Lec 1: Microcontroller Fundamentals

What Is A Microcontroller ?

Microcontroller and Embedded Systems

- Microcontroller is a small computer on a single IC (single chip computer) containing a CPU, RAM, ROM, I/O ports, serial ports, timers, A/D and D/A converters.
- Microcontrollers are designed for embedded applications, in contrast to the microprocessor used in PCs or other general purpose applications.
- The Microcontroller is capable of storing and running a program.

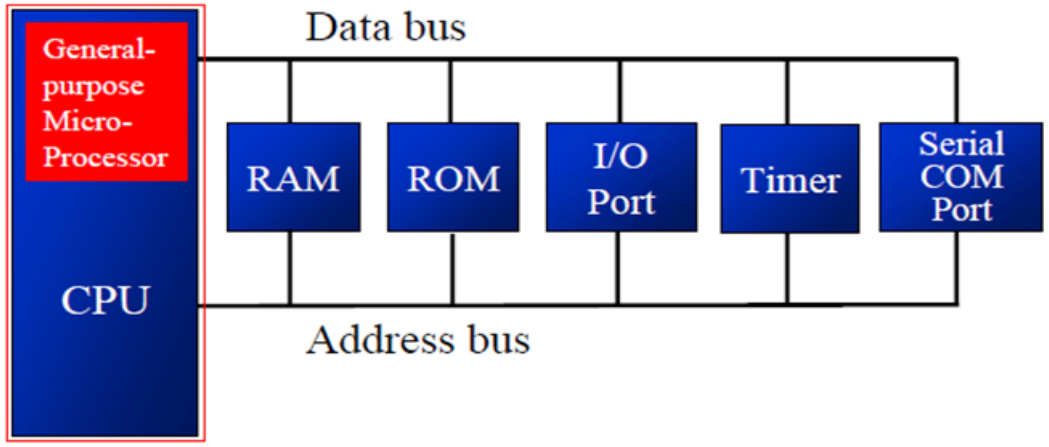
Microcontroller and Embedded Systems

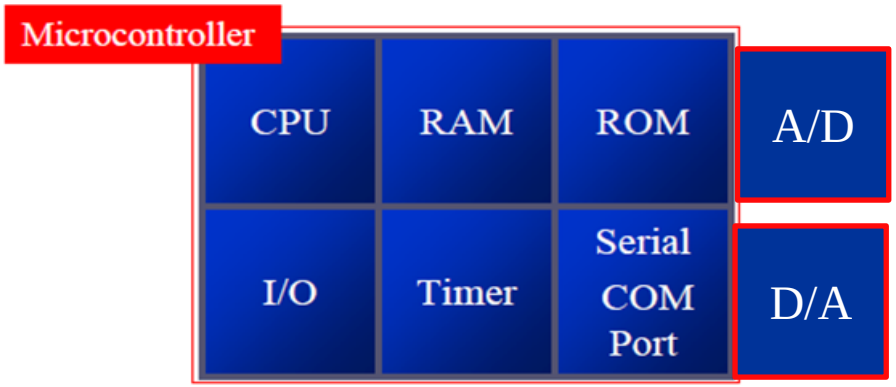
- An **embedded system** is a computer system (using Microcontroller) with a specific function within a larger mechanical or electrical system. It is *embedded* as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices today.

Microcontroller Vs Microprocessor

- Microcontroller differs from a microprocessor in many ways. First and the most important is its functionality.
- To use a microprocessor, other components such as memory, or components for receiving and sending data must be added to it.
- On the other hand, microcontroller is designed to be all of that in one. No other external components are needed, why? Because all necessary components are already built into it.
- Thus, we save the time and space needed to construct devices.

