UMM AL-QURA UNIVERSITY

College of Computer and Information Systems

Computer Engineering Department

1403450

Microcomputer System Design

Lab Manual

Student Name: _____

Student ID: _____

Section: _____ Group: _____

Session (Fall / Spring / Summer) _____

This page is intentionally left blank

TABLE OF CONTENTS

Table of Contents i

Laboratory Safety Guidelines. ii

Experiment No. 1. Introduction to PIC18F452 microcontroller. 1

Experiment No. 2. Internal Architecture of PIC18F452 microcontroller 7

Experiment No. 3. To Introduce EEDT 6.0 multi-microcontroller kit 11

Experiment No. 4. Introduction to mikroC Pro for PIC microcontroller 15

Experiment No. 5. Program to Turn ON and OFF LEDs 22

Experiment No.6. Program to make Binary Counter on LEDs 24

Experiment No. 7. Program to Read from Button 26

Experiment No. 8. Interfacing Seven Segment Display with PIC18F452 28

Experiment No. 9. Interfacing Character LCD with PIC18F452 31

Experiment No. 10. Program to Run stepper motor using PIC microcontroller 34

Experiment No. 11. A-to-D Converter of PIC microcontroller 36

Experiment No. 12. Serial Communication using RS232 38

Prepared By:	Engr. Muhammad Yousuf Irfan Zia	2012
Revised By:	Engr. Muhammad Saqib	2013
Reviewed By:	_____	_____
Approved By:	_____	_____

Umm Al-Qura University

Computer Engineering Department

LABORATORY SAFETY GUIDELINES

A. General Laboratory Safety Rules

1. Personal Safety

- Be familiar with the electrical and fire hazards associated with your workplace.
- Be as careful for the safety of others as for yourself. Think before you act.
- Be tidy and systematic.
- Avoid bulky, loose or trailing clothes. Avoid long loose hair.
- No one is allowed to enter in the lab area bare foot due to increased risk of electric shock.
- Remove metal bracelets, rings or watchstraps when working in the laboratories.
- Avoid working with wet hands and clothing.

2. Food, Beverages and Smoking

- Due to the increased risk of electric shock, no drinking of beverages, consumption or storage of any kind of food is allowed in the laboratory.
- Smoking is prohibited in all laboratories in all timings.

3. Soldering

- No one is allowed to do soldering in any of the computer engineering laboratories except the graduation project design laboratory.
- Anyone doing soldering in the graduation project design laboratory must wear appropriate apparel, socks, gloves, covered shoes and safety goggles to prevent the possibility of severe burns resulting from the splashing or dripping of hot liquefied solder into the face and eyes or on to the exposed skin on the chest, hands, legs, and feet.
- Students who are not so properly attired for these tasks will NOT be allowed to perform any type of soldering in the graduation project design laboratory.

4. Laboratory Operating Hours

- Students are never allowed to work alone in any lab area other than scheduled laboratory operating hours unless either a Lab T/A or Course Instructor is present inside that lab area.
- The laboratory operating hours for students are posted on the entrance doorway and on the notice board of computer engineering department.

5. Power Supply Related Safety

- Voltages above **50-VAC** or **120-VDC** are always dangerous.
- Extra precautions should be considered as voltage levels are increased.
- Never make any changes to circuits or mechanical layout without first isolating the circuit by switching off and removing connections to power supplies.

6. Laboratory Equipment

- Lab equipment may not be removed from the Computer Engineering lab areas without the permission of the Laboratory Supervisors.
- Laboratory bench equipment (except for some lab bench computers) must be turned off before closing down the lab area for the day.
- Never open (remove cover) of any equipment in the laboratories.
- Never "jump," disable, bypass or otherwise disengage any safety device or feature of any equipment in the laboratories.
- Laboratories shall be locked when unoccupied.

7. Waste Management Safety

- Know the correct handling, storage and disposal procedures for batteries, cell, capacitors, inductors and other high energy-storage devices.

8. Equipment Safety

- Before equipment is energized ensure, circuit connections and layout have been checked by a Teaching Assistant (TA) and all colleagues in your group has given their consent.
- Experiments left unattended should be isolated from the power supplies. If for a special reason, it must be left on, a barrier and a warning notice are required.
- Equipment found to be faulty in any way should be reported to the lab supervisor and taken out of service until inspected and declared safe.

