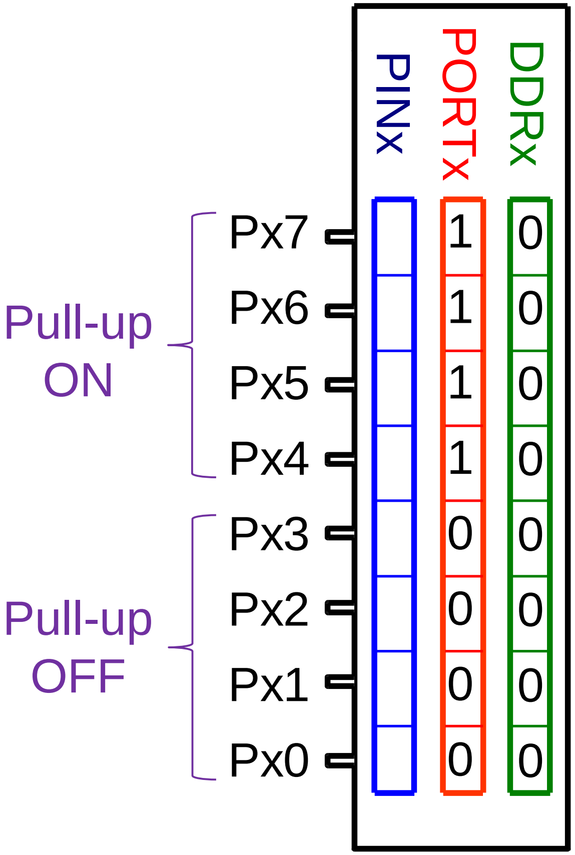
Example: Input (without Pull-up)

- Make all the pins of Port B input
- Make all the pins of Port D output
- Send the reading from Port B to Port D indefinitely, after adding the value 5 to it

```
#include <avr/io.h>

int main(void)
{
    DDRB=0x0;           // PORTB input
    DDRD=0xFF;          // PORTD output
    while (1){
       PORTD=PINB+5;
    }
}
```

Input Mode (with Pull-up)

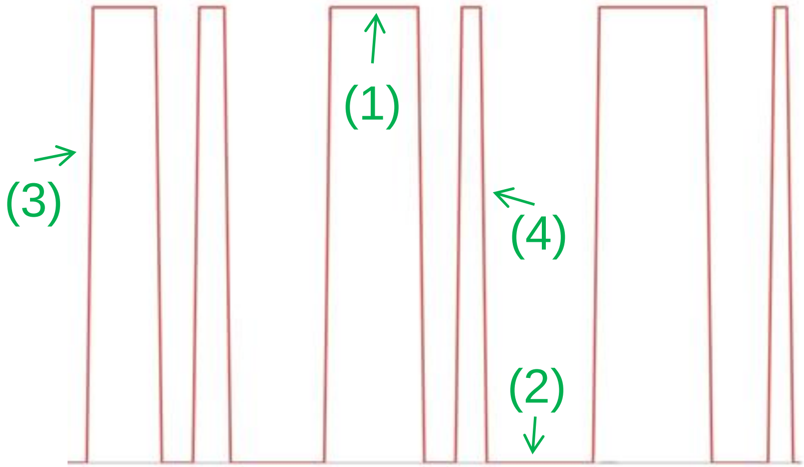
- DDRx = 0
- PINx contains the input logic signal
  - 1: logic HIGH – 5V
  - 0: logic LOW – 0V
- PORTx determines if pull-up resistors are connected or not
  - 1: pull-up ON
  - 0: pull-up OFF



	PINx	PORTx	DDRx
Px7		1	0
Px6		1	0
Px5		1	0
Px4		1	0
Px3		0	0
Px2		0	0
Px1		0	0
Px0		0	0

Transistor–transistor Logic (TTL)

- A digital signal is a waveform that switches between two voltage levels (logic HIGH and LOW)



