Topic 7. Introduction to Microcontrollers

Luis Mejias

Outline

- Microcontroller basic definitions
- Registers on the microcontroller
- Development environment
- Appendix
  - Operations with bits (covered in week 6)

Introduction to microcontrollers

- What is a microcontroller?
  - An integrated chip that is often part of an embedded system.
  - It is programmed to perform a single specific task/application. They are really just “mini-computers”.
  - A micro includes CPU, RAM, ROM, I/O ports and timers all in a single chip.

Introduction to microcontrollers

- What is a microcontroller?
  - They do not need to be powerful because most applications only require a clock of a few Mhz and a small amount of storage.
  - A microcontroller needs to be programmed to be useful
  - Microcontrollers are only as useful as the code written for it.
    - If you wanted to turn on a red light when the temperature reached a certain point, the programmer would have to explicitly specify how that will happen through his code

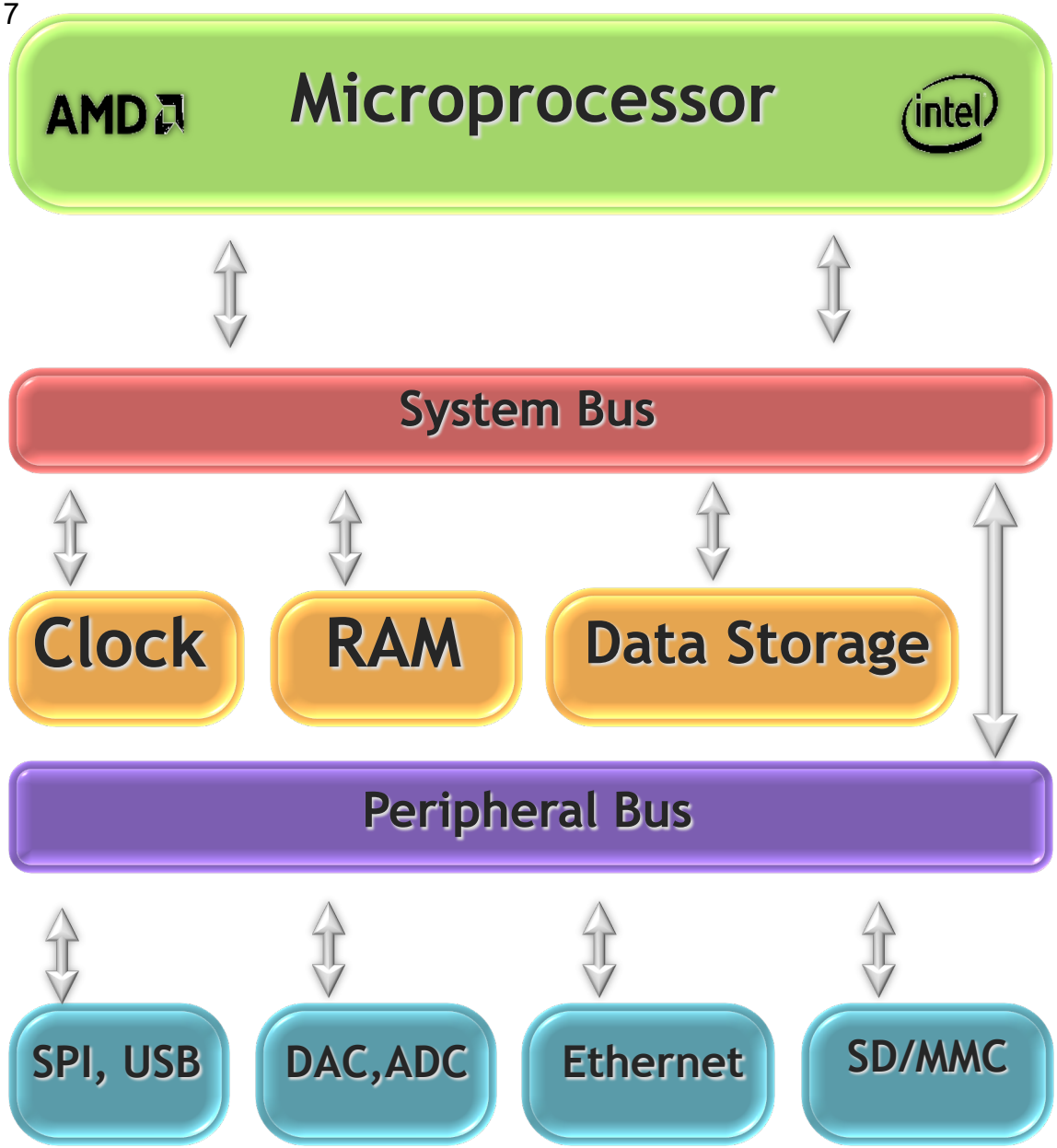
Introduction to microcontrollers

Where do you find them?