9. Equipment Accessories

- Use extension cords only when necessary and only on a temporary basis.
- Request new outlets if your work requires equipment in an area without an outlet.
- Discard damaged cords, cords that become hot, or cords with exposed wiring.

B. Electrical and Fire Emergency Responses

1. Police, Fire or Medical Emergency

- Use the telephone located in the laboratory area and press **0-996** to notify police, fire, and ambulance for emergency help.
- Everyone present in the laboratory area shall be familiar with the locations and operation of safety and emergency equipment, including but not limited to, fire extinguishers, first aid kits, emergency power off system, fire alarm pull stations, and emergency telephones.

2. Electric Shock

- When someone suffers serious electrical shock, he may be knocked unconscious.
- If the victim is still in contact with electrical current, immediately turn off the electrical power source.
- If you cannot disconnect the power source, depress the Emergency Power Off switch.
- Do not touch a victim that is still in contact with a live power source; you could be electrocuted! Have someone call for emergency medical assistance immediately. Administer first-aid, as appropriate.

3. Electrical Fire

- If an electrical fire occurs, try to disconnect the electrical power source, if possible.
- If the fire is small and you are not in immediate danger; and if you have been properly trained in fighting fires, use the correct type of fire extinguisher to extinguish the fire.
- When in doubt, push in the Emergency Power Off button.
- **NEVER use water to extinguish an electrical fire**.

4. Emergency Power Off

- Every lab is equipped with an Emergency Power off System.
- When this switch is depressed, electrical power to the lab will shut off, except for lights.
- Only authorized personnel are permitted to reset power once the Emergency Power Off system has been engaged.

5. Building Evacuation in Emergency

- Everyone present in the laboratory should be familiar to emergency exits & way out plans.
- Use the nearest exit doorway from lab area closest to the stairwell to exit the building.
- Follow the Emergency Exit Signs posted in the hallways. Do not use elevators.
- Lab Teaching Assistants (T/As) or Instructor shall make sure all persons are out of the laboratory area and follow the directions posted at each doorway to the laboratory area.

The above general laboratory safety rules are designed to safeguard you and your co-workers, fellow students and colleagues and are a minimum requirement for individuals working in the computer engineering laboratories at Umm Al-Qura University, Makkah Al-Mukarramah. Specialized training and rules may apply depending on type and scope of activities involved.

This page is intentionally left blank

Lab 1: Introduction to PIC18F452 Microcontroller

Objective:

The objective of this experiment is to introduce PIC18F452 microcontroller.

Introduction:

Peripheral Interface Controller (PIC) is a family of modified Harvard architecture microcontrollers made by Microchip Technology. PICs are popular with both industrial developers and hobbyists due to the availability of low cost development tools and serial programming capability.

Pin outs of PIC 18F452 microcontroller:

