INTEGRATED TECHNICAL EDUCATION CLUSTER
AT ALAMEERIA

E-626-A
Real-Time Embedded Systems (RTES)

Lecture #4
Parallel Ports, Power Supply & Clock Oscillator

Instructor:
Dr. Ahmad El-Banna

SPRING 2015

Agenda

- Parallel Input/Output
- Parallel Interface
- Connecting to the Parallel Port
- Clock Oscillator
- Power Supply
- Example

The main idea – parallel input/output

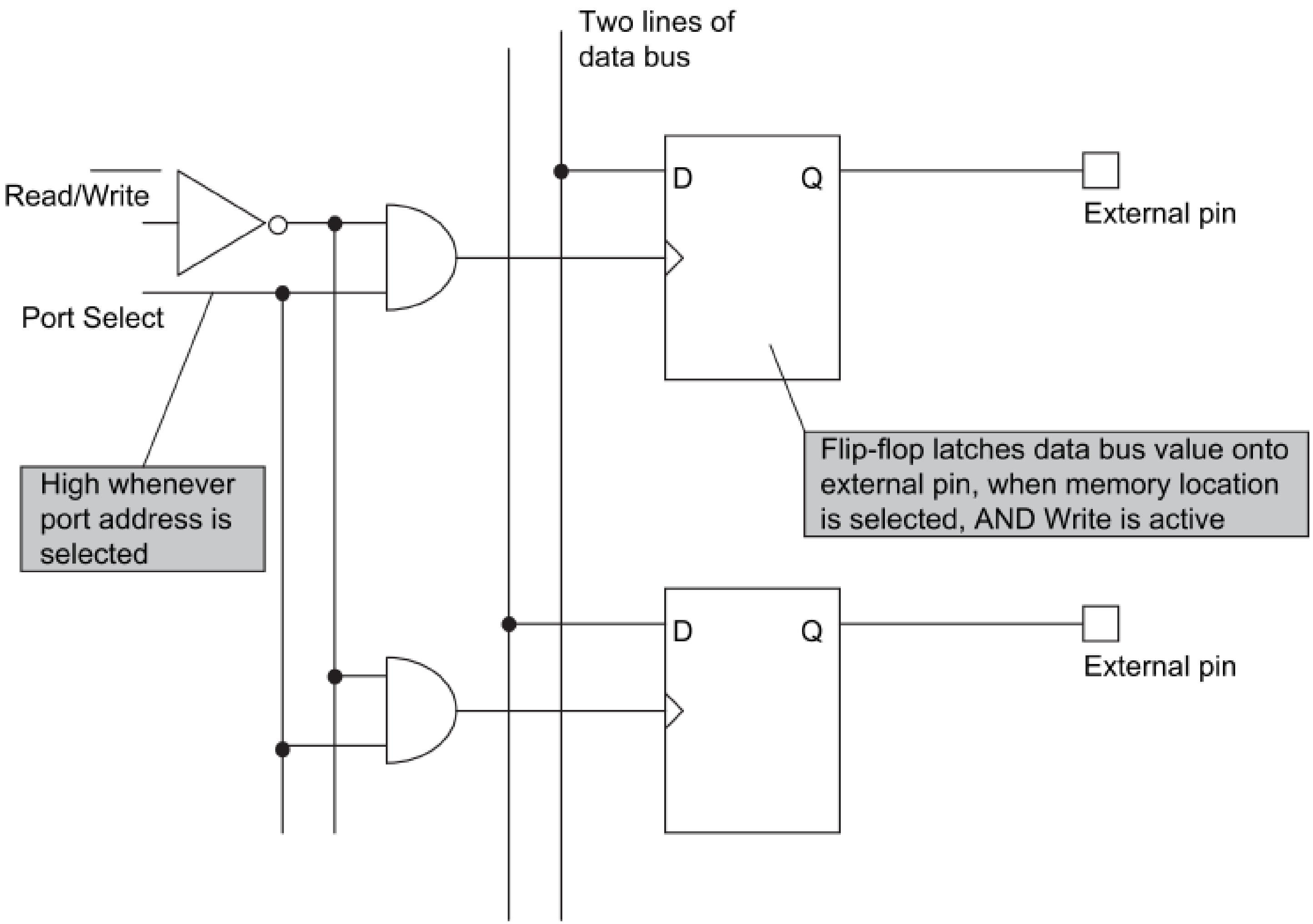
- Almost any embedded system needs to **transfer digital data** between its CPU and the outside world.
- This transfer falls into a number of **categories**, which can be summarized as:
  - **Direct user interface**, including switches, keypads, light-emitting diodes (LEDs) and displays.
  - **Input measurement information**, from external sensors, possibly being acquired through an analog-to-digital converter.
  - **Output control information**, for example to motors or other actuators.
  - **Bulk data transfer** to or from other systems or subsystems, moving in serial or parallel form, for example sending serial data to an external memory.