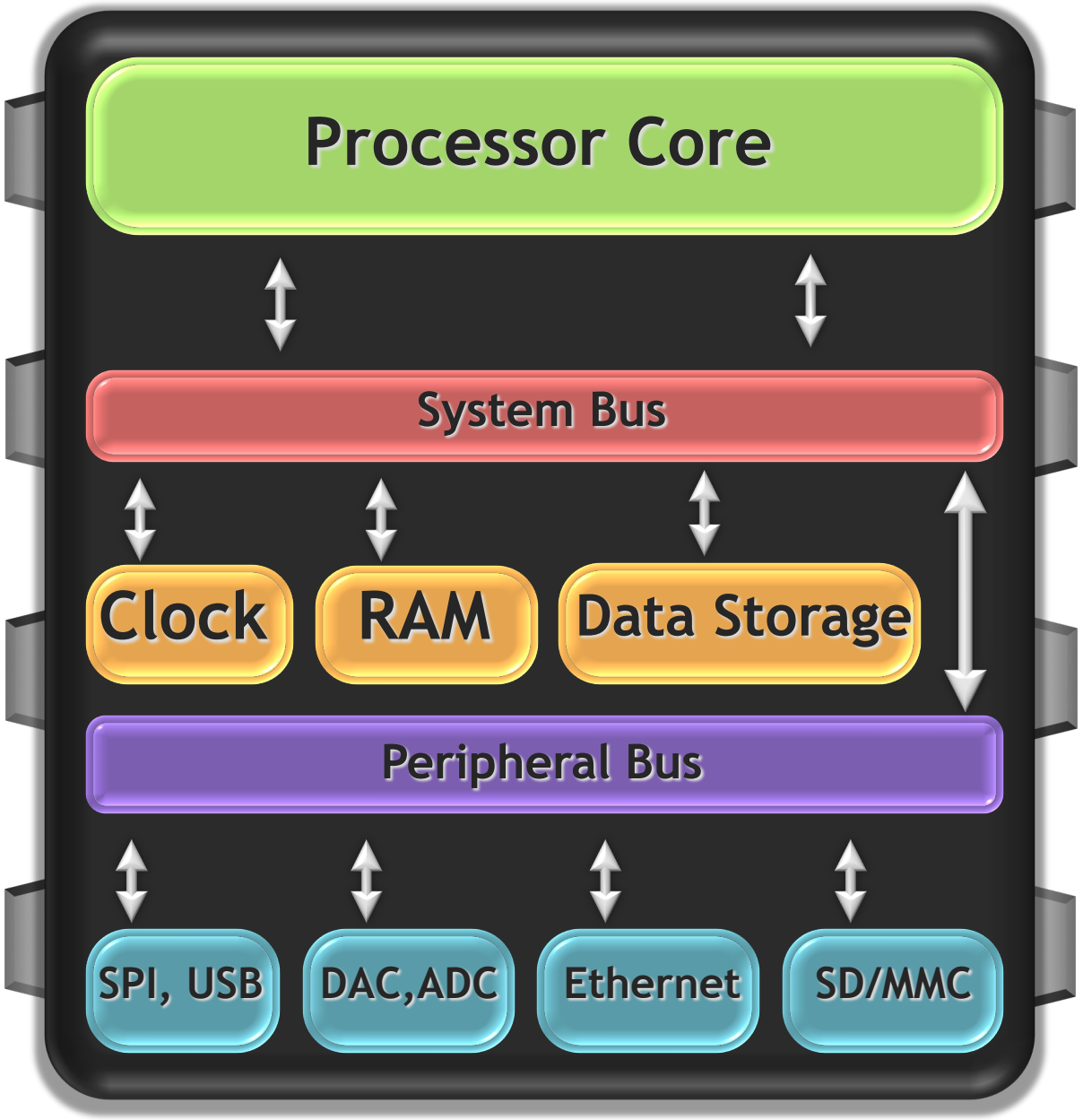
- Microcontrollers are hidden in tons of appliances, gadgets, and other electronics.
- They're everywhere!

Introduction to microcontrollers

- Microcontroller programming
  1. Code is written for the microcontroller in an integrated development environment (IDE), text editor or web-based simulation environment. The code is written in a programming language (e.g c, basic or assembly).
  2. The IDE debugs the code for errors, and then compiles it into binary code which the micro-controller can execute.
  3. A programmer (a piece of hardware, not the person) is used to transfer the code from the pc to the microcontroller. The most common type of programmer is an ICSP (in-circuit serial programmer).
  4. In case, of web-based coding and simulation environment such as TinkerCad, the compilation and executing is performed on a web server.