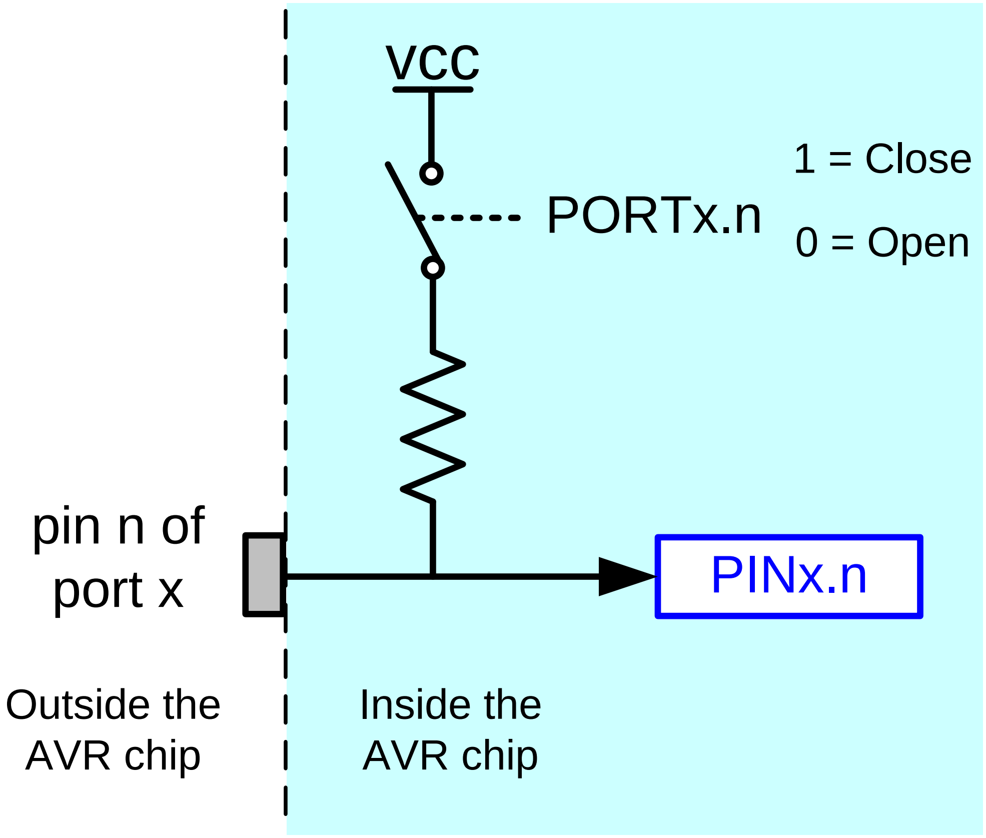
(1) Logic HIGH
(2) Logic LOW
(3) Rising edge
(4) Falling edge

- It is usual to allow some tolerance in the voltage levels

LOW voltage	HIGH voltage	Notes
0 V to 0.8 V	2 V to V_{CC}	V_{CC} is 4.75 V to 5.25 V