Microcontroller Vs Microprocessor

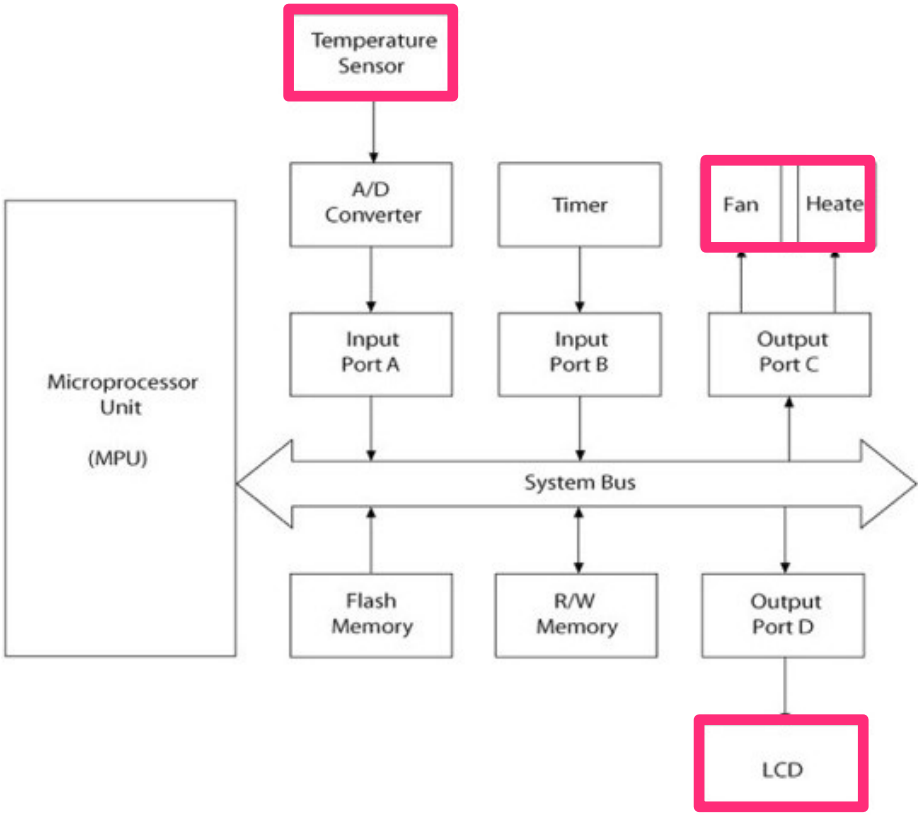




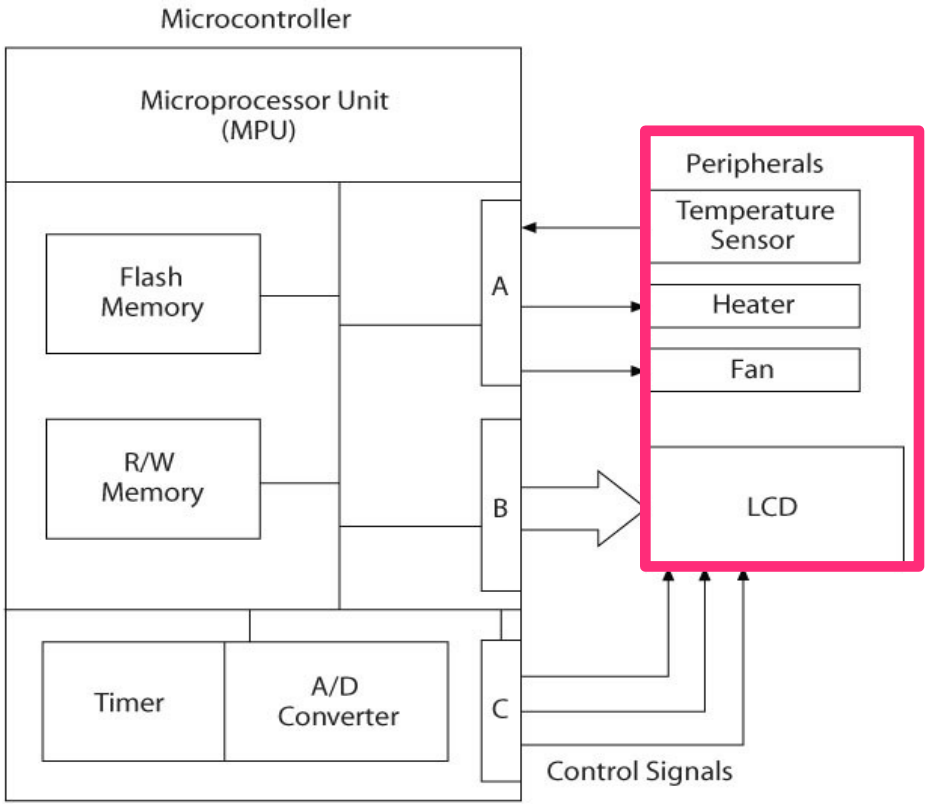
Design Examples …..

Microcontrollers *vs*. Microprocessors

Microprocessor-Based Temperature System



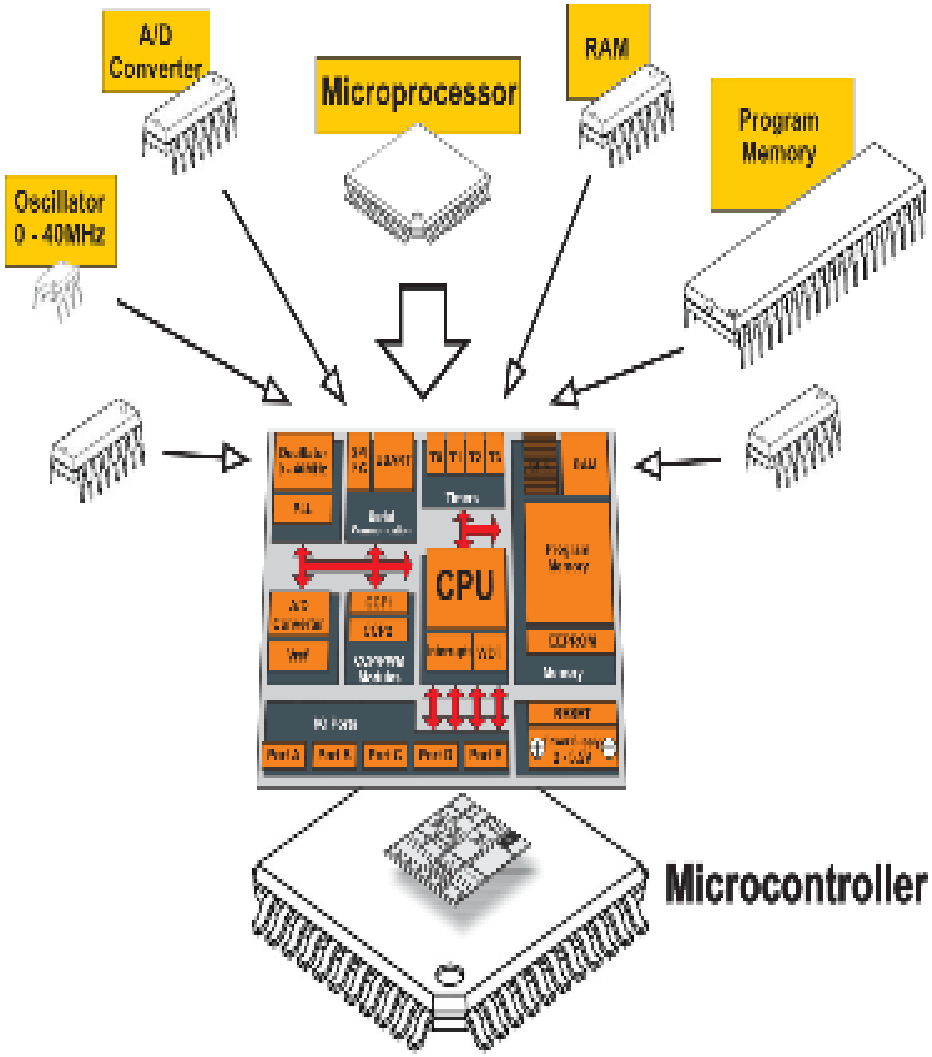
Microcontroller-Based Temperature System

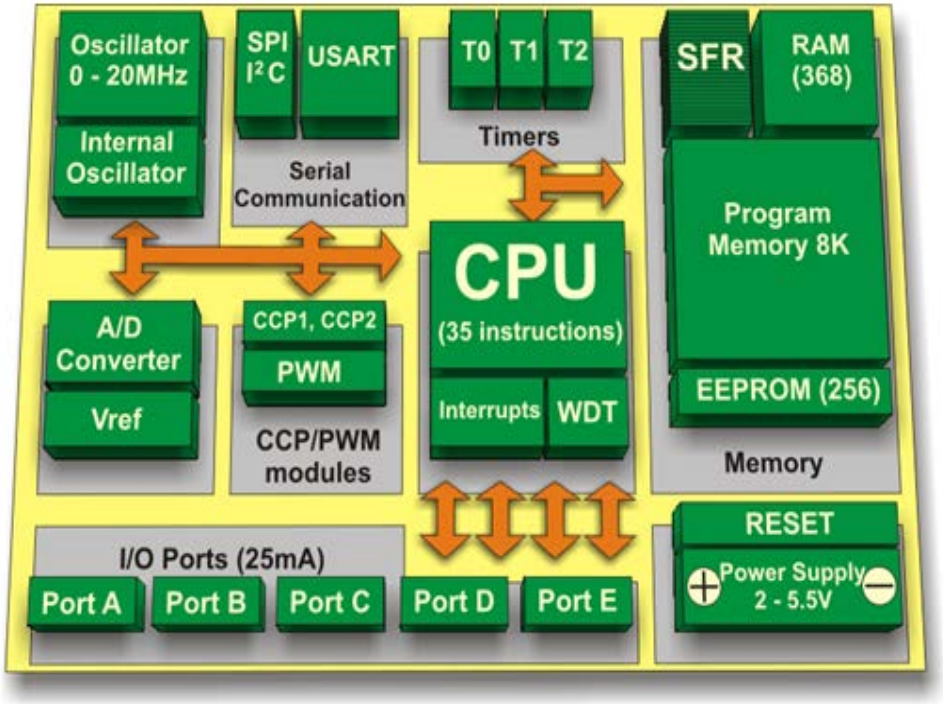


Main components of Microcontroller

- CPU
- Memory Unit
- Input-output unit(I/O)
- Serial communication unit
- Timer
- Watchdog timer
- A/D converter
- D/A converter

Microcontroller Components





Memory Organization

PIC microcontroller has two separate memory blocks, one for data and the other for program.