The technical challenge of parallel input/output

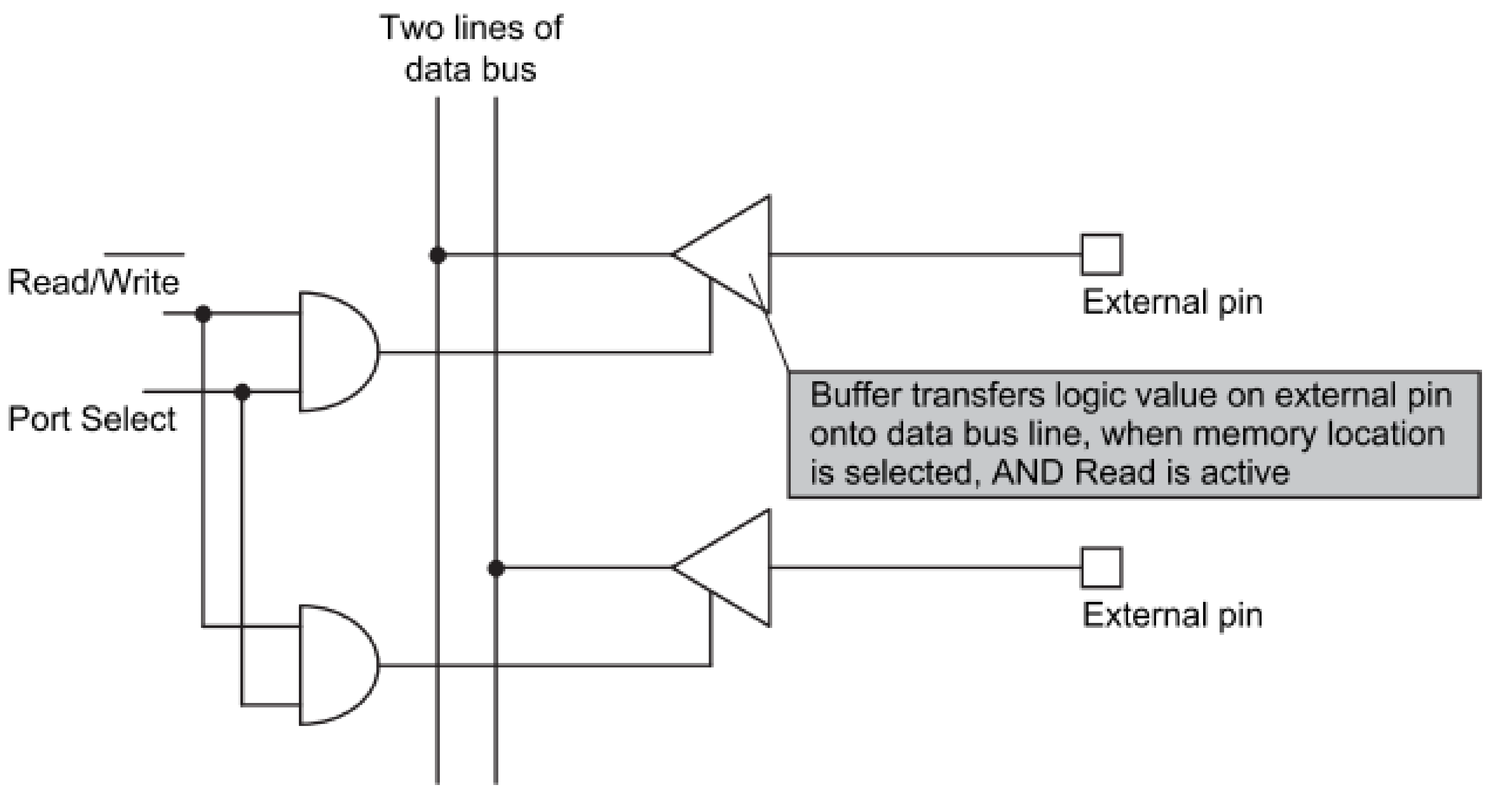
- Our immediate **challenge** is **how to provide the required interface** between the microcontroller data and address buses and the outside world.
- How can we **grab the data we want from the bus**, and transfer it to the outside world, via the parallel port?
- Alternatively, how can we **take external input data** and introduce it onto the data bus, at the right time and place, so that it gets to the right place within the microcontroller?
- Finally, given a port that can do these things, how can we make it really **flexible**, so that it can be used for **input**, or **output**, or a **mixture of both**, and can transfer a combination of data with possibly very different end uses?