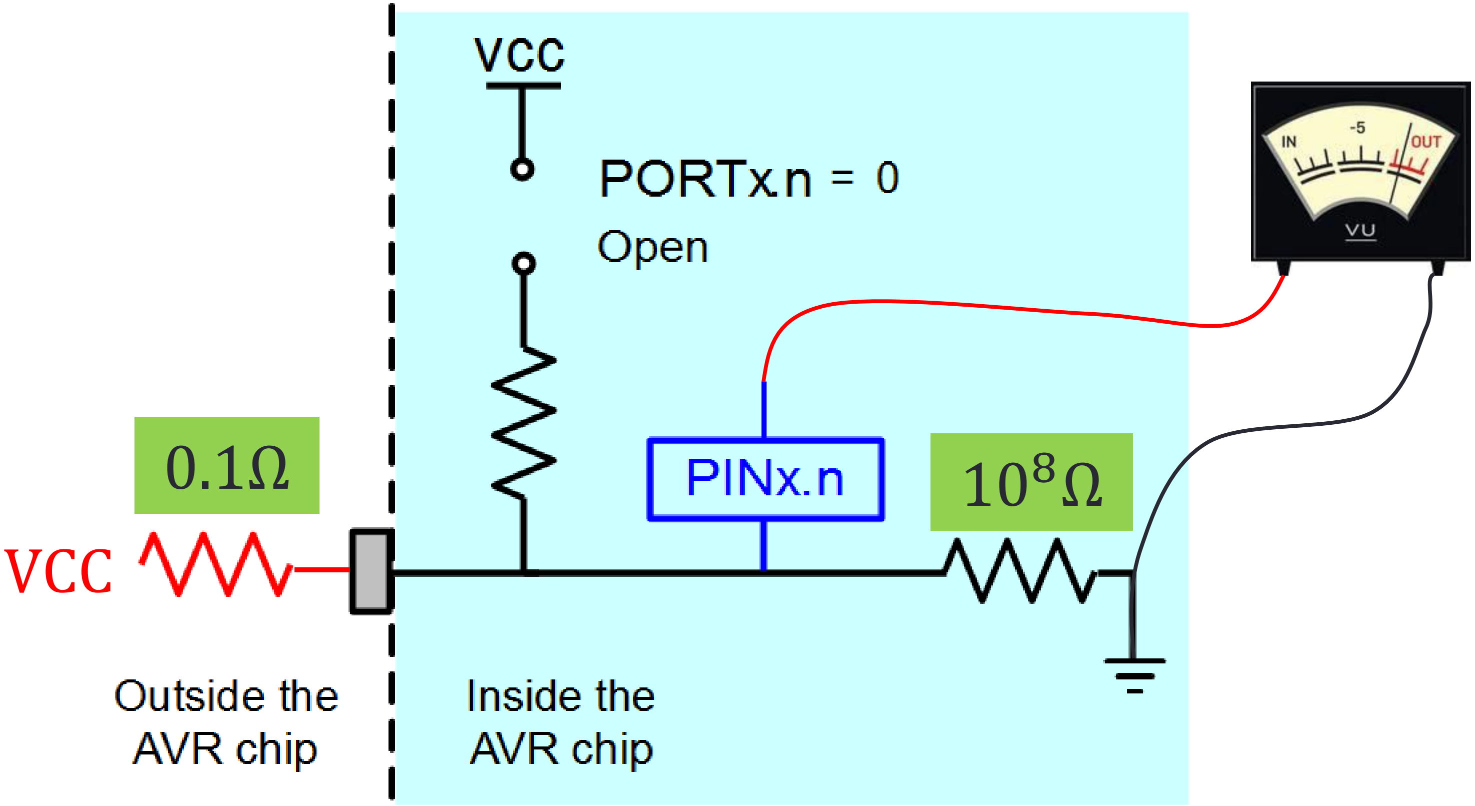
Pull-up Resistor

- Used to ensure that inputs settle at expected logic levels (HIGH or LOW)
- The PORTx determines if the pull-up resistor is "open" or "close" (off) (on)