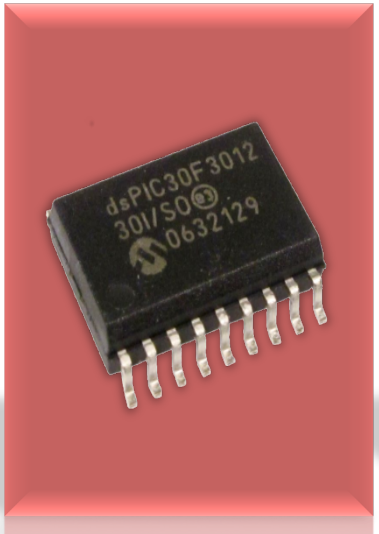
Microprocessor



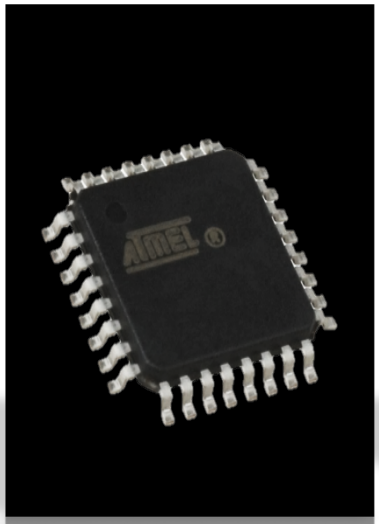
Microcontroller

Microcontroller Packaging

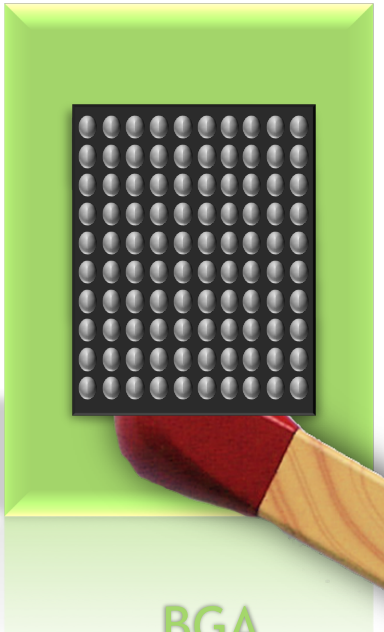
DIP
(Dual Inline Package)
Through hole
8 pins
9mm x 6mm
0.15pins/mm^2



SOIC
(Small Outline IC)
Surface Mount
18 pins
11mm x 7mm
0.23pins/mm^2



QFP
(Quad Flat Package)
Surface Mount
32 pins
7mm x 7mm
0.65pins/mm^2



BGA
(Ball Grid Array)
Surface Mount
100 pins
6mm x 6mm
2.78pins/mm^2

	Microprocessor	Microcontroller
Applications	General computing (i.e. Laptops, tablets)	Appliances, specialized devices
Speed	Very fast	Relatively slow
External Parts	Many	Few
Cost	High	Low
Energy Use	Medium to high	Very low to low
Vendors	intel AMD ARM	ATMEL ST Texas Instruments Microchip