1) Program memory (Flash memory)

Program memory is used for storing a written program. Flash memory can be programmed and cleared more than once.

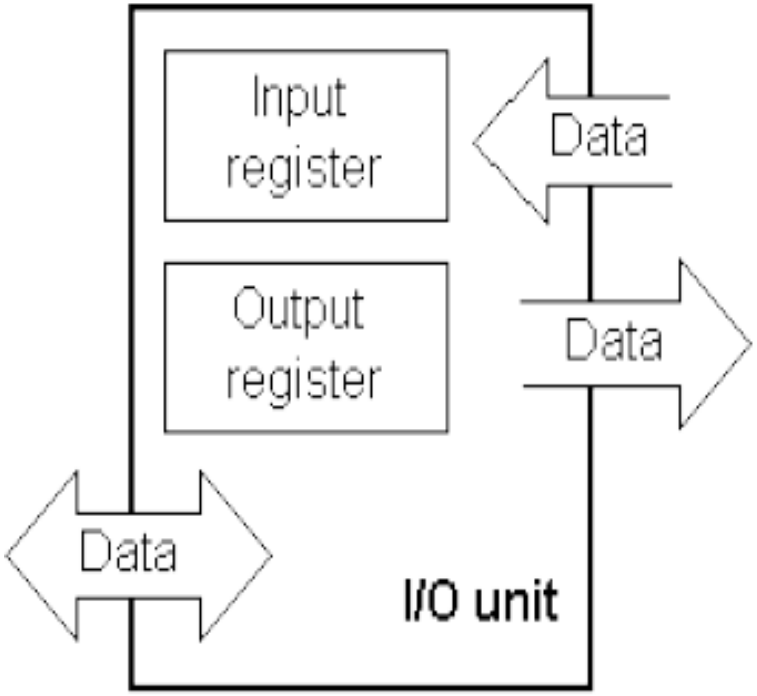
Memory Organization

2) Data memory

Data memory consists of EEPROM and RAM memories.

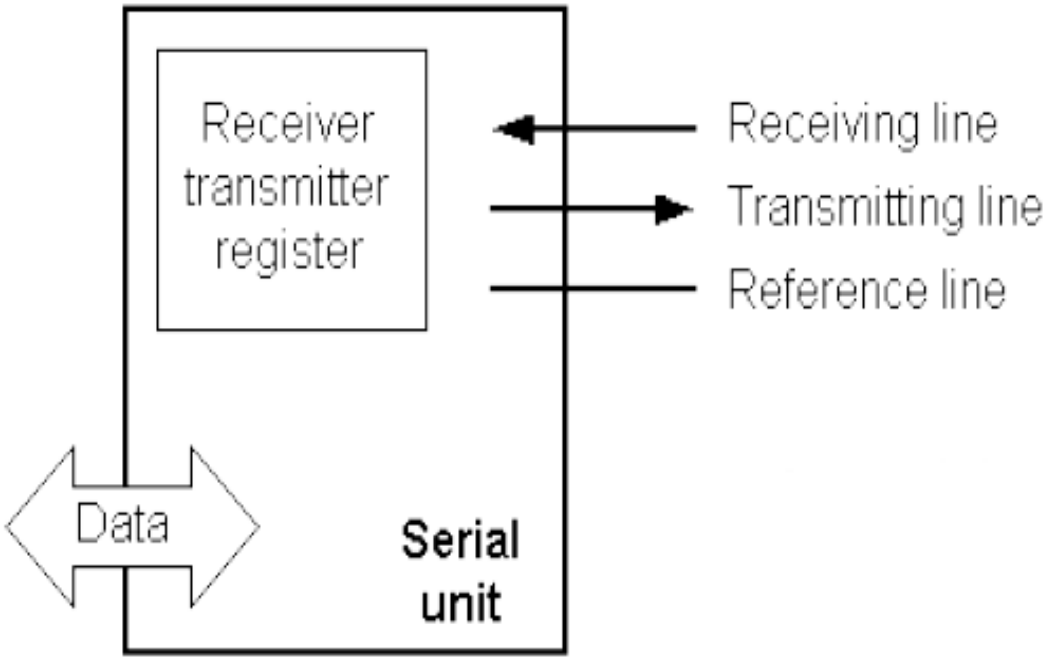
- **EEPROM** is usually used for storing **important data**, e.g. users passwords, that must not be lost when there is no power (power supply suddenly stops).
- A program uses **RAM** during its execution. RAM stores all intermediate-results or **temporary data** during run-time.

- The ports should be configured as input port or output port before using.
- The port pins can be configured individually (any pins can be used as input and other pins of the same port can be used as output)