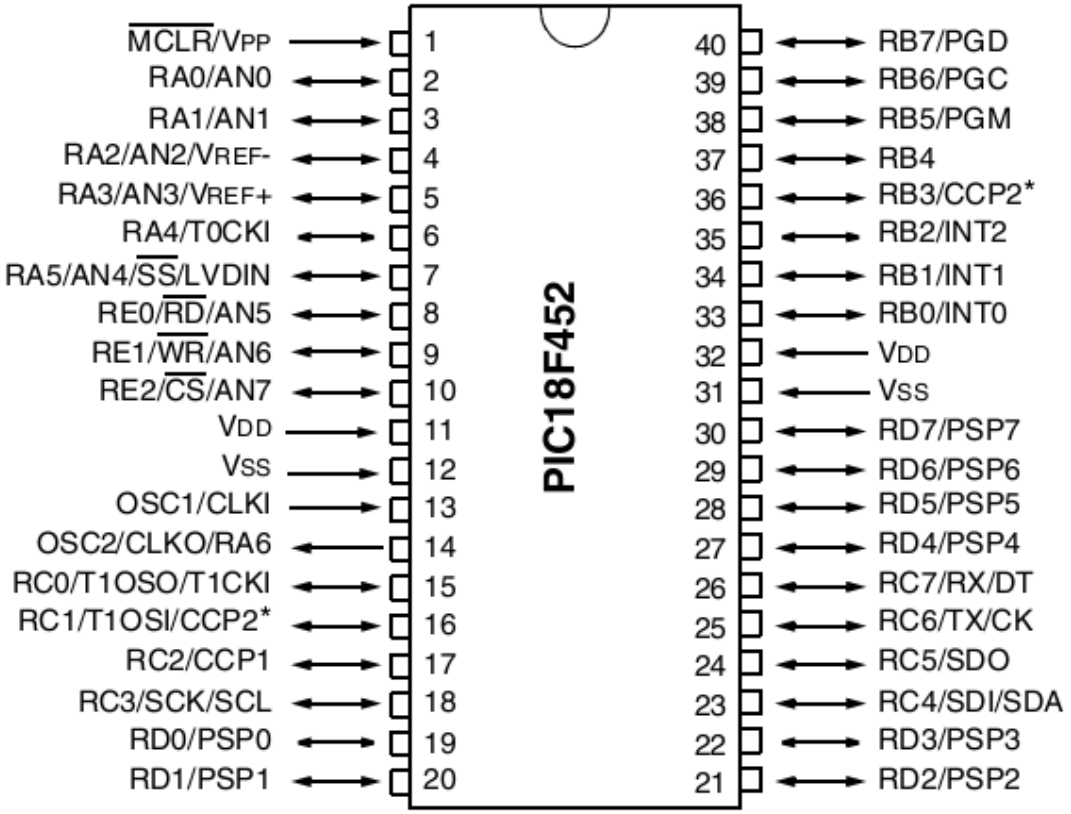


Figure 1.1: Pin outs of PIC 18F452 microcontroller.

Major Features of PIC18F452 microcontroller:

High-Performance RISC CPU

- Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 32 Kbytes
- DC - 40 MHz osc./clock input
- 16-bit wide instructions, 8-bitwide data path
- Priority levels for interrupts

Peripheral Features

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Two 16-bit timer/counter (TMR1, TMR3)
- I ²C Master and Slave mode
- Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

Special Microcontroller Features

- Power-On Reset
- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- Watchdog Timer (WDT) with its own On-Chip RC oscillator
- Programmable code protection
- Single supply 5V In-circuit Serial Programming via two pins
- In-Circuit Debug (ICD

Analog Features

- Compatible 10-bit Analog-to-Digital Converter module (A/D)
- Programmable Low Voltage Detection (PLVD)
- Programmable Brown-out Reset(BOR)

Power Supply, Clock and Port A

Pin Name	Pin	Pin Type	Buffer Type	Description
MCLR/VPP	1			Master Clear (input) or high voltage ICSP programming enable pin.
MCLR		I	ST	Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP		I	ST	High voltage ICSP programming enable pin.
NC	—	—	—	These pins should be left unconnected.
OSC1/CLKI	13			Oscillator crystal or external clock input.
OSC1		I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise.
CLKI		I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
OSC2/CLKO/RA6	14			Oscillator crystal or clock output.
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General Purpose I/O pin.
				PORTA is a bi-directional I/O port.
RA0/AN0	2			
RA0		I/O	TTL	Digital I/O.
AN0		I	Analog	Analog input 0.
RA1/AN1	3			
RA1		I/O	TTL	Digital I/O.
AN1		I	Analog	Analog input 1.
RA2/AN2/VREF-	4			
RA2		I/O	TTL	Digital I/O.
AN2		I	Analog	Analog input 2.
VREF-		I	Analog	A/D Reference Voltage (Low) input.
RA3/AN3/VREF+	5			
RA3		I/O	TTL	Digital I/O.
AN3		I	Analog	Analog input 3.
VREF+		I	Analog	A/D Reference Voltage (High) input.
RA4/T0CKI	6			
RA4		I/O	ST/OD	Digital I/O. Open drain when configured as output.
T0CKI		I	ST	Timer0 external clock input.
RA5/AN4/SS/LVDIN	7			
RA5		I/O	TTL	Digital I/O.
AN4		I	Analog	Analog input 4.
SS		I	ST	SPI Slave Select input.
LVDIN		I	Analog	Low Voltage Detect Input.
RA6				(See the OSC2/CLKO/RA6 pin.)