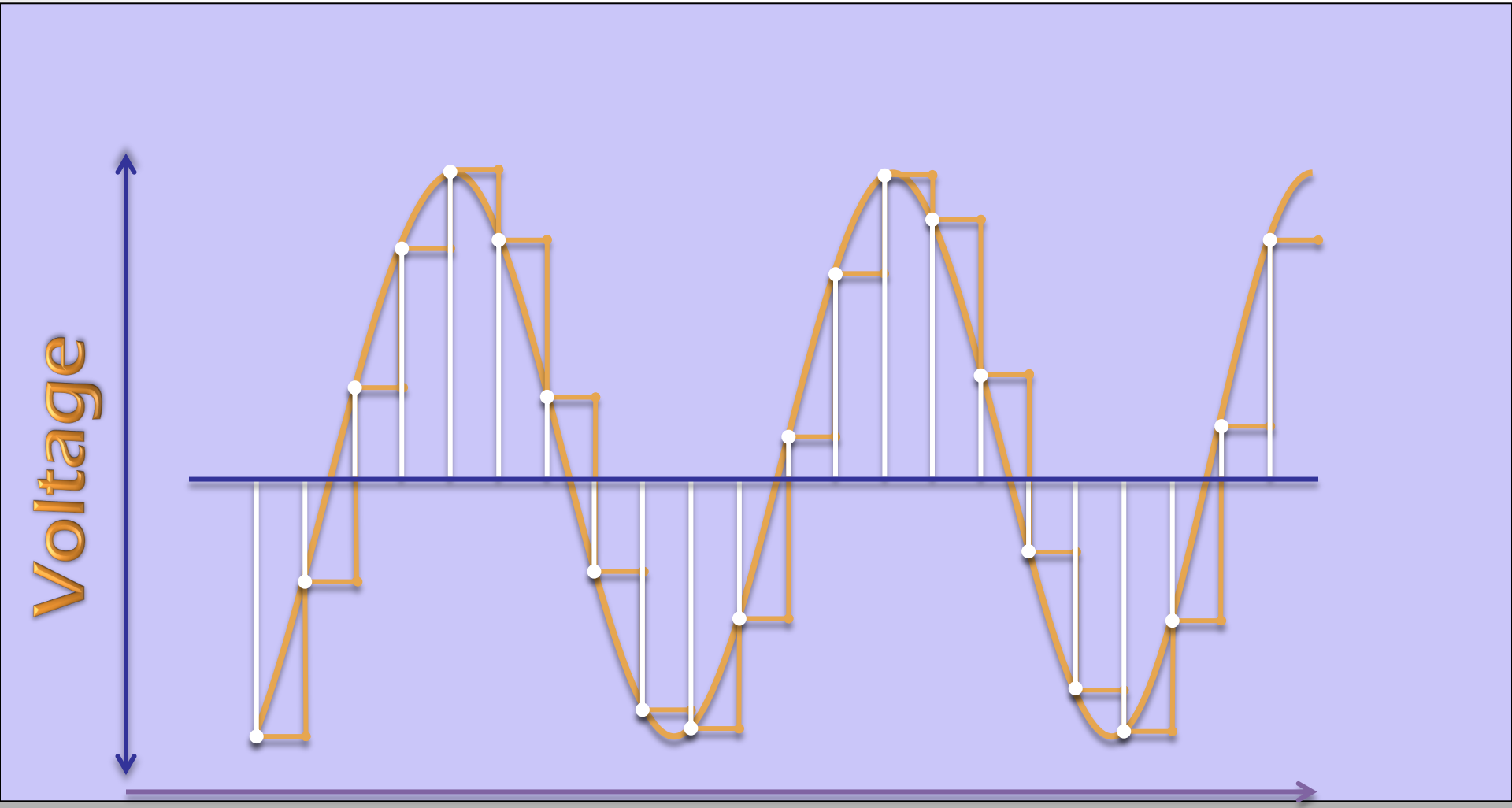
The Analog to Digital Converter (ADC)

- Just about every modern microcontroller contains an ADC(s).

- It converts analog voltages (analog signals) into digital values.

- These digital representations of the signal at hand can be analyzed in code, logged in memory, or used in practically any other way possible.

The Analog to Digital Converter (ADC)



The Digital to Analog Converter (DAC)

- You guessed it! Microcontrollers have accompanying DACs.

- It does exactly the opposite function of an ADC. It takes a digital value and converts it into an pseudo-analog voltage.

- It can be used to do an enormous amount of things. One example is to synthesize a waveform. We can create an audio signal from a microcontroller. Imagine that!

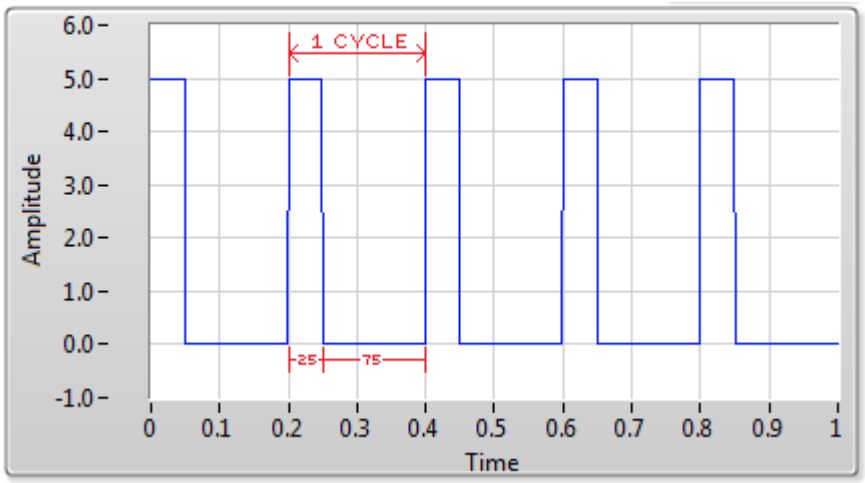
Pulse-Width Modulation (PWM)

- Modulation technique used to encode information into a signal, although its main use is for regulating power supplied to a load.