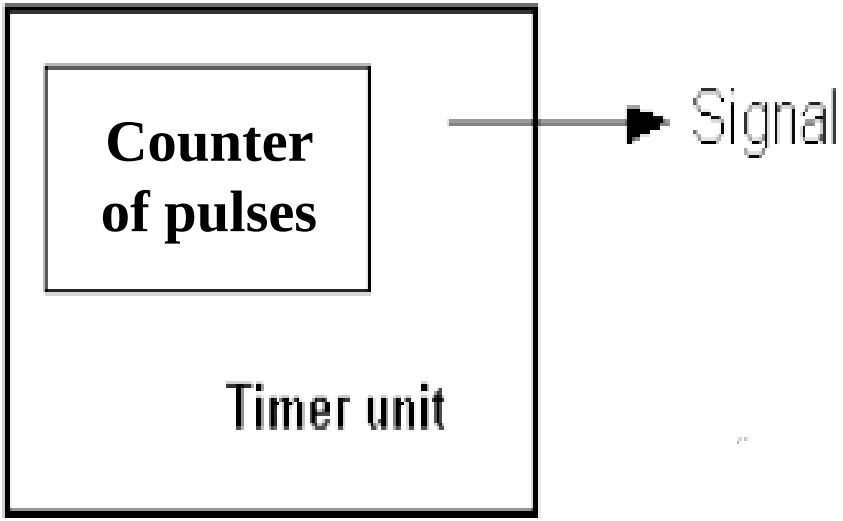


Provides communication with external world

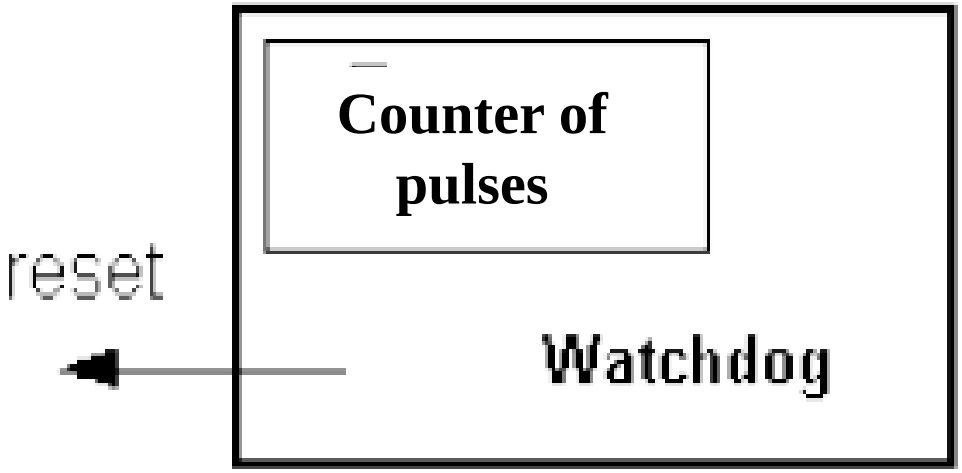
Example of simple I/O unit



Serial Unit is used to send and receive data by only three lines



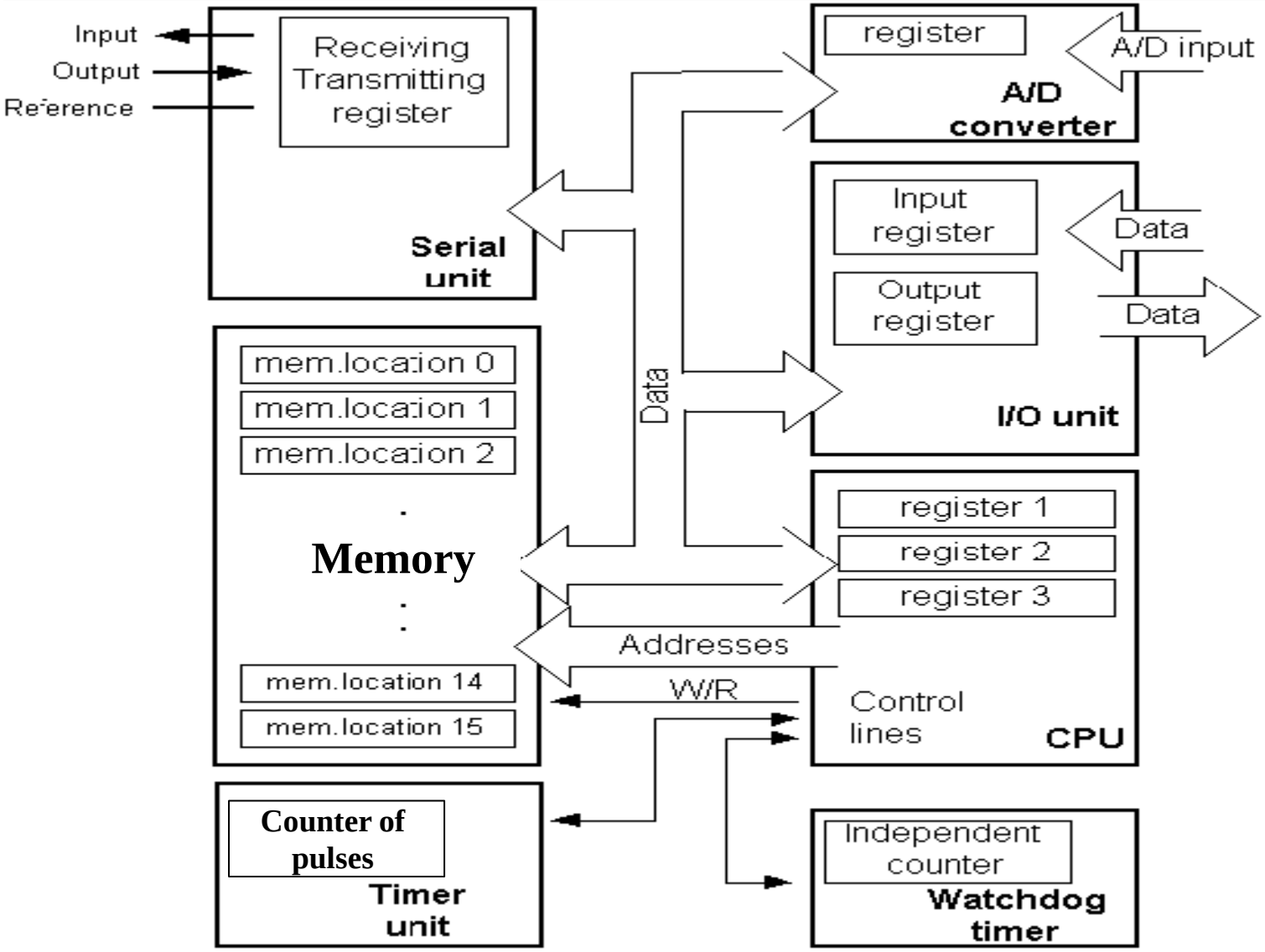
Timer Unit generates signals in regular time intervals



Watchdog Timer (WDT)

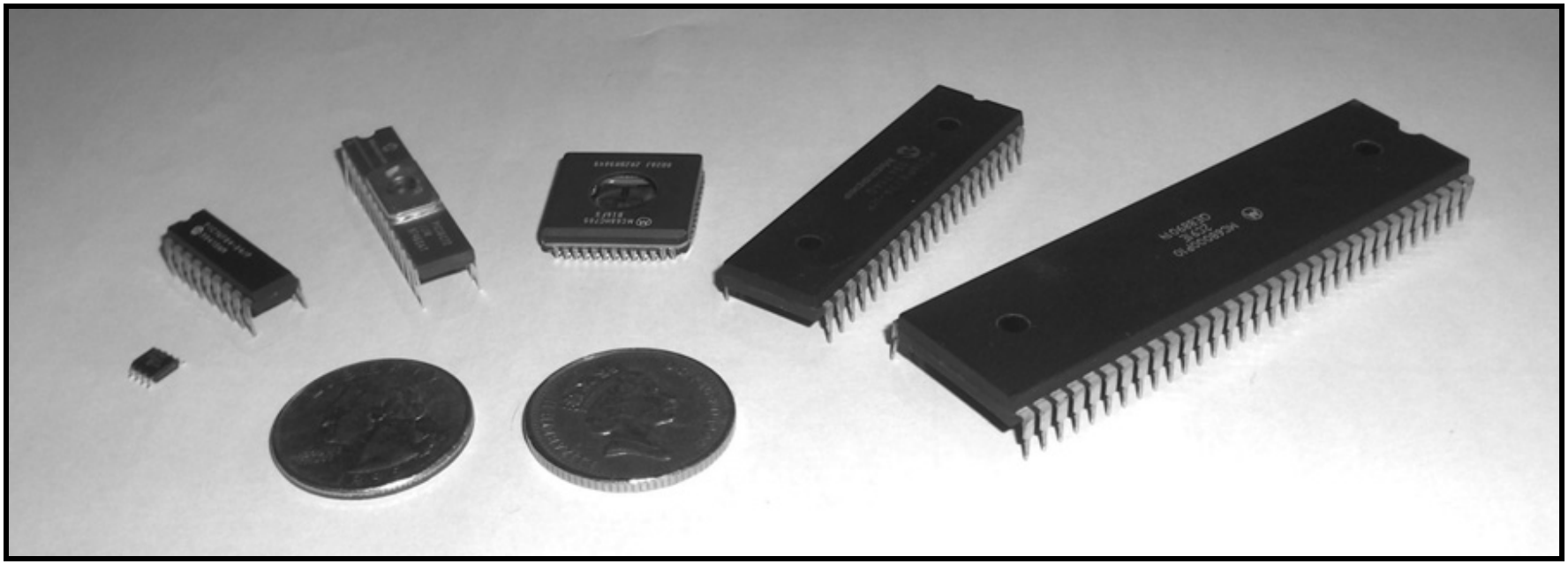
Watchdog Timer (WDT)