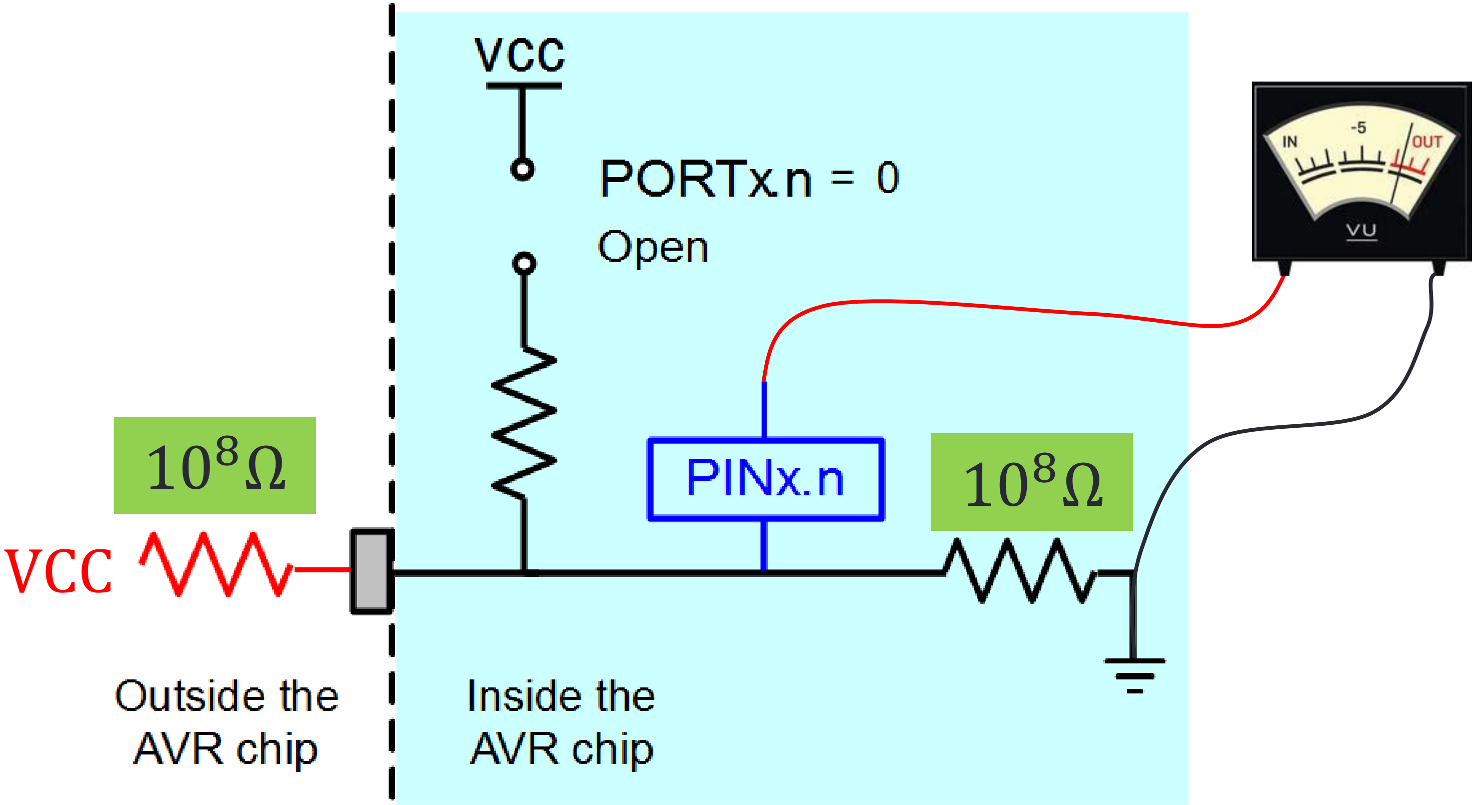
Low Impedance and Pull-up Off

- The input reading logic HIGH when the external device is of low impedance



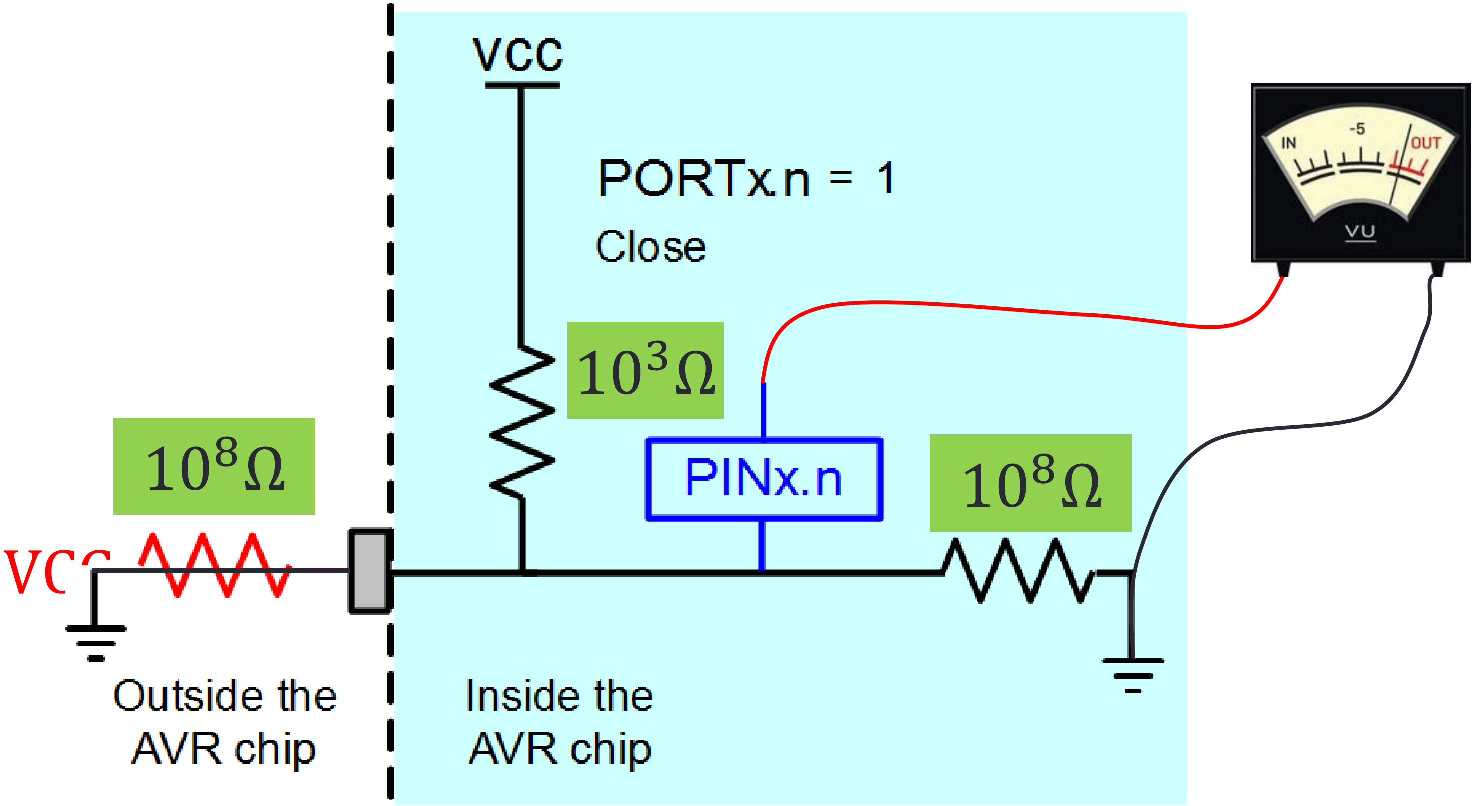
High Impedance and Pull-up Off

- The input reading is “floating” between logic LOW and HIGH when the external device is of high impedance