Building a parallel interface

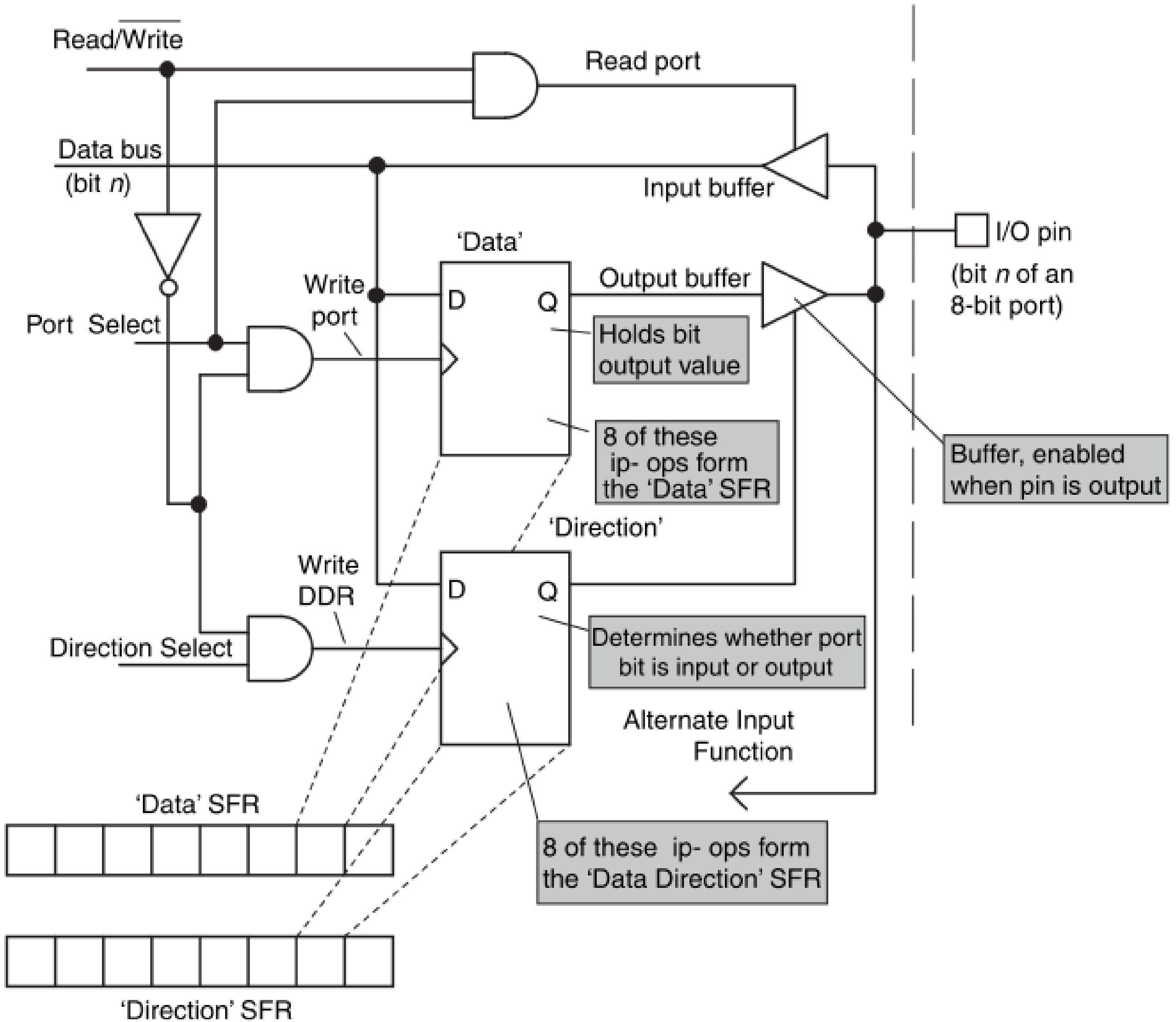
- **Two bits of a possible digital output port**



Two bits of a possible digital input port



A bi-directional port pin driver circuit



Connecting to the Parallel Port Switches & LEDS

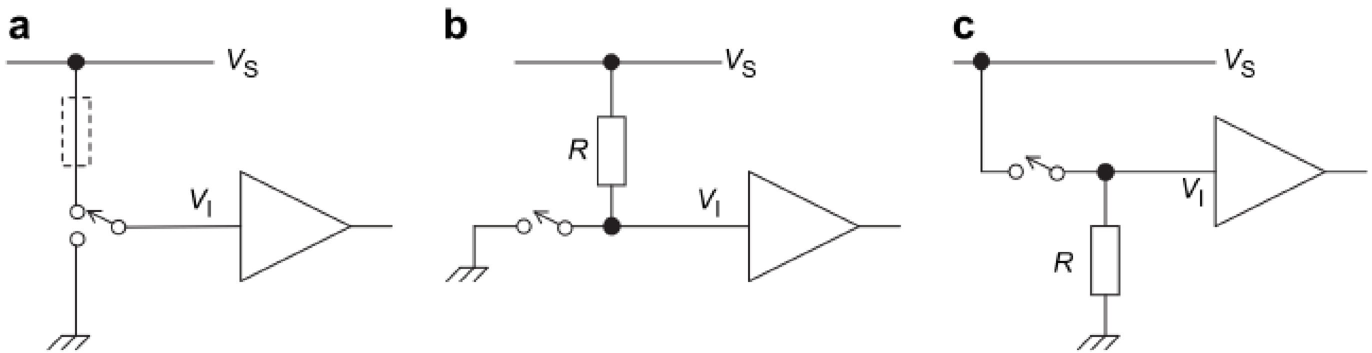


Figure 3.7: Connecting switches to logic inputs. (a) SPDT connection. (b) SPST with pull-up resistor. (c) SPST with pull-down resistor