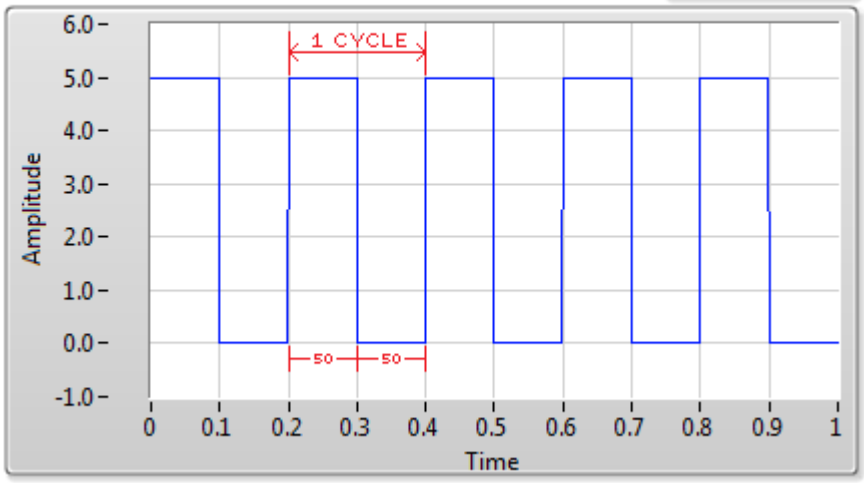
- An Analog signal can be generated using a digital source.

- Consist of two main components that define its behavior: a duty cycle and a frequency.
  - The duty cycle describes the amount of time the signal is in a high (on) stated as a percentage of the total time of it takes to complete one cycle.
  - The frequency determines how fast the PWM completes a cycle (i.e. 1000 Hz would be 1000 cycles per second), and therefore how fast it switches between high and low states.
- By cycling a digital signal off and on at a fast enough rate, and with a certain duty cycle, the output will appear to behave like a constant voltage analog signal when providing power to devices.

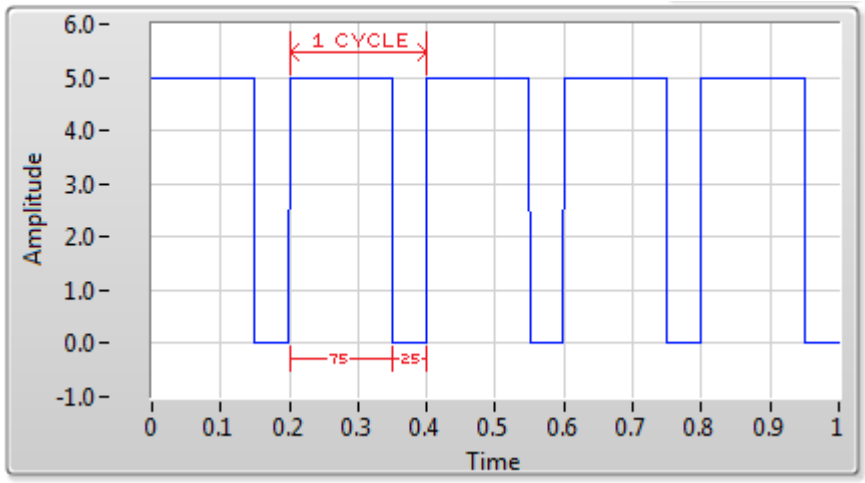
Pulse-Width Modulation (PWM)