- This block is in fact another counter of pulses where our program needs to write a zero in every time it executes correctly.
- In case that program gets "stuck = microcontroller stops executing the program ", 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. That is an important element of every program to be reliable without man's supervision.



Microcontroller outline with its basic elements and internal connections

Microcontroller Packaging

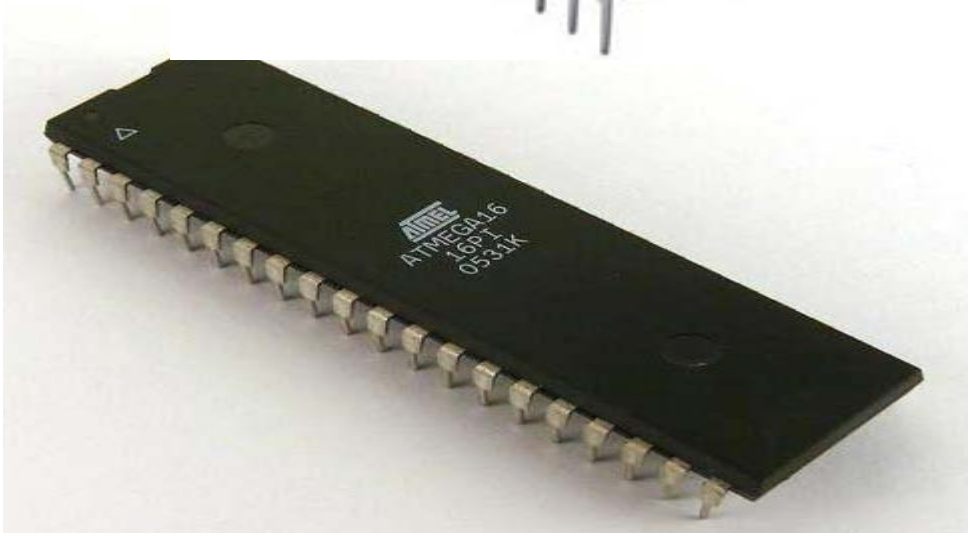


From left to right: PIC 12F508, PIC 16F84A, PIC 16C72, Motorola 68HC05B16, PIC 16F877, Motorola 68000

Microcontroller Manufacturers

- Microchip (**PIC = Programmable Interface Controllers**)
- Atmel
- Hitachi
- Intel
- Motorola
- Zilog

ATMEGA and PIC microcontrollers



Microcontroller Applications

The common applications :

- Telephones and fax machines
- Security systems (for example: finger print device)
- Garage door openers
- Washing machines
- Video games
- Robot
- Air conditioner
- Microwave
- Car Doors
- Mobiles

Programming Languages

There are three types of programming languages can be used to program the microcontroller chip:

1) Machine Language
2) Assembly Language
3) High-Level Languages

Programming Languages

1) Machine Language

- Binary Instructions
- Difficult to write, find errors, and modify
- All programs converted into machine language for execution

Instruction	Hex	Description
10000000	80	Add reg B to Acc
00101000	28	Add Reg R0 to Acc
00011011	1B	Add Acc A and B

Programming Languages

2) Assembly Language

- Machine instructions represented in mnemonics
- Machine dependent

Instruction	Hex	Mnemonic	Description	Processor
10000000	80	ADD B	Add reg B to Acc	Intel 8085
00101000	28	ADD A, R0	Add Reg R0 to Acc	Intel 8051
00011011	1B	ABA	Add Acc A and B	Motorola 6811

Programming Languages

3) High-Level Languages

- BASIC, QBASIC, C, and C++
- Written in statements of spoken languages
- Machine independent
- Easy to write and troubleshoot

Criteria for choosing Microcontrollers

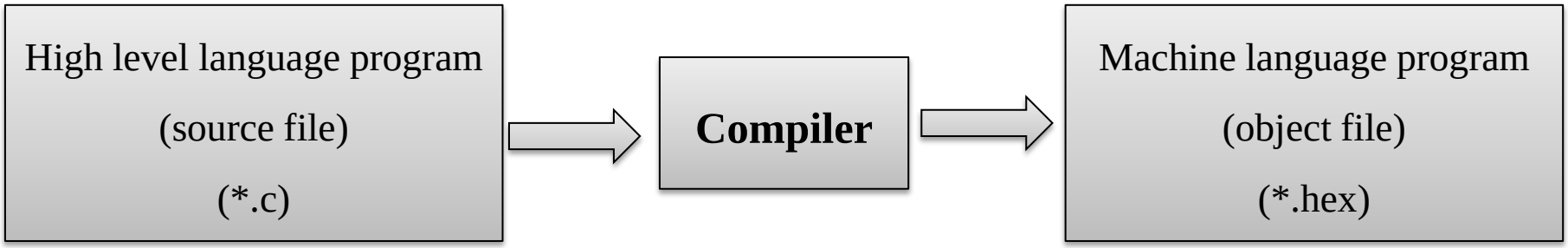
- The amount of RAM and ROM
- The number of I/O pins
- The need of serial port, A/D and D/A (depend on application)
- Cost per unit

Developing a microcontroller-based project

1) Type the program into a PC (using any suitable programming language like C)
2) Compile the program using any compiler. After compilation, the converted program will be stored in a hexadecimal file.
3) Load the program into microcontroller's program memory (this operation is done using a device called the programmer or burner)
4) Design and construct the hardware
5) Test the project.

The compiler function

Converting the program from high level language in to machine language program (always on a hex. format file) which will be loaded to the microcontroller through the programmer device (burner).