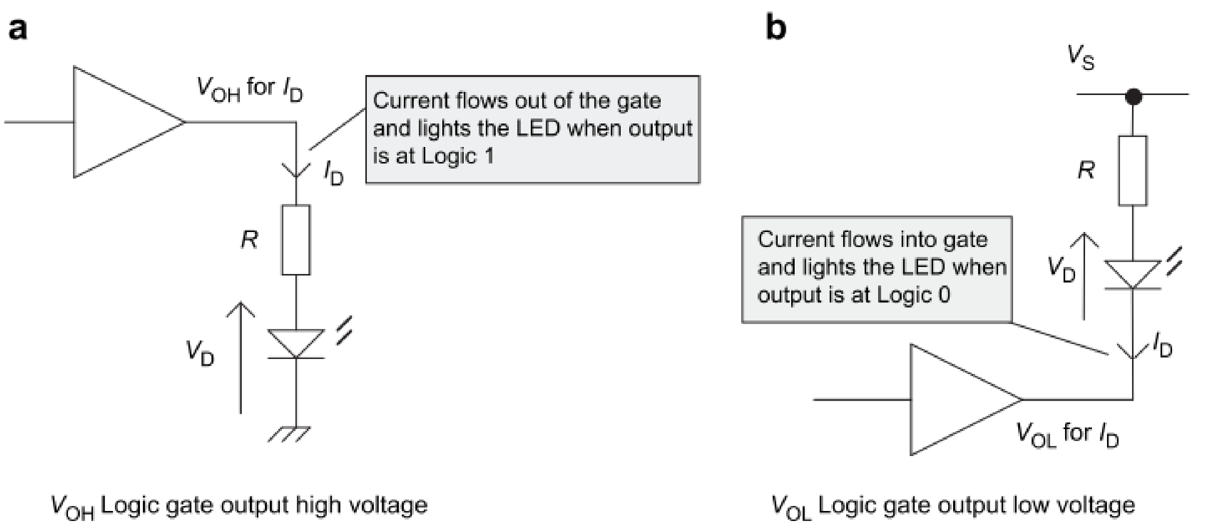


Figure 3.9: Driving LEDs from logic gates. (a) Gate output sourcing current to LED. (b) Gate output sinking current from LED

The clock oscillator

- **Types**

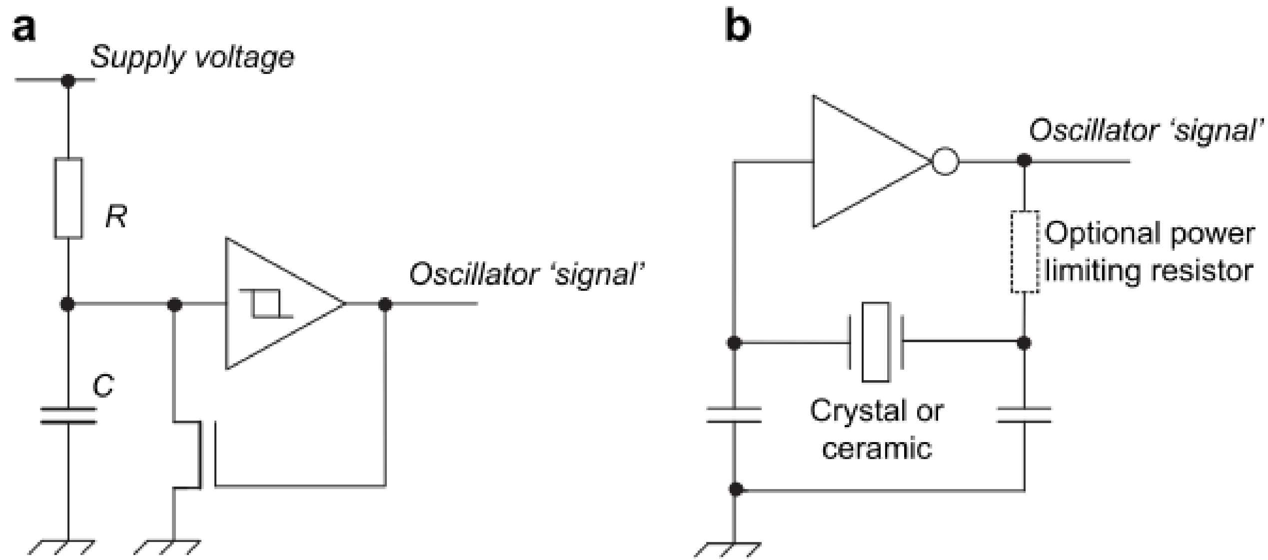


Figure 3.13: Microcontroller oscillator generator circuits. (a) Resistor–capacitor (RC). (b) Crystal or ceramic