Pull-up On

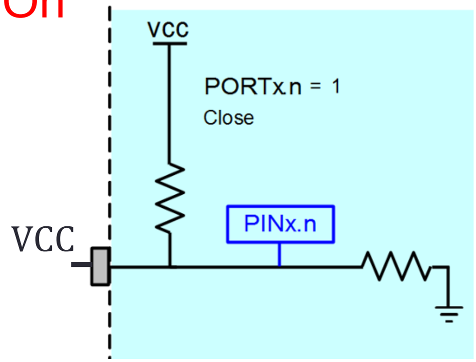
- The pull-up resistor brings an input at expected logic levels no matter what the impedance of the external device is



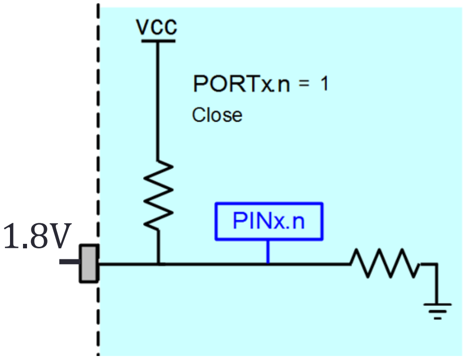
Summary of Pull-up Resistor

Input: VCC Input: 1.8V Input: GND Input: Open

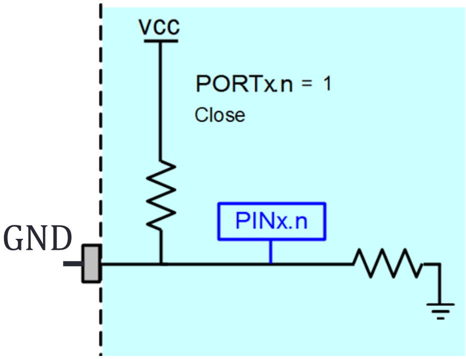
On



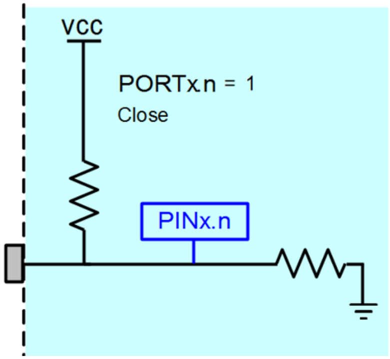
Reading:



Reading:

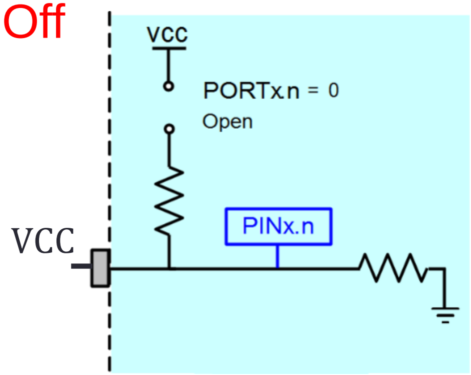


Reading:

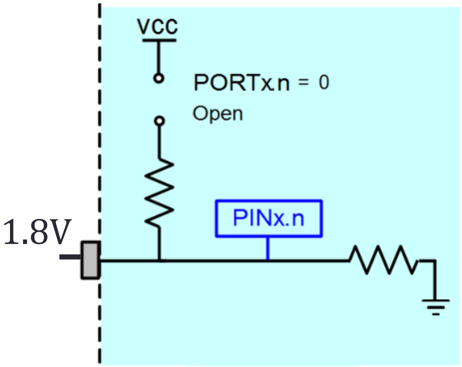


Reading:

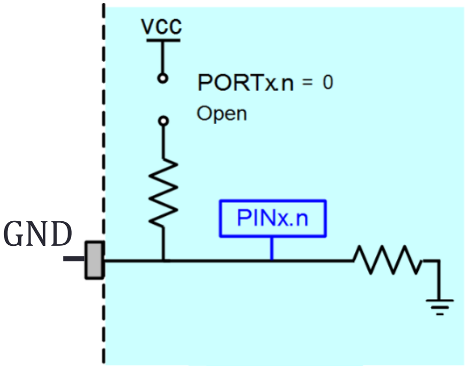
Off



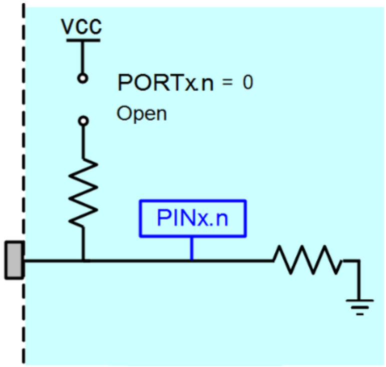
Reading:



Reading:



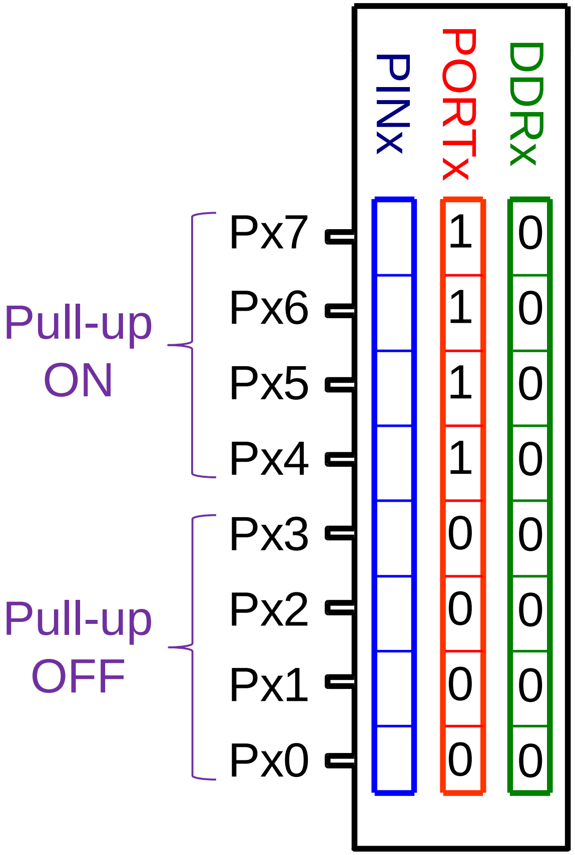
Reading:



Reading:

Input Mode (with Pull-up)

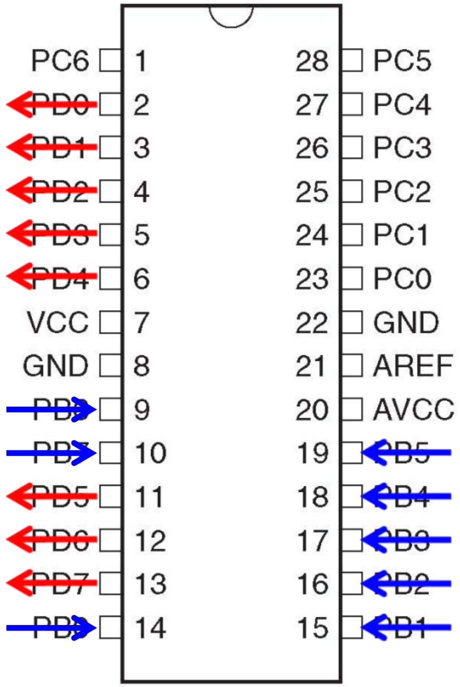
- DDRx = 0
- PINx contains the input logic signal
  - 1: logic HIGH – 5V
  - 0: logic LOW – 0V
- PORTx determines if pull-up resistors are connected or not
  - 1: pull-up ON
  - 0: pull-up OFF



