Embedded Systems

Constraints not found in desktop applications:

- low cost, low power, low amount of memory, no user interface

Characteristics of Embedded Control Systems:

- Interface with external environment – sensors and actuators
- Real time critical – performance and safety – embedded software must execute in synchrony with physical system
- Hybrid behavior – continuous dynamics – discrete states
- Distributed control – networks of embedded microprocessors

Modularity

Modular hardware and software parts

Advantages:

- Components can easily be replaced when broken
- Adding new functions (upgrading) is very simple
- A small step towards standardization

Properties

- **Performance/speed of a processor**
  - Number of clock cycles per second
  - Determines the number of instructions performed each second
  - Single core vs dual core with half the frequency

- **Compatibility**
  - Electronic compatibility (power, noise, communication, etc.)
  - Environmental compatibility

- **Fixed-point/ floating-point device**
  - Categories of processor computation arithmetic.
  - Fixed-point tend to be faster, more power-conscious, and cost-sensitive
  - Floating-point tend to be higher precision and higher dynamic range

Properties

- **Volatile Memory**
  - Cache
    - Small, fast, expensive memory
    - Located close to a processor core
    - Stores copies of the data from frequently used main memory locations.

  - Random Access Memory (RAM)
    - Stores frequently used program instructions to increase the general speed of a system, cheaper and much larger than the cache
    - Dynamic RAM consist of a transistor and a capacitor and must be periodically refreshed
    - Static RAM retains data as long as power is supplied.