- **Oscillator frequency** shows greater or lesser **dependence** on supply voltage, temperature, humidity, PCB layout and possibly other factors.
- **Crystals** in particular are **sensitive** to poor PCB layout.
- It is important to exclude parasitic resistance, capacitance or inductance by having very short PCB tracks, therefore **locating** the crystal **close to the body of the microcontroller**.

Power supply
The need for power, and its sources

- Like any electronic circuit, a microcontroller and the overall embedded system need to be supplied with **electrical power**.
- Traditionally, much logic circuitry is supplied at 5 V, arising from the voltage specified for the **TTL** family.
- With the growth in battery-powered equipment and developments in electronic technology, supply voltages have been **pushed down**, and 3.3 and 3.0 V supplies are now common.
- Operating conditions for electronic components are specified in the manufacturer's **data sheet**.
- In terms of power supply there are **two important issues**:
  - the **supply voltage** required and
  - the **current** that the device will then take from the supply.
- This supply current will be dependent on **operating frequency**. Also given are 'absolute maximum ratings', which give voltage and power dissipation levels beyond which the device must not be taken.

16F84A operating conditions

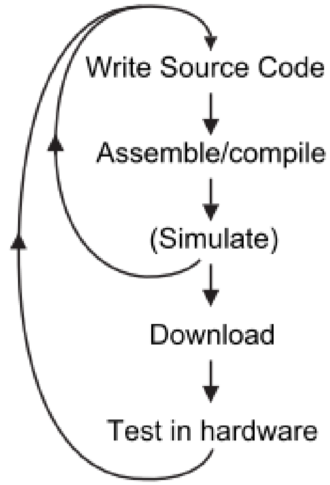
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
	V_{DD}	**Supply Voltage**					
D001		16LF84A	2.0	—	5.5	V	XT, RC, and LP osc configuration
D001 D001A		16F84A	4.0 4.5	— —	5.5 5.5	V V	XT, RC and LP osc configuration HS osc configuration
D002	V_{DR}	**RAM Data Retention Voltage (Note 1)**	1.5	—	—	V	Device in SLEEP mode
D003	V_{POR}	**V_{DD} Start Voltage** to ensure internal Power-on Reset signal	—	V_{SS}	—	V	See section on Power-on Reset for details
D004	S_{VDD}	**V_{DD} Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	
	I_{DD}	**Supply Current (Note 2)**					
D010		16LF84A	—	1	4	mA	RC and XT osc configuration **(Note 3)** F_{OSC} = 2.0 MHz, V_{DD} = 5.5V
D010		16F84A	—	1.8	4.5	mA	RC and XT osc configuration **(Note 3)** F_{OSC} = 4.0 MHz, V_{DD} = 5.5V
D010A			—	3	10	mA	RC and XT osc configuration **(Note 3)** F_{OSC} = 4.0 MHz, V_{DD} = 5.5V (During FLASH programming)
D013			—	10	20	mA	HS osc configuration (PIC16F84A-20) F_{OSC} = 20 MHz, V_{DD} = 5.5V
D014		16LF84A	—	15	45	μA	LP osc configuration F_{OSC} = 32 kHz, V_{DD} = 2.0V, WDT disabled

Note 1: This is the limit to which V_{DD} can be lowered without losing RAM data.
Note 2: Gives further information on factors that influence supply current.
Note 3: Gives guidance on how to calculate current consumed by the external RC network, when this is used.

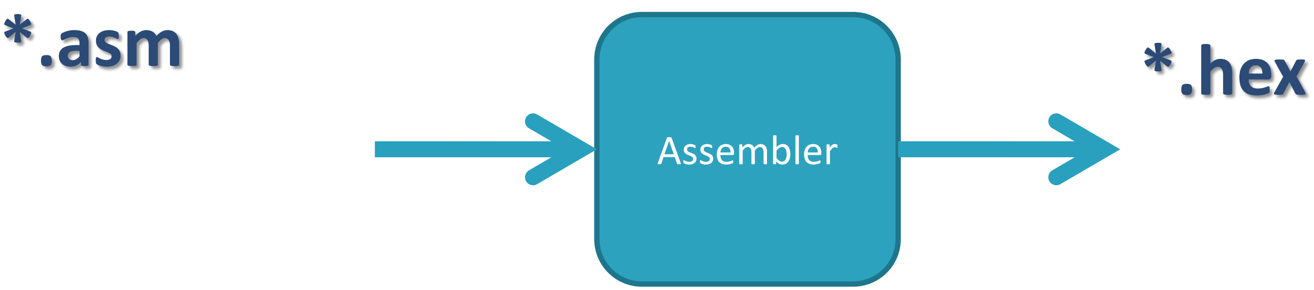
Figure 3.16: The 16F84A basic operating conditions