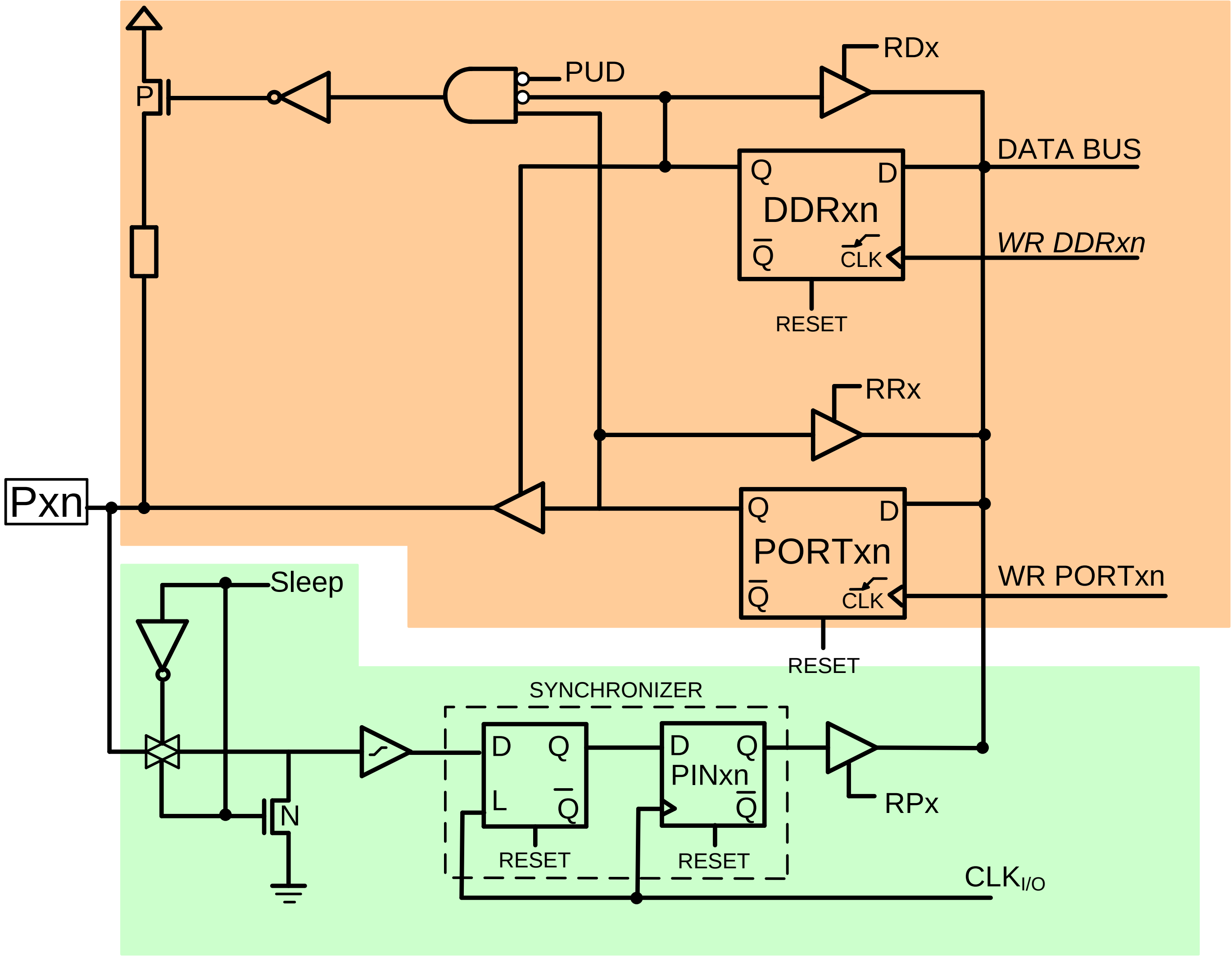
Example: Input (with Pull-up)

- Write a program that reads Port B and writes the readings to Port D indefinitely
- Enable pull-up resistors of Port B