Port B

Pin Name	Pin	Pin Type	Buffer Type	Description
				PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	33			
RB0		I/O	TTL	Digital I/O.
INT0		I	ST	External Interrupt 0.
RB1/INT1	34			
RB1		I/O	TTL	
INT1		I	ST	External Interrupt 1.
RB2/INT2	35			
RB2		I/O	TTL	Digital I/O.
INT2		I	ST	External Interrupt 2.
RB3/CCP2	36			
RB3		I/O	TTL	Digital I/O.
CCP2		I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RB4	37	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB5/PGM	38			
RB5		I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGM		I/O	ST	Low Voltage ICSP programming enable pin.
RB6/PGC	39			
RB6		I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGC		I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/PGD	40			
RB7		I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGD		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Port C

Pin Name	Pin	Pin Type	Buffer Type	Description
				PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15			
RC0		I/O	ST	Digital I/O.
T1OSO		O	—	Timer1 oscillator output.
T1CKI		I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	16			
RC1		I/O	ST	Digital I/O.
T1OSI		I	CMOS	Timer1 oscillator input.
CCP2		I/O	ST	Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	17			
RC2		I/O	ST	Digital I/O.
CCP1		I/O	ST	Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18			
RC3		I/O	ST	Digital I/O.
SCK		I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL		I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23			
RC4		I/O	ST	Digital I/O.
SDI		I	ST	SPI Data In.
SDA		I/O	ST	I^2C Data I/O.
RC5/SDO	24			
RC5		I/O	ST	Digital I/O.
SDO		O	—	SPI Data Out.
RC6/TX/CK	25			
RC6		I/O	ST	Digital I/O.
TX		O	—	USART Asynchronous Transmit.
CK		I/O	ST	USART Synchronous Clock (see related RX/DT).
RC7/RX/DT	26			
RC7		I/O	ST	Digital I/O.
RX		I	ST	USART Asynchronous Receive.
DT		I/O	ST	USART Synchronous Data (see related TX/CK).

