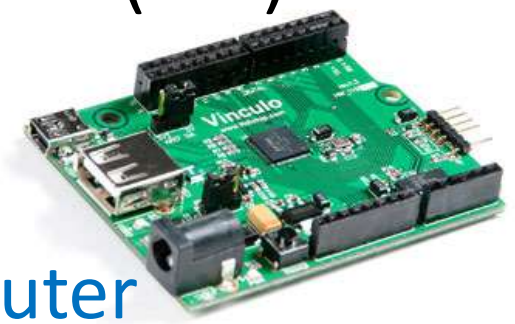
02 - Introduction

- Computer History (mechanical, analog, digital)
- von Neumann architecture
- General view of main components of a computer system
- General overview of the processor architecture

What Is an Embedded System (ES)?



- An embedded system is an applied computer system, as distinguished from other types such as
  - personal computers (PCs) or
  - supercomputers.
- The definition is fluid and difficult to pin down, as
  - it constantly evolves with advances in technology
  - and dramatic decreases in the cost of implementing various hardware and software components.
- In recent years, the field has outgrown many of its traditional descriptions.
  - A few of the more common descriptions of an embedded system are introduced on the next slides.

Limited Functionality

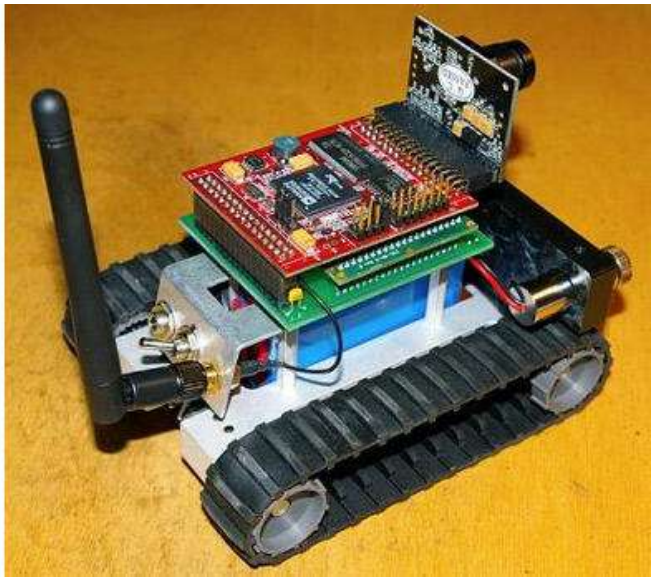
- Embedded systems are more limited in hardware and/or software functionality than a personal computer (PC).
  - This holds true for a significant subset of the embedded systems family of computer systems.
  - In terms of hardware limitations, this can mean limitations
    - in processing performance,
    - power consumption,
    - memory,
    - hardware functionality, and so forth.
  - In software, this typically means limitations relative to a PC
    - fewer applications,
    - scaled-down applications,
    - no operating system (OS) or a limited OS, or
    - less abstraction-level code.

Limited Functionality

- However, this definition is only partially true today as
  - boards and software typically found in PCs of past and present have been repackaged into more complex embedded system designs.

Dedicated Function

- An ES is designed to perform a dedicated function.
- Most devices are primarily designed for one specific function.
  - devices such as tablets / smartphones, which are embedded systems designed to be able to do a variety of primary functions.
  - the latest digital TVs include interactive applications that perform a wide variety of general functions unrelated to the “TV” function but just as important, such as
    - e-mail,
    - web browsing, and
    - games.



Higher Quality And Reliability Requirements

- higher quality and reliability requirements than other types of computer systems.
  - Some have a very high threshold of quality and reliability requirements.
  - For example, if a car's engine controller crashes while driving on a busy freeway or a critical medical device malfunctions during surgery, very serious problems result.
  - However, there are also embedded devices, such as TVs, games, and cell phones, in which a malfunction is an inconvenience but not usually a life-threatening situation.

Embedded Systems And Their Markets

Market	Embedded Device
Automotive	Ignition System; Engine Control Brake System (i.e., Antilock Braking System)
Consumer Electronics	Digital and Analog Televisions; Set-Top Boxes (DVDs, VCRs, Cable Boxes) Personal Data Assistants (PDAs) Kitchen Appliances (Refrigerators, Toasters, Microwave Ovens) Toys/Games Telephones/Cell Phones/Pagers: Global Positioning Systems (GPS) Cameras
Industrial	Control Robotics and Control Systems (Manufacturing)
Medical	Infusion Pumps Dialysis Machines Prosthetic Devices Cardiac Monitors
Networking	Routers; Hubs Gateways
Office Automation	Fax Machine; Printers; Photocopier Monitors Scanners

Embedded Systems Design and Development Lifecycle Model

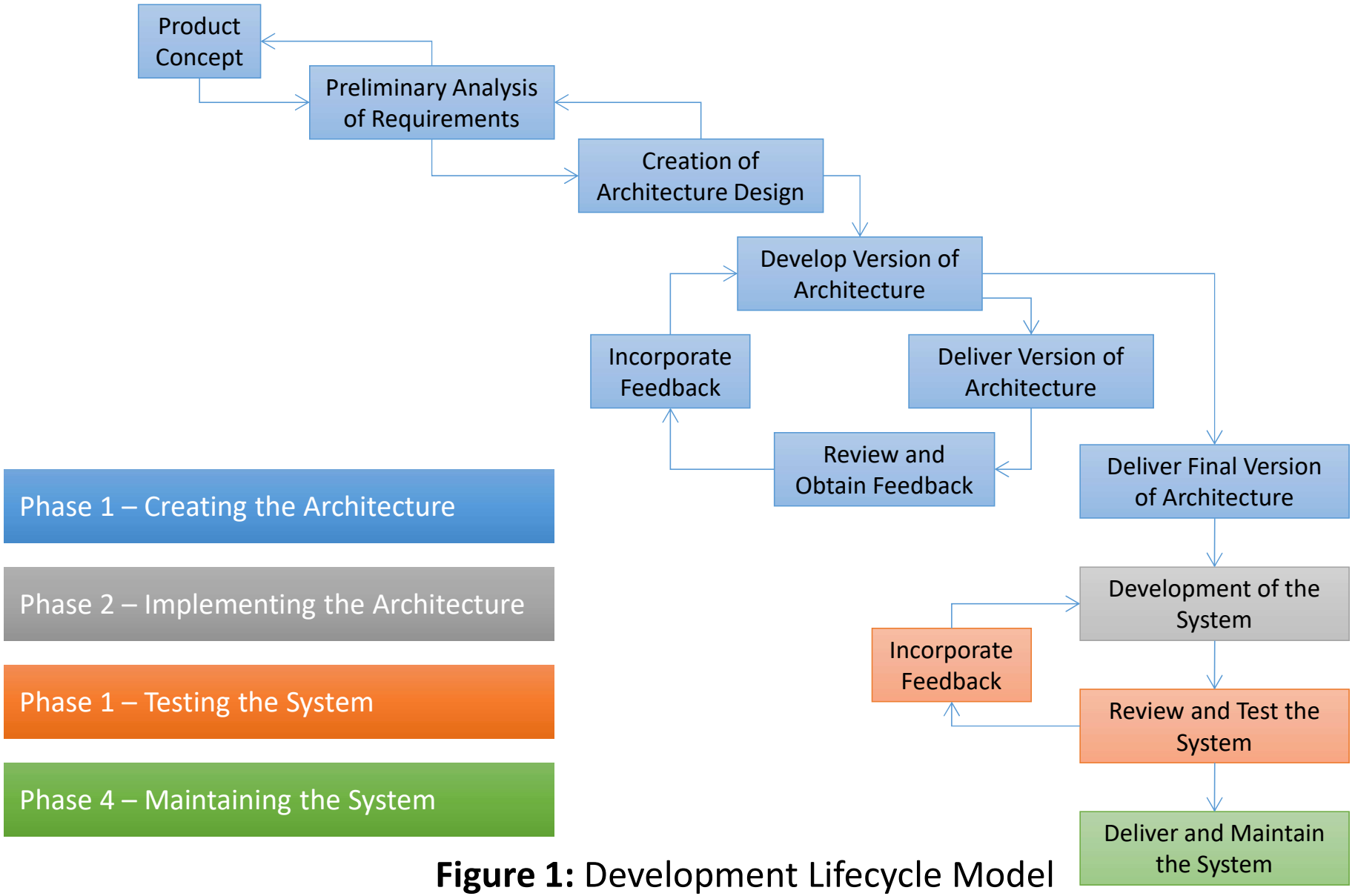


Figure 1: Development Lifecycle Model

The Embedded Systems Model

- All embedded systems have at least one layer (hardware) or all layers (hardware, system software and application software) into which all components fall.
  - The **hardware layer** contains all the major physical components located on an embedded board,
  - the **system** and **application software layers** contain all of the software located on and being processed by the embedded system.

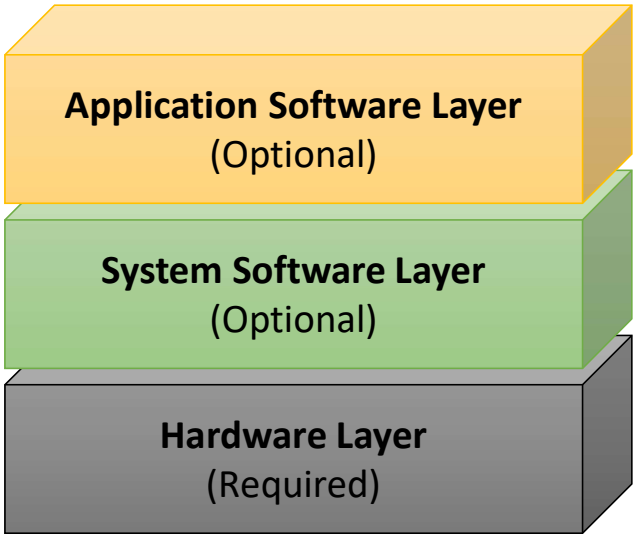


Figure 2: The Embedded System Model

Know Your Standards

- Standards dictate
  - how these components should be designed, and
  - what additional components are required in the system to allow for their successful integration and function.
- Standards can define functionality that is specific to each of the layers of the embedded systems model,
- Standards can be classified as
  - market-specific standards,
  - general-purpose standards
  - standards that are applicable to both categories.

