Distributed & Embedded Real-Time Systems (0640751)

Lecture (6)

DERTS Design Requirements (2): Interrupts and Timer/Counter Applications

Prof. Kasim M. Al-Aubidy
Philadelphia University

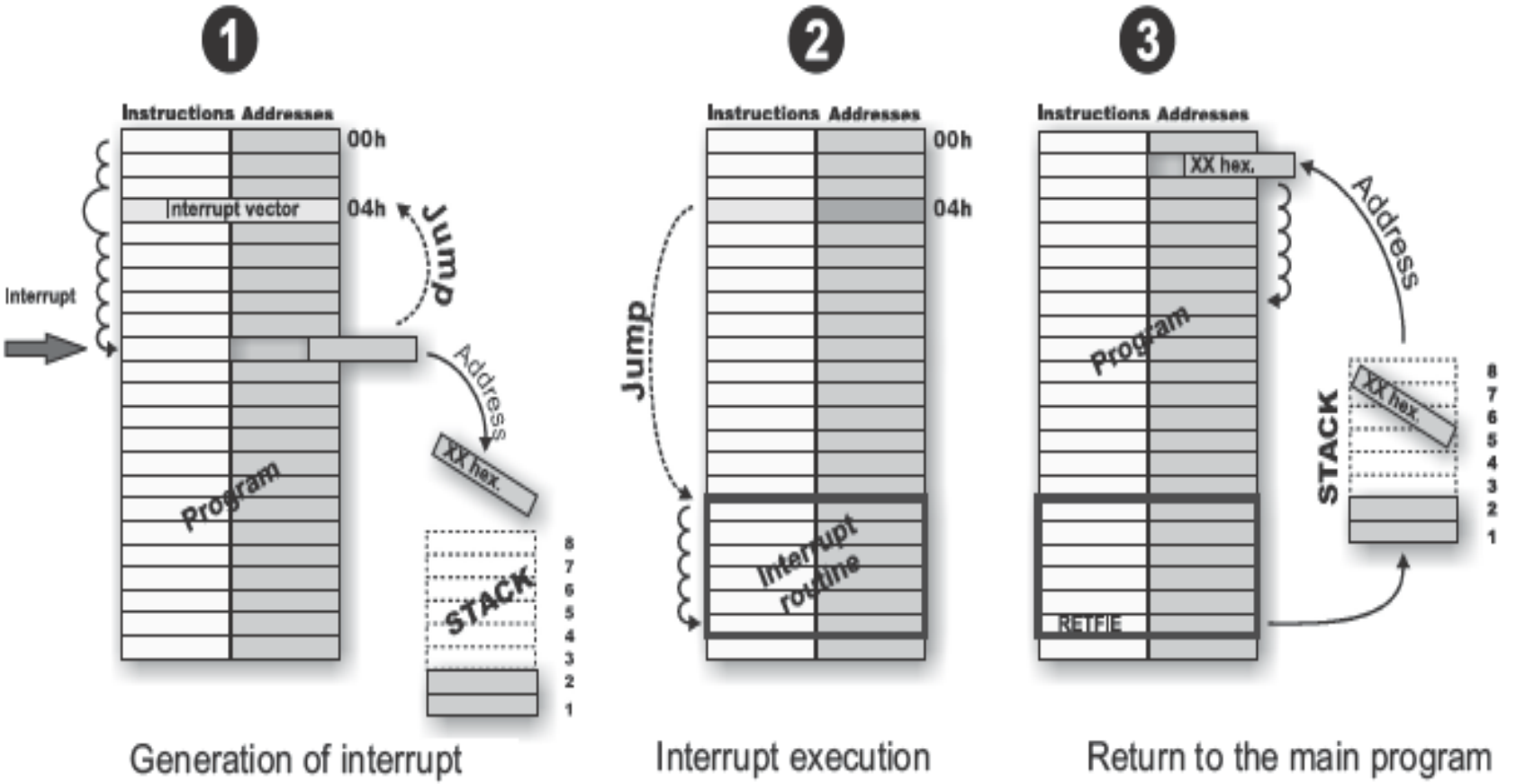
Lecture Outline:

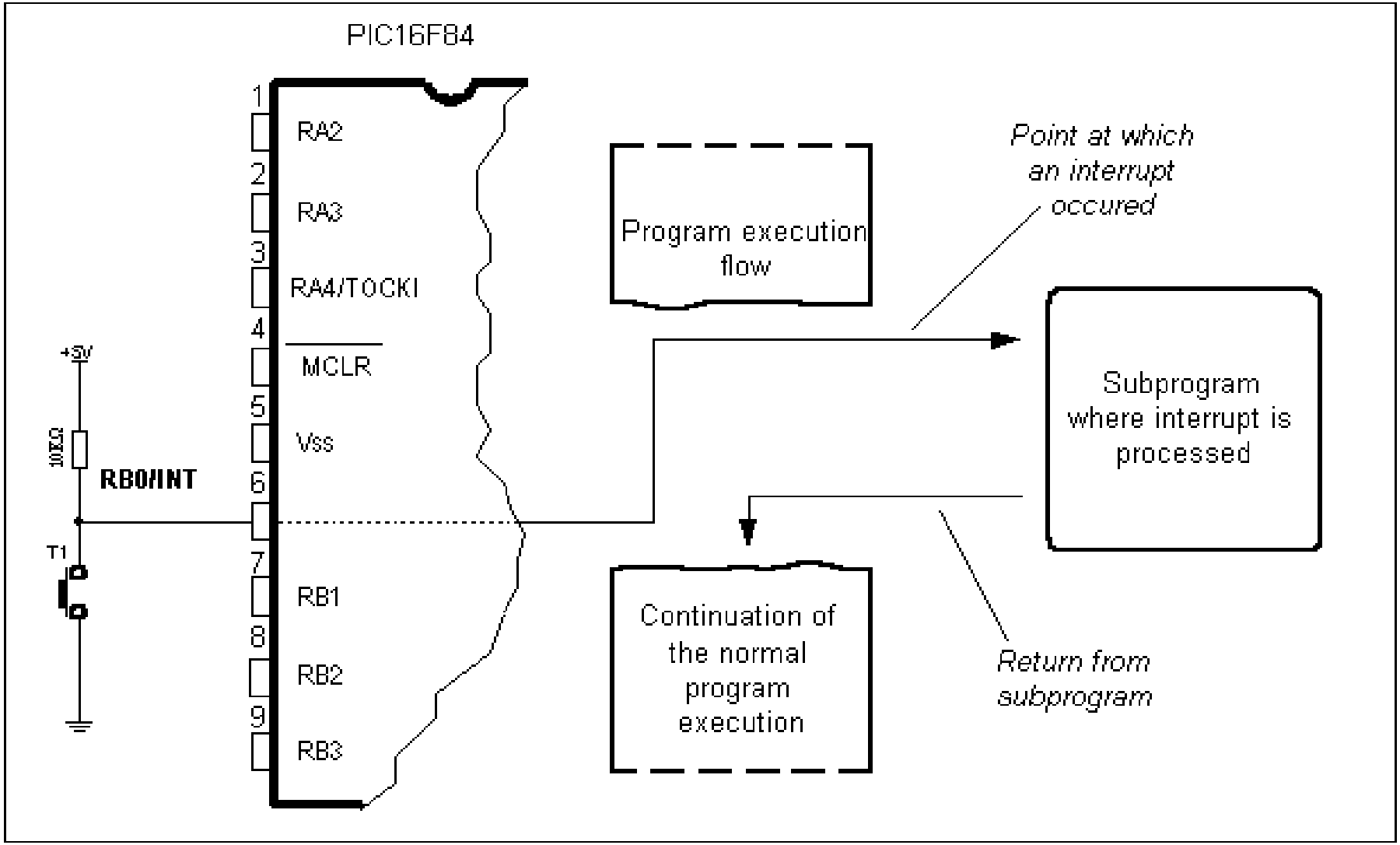
- Features of microcomputers and microcontrollers.
- Microcontroller Interrupts.
- Input/Output Ports.
- Programmable Timer/Counter.
- Standard interfacing techniques.
- Digital input/output interface.
- Analog input/output interface.
- Pulse input/output interface.
- Data acquisition system design.
- Management of data acquisition system.

Interrupts:

Interrupts are a mechanism which enables MC to respond to some events, regardless of what MC is doing at that time.

Each interrupt changes the program flow, interrupts it and after executing an interrupt routine, it continues from that same point on.




PIC16F84
1
RA2
2
RA3
3
RA4/TOCKI
4
MCLR
5
Vss
6
RB0/INT
7
RB1
8
RB2
9
RB3
+5V
10KΩ
T1
Program execution flow
Point at which an interrupt occured
Subprogram where interrupt is processed
Continuation of the normal program execution
Return from subprogram

INTCON Register:

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							

Bit 7: **GIE** (*Global Interrupt Enable*): enables or disables all INTs.

Bit 6: **EEIE** (*EEPROM Write Complete Interrupt Enable bit*): enables an INT at the end of a writing routine to EEPROM.

If EEIE and EEIF are set simultaneously , an INT will occur.

bit 5: **T0IE** (*TMR0 Overflow Interrupt Enable*): enables INTs during TMR0 overflow.

If T0IE and T0IF are set simultaneously, interrupt will occur.

Bit 4: **INTE** (*INT External Interrupt Enable bit*): enables external INT from pin RB0/INT.

Bit 3 **RBIE** (*RB port change Interrupt Enable*): enables INTs to occur at the change of status of pins 4, 5, 6, and 7 of port B.

If RBIE and RBIF are simultaneously set, an INT will occur.