25% duty cycle



50% duty cycle



75% duty cycle

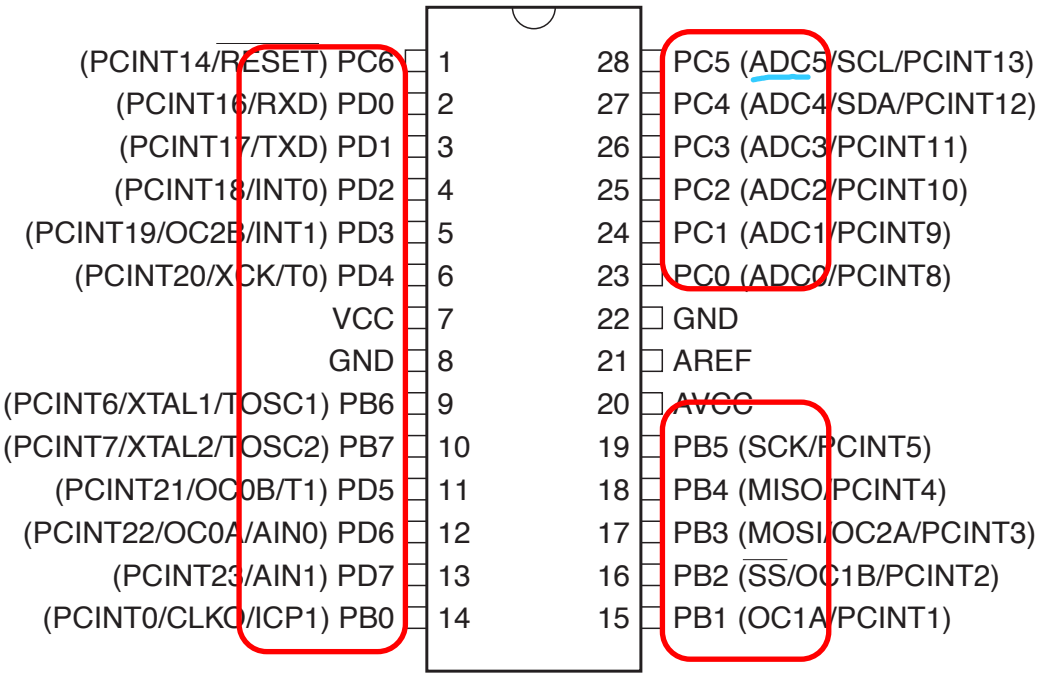
Microcontroller Applications

- Many robots use microcontrollers to allow robots to interact with the real world.

- Ex. If a proximity sensor senses an object near by, the microcontroller will know to stop its motors and then find an unobstructed path.

ATmega328P pinout

- Atmega328P has three 8-bit digital IO ports
  - Ports B, C, and D
- Each port has 8 data pins
- Each port is bidirectional
  - Input or
  - Output



Atmega328P Datasheet

Features

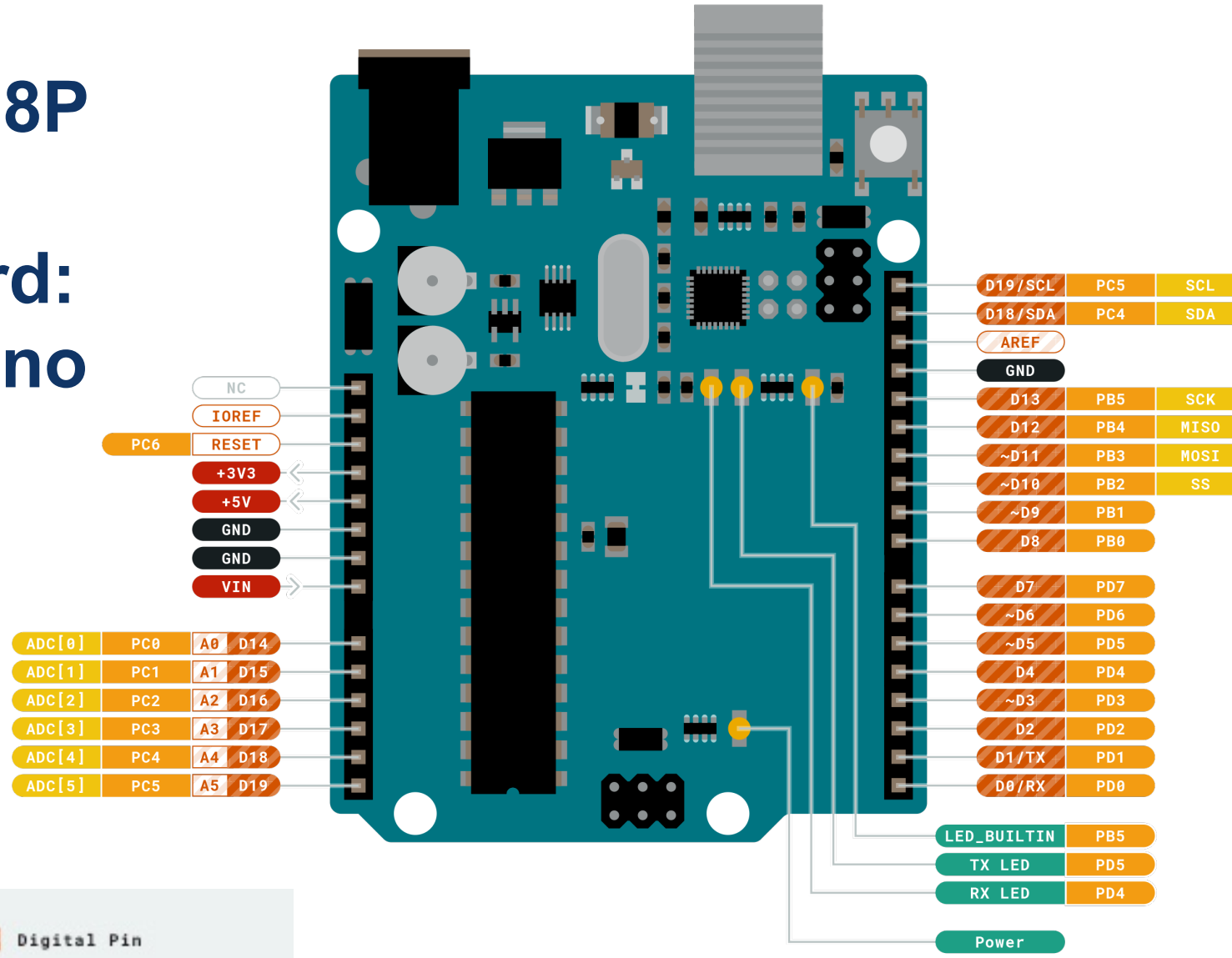
- High Performance, Low Power AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
  - 256/512/512/1KBytes EEPROM
  - 512/1K/1K/2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix™ acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