Know Your Standards

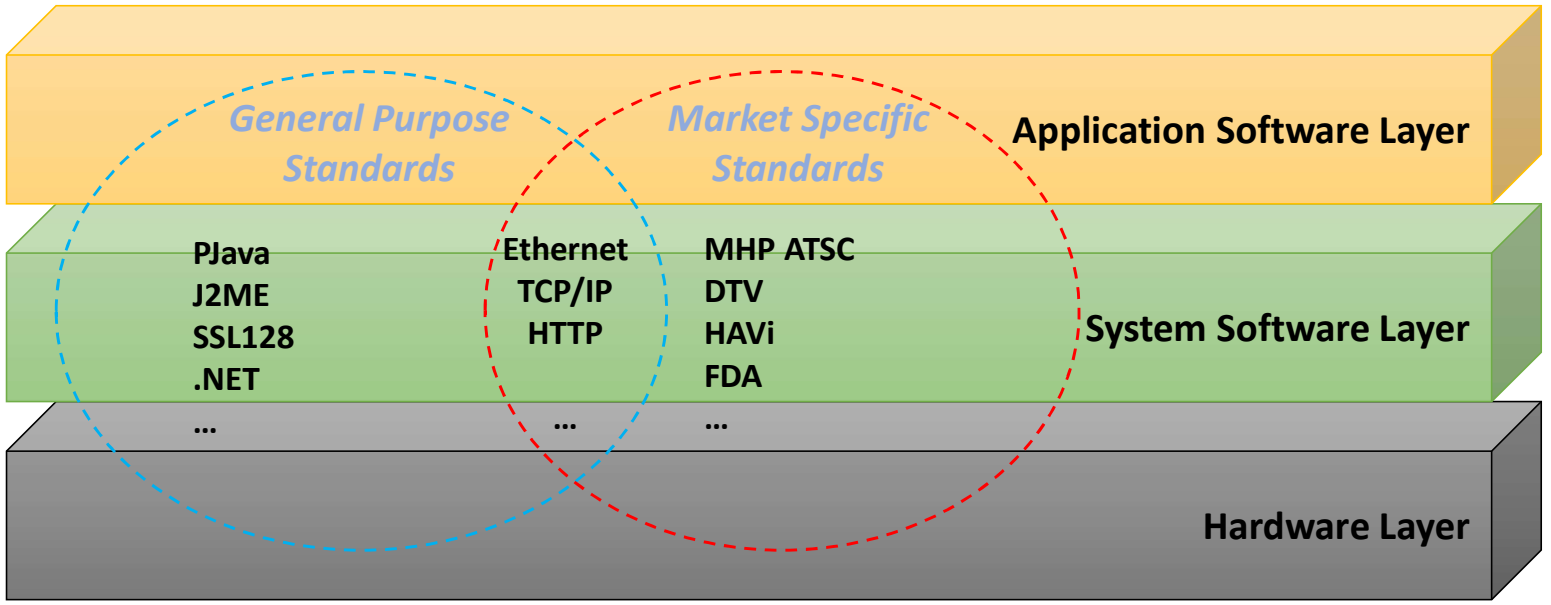


Figure 3: Standards diagram

Introduction

- Basic terms
- Design example
- Architectures
- Main Features
- Peripherals

Introduction – basic terms

Microprocessor	a circuit that has millions of transistors, which functions is based on a program read from memory;
Microcontroller	a circuit that contains on the same chip a central processing unit (equivalent to a microprocessor), ROM (Flash), RAM memory, timers and input/output ports;
Bus	a collection of wires of the same type (examples: data bus, address bus, control bus);
RISC processors	**R**educed **I**nstructions **S**et (**C**omputer), processed very fast;
CISC processors	**C**omplex **I**nstructions **S**et (**C**omputer) , but usually slower.

uP / uC

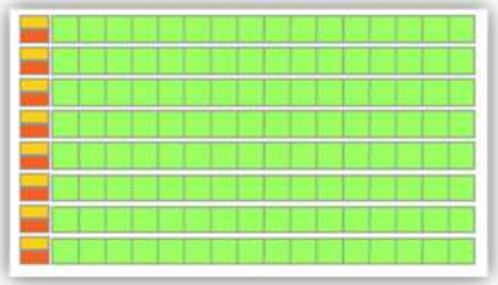
- Embedded boards are designed around the master processor.
- The complexity of the master processor usually determines whether it is classified as
  - a **microprocessor** or
  - a **microcontroller.**
- Traditionally,
  - microprocessors contain a minimal set of integrated memory and I/O components,
  - microcontrollers have most of the system memory and I/O components integrated on the chip.
- However, these definitions may not strictly apply to recent processor designs.
  - microprocessors are increasingly becoming more integrated.

CPU vs GPU

CPU

* Low compute density
* Complex control logic
* Large caches (L1$/L2$, etc.)
* Optimized for serial operations
  - Fewer execution units (ALUs)
  - Higher clock speeds
* Shallow pipelines (<30 stages)
* Low Latency Tolerance
* Newer CPUs have more parallelism

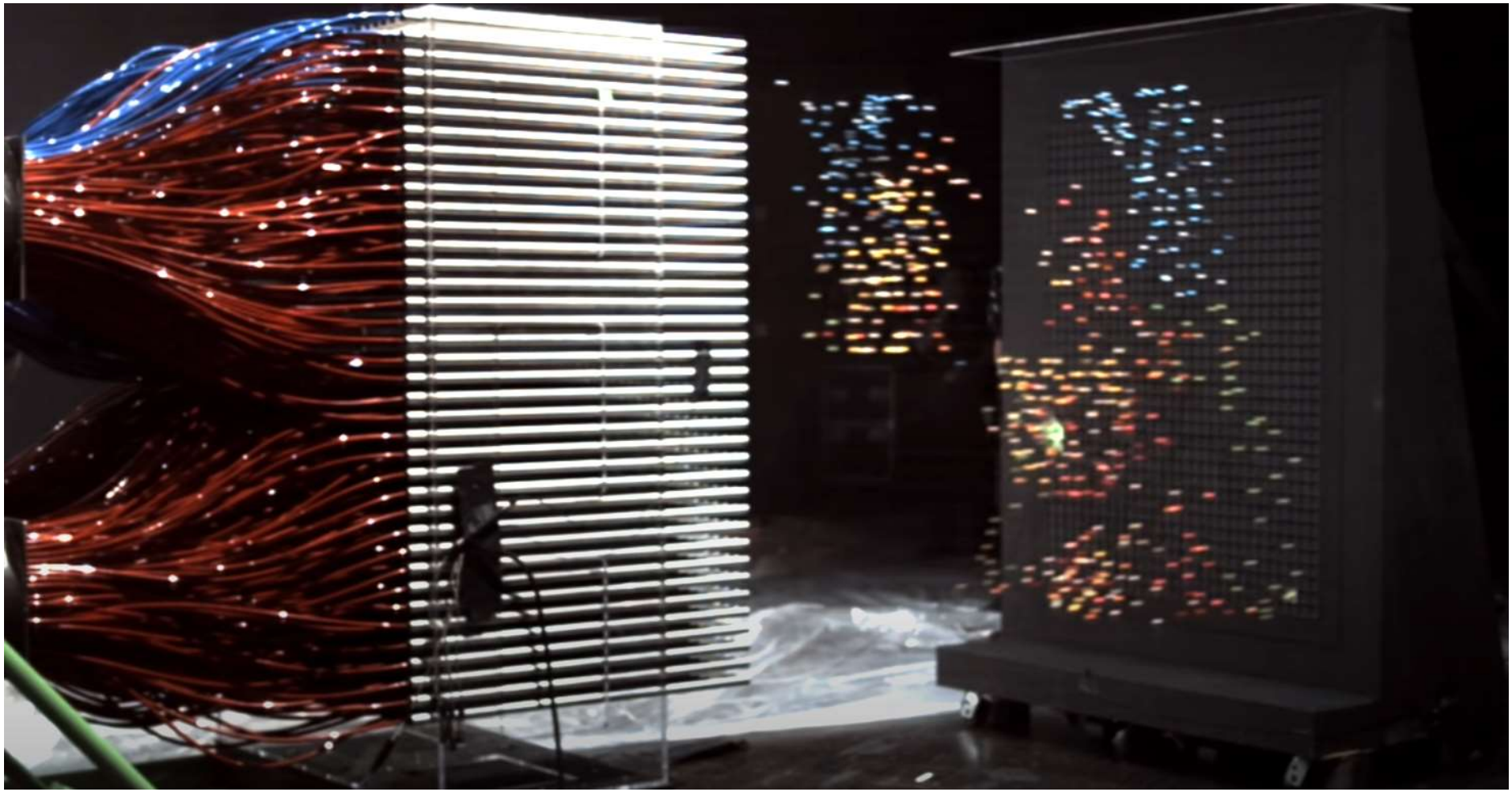
GPU



* High compute density
* High Computations per Memory Access
* Built for parallel operations
  - Many parallel execution units (ALUs)
  - Graphics is the best known case of parallelism
* Deep pipelines (hundreds of stages)
* High Throughput
* High Latency Tolerance
* Newer GPUs:
  - Better flow control logic (becoming more CPU-like)
  - Scatter/Gather Memory Access
  - Don't have one-way pipelines anymore

Myth Busters

- https://youtu.be/-P28LKWTzrI

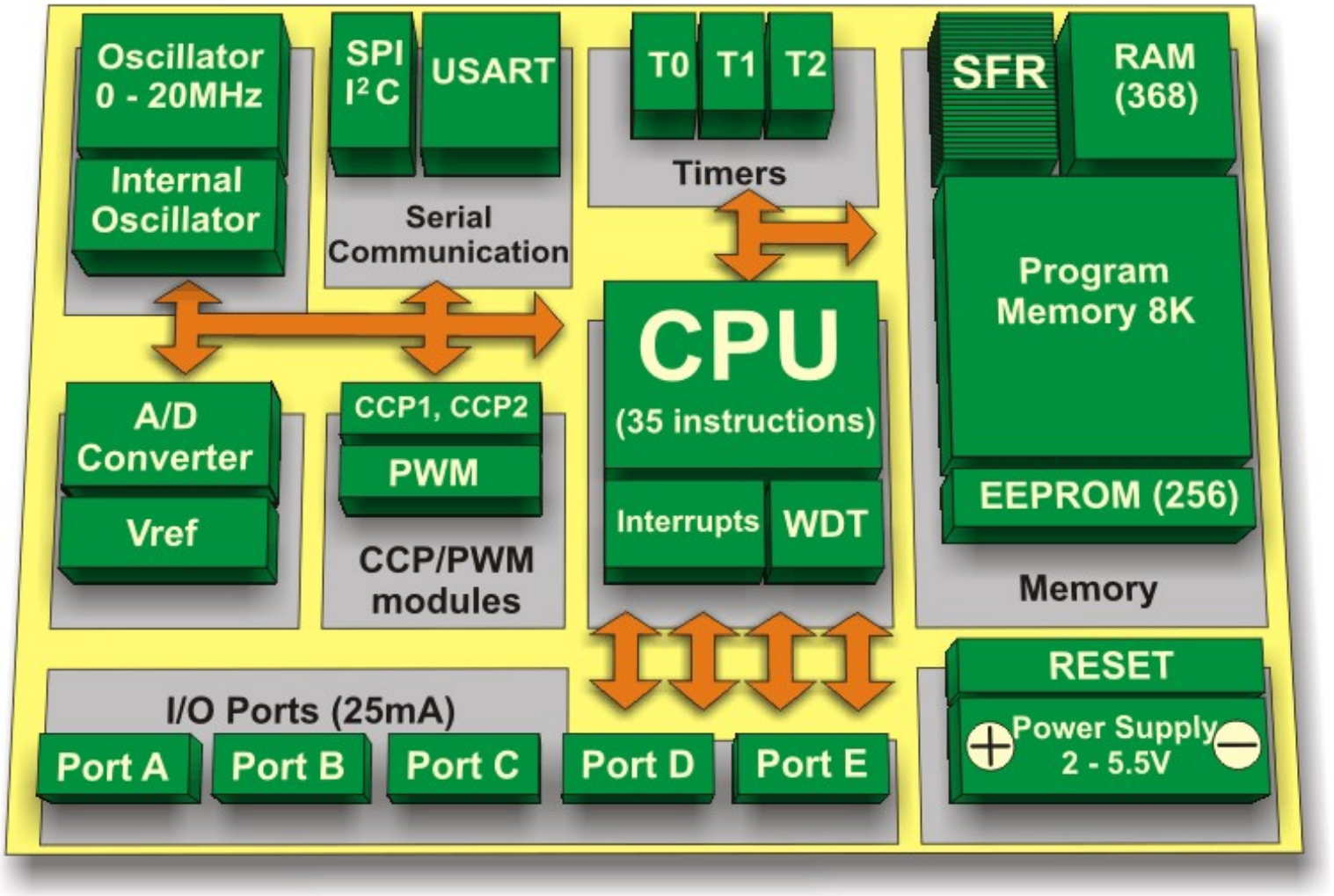


Why Use an Integrated Processor?