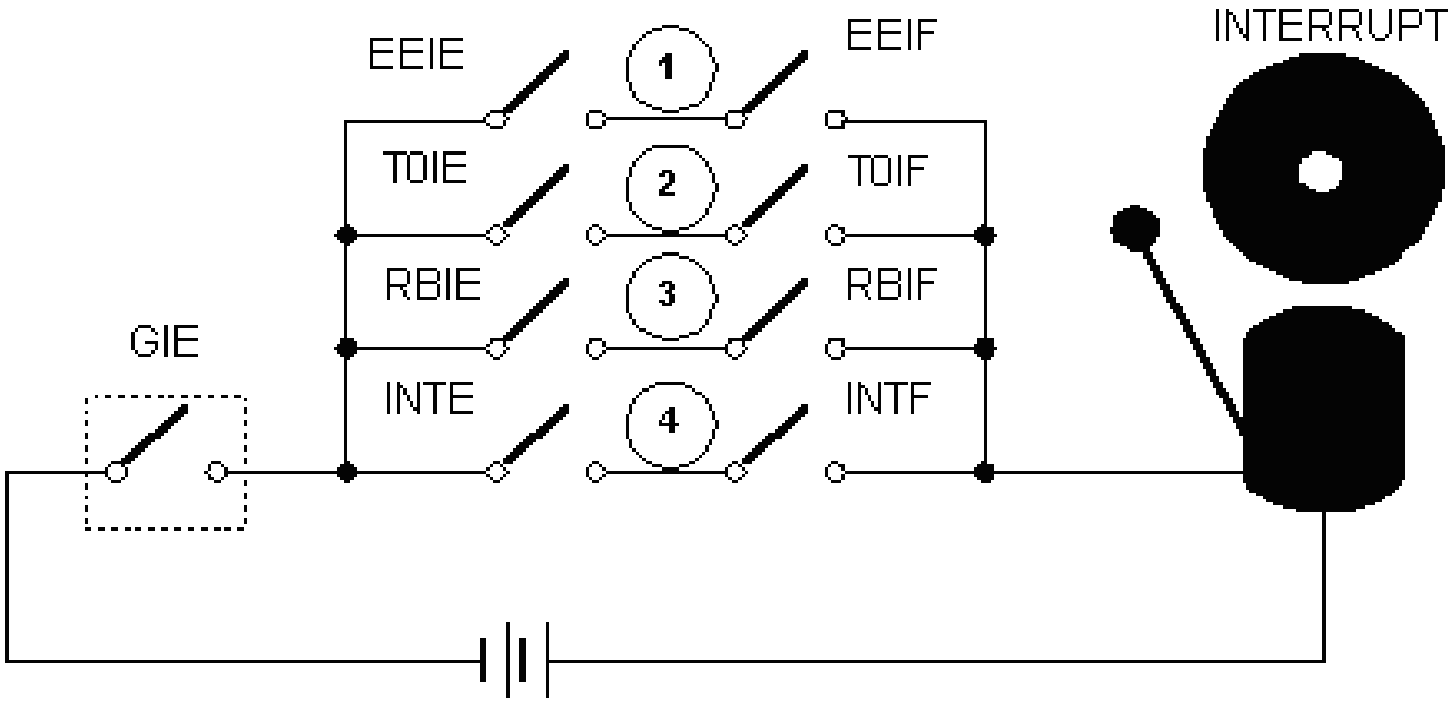
Bit 2: **T0IF** (*TMR0 Overflow Interrupt Flag*): Overflow of counter TMR0.

Bit must be cleared in program in order for an INT to be detected.

Bit 1: **INTF** (*INT External Interrupt Flag*) External INT occurred.

If a rising or falling edge was detected on pin RB0/INT, INTF is set.

Bit 0: **RBIF** (*RB Port Change Interrupt Flag*): informs about changes on pins 4, 5, 6 and 7 of port B.



Simplified outline of PIC16F84 microcontroller interrupt

PIC16F84 has four interrupt sources:

1. Termination of writing data to EEPROM
2. TMR0 interrupt caused by timer overflow
3. Interrupt during alteration on RB4, RB5, RB6 and RB7 pins of port B.
4. External interrupt from RB0/INT pin of microcontroller.

External interrupt on RB0/INT pin:

External interrupt on RB0/INT pin is triggered by rising signal edge (if bit INTEDG=1 in OPTION<6> register), or falling edge (if INTEDG=0).
When correct signal appears on INT pin, INTF bit is set in INTCON register.
INTF bit (INTCON<1>) must be cleared in interrupt routine, so that interrupt wouldn't occur again while going back to the main program.
Interrupt can be turned off by resetting INTE control bit (INTCON<4>).

Interrupt during a TMR0 counter overflow:

Overflow of TMR0 counter (from FFh to 00h) will set T0IF (INTCON<2>) bit.
This is very important interrupt because many real problems can be solved using this interrupt.

Interrupt upon a change on pins 4, 5, 6 and 7 of port B

•Change of input signal on PORTB <7:4> sets RBIF (INTCON<0>) bit.

•Four pins RB7, RB6, RB5 and RB4 of port B, can trigger an interrupt which occurs when status on them changes from logic one to logic zero, or vice versa.

•For pins to be sensitive to this change, they must be defined as input. If any one of them is defined as output, interrupt will not be generated at the change of status. If they are defined as input, their current state is compared to the old value which was stored at the last reading from port B.

Interrupt upon finishing write-subroutine to EEPROM

This interrupt is of practical nature only. Since writing to one EEPROM location takes about 10ms (which is a long time in the notion of a MC), it doesn't pay off to a MC to wait for writing to end.

Thus interrupt mechanism is added which allows the MC to continue executing the main program, while writing in EEPROM is being done in the background.

When writing is completed, interrupt informs the MC that writing has ended. EEIF bit, through which this informing is done, is found in EECON1 register. Occurrence of an interrupt can be disabled by resetting the EEIE bit.

Interrupt initialization:

To use an interrupt mechanism of a MC, some initialization tasks need to be performed.

By initialization we define to what interrupts the MC will respond, and which ones it will ignore.

```
clrf INTCON              ; all interrupts disabled
movlw B'00010000'        ; external interrupt only is enabled
bsf INTCON, GIE          ; occurrence of interrupts allowed
```

The above example shows initialization of external interrupt on RB0 pin of a MC.

GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
bit7							

Input/Output Ports:

- The 16F84 MC has 13 programmable I/O lines (pins).
- Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

PORTA and the TRISA Register

- PORTA is a 5-bit wide, bi-directional port.
- The corresponding data direction register is TRISA.
- Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin.
- The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output.
- All other RA port pins have TTL input levels and full CMOS output drivers.

INITIALIZING PORTA

```
BCF     STATUS, RP0 ;
CLRF    PORTA       ; Initialize PORTA by
                    ; clearing output
                    ; data latches
BSF     STATUS, RP0 ; Select Bank 1
MOVLW   0x0F        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISA       ; Set RA<3:0> as inputs
                    ; RA4 as output
                    ; TRISA<7:5> are always
                    ; read as '0'.
```

PORTA FUNCTIONS

Name	Bit0	Buffer Type	Function
RA0	bit0	TTL	Input/output
RA1	bit1	TTL	Input/output
RA2	bit2	TTL	Input/output
RA3	bit3	TTL	Input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0.

SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
05h	PORTA	RA4/T0CKI	RA3	RA2	RA1	RA0
85h	TRISA	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0

PORTB and TRISB Registers

- PORTB is an 8-bit wide, bi-directional port.
- The corresponding data direction register is TRISB.
- Four pins (RB7:RB4) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison).
- The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).
- This interrupt can wake the device from SLEEP.
- The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

  a) Any read or write of PORTB. This will end the mismatch condition.

  b) Clear flag bit RBIF.

INITIALIZING PORTB

```
BCF     STATUS, RP0 ;
CLRF    PORTB       ; Initialize PORTB by
                    ; clearing output
                    ; data latches
BSF     STATUS, RP0 ; Select Bank 1
MOVLW   0xCF        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISB       ; Set RB<3:0> as inputs
                    ; RB<5:4> as outputs
                    ; RB<7:6> as inputs
```

PORTB FUNCTIONS

Name	Bit	Buffer Type	I/O Consistency Function
RB0/INT	bit0	TTL/ST	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3	bit3	TTL	Input/output pin. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6	bit6	TTL/ST	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7	bit7	TTL/ST	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

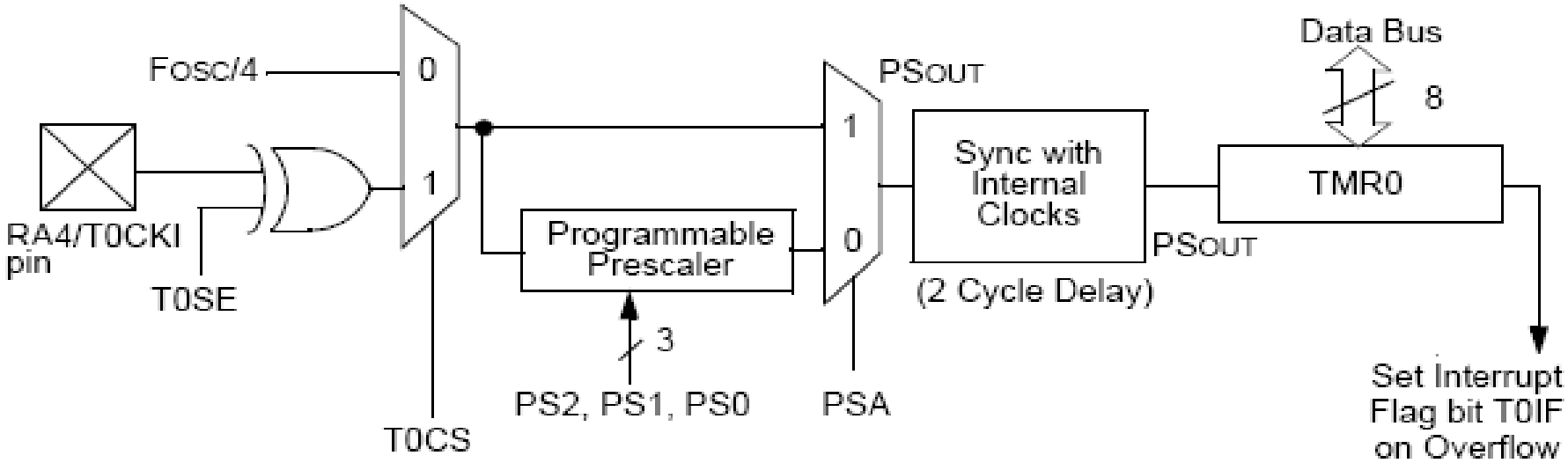
Programmable Timer/Counter:

- The Timer0 module timer/counter has the following features:
  - 8-bit timer/counter
  - Readable and writable
  - Internal or external clock select
  - Edge select for external clock
  - 8-bit software programmable prescaler
  - Interrupt-on-overflow from FFh to 00h
- Timer0 can operate as a timer or as a counter.

- Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler).
- Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In Counter mode, Timer0 will increment, either on every rising or falling edge of pin RA4/T0CKI.

Prescaler

- An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer.
- The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.
- Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable.
- Setting bit PSA will assign the prescaler to the Watchdog Timer (WDT). When the prescaler is assigned to the
- WDT, prescale values of 1:1, 1:2, ..., 1:128 are selectable.
- When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., CLRF 1, MOVWF 1, BSF 1,etc.) will clear the prescaler.
- When assigned to WDT, a CLRWDT instruction will clear the prescaler along with the WDT



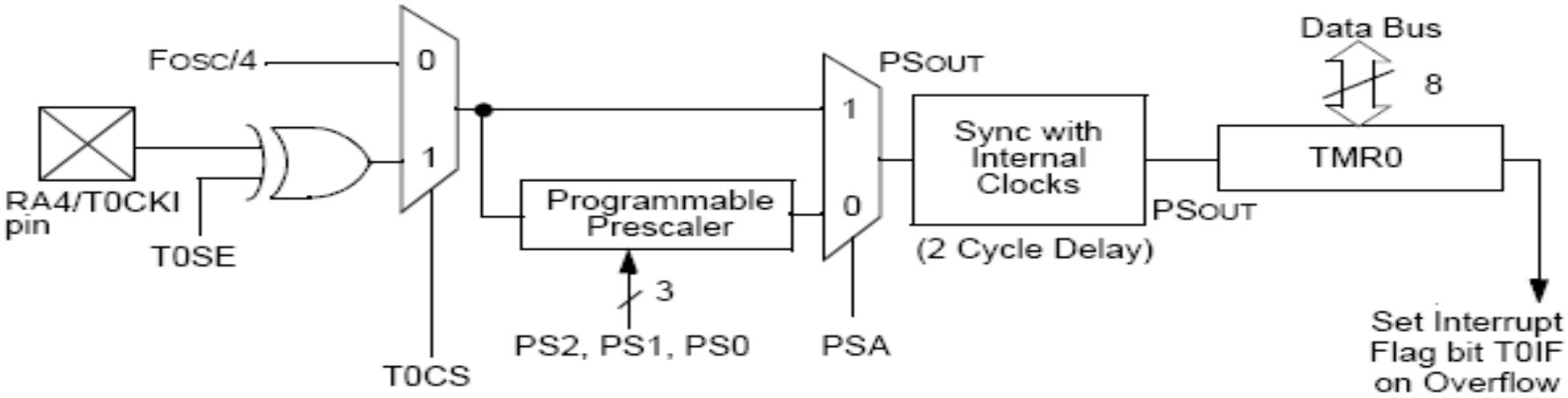
Note: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).

OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7	**$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
bit 6	**INTEDG**: Interrupt Edge Select bit
bit 5	**T0CS**: TMR0 Clock Source Select bit
bit 4	**T0SE**: TMR0 Source Edge Select bit
bit 3	**PSA**: Prescaler Assignment bit
bit 2-0	**PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128



Note: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).

INTCON Reg

7	6	5	4	3	2	1	0
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Bit 7: Global interrupt control
1: Enable all unmasked interrupts
0: Disable all interrupts

Bit 6: EEPROM write complete interrupt
1: Enable EEPROM interrupt
0: Disable EEPROM interrupt

Bit 5: TMR0 overflow interrupt
1: Enable TMR0 interrupt
0: Disable TMR0 interrupt

Bit 4: INT external interrupt control
1: Enable INT External interrupt
0: Disable INT External Interrupt

Bit 3: RB4–RB7 port change interrupt control
1: Enable RB4–RB7 port change interrupt
0: Disable RB4–RB7 port change interrupt

Bit 1: INT interrupt flag
1: INT interrupt occurred
0: INT interrupt did not occur

Bit 0: RB4–RB7 port change interrupt flag
1: One or more of RB4–RB7 pins changed state
0: None of RB4–RB7 pins changed state

Timer0 Interrupt

- The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>).
- The interrupt can be masked by clearing bit T0IE (INTCON<5>).
- Bit T0IF must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt.
- The TMR0 interrupt cannot awaken the processor from SLEEP since the timer is shut-off during SLEEP.

REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01h	TMR0	Timer0 Module Register							
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
85h	TRISA	—	—	—	PORTA Data Direction Register				

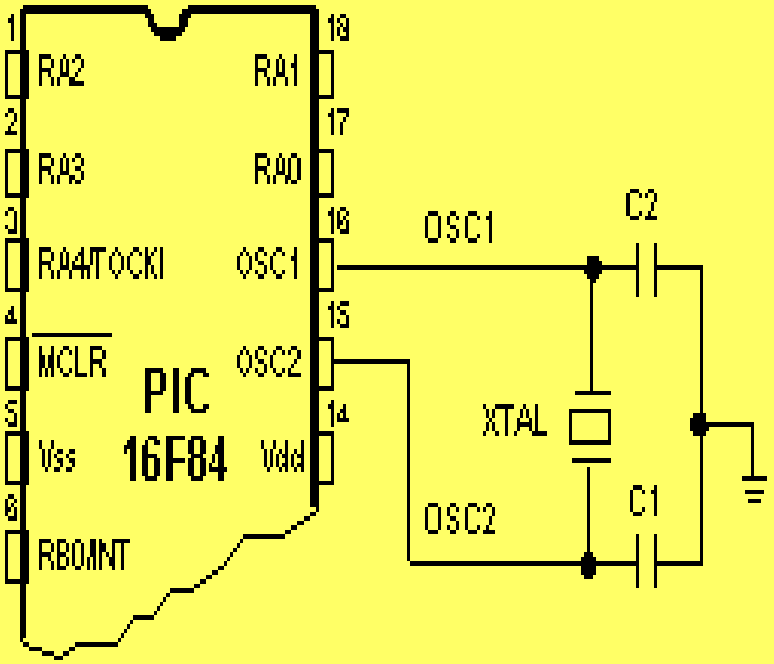
Design Examples: Using Timer/Counter in Hardware design of real-time mechatronics systems.

Clock generator - oscillator

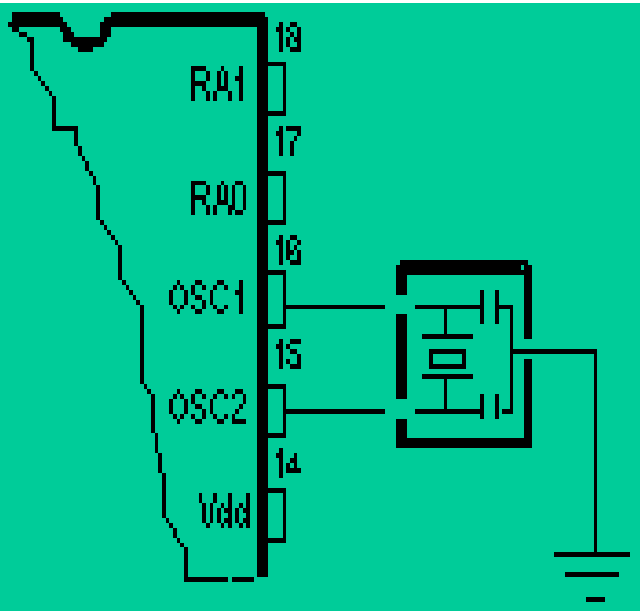
Oscillator circuit is used for providing a MC with a clock.

Types of oscillators:

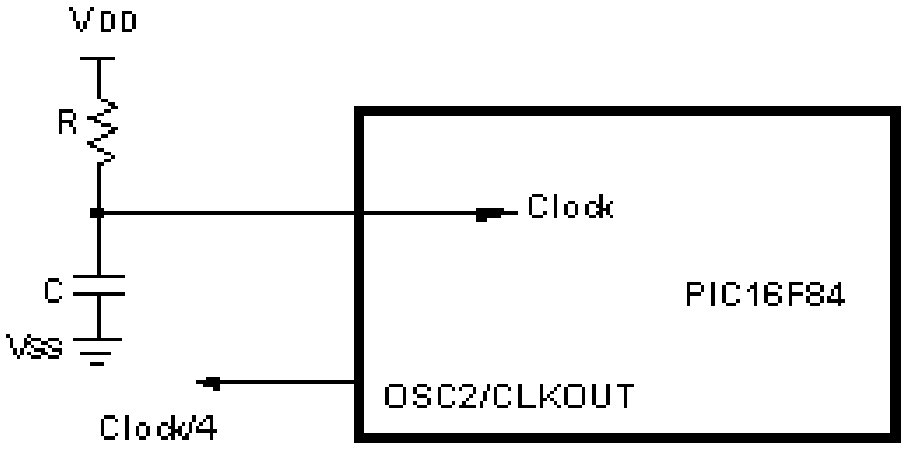
- PIC16F84 can work with four different configurations of an oscillator.