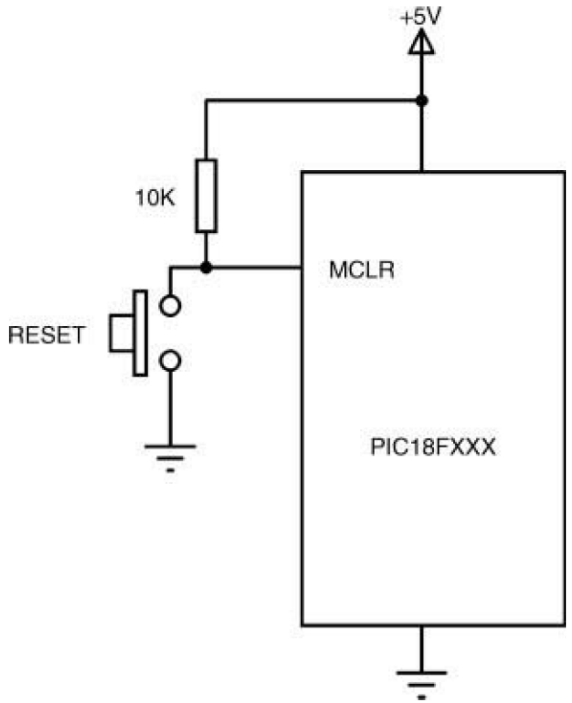
Reset Microcontroller

- Microcontroller can behave inaccurately under certain undesirable conditions.
- In order to continue its proper functioning it has to be reset, meaning the program will be executed again from the beginning.
- A reset action can be automatic by software (e.g. when the watchdog is enabled but not refreshed),
- OR an external button can be used to reset the microcontroller.
- After a reset, the program starting from memory address 0 of the microcontroller is executed.

How Can We Reset Microcontroller

1) Reset during regular work by bringing logical zero to $\overline{\text{MCLR}}$ microcontroller's pin.
2) Reset at watchdog timer (WDT) overflow