- While some components, like I/O, may show a decrease in performance when integrated into a master processor as opposed to remaining a dedicated slave chip,
  - many others show an **increase in performance** because they no longer have to deal with the latencies involved with transmitting data over buses between processors.
- An integrated processor also **simplifies the entire board design** since there are fewer board components, resulting in a board that is simpler to debug (fewer points of failure at the board level).
- The **power requirements** of components integrated into a chip are typically a lot less than those same components implemented at the board level.
  - With fewer components and lower power requirements, an integrated processor may result in a **smaller and cheaper board**.
- On the flip side, there is **less flexibility** in adding, changing, or removing functionality since components integrated into a processor cannot be changed as easily as if they had been implemented at the board level.

What is a microcontroller?

- A computer on a single chip
- Contains a CPU and some peripherals



Design Example

Design Example: Wiper Windshield

Requirements:

- Rain triggered
- Speed control (3 ranges)
- Intermittent mode (potentiometer)
- Car speed correlation
- Intelligent control (delays, end of turn, etc)

Wiper Windshield: Logic Diagram

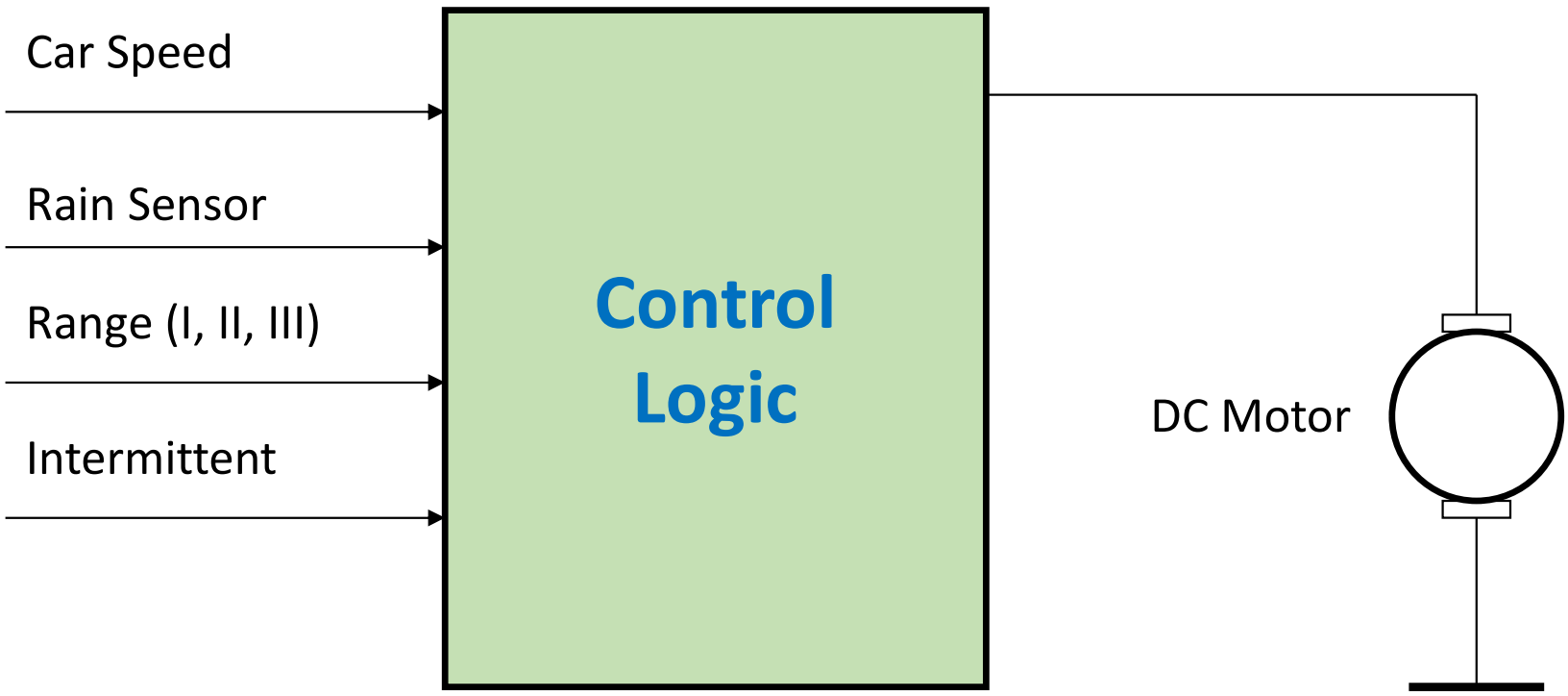


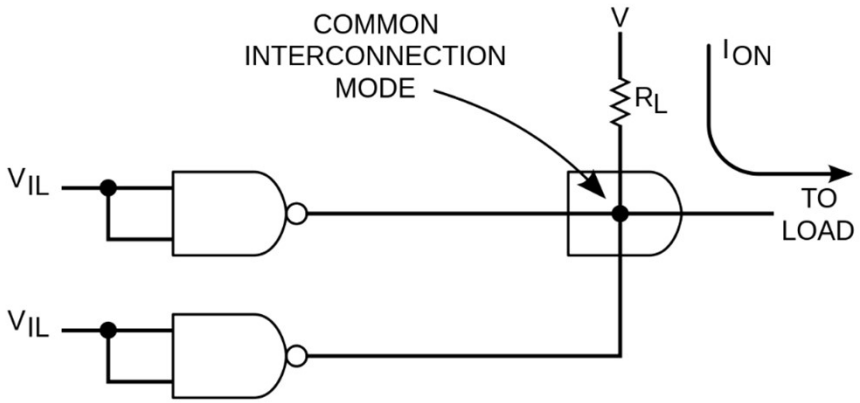
Figure 4: Design Example

Choices in Digital Design Implementation

Solution type	Advantages	Drawbacks
Wired logic	Very high speed Cheap components	Only for simple circuits Difficult to expand High overall cost
Programmable logic	High speed Able to handle complex digital signals	Limited number processing Moderate cost
Microprocessor	Powerful Wide choice of models Configurable in wide limits Allows almost all popular programming languages	Many components even for simple systems Moderate cost
Microcontroller	Simple electronic circuits are possible with few components Allows the most popular programming languages such as BASIC or C. Even MATLAB or LABVIEW solutions possible	Standard configurations rarely exactly fit the application's needs implying the use of over-sized models Special configurations available, but only for large quantities.

Wiper Example – Wired Logic

- Cheap components
- Speed control: potentiometer + logic, One shot: glue logic
- Intermittent action: multivibrator
- Speed correlation: serious logic
- Delay+ one more time: timers



Serious drawbacks

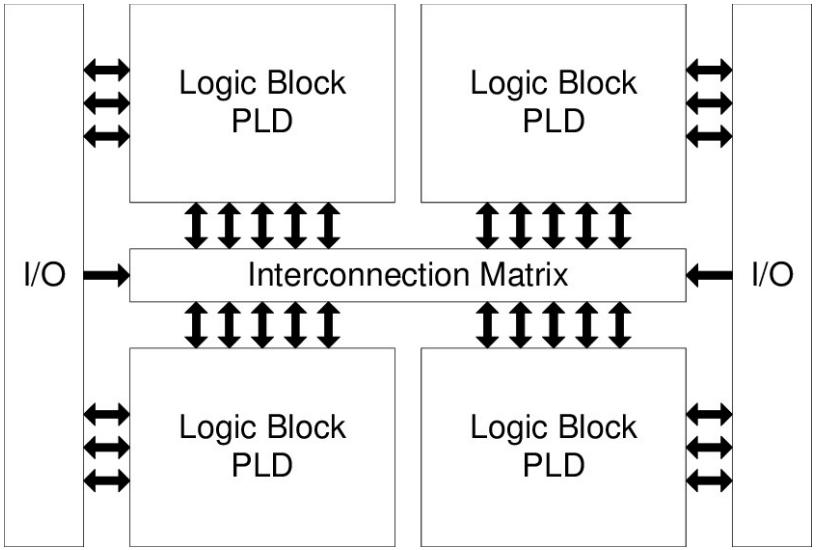
- Size (A4); high overall cost
- Lack of ease for modifications / upgrades
- Large analog blocks and components, unstable in time
- Large number of components
- No future extensions possible without redesigning
- No communication mean with other systems

Wiper Example - Programmable Logic

- Only 1 circuit solution is possible - size is greatly reduced
- No major analog components
- Future extensions and upgrades possible
- Low Power design, ultra high speed achievable
- Future hardware reconfiguration possible

Drawbacks

- Programming in VHDL, EDK or Labview (reduced number of specialists)
- Moderate cost (low efficiency at large quantities)