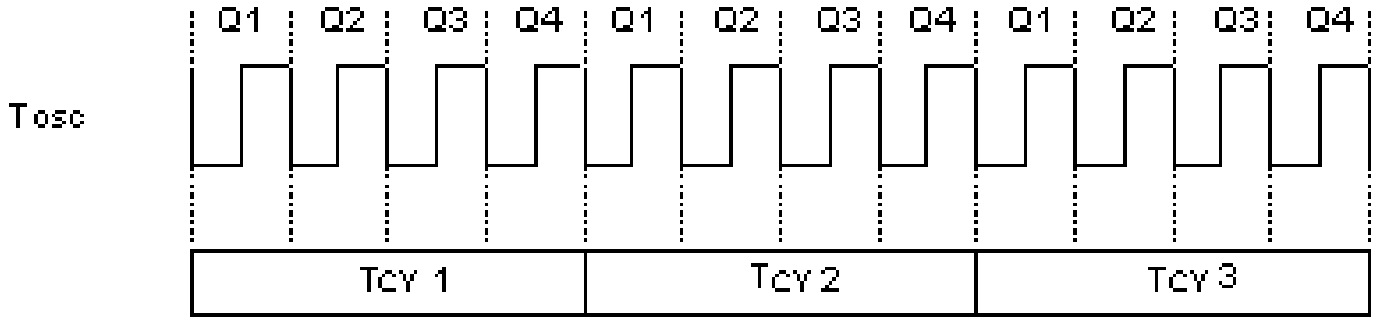
Connecting the quartz oscillator to give clock to a microcontroller



Connecting a resonator onto a microcontroller



Note: This pin can be configured as input/output pin

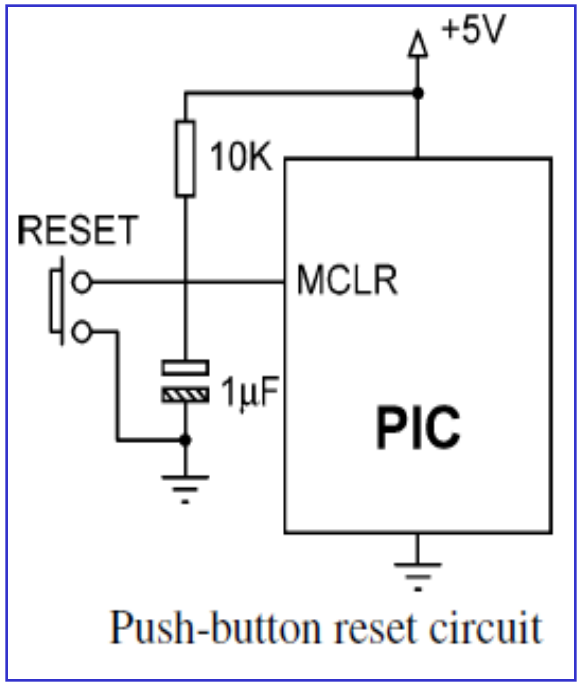


Relationship between a clock and a number of instruction cycles

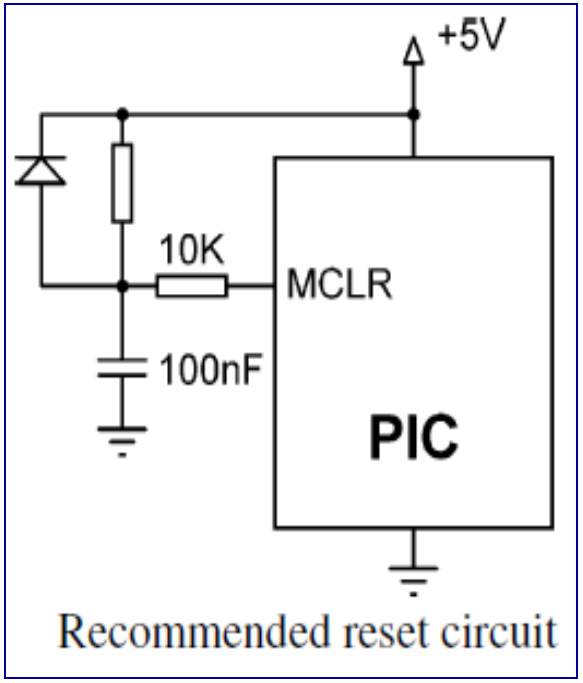
Reset MC:

Microcontroller PIC16F84 knows several sources of resets:

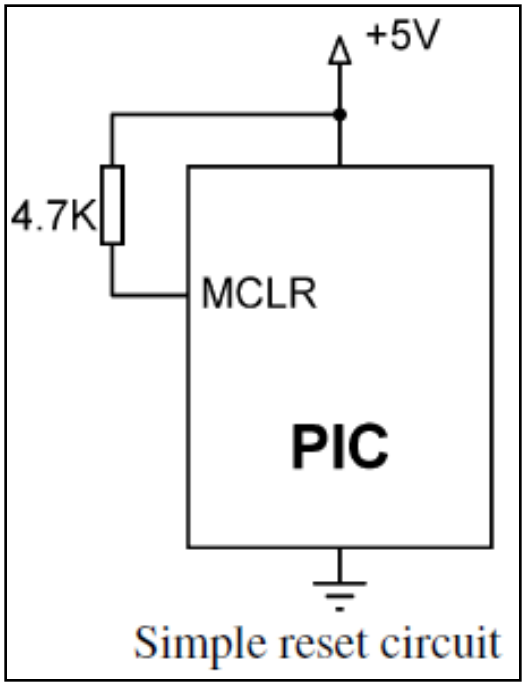
- Reset during power on, POR (Power-On Reset)
- Reset during regular work by bringing logical zero to MCLR microcontroller's pin.
- Reset during SLEEP regime.
- Reset at watchdog timer (WDT) overflow.
- Reset during at WDT overflow during SLEEP work regime.



Push-button reset circuit



Recommended reset circuit



Simple reset circuit

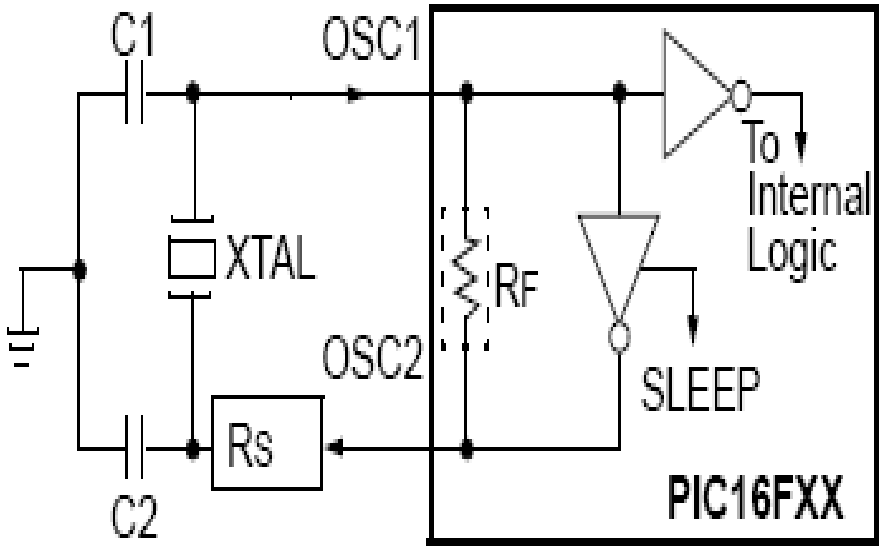
Oscillator Configurations

OSCILLATOR TYPES:

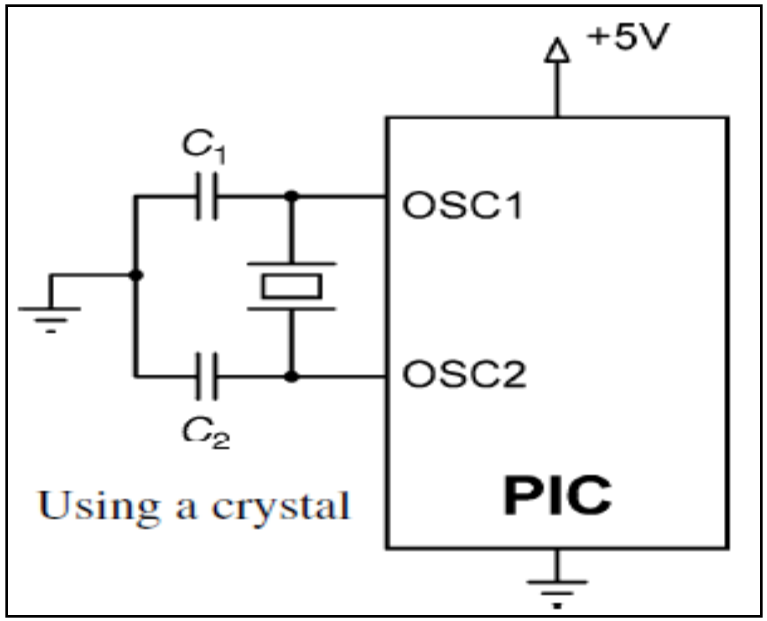
- The PIC16F84A can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1 and FOSC0) to select one of these four modes:
    - LP Low Power Crystal
    - XT Crystal/Resonator
    - HS High Speed Crystal/Resonator
    - RC Resistor/Capacitor

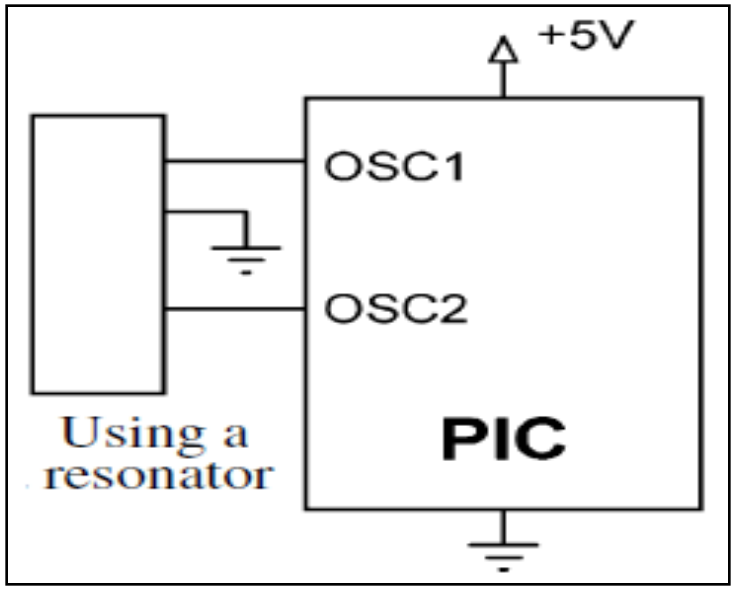
CRYSTAL OSCILLATOR/ CERAMIC RESONATORS

In XT, LP, or HS modes, a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation