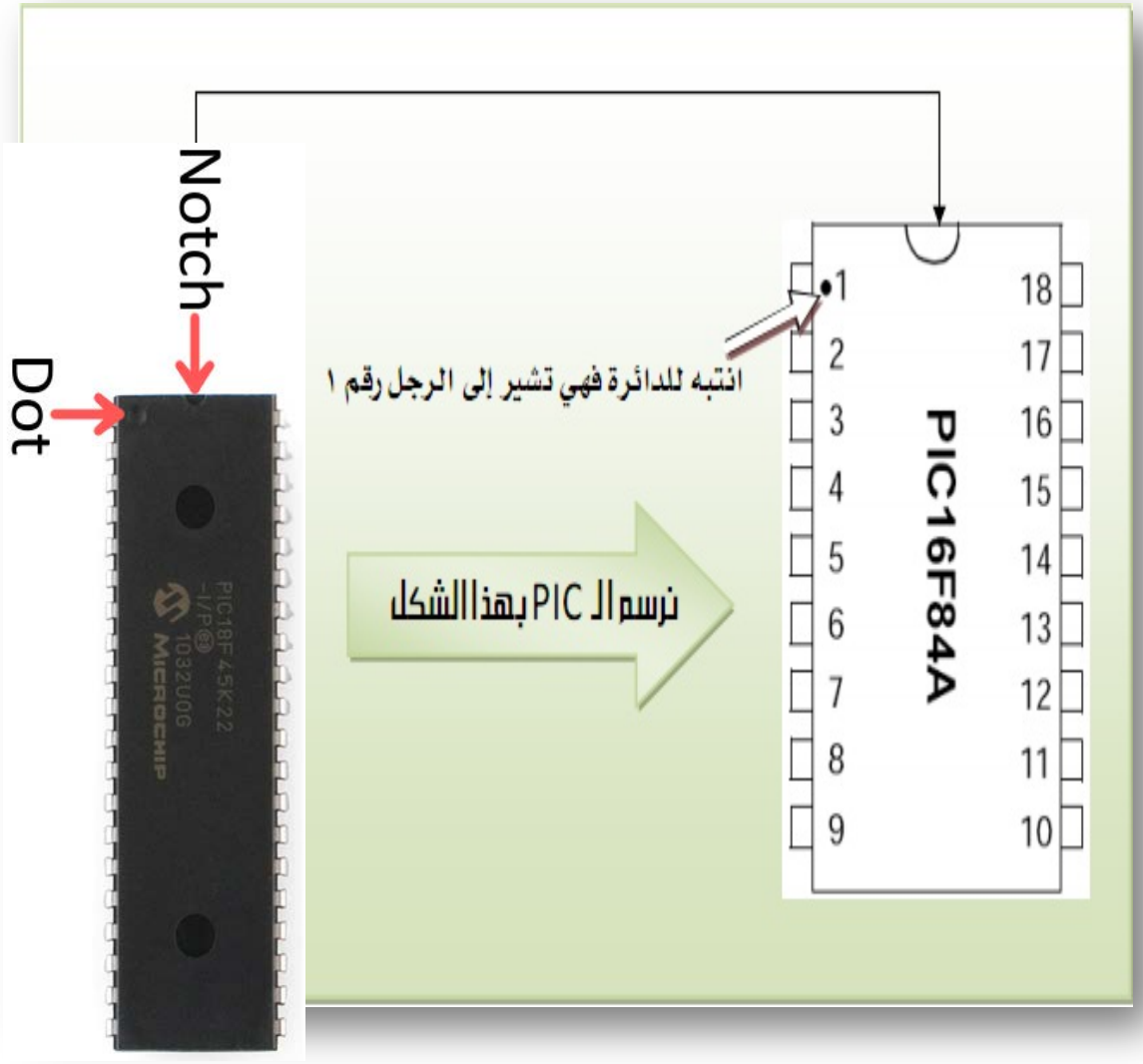
R (10K) is a Resistor



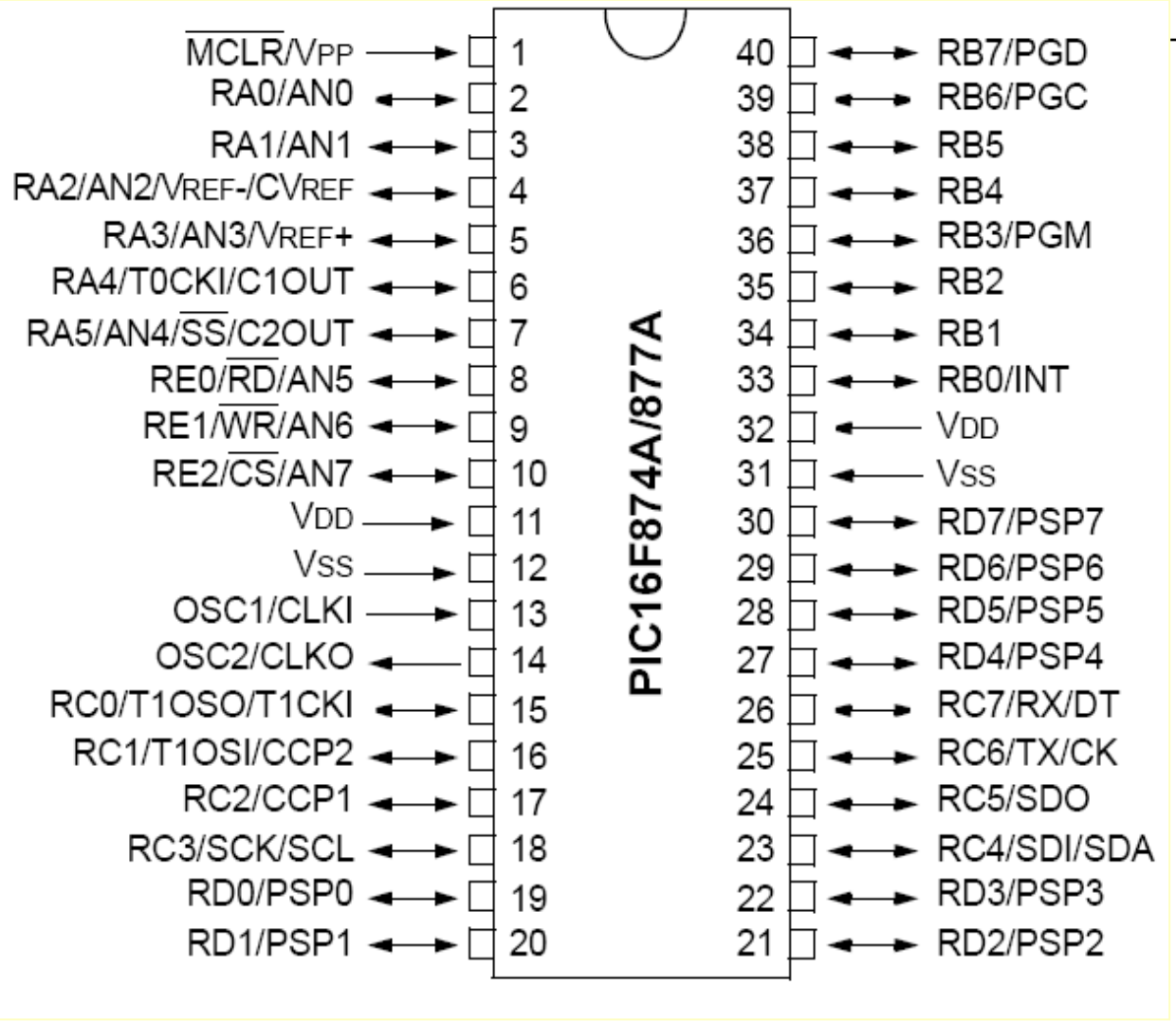
External Reset circuit

Specifications of “PIC16F877A” microcontroller

1. How to read micro pins.
2. Oscillator (max frequency) = 20 MHZ.
3. 4 MHZ & 8 MHZ crystal oscillators are in common use.
4. Max. I/O current = 25mA.
5. Pins description.



PIC16F877A Microcontroller



Specifications:

- Has five ports: **Port A** (6 pins) and **Port B , C, D** (8 pins) , **E** (3 pins)
- 8K words (14 bits) of rewritable memory (flash memory)
- 368 bytes of RAM
- 256 bytes of EEPROM
- A/D Converter

Name	Number (DIP 40)	Function	Description
RE3/MCLR/Vpp	1	RE3	General purpose input Port E
		MCLR	Reset pin. Low logic level on this pin resets microcontroller.
		Vpp	Programming voltage
RA0/AN0/ULPWU/C12IN0-	2	RA0	General purpose I/O port A
		AN0	A/D Channel 0 input
		ULPWU	Stand-by mode deactivation input
		C12IN0-	Comparator C1 or C2 negative input
RA1/AN1/C12IN1-	3	RA1	General purpose I/O port A
		AN1	A/D Channel 1
		C12IN1-	Comparator C1 or C2 negative input
RA2/AN2/Vref-/CVref/C2IN+	4	RA2	General purpose I/O port A
		AN2	A/D Channel 2
		Vref-	A/D Negative Voltage Reference input
		CVref	Comparator Voltage Reference Output
		C2IN+	Comparator C2 Positive Input
RA3/AN3/Vref+/C1IN+	5	RA3	General purpose I/O port A
		AN3	A/D Channel 3
		Vref+	A/D Positive Voltage Reference Input
		C1IN+	Comparator C1 Positive Input

RA3/AN3/Vref+/C1IN+	5	RA3	General purpose I/O port A
		AN3	A/D Channel 3
		Vref+	A/D Positive Voltage Reference Input
		C1IN+	Comparator C1 Positive Input
RA4/T0CKI/C1OUT	6	RA4	General purpose I/O port A
		T0CKI	Timer T0 Clock Input
		C1OUT	Comparator C1 Output
RA5/AN4/SS/C2OUT	7	RA5	General purpose I/O port A
		AN4	A/D Channel 4
		SS	SPI module Input (*Slave Select*)
		C2OUT	Comparator C2 Output
RE0/AN5	8	RE0	General purpose I/O port E
		AN5	A/D Channel 5
RE1/AN6	9	RE1	General purpose I/O port E
		AN6	A/D Channel 6
RE2/AN7	10	RE2	General purpose I/O port E
		AN7	A/D Channel 7
Vdd	11	+	Positive supply
Vss	12	-	Ground (GND)

Name	Number (DIP 40)	Function	Description
RA7/OSC1/CLKIN	13	RA7	General purpose I/O port A
		OSC1	Crystal Oscillator Input
		CLKIN	External Clock Input
RA6/OSC2/CLKOUT	14	OSC2	Crystal Oscillator Output
		CLKO	Fosc/4 Output
		RA6	General purpose I/O port A
RC0/T1OSO/T1CKI	15	RC0	General purpose I/O port C
		T1OSO	Timer T1 Oscillator Output
		T1CKI	Timer T1 Clock Input
RC1/T1OSO/T1CKI	16	RC1	General purpose I/O port C
		T1OSI	Timer T1 Oscillator Input
		CCP2	CCP1 and PWM1 module I/O
RC2/P1A/CCP1	17	RC2	General purpose I/O port C
		P1A	PWM Module Output
		CCP1	CCP1 and PWM1 module I/O
RC3/SCK/SCL	18	RC3	General purpose I/O port C
		SCK	MSSP module Clock I/O in SPI mode
		SCL	MSSP module Clock I/O in I^2C mode
RD0	19	RD0	General purpose I/O port D
RD1	20	RD1	General purpose I/O port D
RD2	21	RD2	General purpose I/O port D
RD3	22	RD3	General purpose I/O port D
RC4/SDI/SDA	23	RC4	General purpose I/O port A
		SDI	MSSP module *Data* input in SPI mode
		SDA	MSSP module *Data* I/O in I^2C mode
RC5/SDO	24	RC5	General purpose I/O port C
		SDO	MSSP module *Data* output in SPI mode
RC6/TX/CK	25	RC6	General purpose I/O port C
		TX	USART Asynchronous Output
		CK	USART Synchronous Clock
RC7/RX/DT	26	RC7	General purpose I/O port C
		RX	USART Asynchronous Input
		DT	USART Synchronous Data