A Typical Microprocessor System

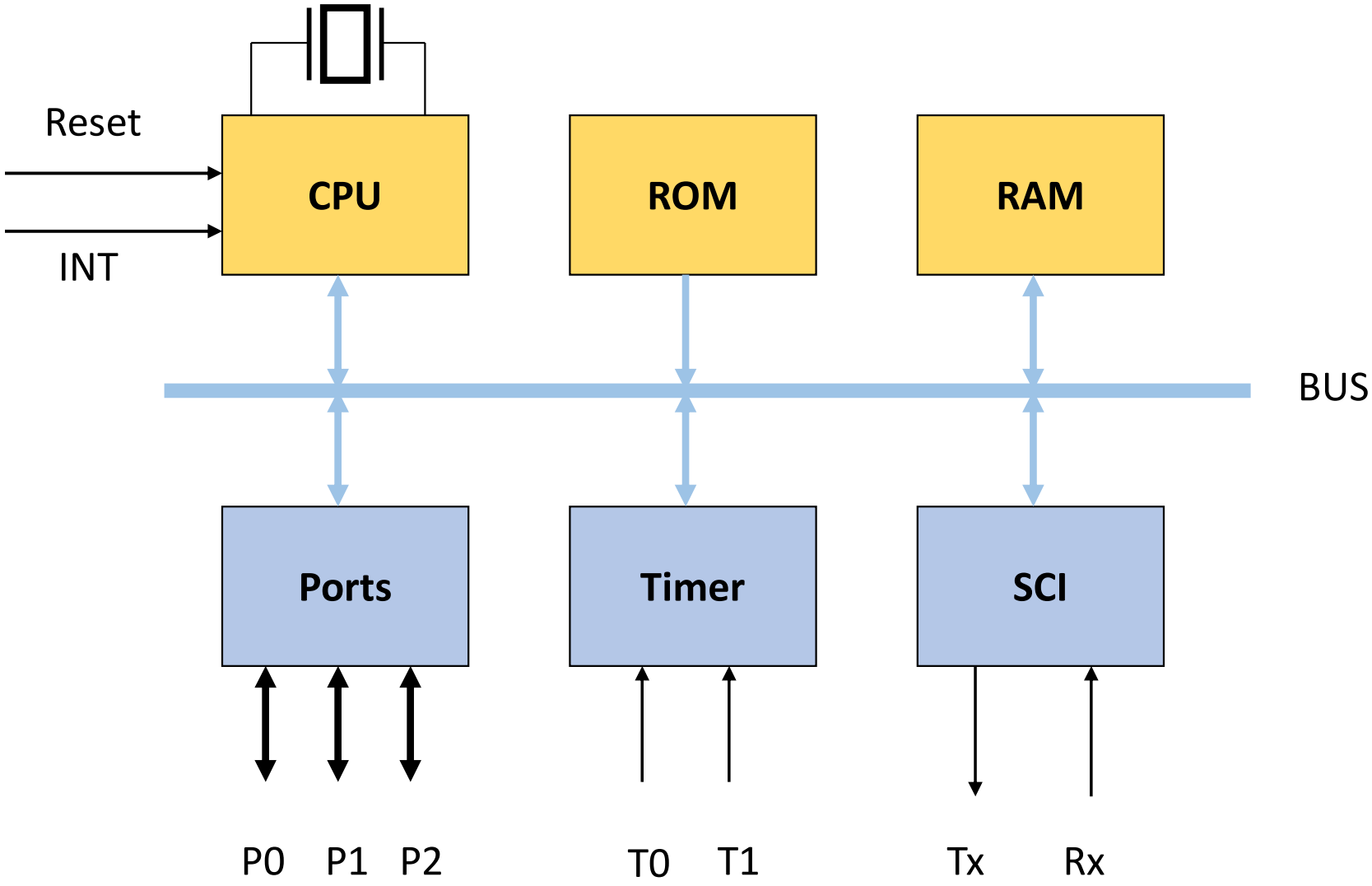


Figure 5: A Typical Microsystem

Wiper Example – Microprocessor

- Ease of implementation
- Future extensions possible without redesigning
- Communication with other systems using just a dedicated controller

Drawbacks

- Large number of components (ROM, RAM, ADC, DAC)
- Size (A5)
- Lower reliability due to the increased number of components
- Moderate cost, for both low and high volume

A Typical Microcontroller

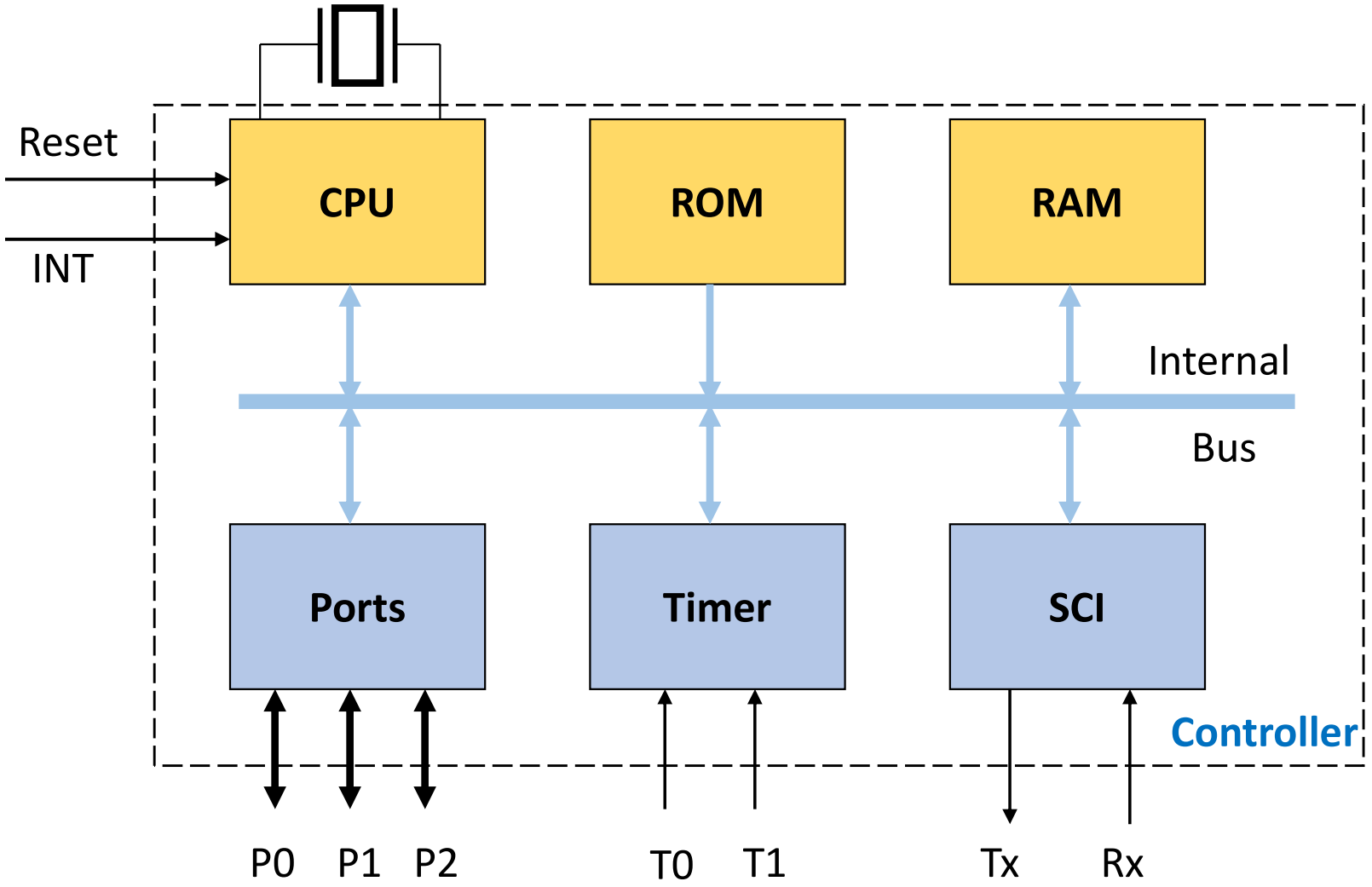


Figure 6: A Typical Microcontroller

Intel 8051 – The First Microcontroller

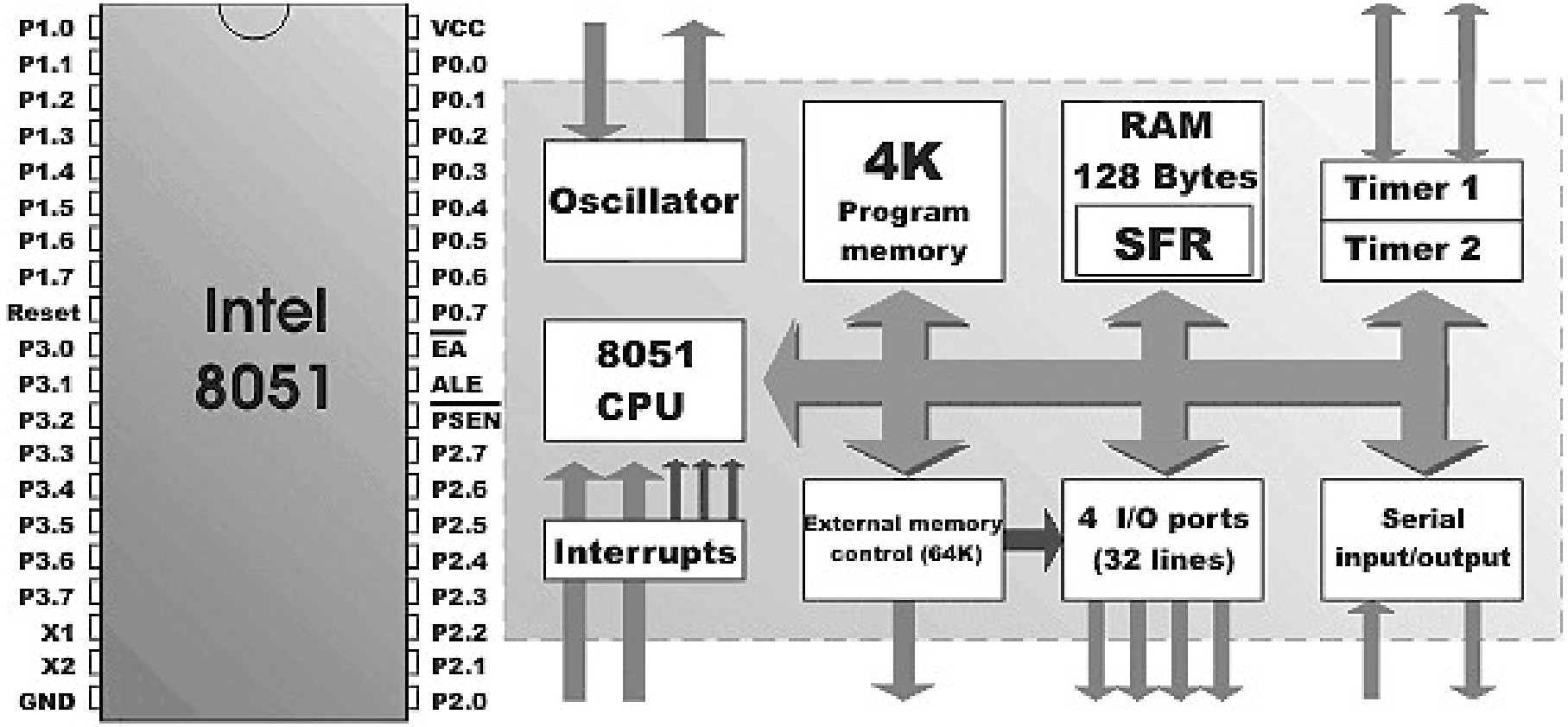


Figure 7: Intel 8051

Wiper Example – Microcontroller

- 1 chip overall solution possible
- Future software extensions possible without redesigning
- Extra functions may be combined on the same board (doors lock, alarm, etc).
- Communication with other systems possible with no extra circuits
- Low size (A6), low cost
- High reliability

Drawbacks

- A hardware and software design is still required ☺

The ASIC Connection

- The Microcontroller implementation is not always the best solution!
- For example, in a wrist watch, an ASIC is an unbeatable implementation!