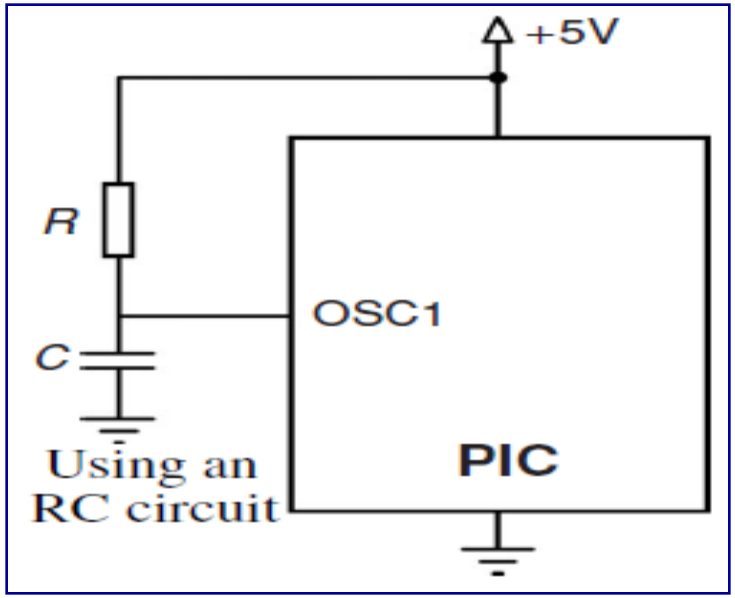
Note: A series resistor (Rs) may be required for AT strip cut crystals.





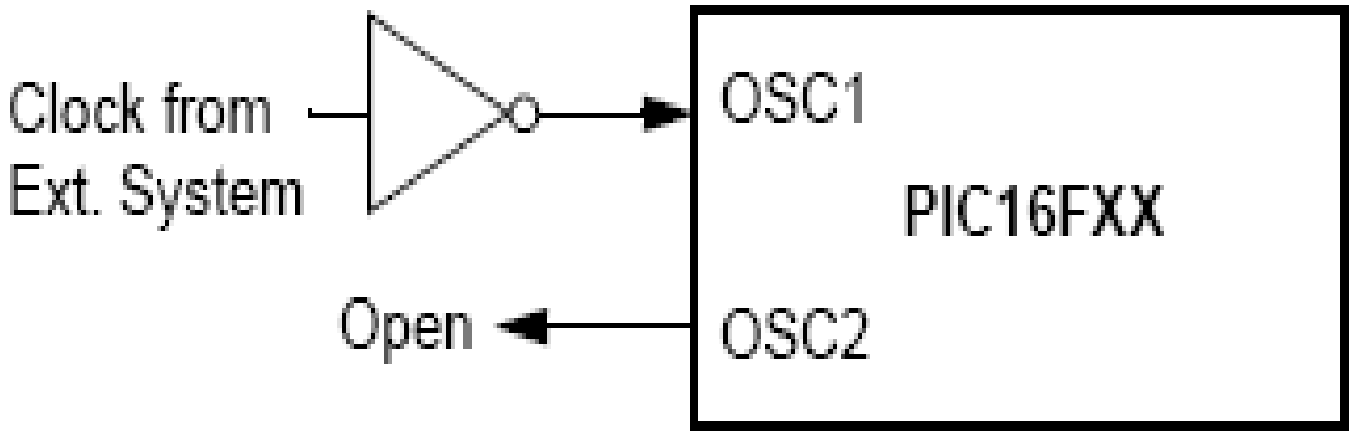
Capacitor values for crystal operation

Mode	Frequency	C1, C2 (pF)
LP	32 kHz	68–100
LP	200 kHz	15–33
XT	100 kHz	100–150
XT	2 MHz	15–33
XT	4 MHz	15–33
HS	4 MHz	15–33
HS	10 MHz	15–33



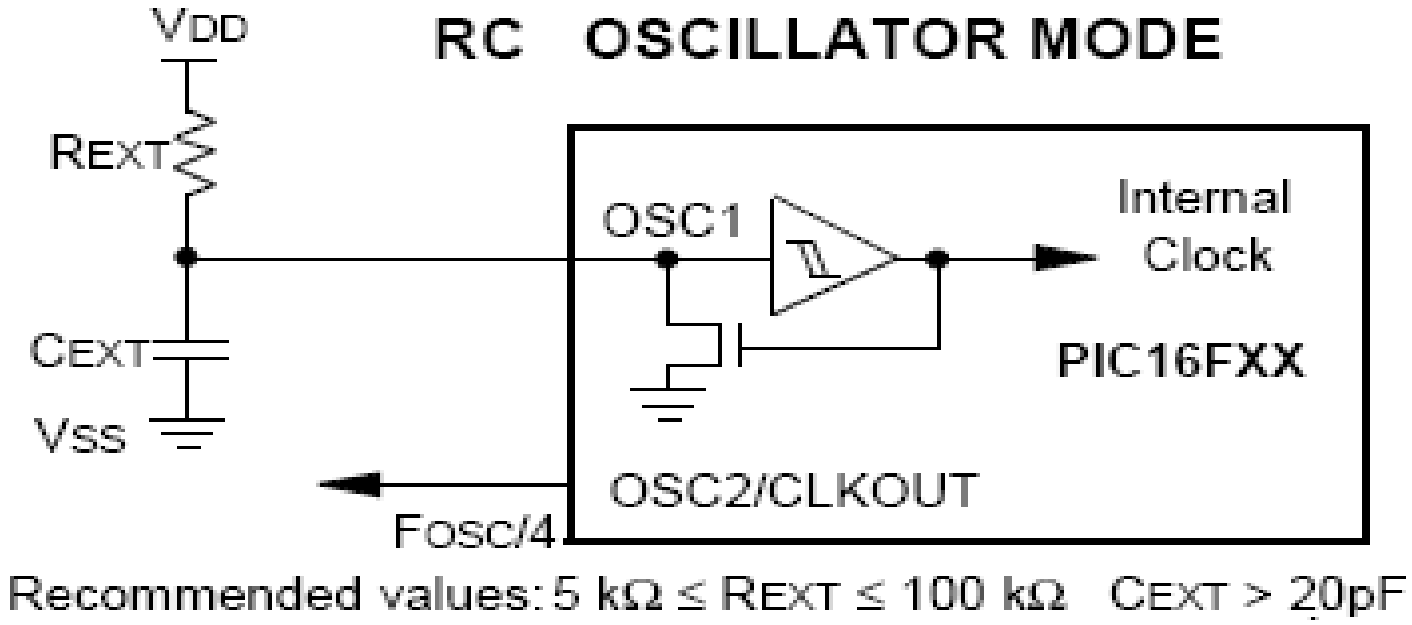
External Clock:

- The PIC16F84A oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications. When in XT, LP, or HS modes, the device can have an external clock source to drive the OSC1/CLKIN pin



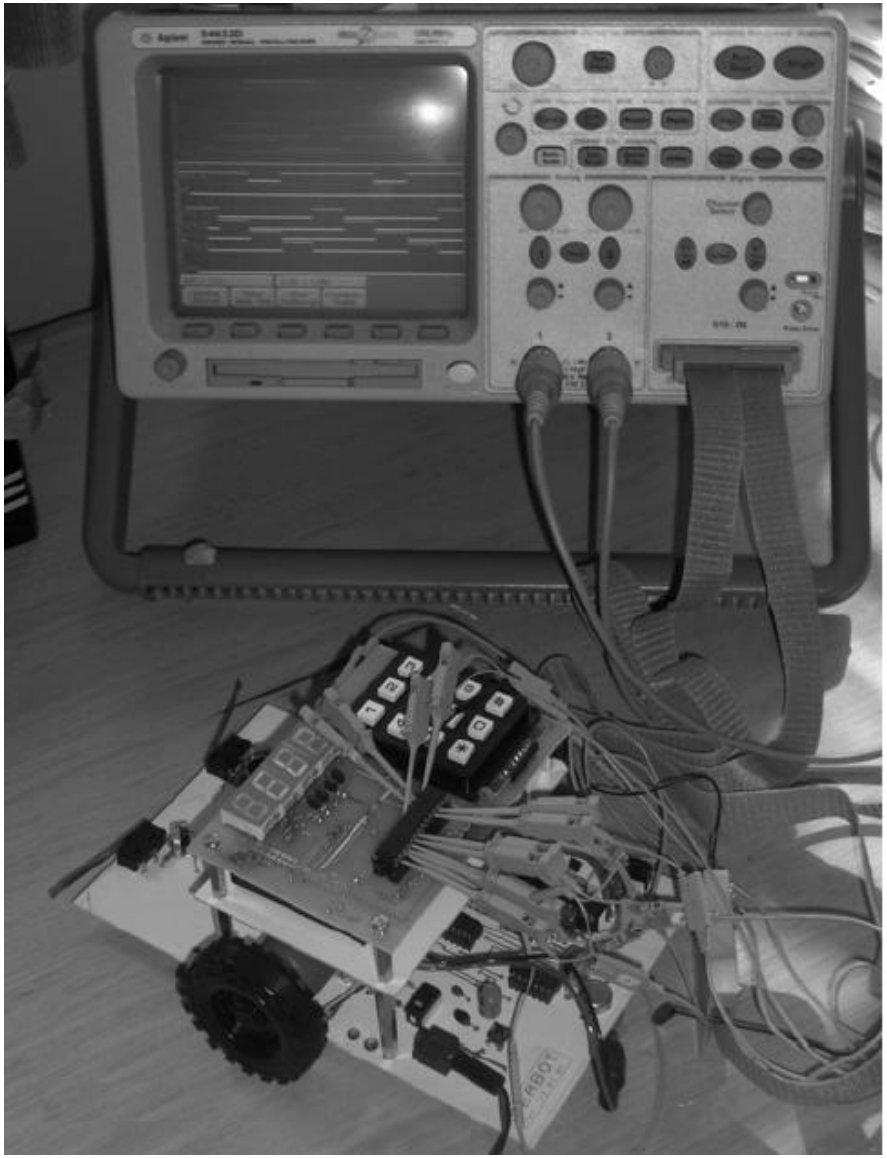
RC Oscillator:

- The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) values, capacitor (CEXT) values, and the operating temperature.
- The oscillator frequency will vary from unit to unit due to normal process parameter variation.
- The difference in lead frame capacitance between package types also affects the oscillation frequency, especially for low CEXT values.
- The user needs to take into account variation, due to tolerance of the external R and C components.