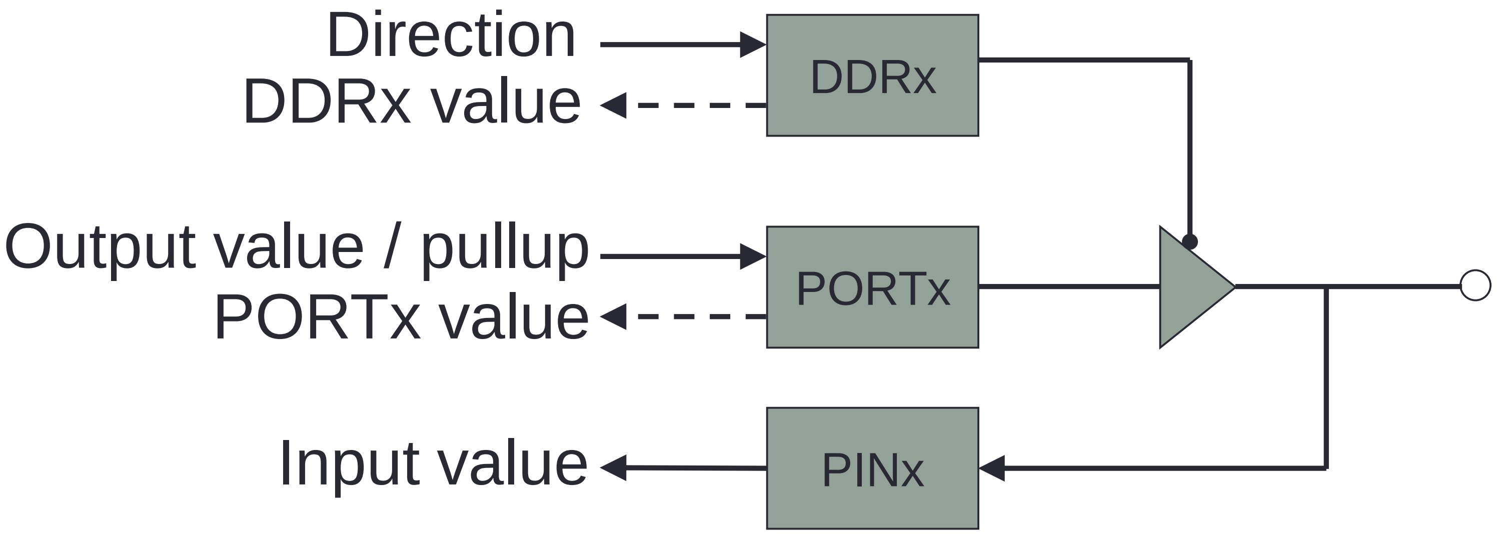
```
#include <avr/io.h>

int main(void)
{
    DDRB=0x0;        // PORTB input
    PORTB=0xFF;      // Pull-up enabled
    DDRD=0xFF;       // PORTD output
    while (1){
       PORTD=PINB;
    }
}
```



The Structure of I/O Pins



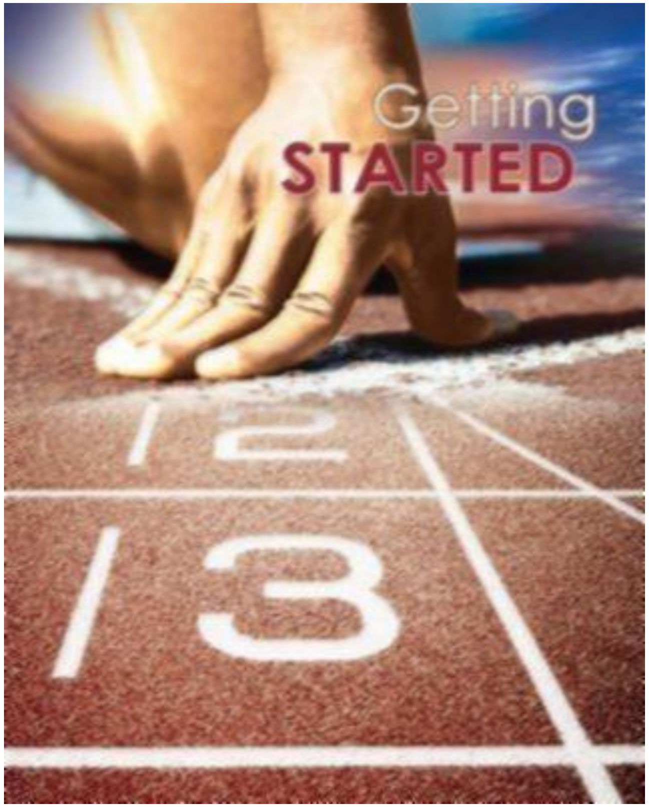
Summary of I/O



I/O function	DDRx	PORTx	Pull-up	Comment
Input	0	0	No	
Input	0	1	Yes	Pin will source current if external pulled LOW
Output	1	0	N/A	Output LOW
Output	1	1	N/A	Output HIGH

Outline (Cont’d)

- AVR microcontrollers
  - Family
  - ATmega328P
  - Features
- AVR I/O
  - Pinout of ATmega328P
  - I/O registers
  - I/O programming
  - Program download
- Getting started



Further Reading

- AVR studio:
  - http://www.atmel.com/microsite/avr_studio_5/default.aspx
- ATmega328p:
  - http://www.atmel.com/devices/ATMEGA328P.aspx
- Others:
  - http://www.avrfreaks.net
  - http://www.wrighthobbies.net

Reference

- ATmega328P data sheet
- AVR 8-bit instruction set
- M. A. Mazidi, S. Naimi, and S. Naimi, *The AVR Microcontroller and Embedded Systems: Using Assembly and C*, Prentice Hall, 2010
- AVR GCC library help http://nongnu.org/avr-libc/user-manual/modules.html