Figure 8: Some watches

A Typical Block Diagram of a Microcontroller

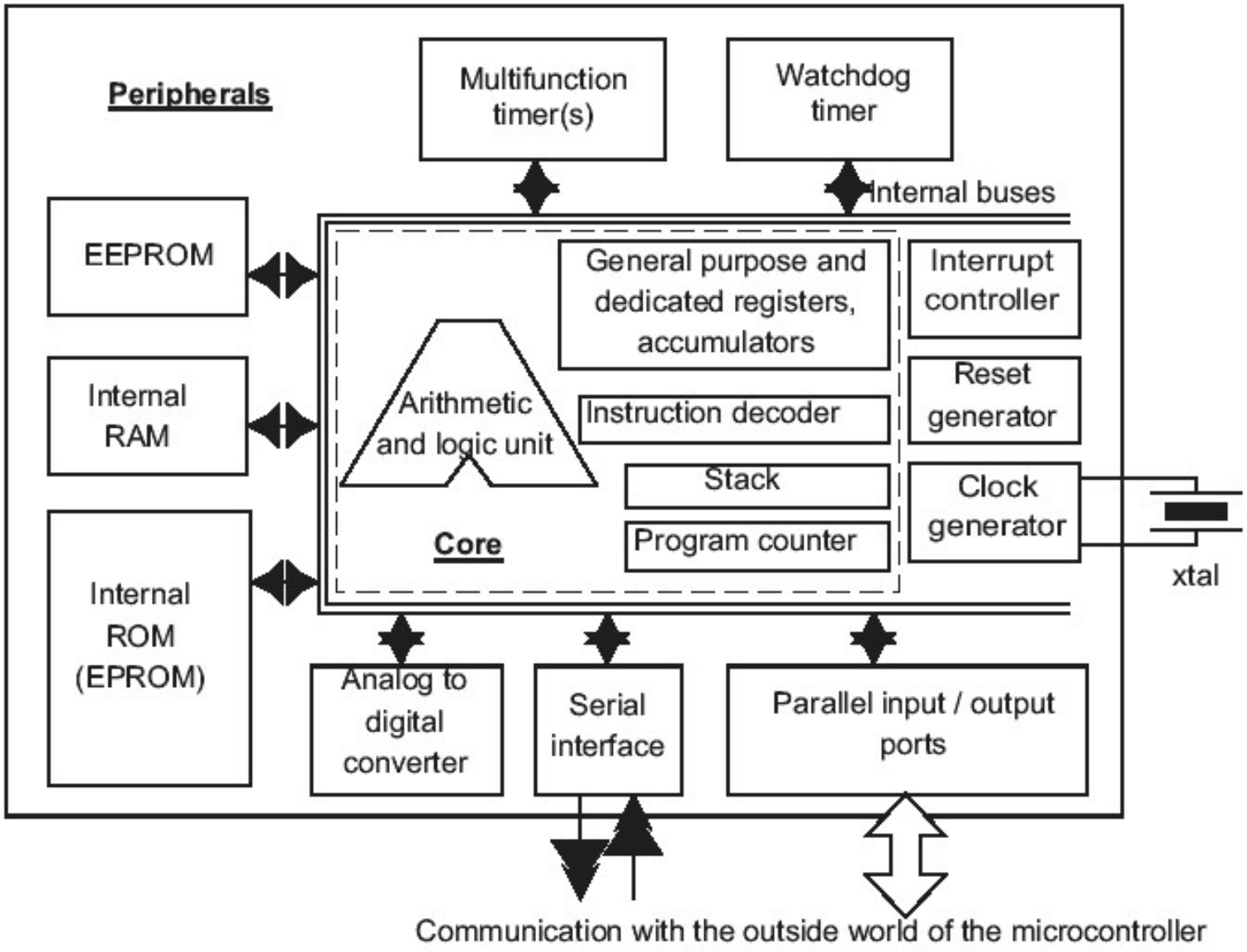
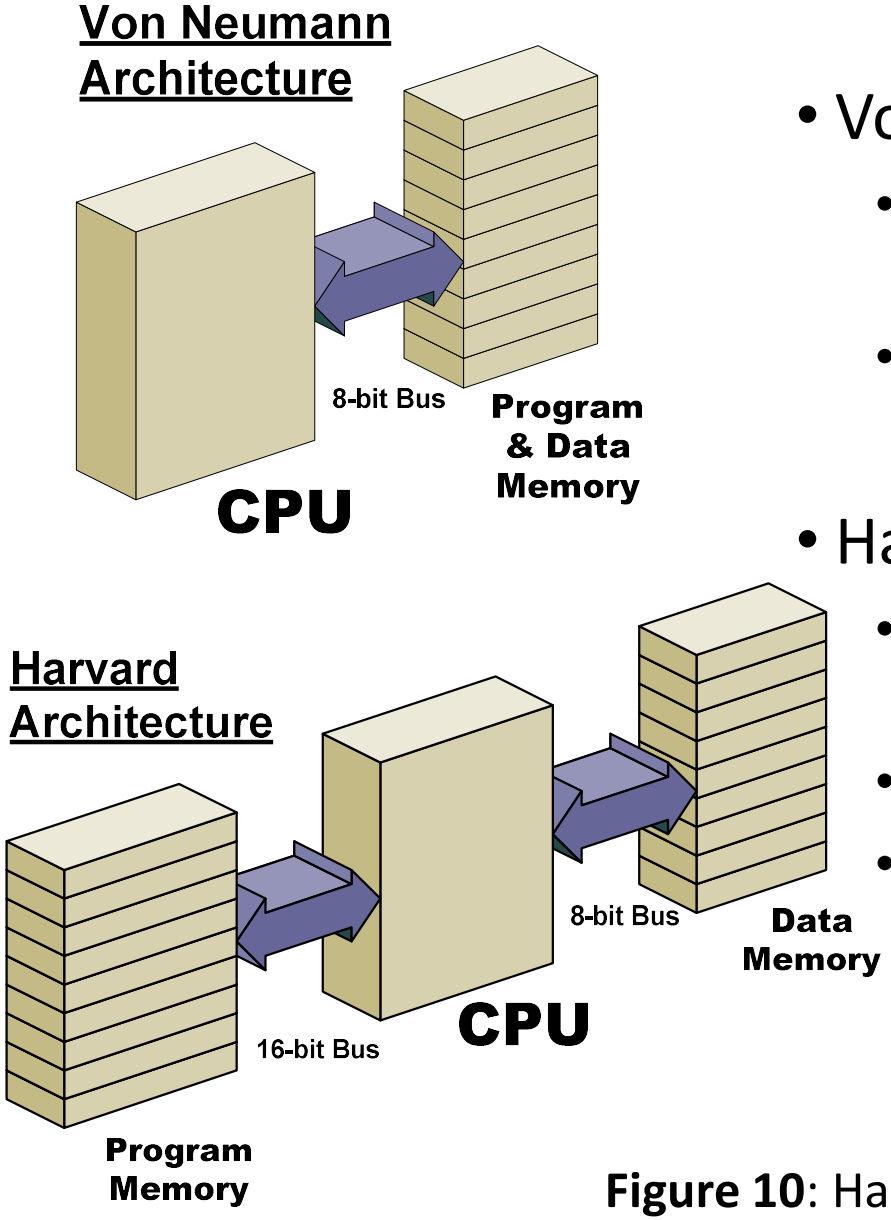


Figure 9: A Typical Microcontroller

Computers (CPU) Architecture



- Von Neumann Architecture:
  - Fetches instructions and data from a single memory space
  - Limits operating bandwidth
- Harvard Architecture:
  - Uses two separate memory spaces for program instructions and data
  - Improved operating bandwidth
  - Allows for different bus widths

Figure 10: Harvard vs. von Neumann

PIC 8-bit microcontroller versions

- the data bus is 8-bits wide.
- because the program bus is separate, it can be optimized to any width that fits within the design goals of the microcontroller.
- Microchip's PICmicro family uses three **program bus widths**:
  - **12-bit** - *base-line* (PIC16F5x/PIC12F5xx)
  - **14-bit** - *mid-range* (PIC16Fxxx/PIC12Fxxx)
  - **16-bit** - *high end* (PIC18) family.

External Buses

- All microprocessors have external busses
- An external bus is present on some microcontrollers
- Most small today microcontrollers do not have external busses

Program Storage

- ROMless
- OTP
- Masked
- Flash

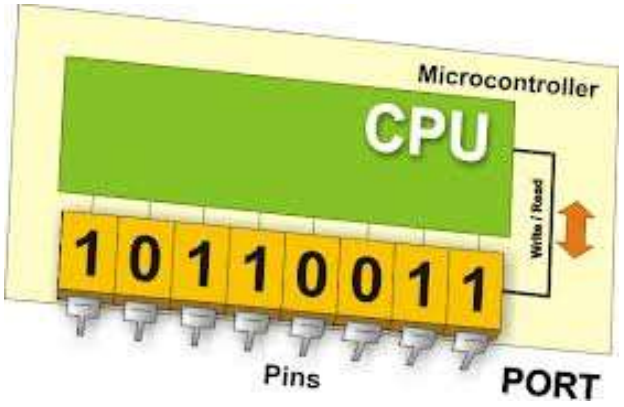
Micro Features

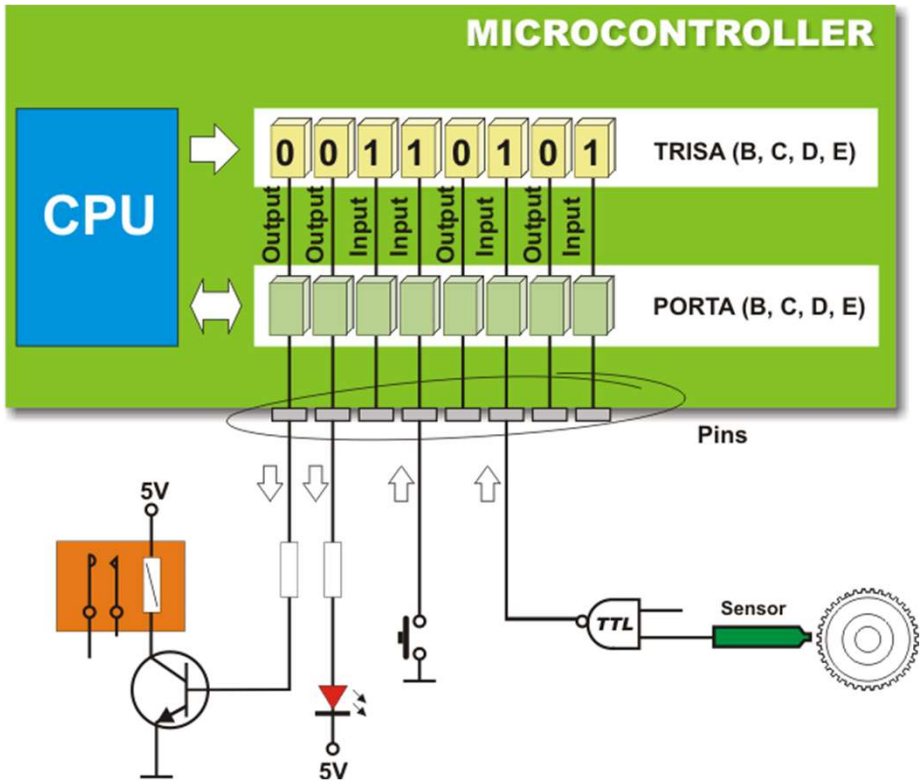
- Supply Voltage Range
- The Clock
- The Reset Input
- The Brown-Out Detector
- The Watchdog

I/O Ports

- Can be used as digital input or output.
- An I/O port is generally a group of up to 8 I/O lines which are addressed together as a byte.
- Usually any line can be configured as an input or output, with optional weak pull-up (effectively a large resistors connected to VCC) often available for inputs.
- It is common for I/O lines to have an alternate (special) function related to a peripheral; it's your choice whether that line is used by that peripheral or as standard digital I/O.
- Some I/O lines may have higher current capacity than others, for driving LEDs and other heavy loads; sinking 20mA is a typical figure for a high-current output.
  - Always observe the micro's total maximum current rating!