Process Related Interface:

- Instruments and actuators connected to the plant can take a wide variety of forms; they may be used for measuring a variable, they could be used to control an actuator.
- There is a need to convert a digital quantity to a physical quantity, or an analog signal generated from a sensor into a digital quantity.
- Most devices can be allocated to one of the following categories;
  1. Digital quantities.
  2. Analog quantities.
  3. Pulses and pulse rates.
  4. Telemetry.



SENSORS USED IN RT SYSTEMS:

- A sensor is a device that outputs a signal which is related to the measurement of a physical quantity such as temperature, speed, force, pressure, displacement, acceleration, torque, flow, light or sound.
- Sensors are used in RT systems in the feedback loops, and they provide information about the actual output of a plant. For example, a speed sensor gives a signal proportional to the speed of a motor.
- Sensors can be classified as analog or digital;
  - **Analog sensors** are more widely available, and their outputs are analog voltages. For example, the output of an analog temperature sensor may be a voltage proportional to the measured temperature. Analog sensors can only be connected to a computer by using an A/D converter.
  - **Digital sensors** are not very common and they have logic level outputs which can directly be connected to a computer input port.
- The choice of a sensor for a particular application depends on many factors such as the cost, reliability, required accuracy, resolution, range and linearity of the sensor.

The choice of a sensor:

- **Range:** The range of a sensor specifies the upper and lower limits of the measured variable for which a measurement can be made. For example, if the range of a temperature sensor is specified as 10–60 ◦C then the sensor should only be used to measure temperatures within that range.
- **Resolution:** The resolution of a sensor is specified as the largest change in measured value that will not result in a change in the sensor's output, i.e. the measured value can change by the amount quoted by the resolution before this change can be detected by the sensor. In general, the smaller this amount the better the sensor is, and sensors with a wide range have less resolution. For example, a temperature sensor with a resolution of 0.001K is better than a sensor with a resolution of 0.1 K.
- **Repeatability:** The repeatability of a sensor is the variation of output values that can be expected when the sensor measures the same physical quantity several times. For example, if the voltage across a resistor is measured at the same time several times we may get slightly different results.
- **Linearity:** An ideal sensor is expected to have a linear transfer function, i.e. the sensor output is expected to be exactly proportional to the measured value. However, in practice all sensors exhibit some amount of nonlinearity depending upon the manufacturing tolerances and the measurement conditions.
- **Dynamic response:** The dynamic response of a sensor specifies the limits of the sensor characteristics when the sensor is subject to a sinusoidal frequency change. For example, the dynamic response of a microphone may be expressed in terms of the 3-dB bandwidth of its frequency response.

Analog Input/Output Interfacing:

Some properties of analog and digital quantities

Property	Analog	Digital
Means of (electrical) representation	A continuously variable voltage, or current, represents the variable.	Variable is represented by a binary number.
Precision of representation	Can take infinite range of values; absolute precision is theoretically possible, as long as signal is kept completely uncorrupted.	Only a fixed number of digit combinations are available to represent measure; for example, an 8-bit number has only 256 different combinations. 'Continuously variable' quality of analog signal cannot be replicated.
Resistance to signal degradation	Almost inevitably suffers from drift, attenuation, distortion, interference. Cannot completely recover from these.	Digital representation is intrinsically tolerant of most forms of signal degradation. Error checking can also be introduced and with appropriate techniques complete recovery of a corrupted signal can be possible.
Processing	Analog signal processing using op amps and other circuits has reached sophisticated levels, but is ultimately limited in flexibility and always suffers from signal degradation.	Fantastically powerful computer-based techniques available.
Storage	Genuine analog storage for any length of time is almost impossible.	All major semiconductor memory technologies are digital.

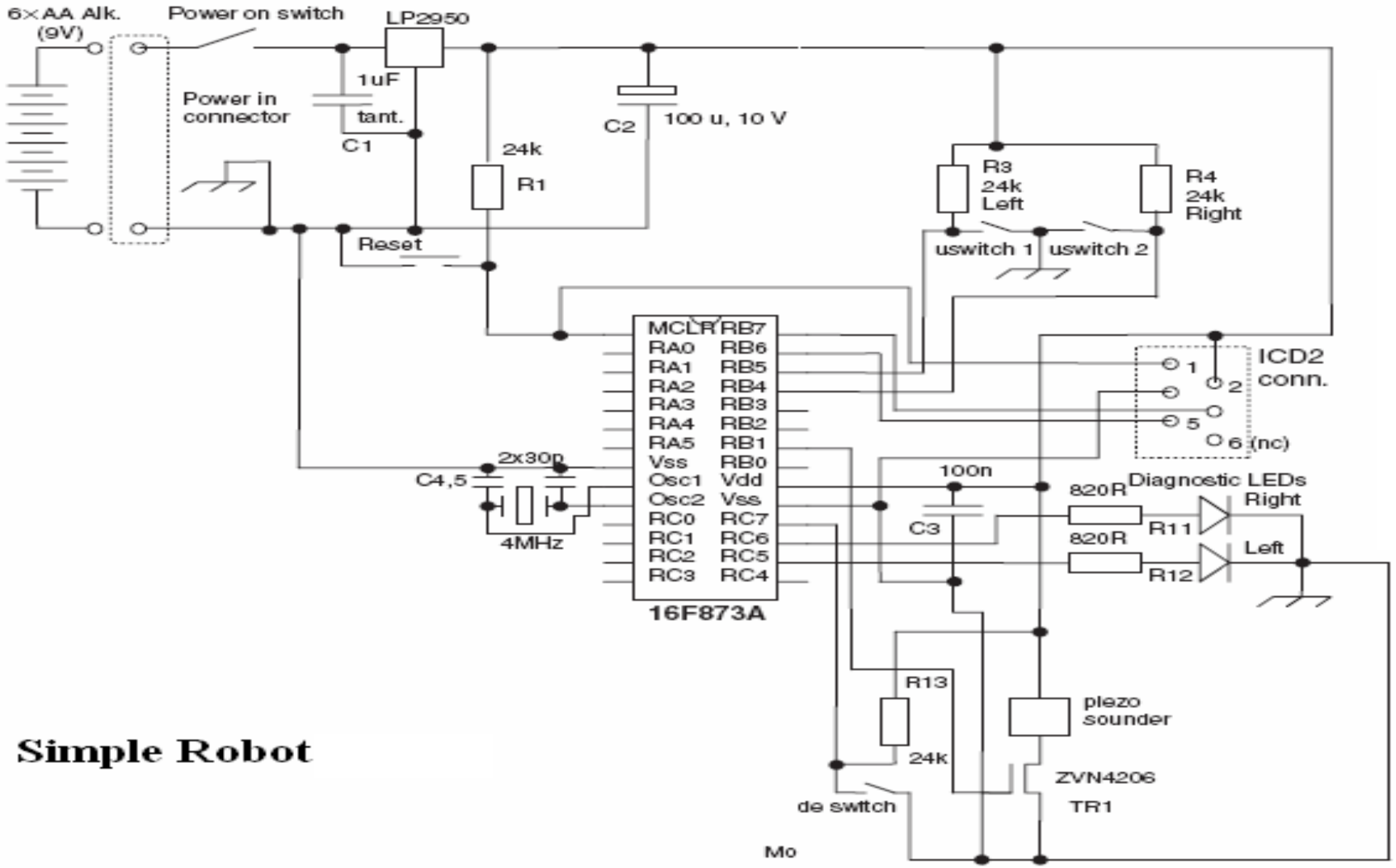
Pulse Input/Output Interfacing:

- Reading sequence of pulses generated from a sensor.
- Reading the width of a pulse width modulated signal.
- Generating number of pulses with fixed frequency.
- Generating a controllable pulse width modulated signal.