Name	Number (DIP 40)	Function	Description
RD4	27	RD4	General purpose I/O port D
RD5/P1B	28	RD5	General purpose I/O port D
		P1B	PWM Output
RD6/P1C	29	RD6	General purpose I/O port D
		P1C	PWM Output
RD7/P1D	30	RD7	General purpose I/O port D
		P1D	PWM Output
Vss	31	-	Ground (GND)
Vdd	32	+	Positive Supply
RB0/AN12/INT	33	RB0	General purpose I/O port B
		AN12	A/D Channel 12
		INT	External Interrupt
RB1/AN10/C12INT3-	34	RB1	General purpose I/O port B
		AN10	A/D Channel 10
		C12INT3-	Comparator C1 or C2 Negative Input
RB2/AN8	35	RB2	General purpose I/O port B
		AN8	A/D Channel 8
RB3/AN9/PGM/C12IN2-	36	RB3	General purpose I/O port B
		AN9	A/D Channel 9
		PGM	Programming enable pin
		C12IN2-	Comparator C1 or C2 Negative Input
RB4/AN11	37	RB4	General purpose I/O port B
		AN11	A/D Channel 11
RB5/AN13/T1G	38	RB5	General purpose I/O port B
		AN13	A/D Channel 13
		T1G	Timer T1 External Input
RB6/ICSPCLK	39	RB6	General purpose I/O port B
		ICSPCLK	Serial programming Clock
RB7/ICSPDAT	40	RB7	General purpose I/O port B
		ICSPDAT	Programming enable pin

Oscillator

➢Clock generator - oscillator

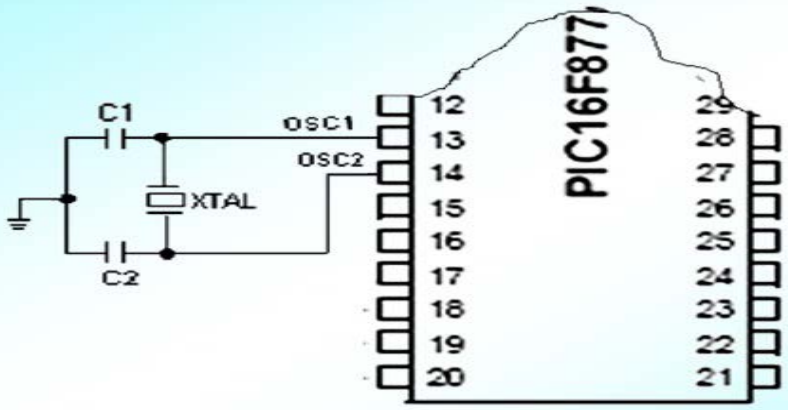
Crystal controlled oscillator connected with two pins "9&10 for 28 pins; 13&14 for 40 pins".

The PIC16F8X can be operated in four different oscillator modes:

- **LP** Low Power Crystal
- **XT** Crystal/Resonator
- **HS** High Speed Crystal/Resonator
- **RC** Resistor/Capacitor

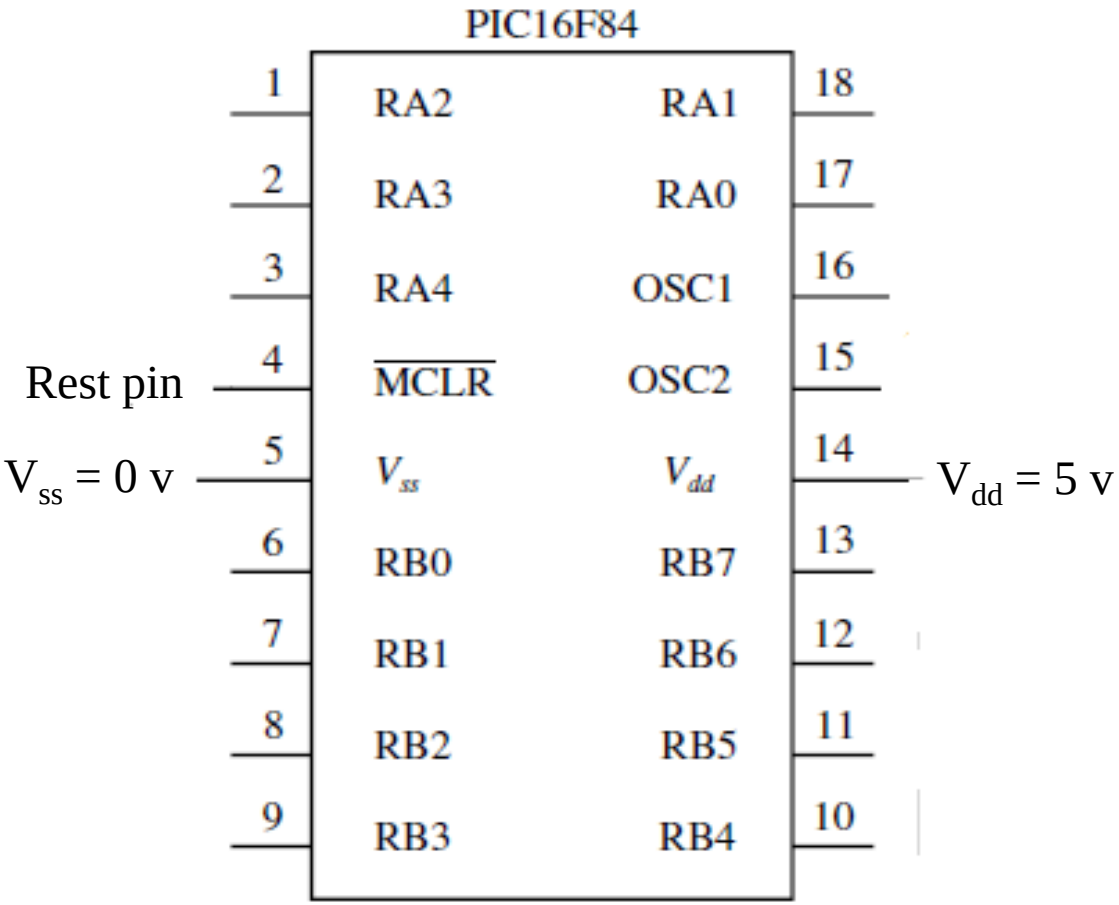
➢Capacitor selection for crystal oscillator

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF



Connecting the crystal oscillator to give clock to a microcontroller

Another Model (PIC16F84 Microcontroller)



Specifications:

- Has two ports: **Port A** (5 pins) and **Port B** (8 pins)
- 1K words (14 bits) of rewritable memory (flash memory)
- 68 byte of RAM