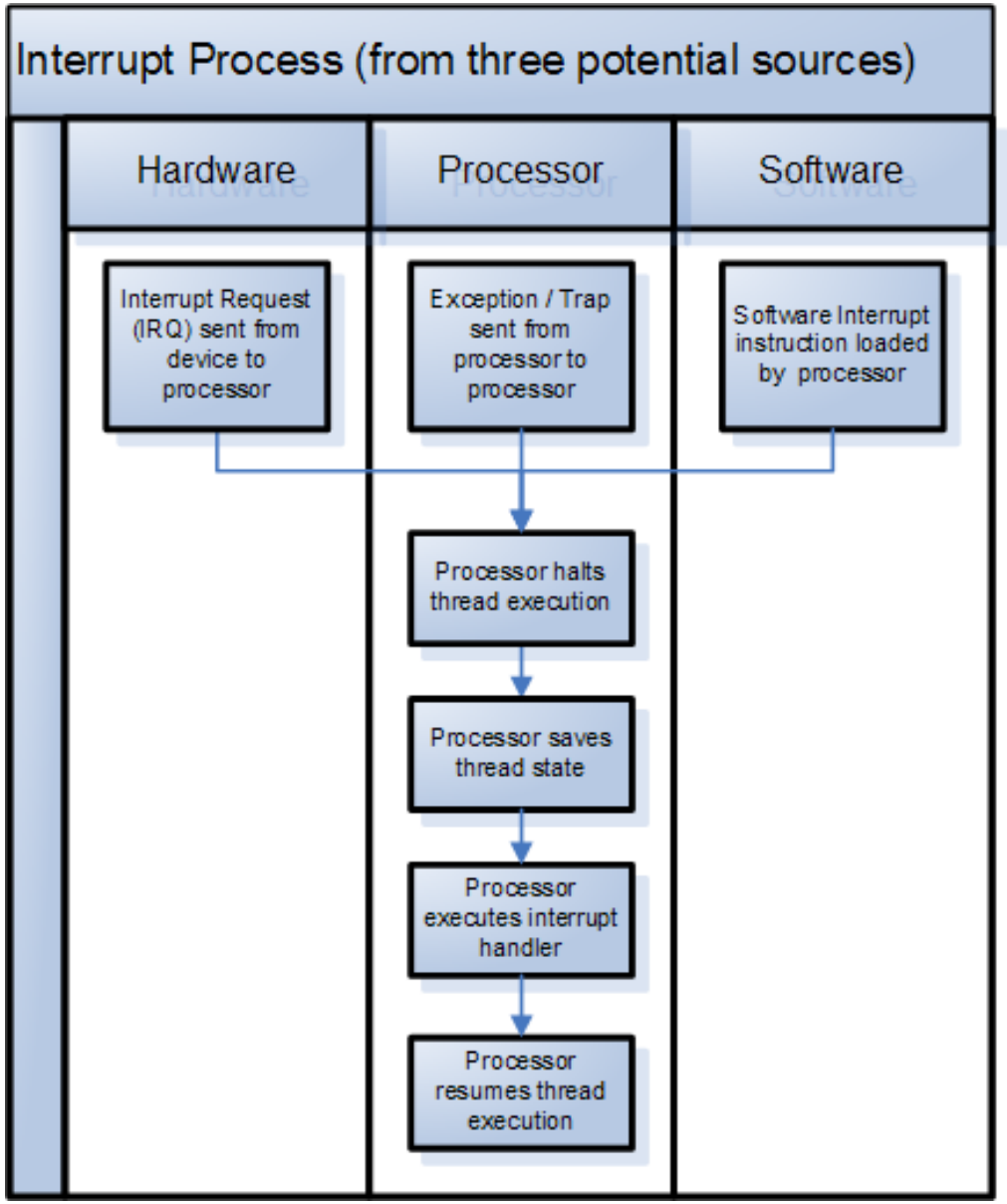
Properties

- **Non-Volatile Memory**
  - Read Only Memory (ROM)
    - Non-volatile memory, typically used to store firmware, etc.
    - Small number of write cycles possible
  - (Electrically-) Erasable Programmable Read-Only Memory ((E-)EPROM)
    - Based on AND-gates
    - Can be (completely) erased and re-programmed
    - Slow speed and may require special equipment to achieve
    - Typically only possible a limited number of times
  - Flash Memory
    - Based on NAND and NOR-gates
    - Allows you to write or read blocks of the memory (faster)

Properties

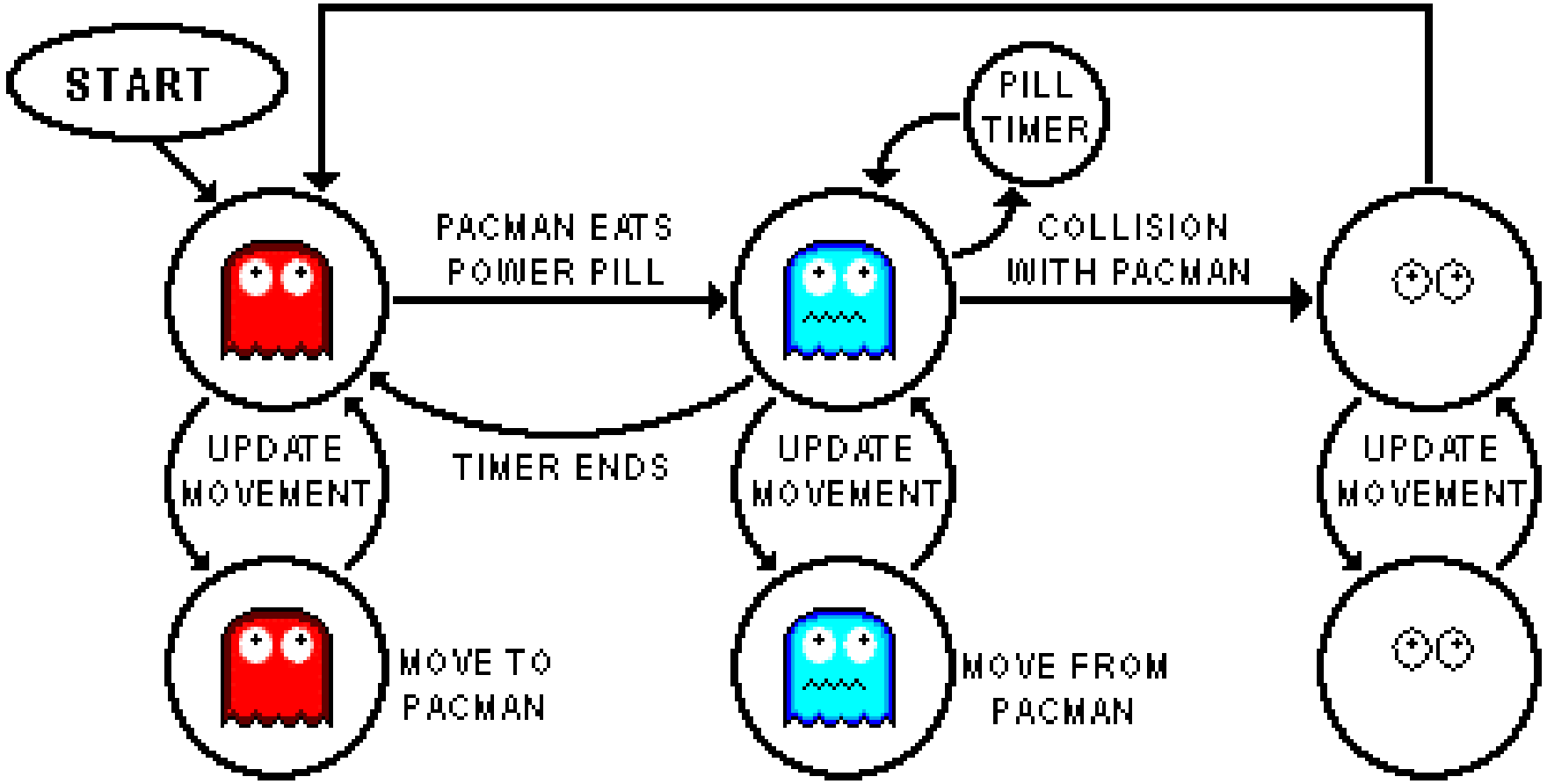
- **Real-Time Aspects**
  - *Interrupts and interrupt vectors*