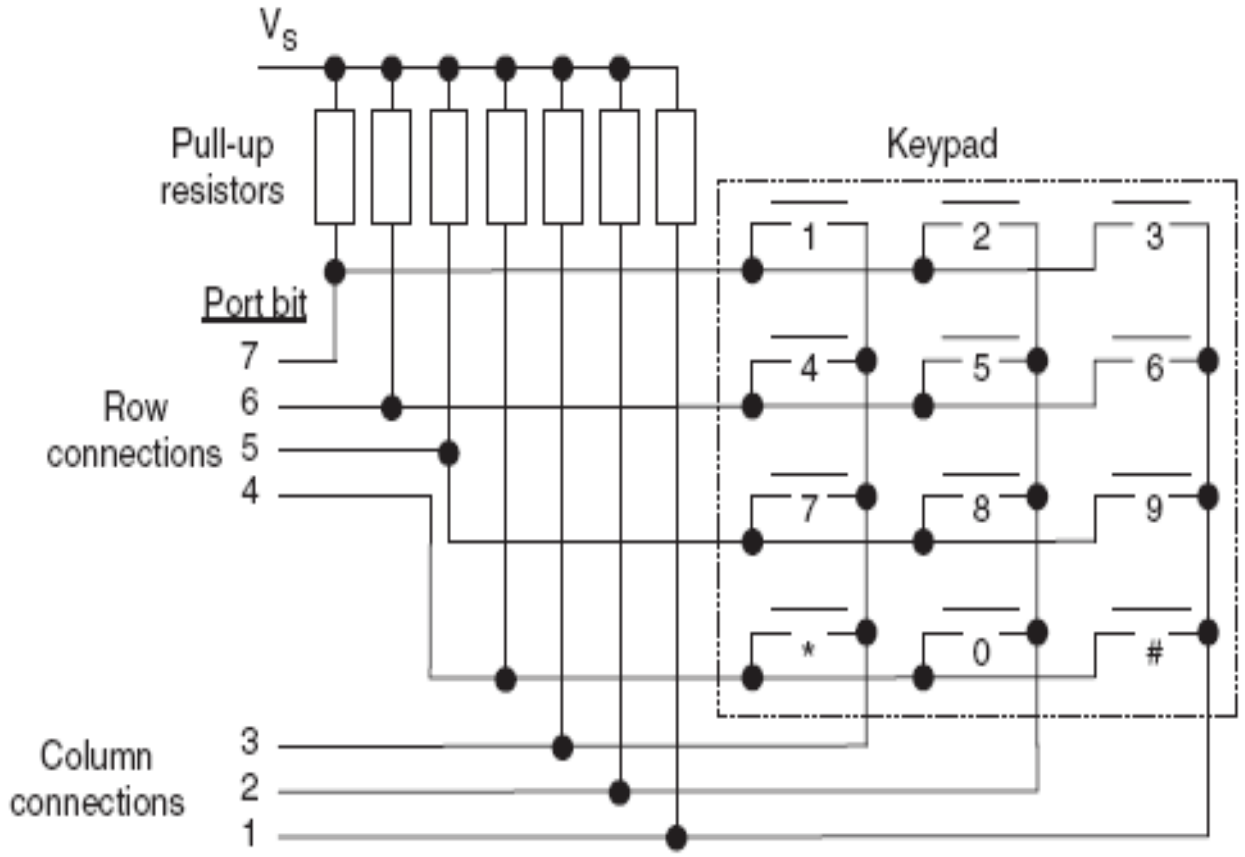
➢ Several design circuits will be considered during lecture.

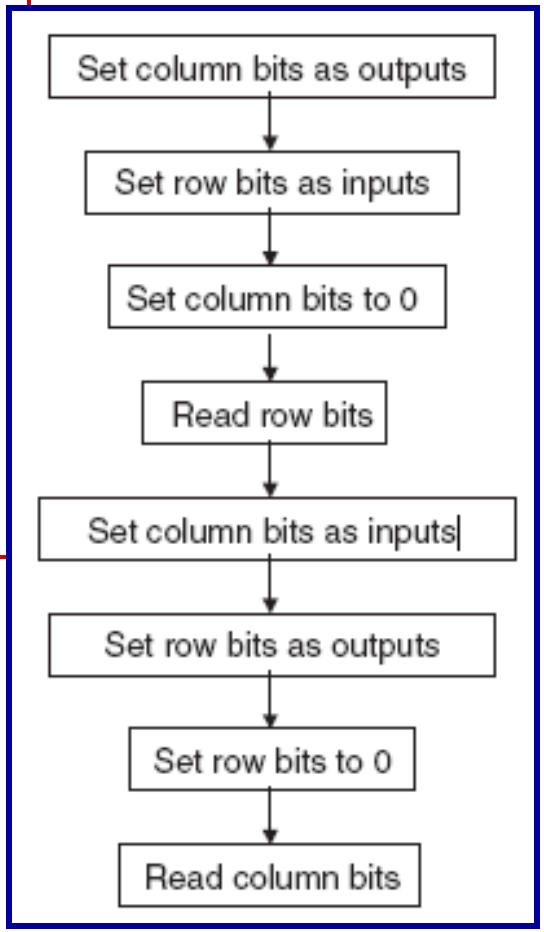
Example:

Simple Robot System: Hardware interfacing with an 8-bit microcontroller.



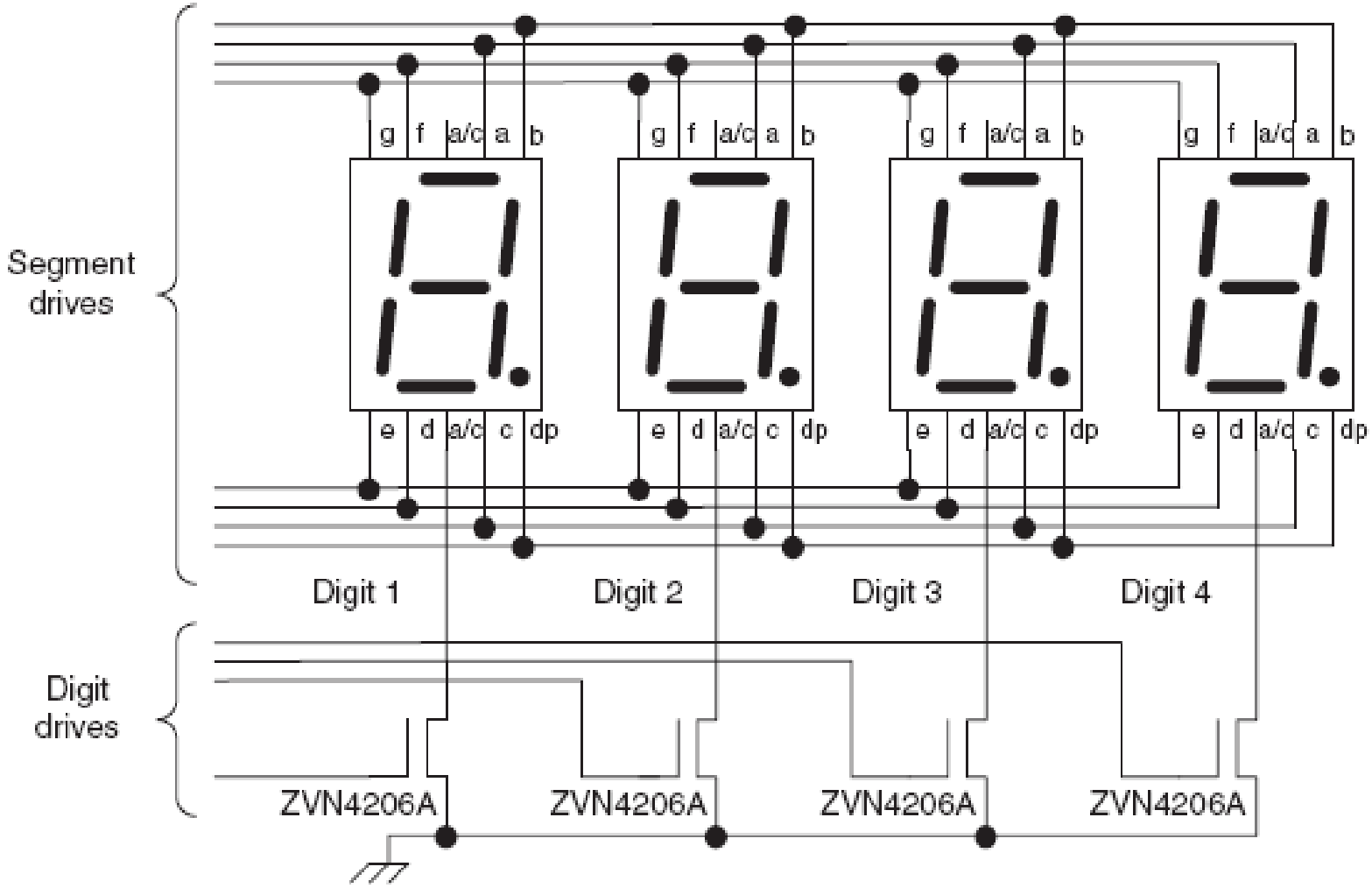
Keypad circuit diagram with pull-up resistor:





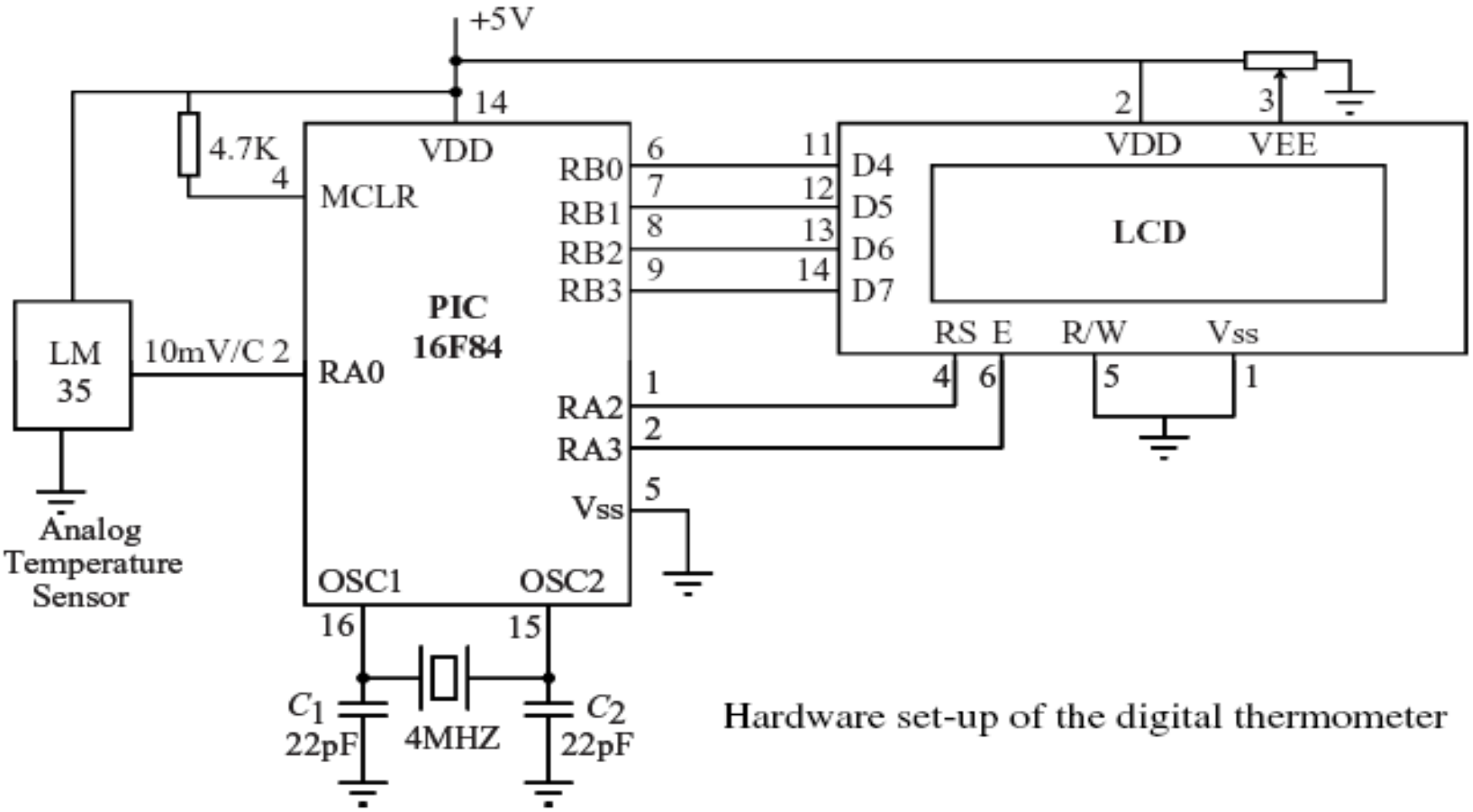
Example:

Four-digit display unit design.



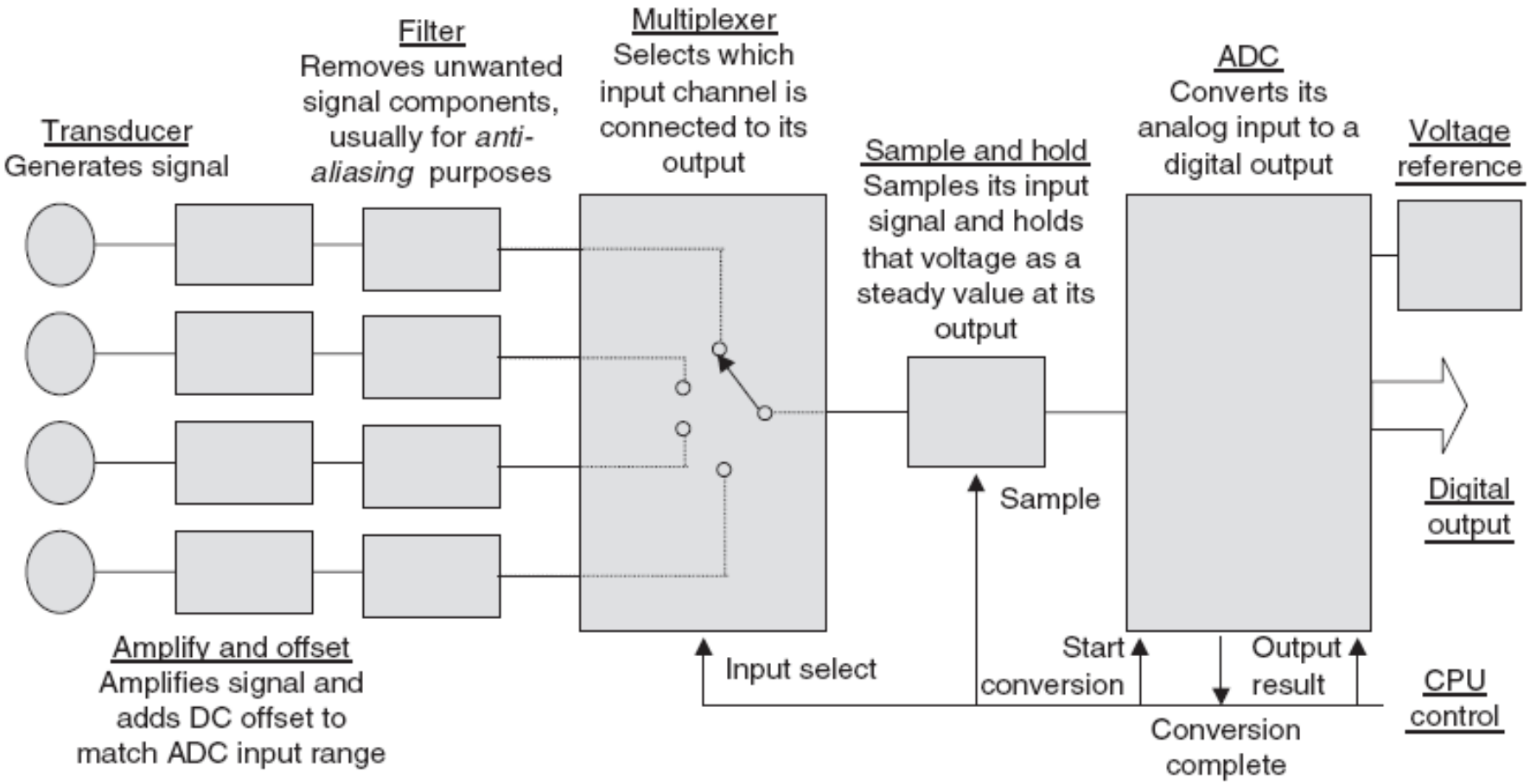
Example:

LCD interfacing with an 8-bit microcontroller.



Hardware set-up of the digital thermometer

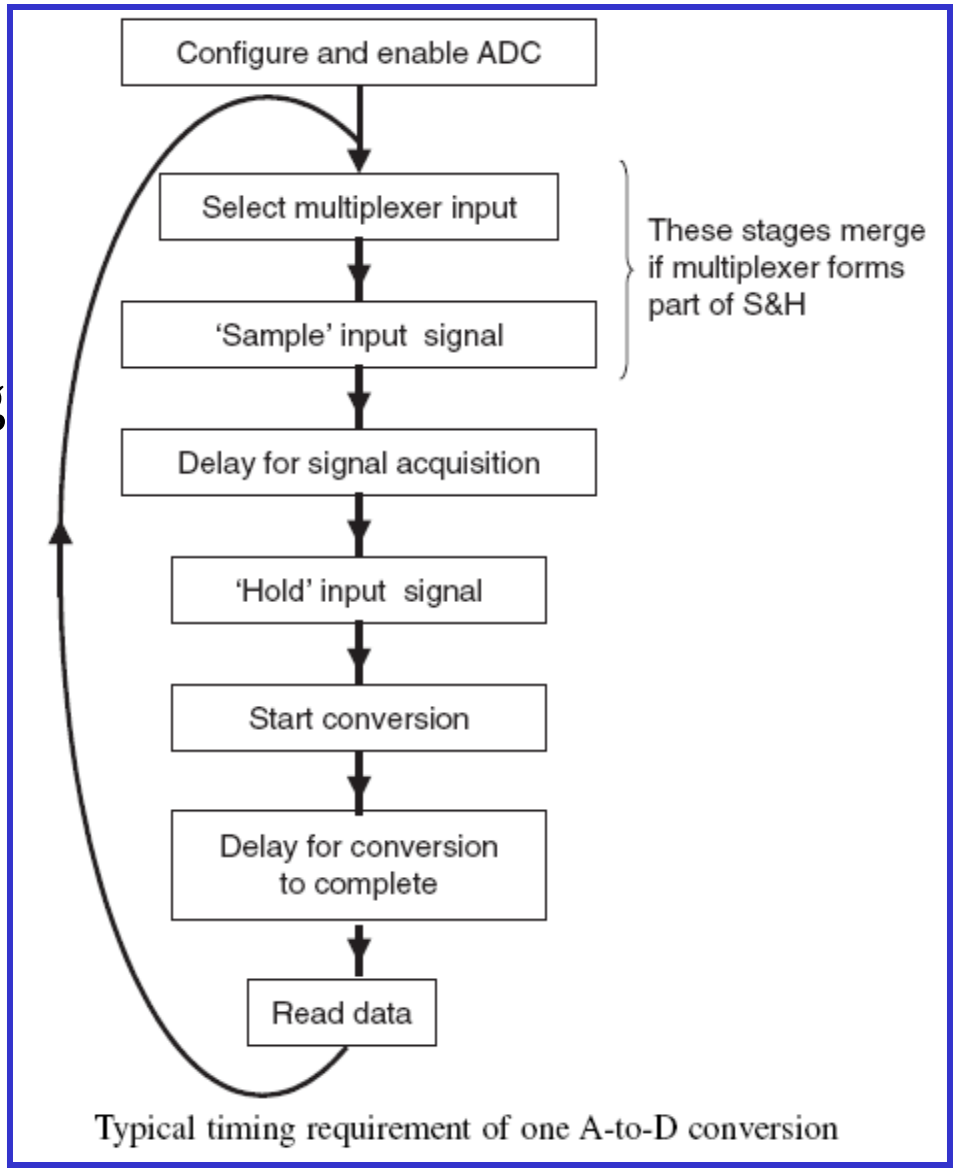
Data Acquisition System Design:



Elements of a (four-channel) data acquisition system

DAS: Software Design:

- Using flowchart.
- Writing an assembly program.
- Different sampling rates.
- Selecting the suitable sampling frequency.
- Task execution time.
- Dealing with interrupts.
- Memory management.



Typical timing requirement of one A-to-D conversion