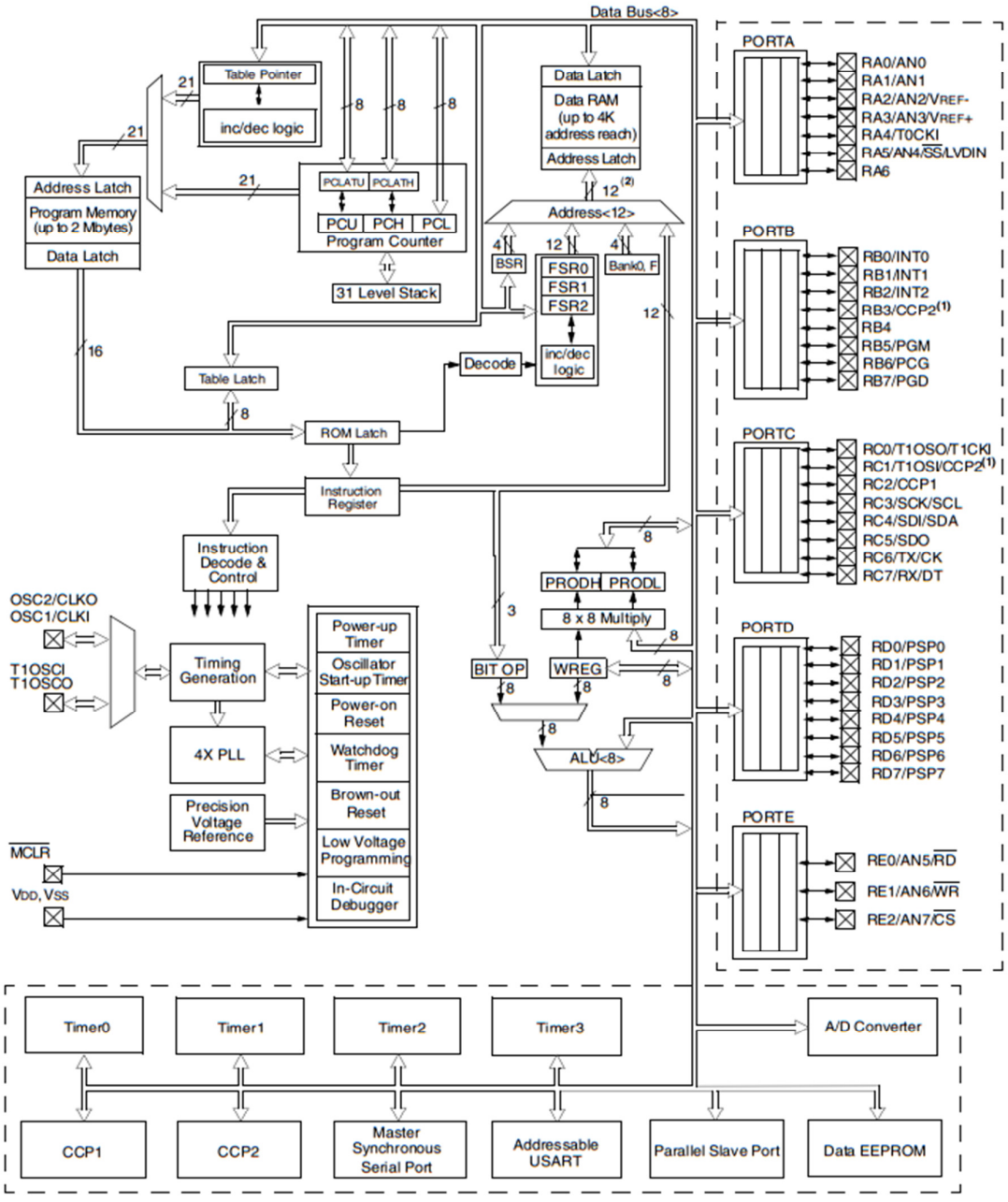
Port D

Pin Name	Pin	Pin Type	Buffer Type	Description
				PORTD is a bi-directional I/O port, or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/PSP0	19	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD1/PSP1	20	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD2/PSP2	21	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD3/PSP3	22	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD4/PSP4	27	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD5/PSP5	28	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD6/PSP6	29	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD7/PSP7	30	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
				PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5 RE0 $\overline{RD}$ AN5	8	I/O	 ST TTL Analog	 Digital I/O. Read control for parallel slave port (see also $\overline{WR}$ and $\overline{CS}$ pins). Analog input 5.
RE1/$\overline{WR}$/AN6 RE1 $\overline{WR}$ AN6	9	I/O	 ST TTL Analog	 Digital I/O. Write control for parallel slave port (see $\overline{CS}$ and $\overline{RD}$ pins). Analog input 6.
RE2/$\overline{CS}$/AN7 RE2 $\overline{CS}$ AN7	10	I/O	 ST TTL Analog	 Digital I/O. Chip Select control for parallel slave port (see related $\overline{RD}$ and $\overline{WR}$). Analog input 7.
VSS	12, 31	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	P	—	Positive supply for logic and I/O pins.

Lab 2: Internal Architecture of PIC18F452 Microcontroller

Objective:

To understand the internal architecture of PIC18F452 microcontroller.



Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.
2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the MOVFF instruction).
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