I/O Ports

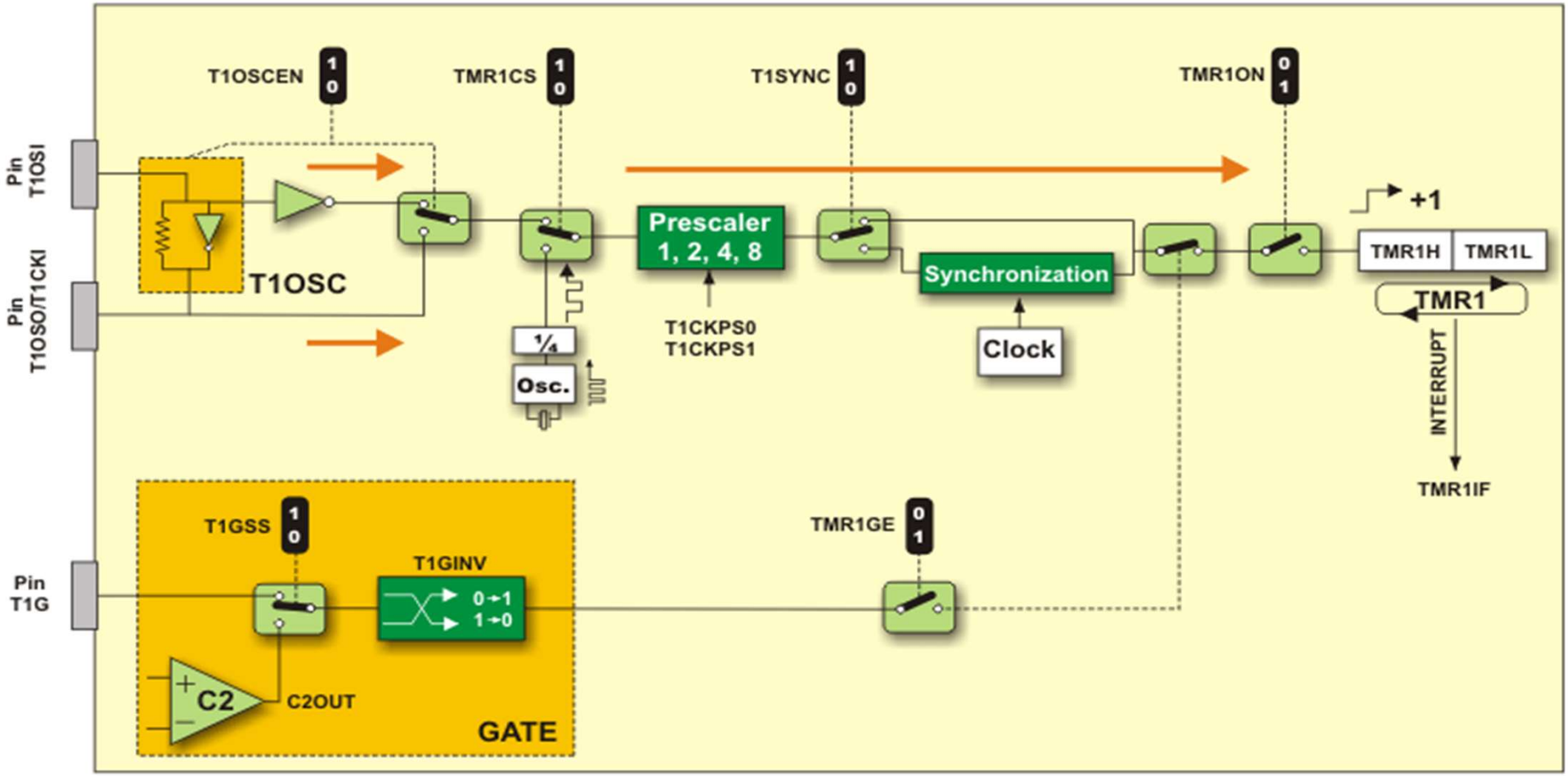

Microcontroller
CPU
1 0 1 1 0 0 1 1
Write / Read
Pins
PORT


MICROCONTROLLER
CPU
0 0 1 1 0 1 0 1
TRISA (B, C, D, E)
Output
Output
Input
Input
Output
Input
Output
Input
PORTA (B, C, D, E)
Pins
5V
5V
TTL
Sensor

Timers

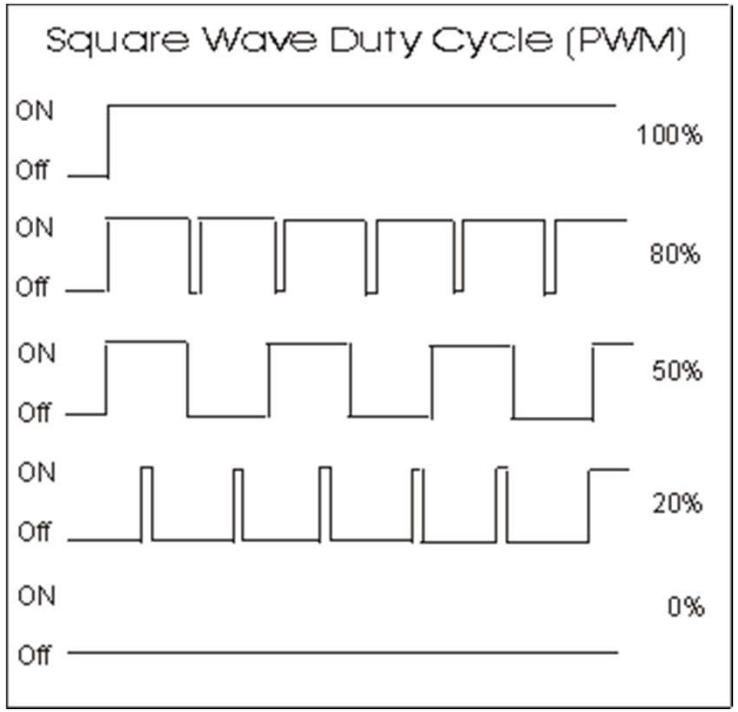
- A timer is a counter driven by some division of the clock sometimes used with a user-settable divider called **prescaler**; both up and down-counting timers exist.
- You can usually
  - start and stop a timer,
  - read and set its value and
  - generate a interrupt when it reaches zero (*overflows*),
    - which allows you to generate accurate time intervals.
- More powerful 16-bit timers offer
  - capture and/or compare, usually **both**.
  - auto-reload timers, which automatically reload themselves with a specified value and start again when they reach zero, optionally generating an interrupt.

PIC16/18 Timer1 Overview



Timers - PWM

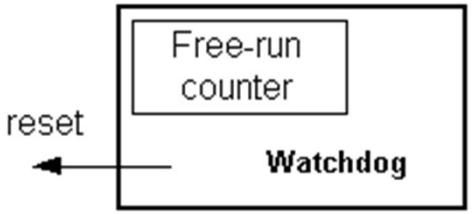
- Related to these are *PWM (*Pulse Width Modulation*)* facilities, which produce a pulse stream with a precisely defined duty cycle.



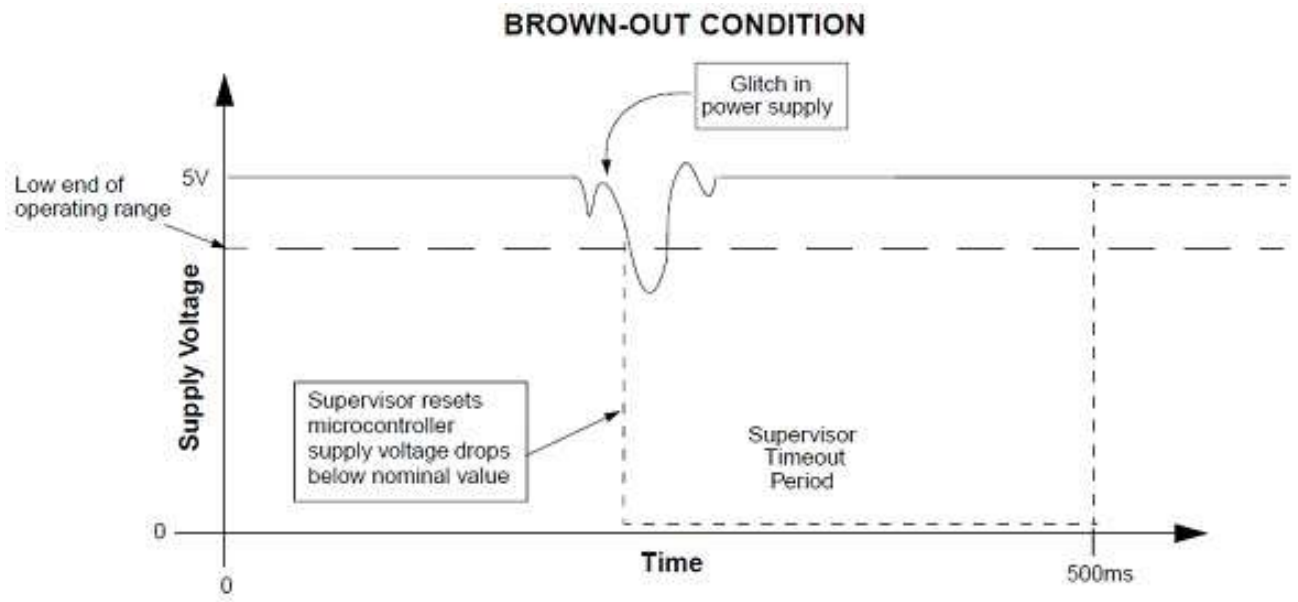
Watchdog

- This block is in fact another free-run counter where our program needs to write a zero (clears the counter) in every time it executes correctly.
- In case that program gets "stuck", zero will not be written in, and counter alone will reset the microcontroller upon achieving its maximum value.
- This will result in executing the program again, and correctly this time around. That is an important element of every program to be reliable without man's supervision

Figure 11: A Typical Watchdog



Brown-Out Detection

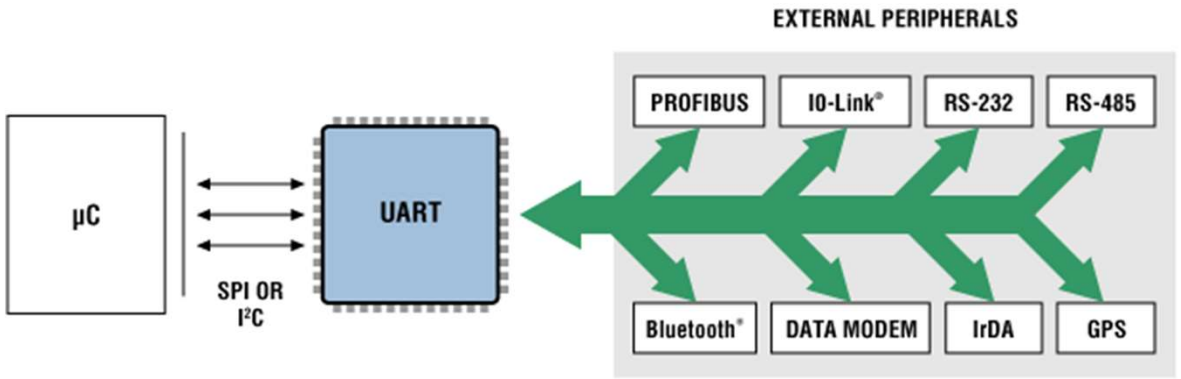


Analog Input and Output

- Since a micro is a very noisy environment for A/D conversion many micros allow you literally switch off other peripherals while conversions are taking place.
- Most can generate interrupts when conversions are complete.
- If conversion speed is not important, serial A/Ds like the TLC549 can do the job without consuming many I/O lines.
- A few micros also offer D/A converters for analog output.
- There are other ways of implementing analog input and output.