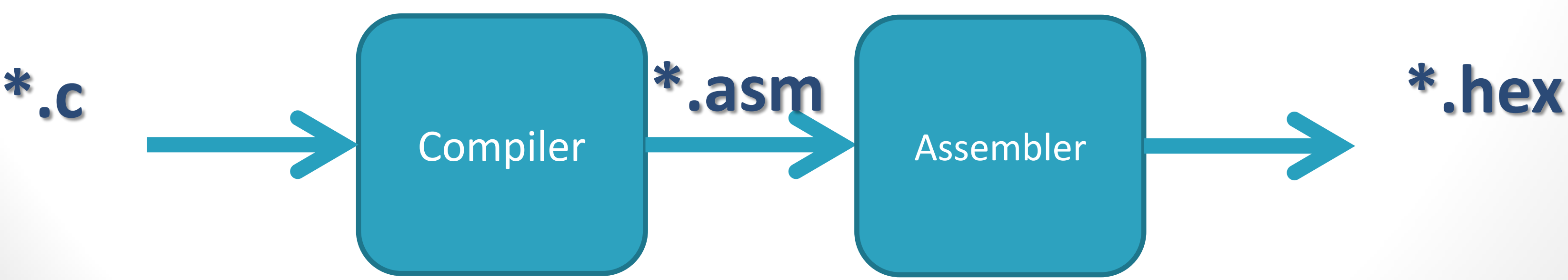
SIMPLE PROJECT

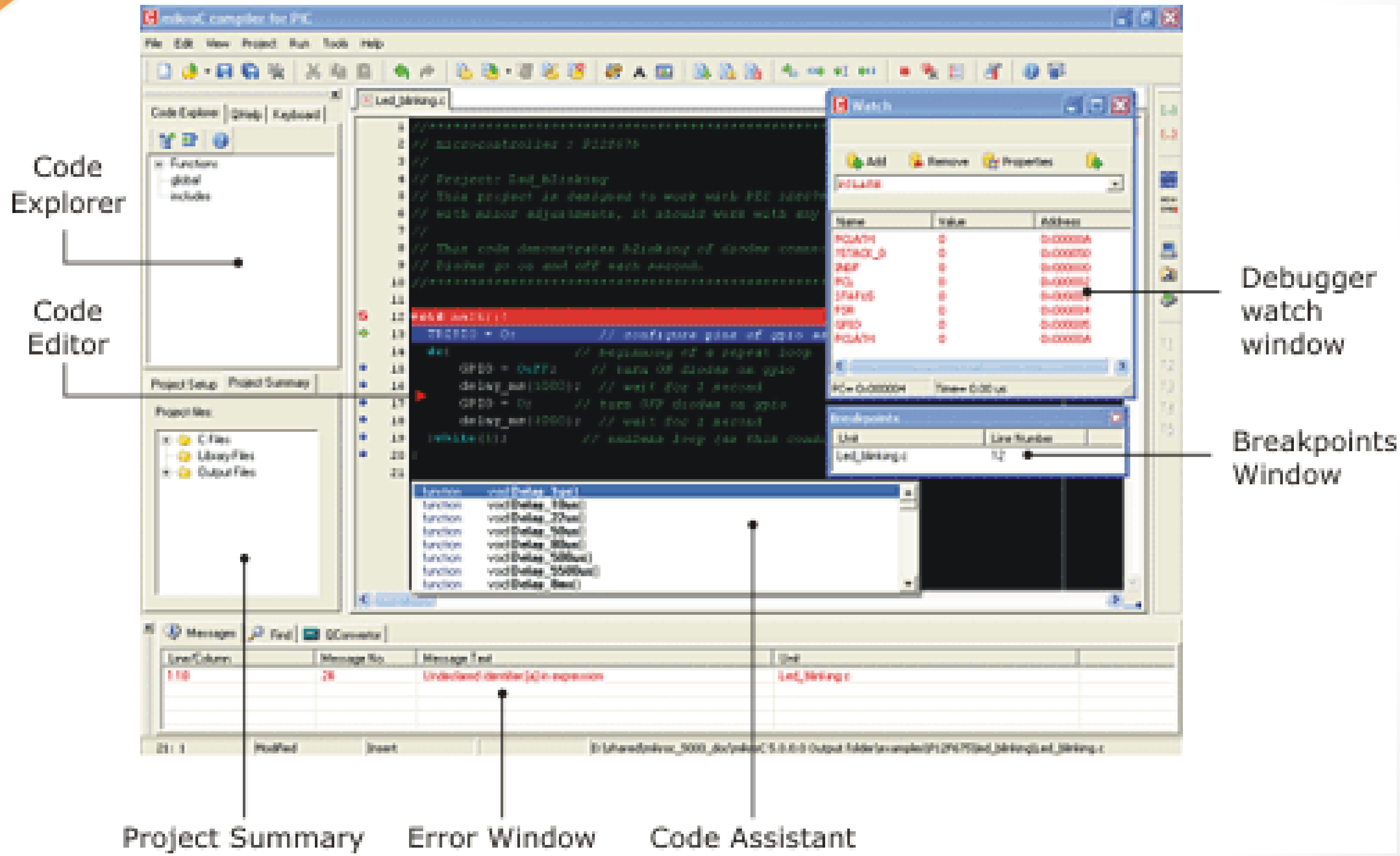
IDE



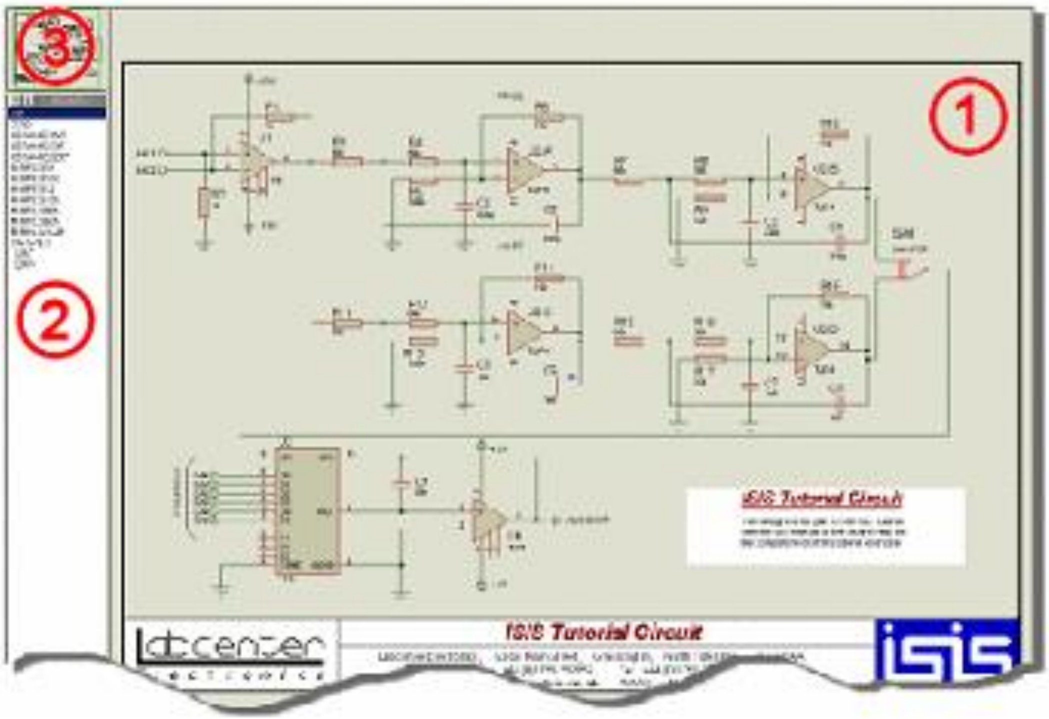
3: Developing a simple project






Code Explorer
Code Editor
Debugger watch window
Breakpoints Window
Project Summary
Error Window
Code Assistant