- Datasheet can be found on blackboard under Learning Resources – Microcontrollers

and under

- Topic 7 – Intro to Microcontrollers

Atmega328P pinout Host board: Arduino uno



Atmega328P Registers

- Registers are special storage with 8 bits capacity

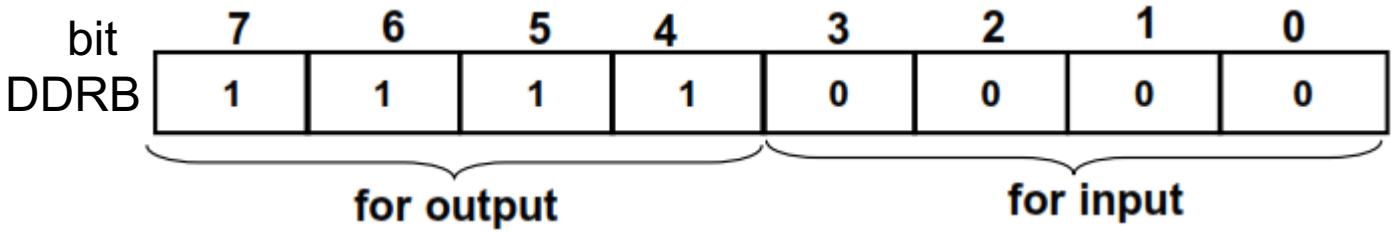
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

- The special character of registers, compared to other storage locations, is that
  - they are connected directly to the central processing unit called the accumulator,
  - they can be used directly in assembler instructions, either as a target register for the result or as read register for a calculation or transfer,
  - operations on their content require only a single instruction.

Atmega328P
Data Direction Register (DDRx)

- Configures pins in a port to be output (1) or input (0)
  - For example, if we want to set Port B pins 0 to 3 for input and pins 4-7 for output, we would write the following C code:

```
DDRB = 0b11110000;
```



Atmega328P
Data Register (PORTx)

- Writes output data to port
  - For example, if want to write a binary 0 to output pin 6, binary 1 to other pins of Port B, we write C code:

```
PORTB = 0b10111111;
```

Atmega328P
Input Pins Address (PINx)

- Reads input data from port
  - For example, if we want to read the input pins of Port B, we write C code:

```
unsigned char temp; // temporary variable
temp = PINB;        // read input
```

Development Environment

We will be using TinkerCad Circuits

www.tinkedcad.com

This environment will allow to design a circuit using microcontrollers, program it and execute the code.

You will need to create an account.

See lecture notes for additional details.

Summary

- Key learning objectives:
  - Differences between microcontroller and CPU
  - Registers on the AVR Atmega328P.
  - TinkedCad circuits for project design and code execution.
  - Appendix:
    - Operations with bits: bitwise operator to handle reading and setting individual pins in a given port

Appendix

Operations with Bits – Intro in the context of microcontrollers

- Covered in detail in Lecture / tutorial week 6.
- Binary operations
  - Bit shift: this operation move/shift bits in a particular direction (either left or right)
  - There are two bit shift operators
  - The left shift operator <<
  - The right shift operator >>
  - These operators cause the bits in the left operand to be shifted left or right, by the number of positions specified by the right operand.

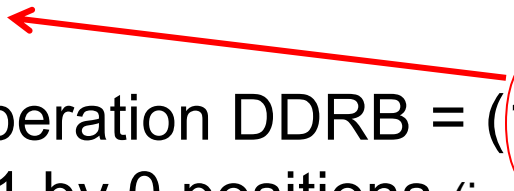
    In a left shift, bits get "shifted out" on the left and 0 bits get "shifted in" on the right.
    The opposite goes for a right shift

Operations with Bits - Intro

- Binary operations
  - Bit shift:
  - Let's say DDRB has an initial value of 0b00000000

In binary is 0b00000001

DDRB = 0b00000000;



  - The operation DDRB = (1<<0); shift bits in the binary value 1 by 0 positions (in practice nothing has been shifted)

DDRB = 0b00000001;