Serial Interfaces

- A UART (Universal Asynchronous Receiver/Transmitter) is a peripheral that implements an asynchronous serial interface, as used in standard RS-232 connections.
- If you have a need for speed, more modern micros can offer up to 1Mb/s
- Some micros offer an SPI (Serial Peripheral Interface) or I^2C (Integrated InterConnect) interface, synchronous.
  - Both standards allow connection to multiple peripheral chips, especially time-of-day clocks and EEPROMs, with the use of only 2 or 3 I/O lines, and can operate at very high speeds.



EEPROM

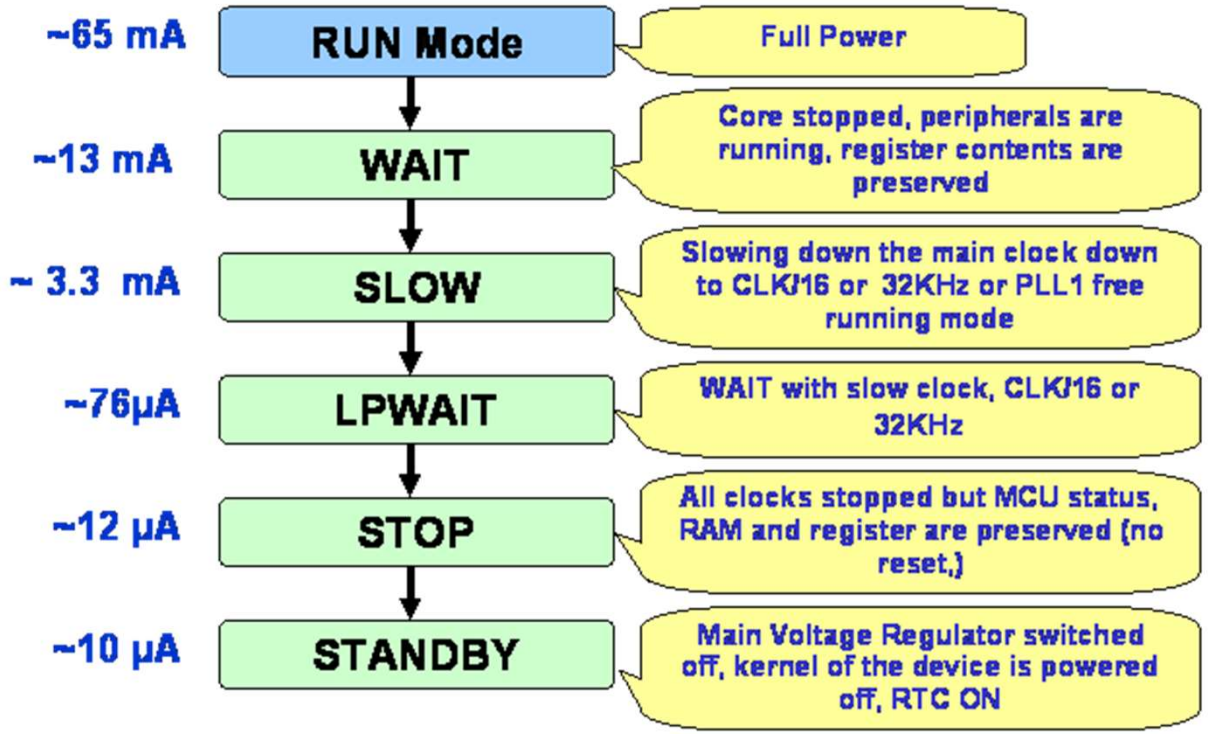
- EEPROM is non-volatile memory - it retains its contents when power is removed. Some micros include a small amount of EEPROM storage for use by your code.
- In the past on-chip EEPROM was relatively rare and expensive compared to an external EEPROM chip, but it appears that flash micros can offer EEPROM at little additional cost due to the similarity with flash storage.
- The use of EEPROM requires some care.
  - Erasing and rewriting an EEPROM location takes several milliseconds, for which time your code must wait if writing multiple locations, and loss of power can corrupt an EEPROM write in progress if the micro doesn't have an internal or external brownout detector.

Analog Comparator

- The inclusion of analog comparators on micros is a fairly recent wrinkle, but they can be very handy in circuits which have to interface with an analog world.
- Typically the output of the comparator appears to your code as a port pin, and may be able to generate an interrupt.
- Since micros are quite noisy internally, don't assume they will be quite as precise as an external comparator.

Low-Power Modes

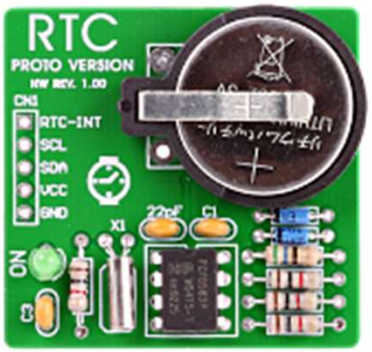
- one or more low-power states
- program does not run and very little power is drawn
- these modes are typically called Wait, Idle, Sleep etc,
- can reduce power supply current to several µA



ARM7
Low Power Modes

Real-Time Clocks

- RTC helps you keep track of the time of day even when the micro is in a low-power mode,
- RTC runs from a separate 32768Hz crystal
- Typically it generates an interrupt at 1/2 second intervals; the interrupt handler can increment time-of-day counters.



Specialized Display Drivers

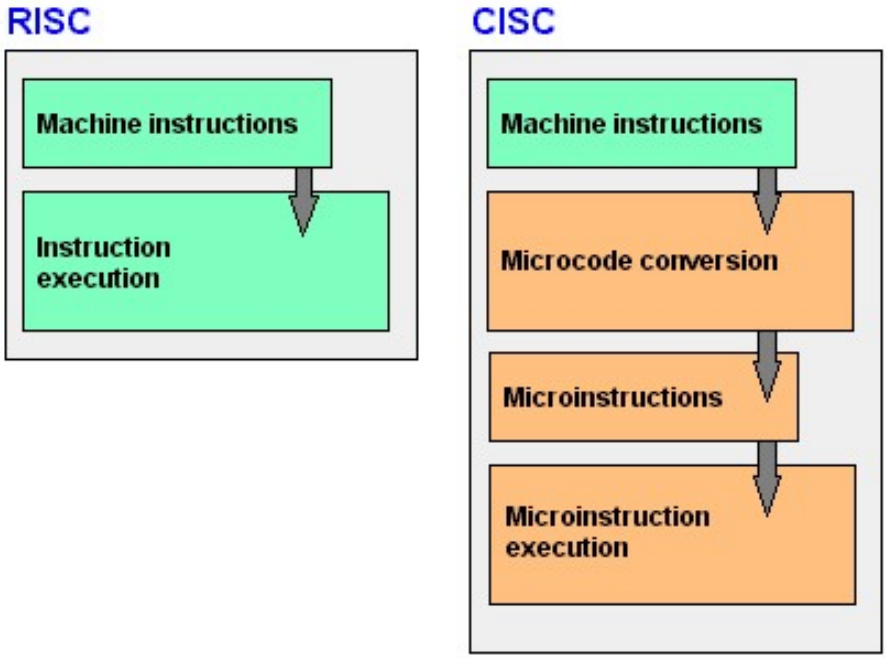
- Micros targeted at modern appliances using LCD and gas plasma displays often include drivers to support them.
- Since these displays are custom-designed for specific applications they're usually of interest only to high-volume appliance designers.
- Japanese micro manufactures are particularly strong in display drivers.

External Peripherals

- **Timer** (bus interface).
- **UART** (bus interface).
- **I/O extender** (bus or serial interface): Provides more general digital I/O.
- **A/D and D/A converters** (bus or serial interface).
- **Real-Time Clock** (serial interface).
- **Non-Volatile RAM** (bus or serial interface): Flash or EEPROM memory.
- standard 74-series **logic chips** (most commonly 74HC CMOS).

CISC and RISC

- **RISC**
  - instruction words are more than 8 bits long (eg 12, 14, 16 bits)
  - most instructions only require one word
- **CISC**
  - have 8-bit instruction words and
  - many instructions that require more than one of them



Spaces

- **Code**: Where the program is stored.
- **Data**: RAM locations available to your program. The I/O space may be mapped into this space rather than separate.
- **Registers**: May be mapped into the Data space or made entirely separate in RISC-type architectures. Access to registers is faster and more capable than access to Data.
- **I/O**: Special locations which form the program's interface with Port I/O and peripherals.

Arithmetic and Logic Unit (ALU)

- This is where all the computations take place.
- The basic set of operations available to all ALUs is:
  - Addition and addition with carry, to provide for multiple precision calculations
  - Subtraction and subtraction with carry
  - Increment and decrement
  - Bitwise shift, leftward or rightward, straight or circular (the outgoing bit in re-injected at the other end of the data word)
  - Logical bitwise OR, AND and EXclusive-OR
  - Logical complement
  - Some ALUs provide additional operations like:
    - Multiplication, Division and more

Arithmetic and Logic Unit (ALU)