Proteus Design Suite

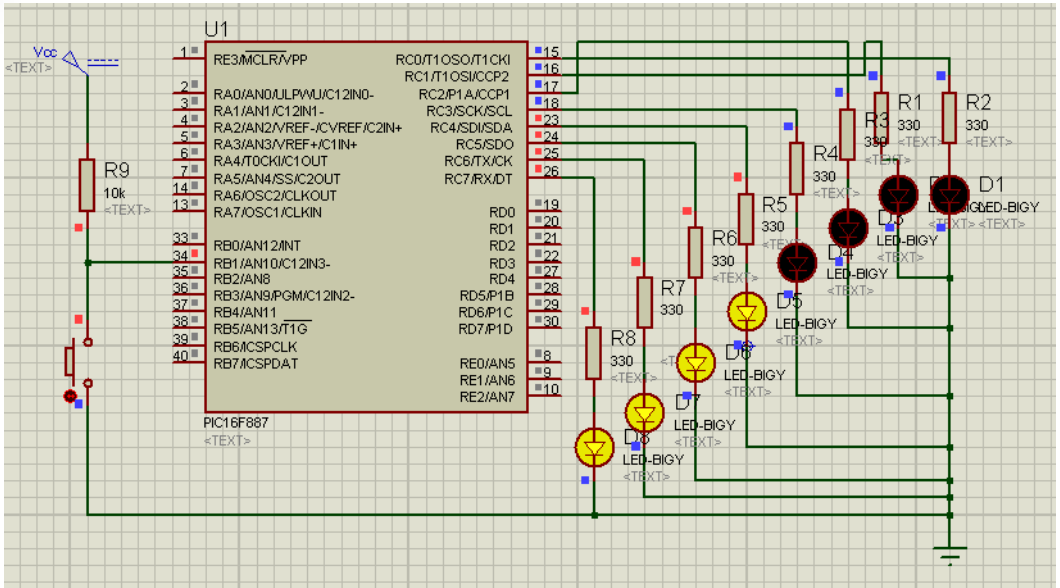


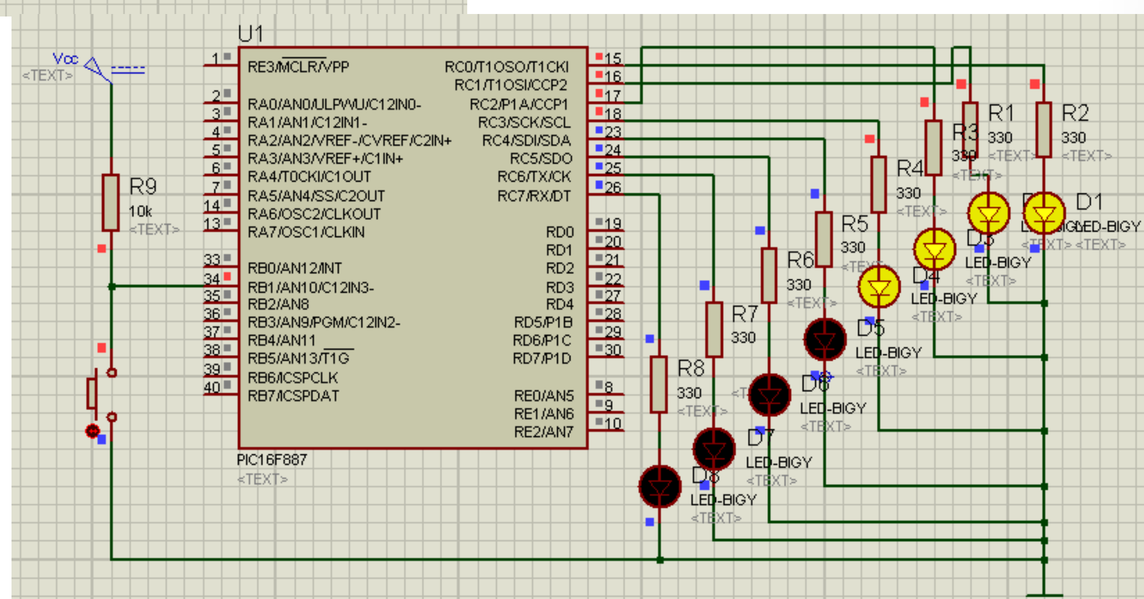
1. **Editing Window**
2. **Object Selector**
3. **Overview Window**

Simple Project: Toggle Up on Press

- Requirement analysis
  - Toggle leds on port c when push button on RB1 is pressed
- Design
  - Need a micro, crystal, resistors, leds and a push button with pull up resistor
- Developing
  - Write and run the code, and build the hardware
- Testing
  - Integrate the h/w and s/w and watch the operation
- Validations &Verification
  - Make sure that the final product matches exactly the requirements.

Circuit view





Pseudo code

- Start
- Configure port c direction as output
- Configure port B direction as input
- Initialize port value
- If key pressed, Toggle the port
- Delay to watch
- Loop infinitely

code

proj2_PB1

```
      MCU:              PIC16F887
      Dev.Board:        EasyPIC5
      Oscillator:       HS, 08.0000 MHz
      Ext. Modules:     -
      SW:               mikroC v8.0
 * NOTES:
      None.
*/



void main() {
  ANSEL  = 0;                                  // Configure AN pins as digital I/O
  ANSELH = 0;
  PORTC = 0xF0;                                   // initialize PORTC
  TRISC = 0;                                   // configure PORTC as output
  TRISB = 255;                                 //// configure PORTB as input

  do {
    if (PORTB.F1 == 0) {                           // if PB is pressed
      PORTC = ~ PORTC;
      Delay_ms(100);     // to ignore bouncing
    //Delay_ms(1000);    // indicate a suitable value
    }

  } while (1);
}
```

Your turn !

- Increment leds on every key press

- Go go ☺

- For more details, refer to:
  - Chapter 3, T. Wilmishurst, **Designing Embedded Systems with PIC Microcontrollers**, 2010.
- The lecture is available online at:
  - http://bu.edu.eg/staff/ahmad.elbanna-courses/12134
- For inquires, send to:
  - ahmad.elbanna@feng.bu.edu.eg