- **Examples for code execution on a Microcontroller**
  - Loop, execution time depends on instructions in the loop
  - Round-Robin
  - Event/interrupt driven
  - Combination



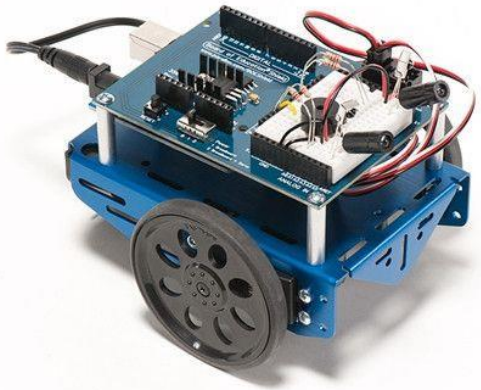
State Machines

- Made of states and transitions



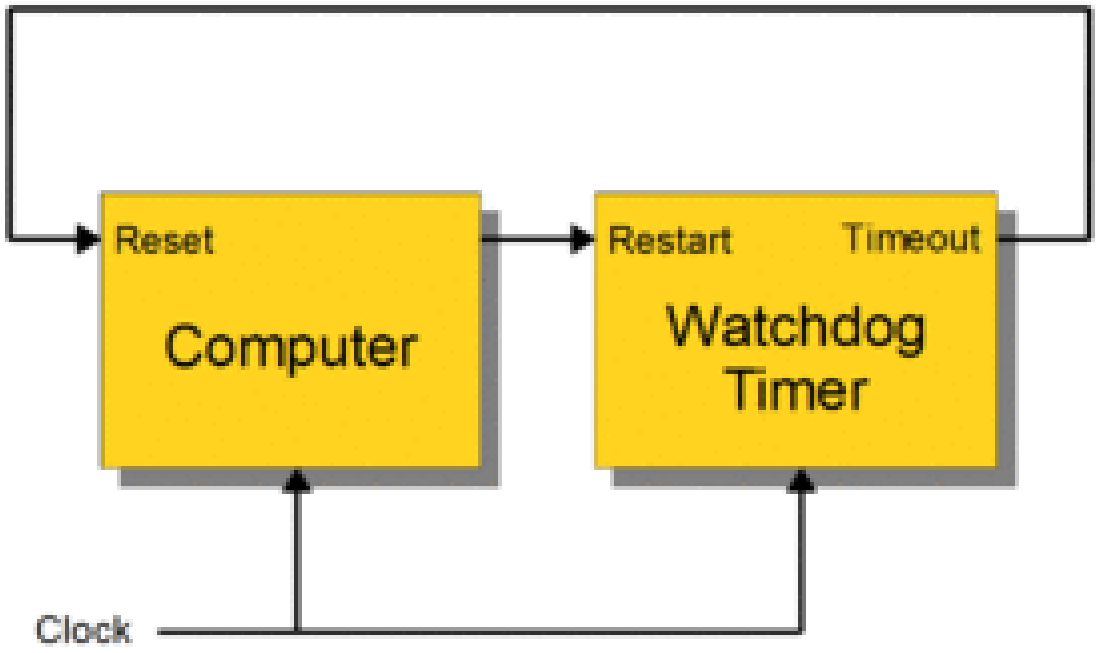
State Machines

- Robots need to process information and react to the environment
- Model robot as a state machine
  - Useful for organizing code/debugging



Interrupts

- Polling vs. Interrupts
- Some uses:
  - Timers
    - Separate loop into intervals
    - Setup 5 separate registers
  - Input Pin
  - Watchdog



Memory

- Memory is a limited resource
- Some tips:
  - Avoid recursive functions
  - Pay attention to the size of types
    - char = 1 byte, int = 2 or 4 bytes, short = 2 bytes, long = 4 bytes, …
    - Signed vs. unsigned
    - Merge bits into a single variable

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

How to write GOOD code?

Why?

- Problems become easier to solve, waste less time on maintenance, communicate ideas more clearly

What to do?

- Use descriptive function and variable names
- Give each class/functions one purpose (modularity)
- Delete unnecessary code
- Readability (don't compress 10 lines into 1 if it makes it impossible to read)
- Consistent Style
- Write good and *concise* comments
    - Refactor!

Go Build Robots!