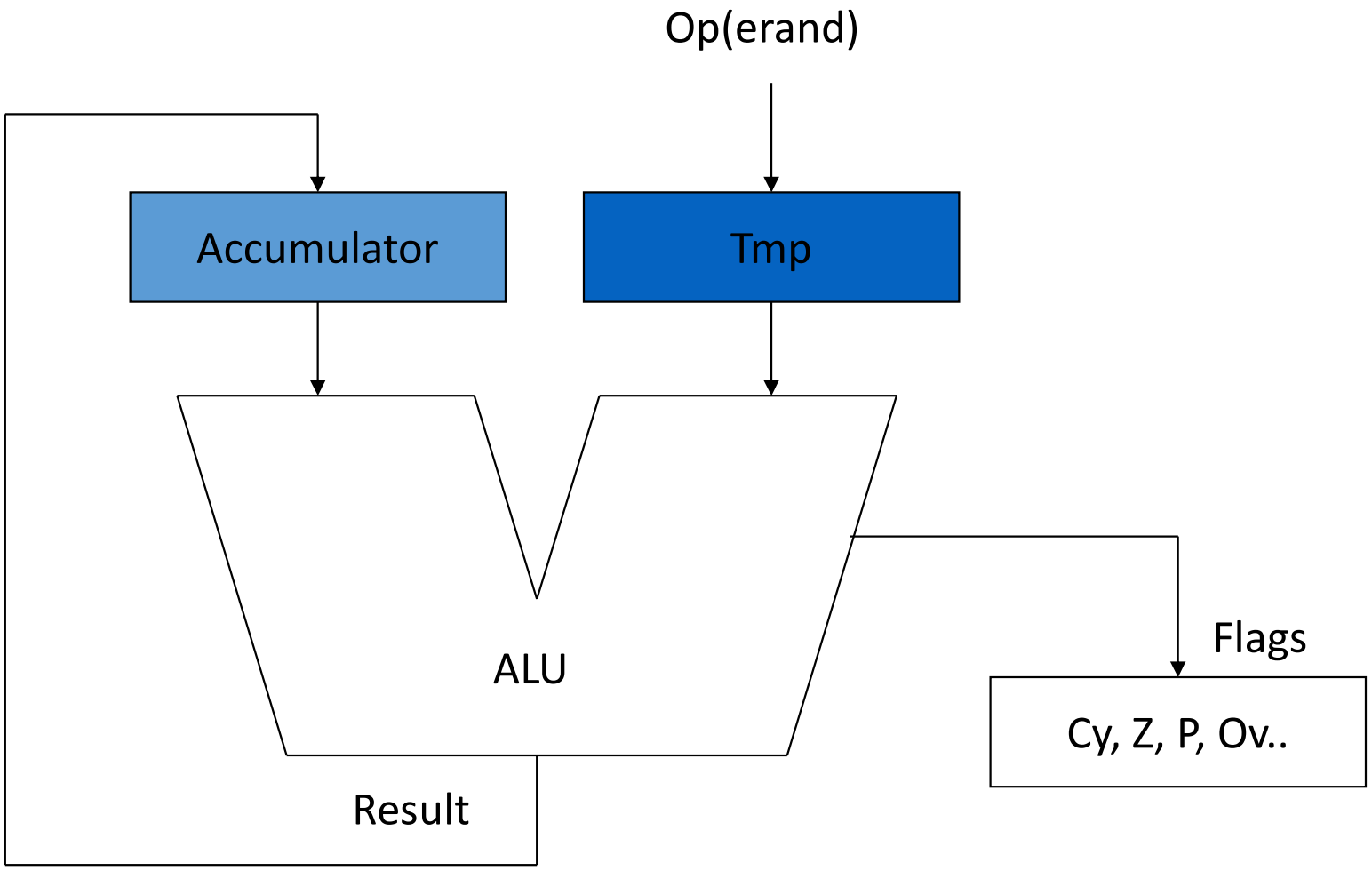
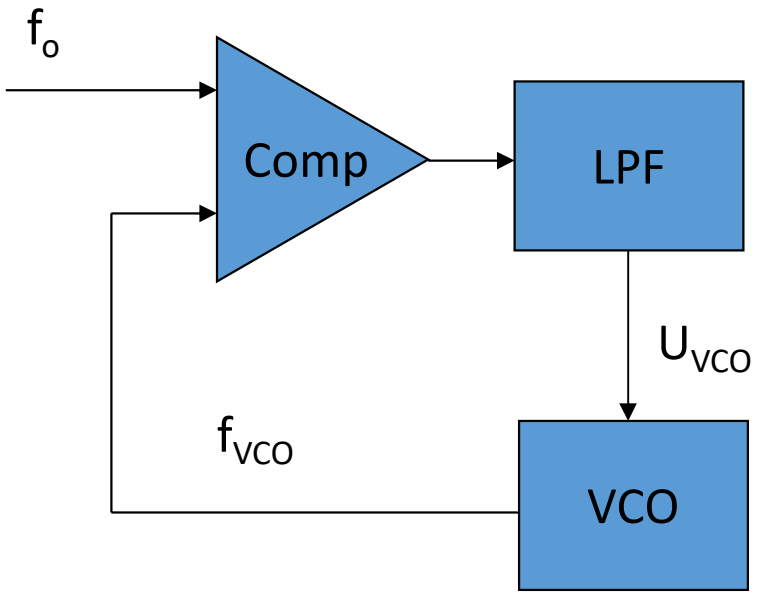


Figure 11: ALU's Architecture

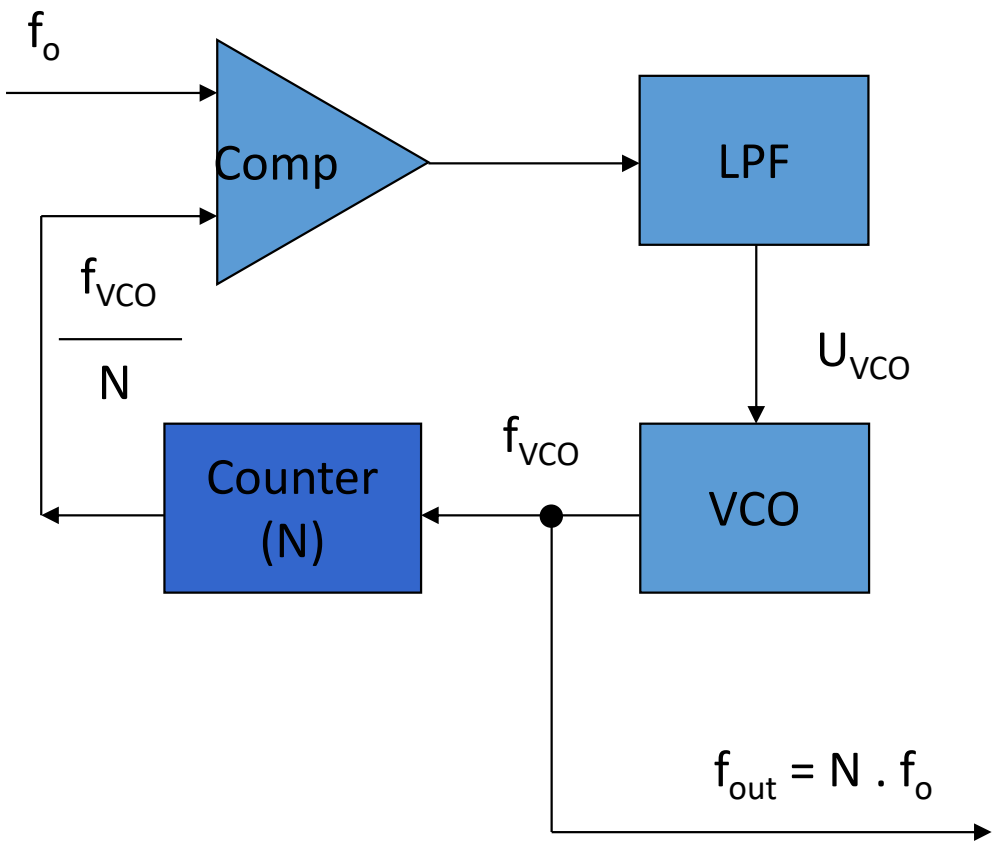
The PLL

- The Phase Locked Loop (**PLL**) circuit is a block that ensures a permanent equality between the phase (and frequency) of an input signal (f_o) and an internal signal (f_{VCO}).
- Adding a counter in the PLL loop, it is possible to multiply the input signal frequency (f_o).



PLL – Block Diagram

PLLs – continued



PLL – Multiplying a Frequency

Figure 12: Multiplying a Frequency

Mid-Range PIC Block Diagram

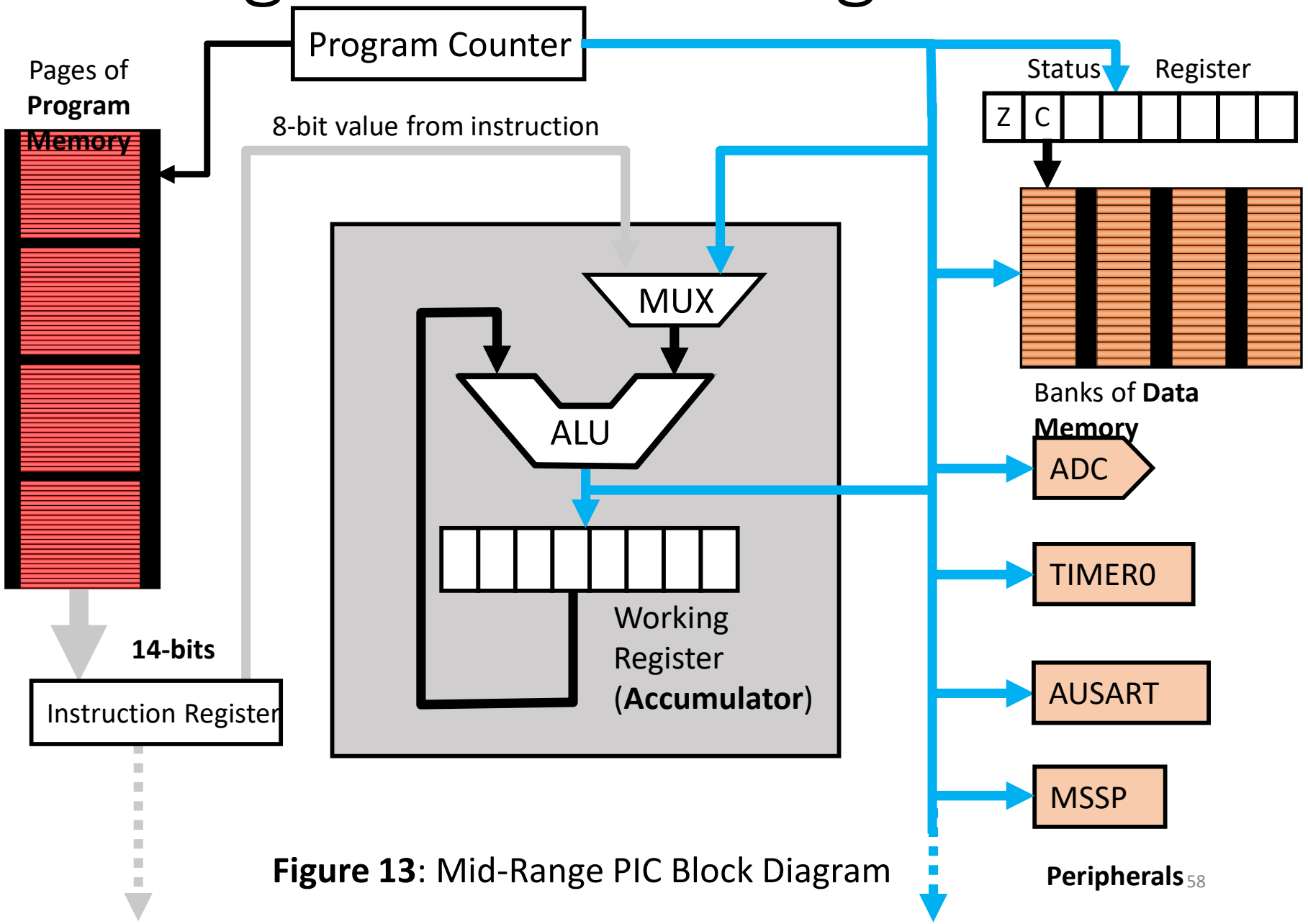


Figure 13: Mid-Range PIC Block Diagram

Choosing a Micro

- Availability and cost
- Support
- Tools
- Variety
- Reliability
- Flexibility
- Package
- Noise Immunity