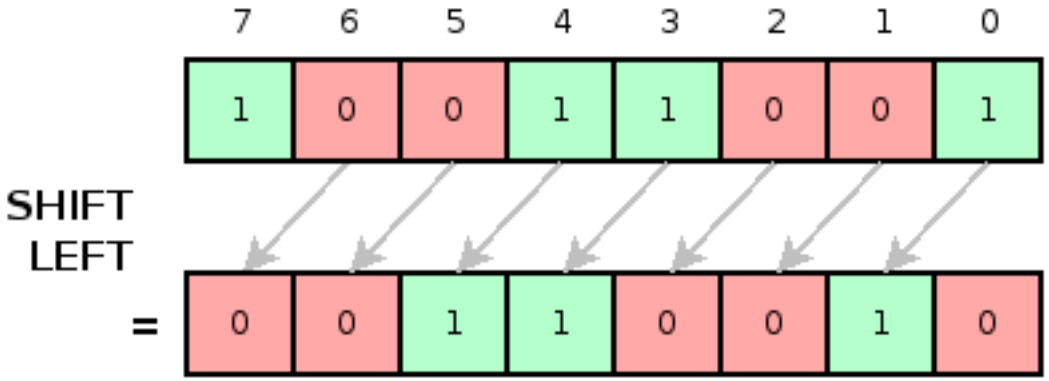
Operations with Bits - Intro

- Binary operations
  - Bit shift:
  - Let's say DDRB has an initial value of 0b10011001

DDRB = 0b10011001;

  - The operation DDRB = (DDRB<<1); shift bits in the binary value DDRB by 1 position

DDRB = 0b00110010;



Operations with Bits - Intro

- Binary operations
  - NOT operation: Also known as a one's complement. The NOT operation will simply negate each bit. Every 1 becomes 0 and every 0 becomes 1.

```
DDRB = ~(0b10011001<<1);
```

```
(0b10011001<<1) =    0b00110010
~(0b10011001<<1) = ~(0b00110010) =
                          = 0b11001101
```

Operations with Bits - Intro

- Binary operations
  - NOT operation:

DDRB = ~(1<<3);

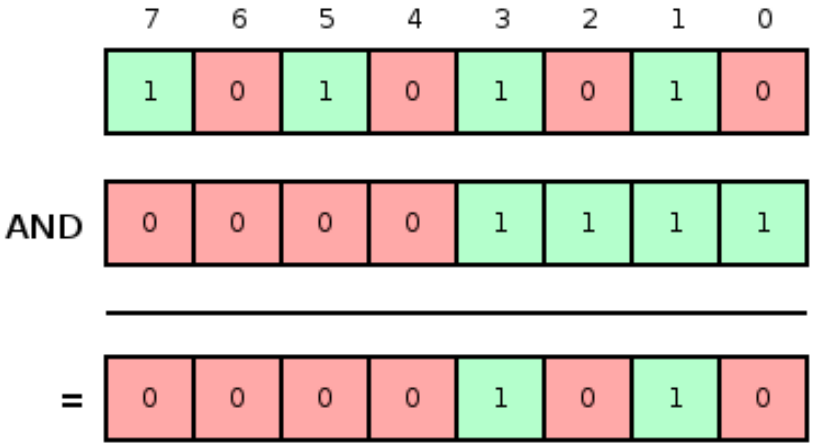
(0b00000001<<3) = 0b00001000

~(0b00000001<<3) = ~(0b00001000) =

= 0b11110111

Operations with Bits - Intro

- Binary operations
  - AND operation: this operation results in bits being set only if the same bits are set in both of the operands. In other words: bit n will be set in the result if bit n is set in the first operand and the second operand.

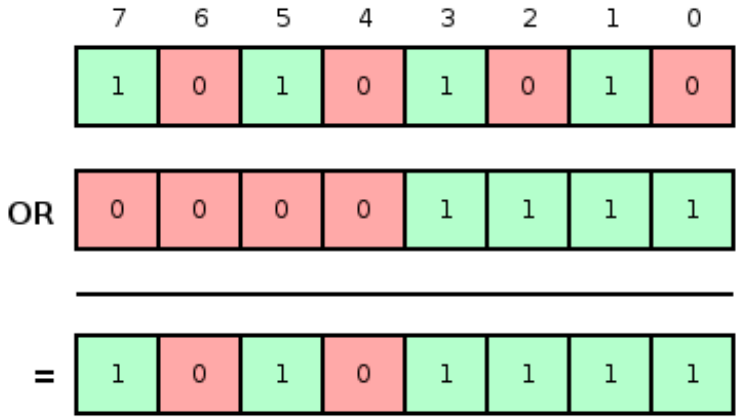


DDRB = 0b10101010 & 0b00001111

DDRB = 0b00001010

Operations with Bits - Intro

- Binary operations
  - OR operation: This results in bits being set if the same bits are set in either of the operands. In other words: bit n will be set in the result if bit n is set in the first operand or the second operand



DDRB = 0b10101010 || 0b00001111

DDRB = 0b10101111