Figure 2.1: Internal Architecture of PIC18F452 Microcontroller

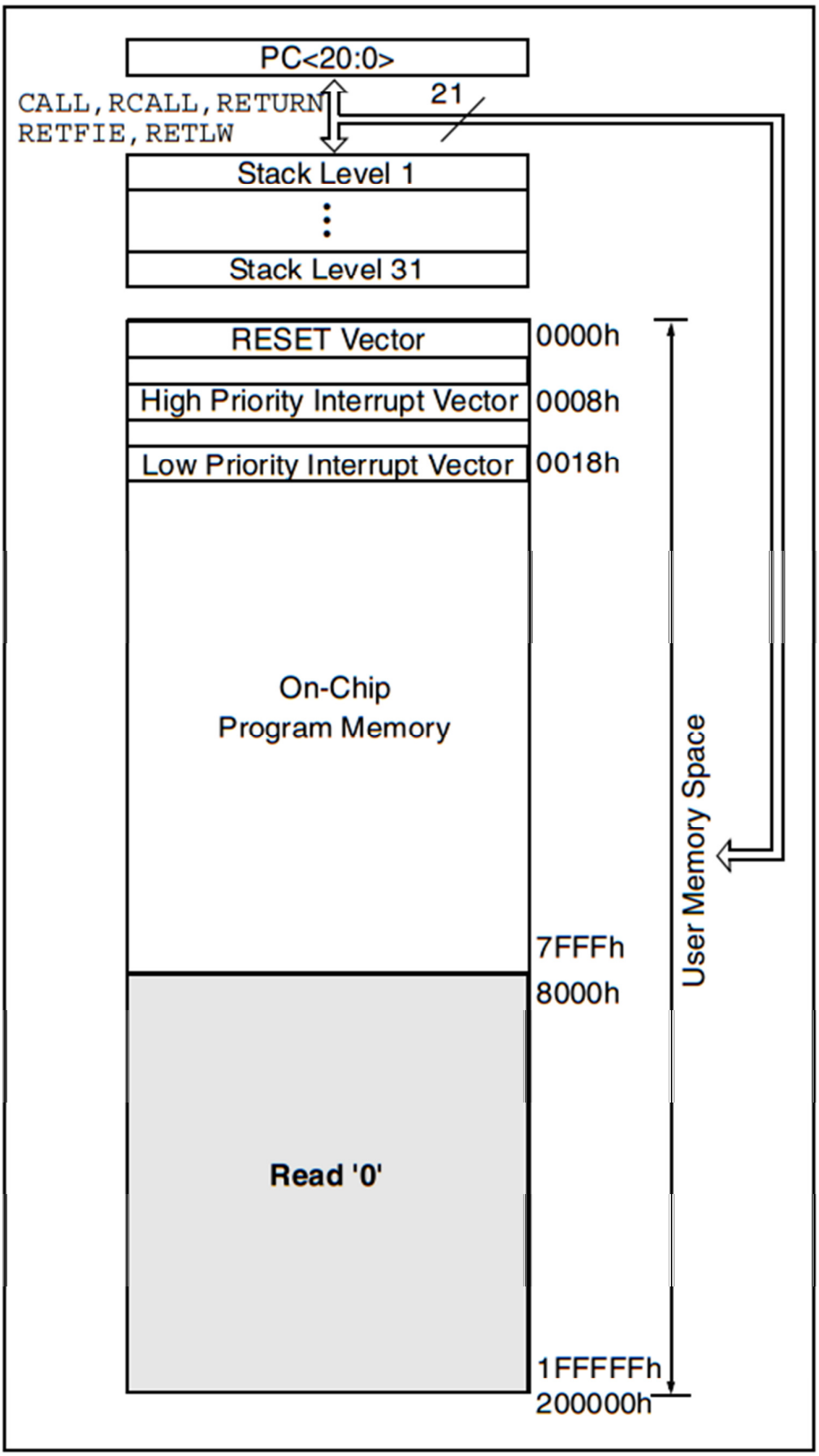


Figure 2.2: Program memory map and stack for PIC18F452 microcontroller

Oscillator Configurations:

The Peripheral Interface Controller (PIC) can be operated in eight different Oscillator modes. The user can program three configuration bits (FOSC2, FOSC1, AND OFSC0) to select one of these eight modes:

1. LP Low Power Crystal
2. XT Crystal / Resonator
3. HS High Speed Crystal / Resonator
4. HS + PLL High Speed Crystal / Resonator with PLL enabled
5. RC External Resistor / Capacitor
6. RCIO External Resistor / Capacitor with I/O pin enabled
7. EC External Clock
8. ECIO External Clock with I/O pin enabled

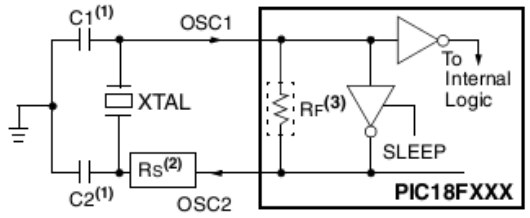


Figure 2.3: Crystal / Ceramic Resonator Operation (HS, XT or LP Configuration

OSCON Register:

The Peripheral Interface Controller (PIC) can be operated in eight different Oscillator modes. The user can program three configuration.

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-1
—	—	—	—	—	—	—	SCS
bit 7							bit 0

bit 7-1 : Un implemented: Read as ‘0’

bit 0 : SCS: System Clock Switch bit

When $\overline{\text{OSCSEN}}$ configuration bit = '0' and T1OSCEN bit is set:
1 = Switch to Timer1 oscillator/clock pin
0 = Use primary oscillator/clock input pin
When $\overline{\text{OSCSEN}}$ and T1OSCEN are in other states:
bit is forced clear

Figure 2.4: OSCON Register of PIC18F452 microcontroller

RESET:

The PIC18F452 differentiate between various kinds of RESET:

a) Power-on-Reset (POR).

b) $\overline{\text{MCLR}}$ Reset during normal operation

c) $\overline{\text{MCLR}}$ Reset during SLEEP.

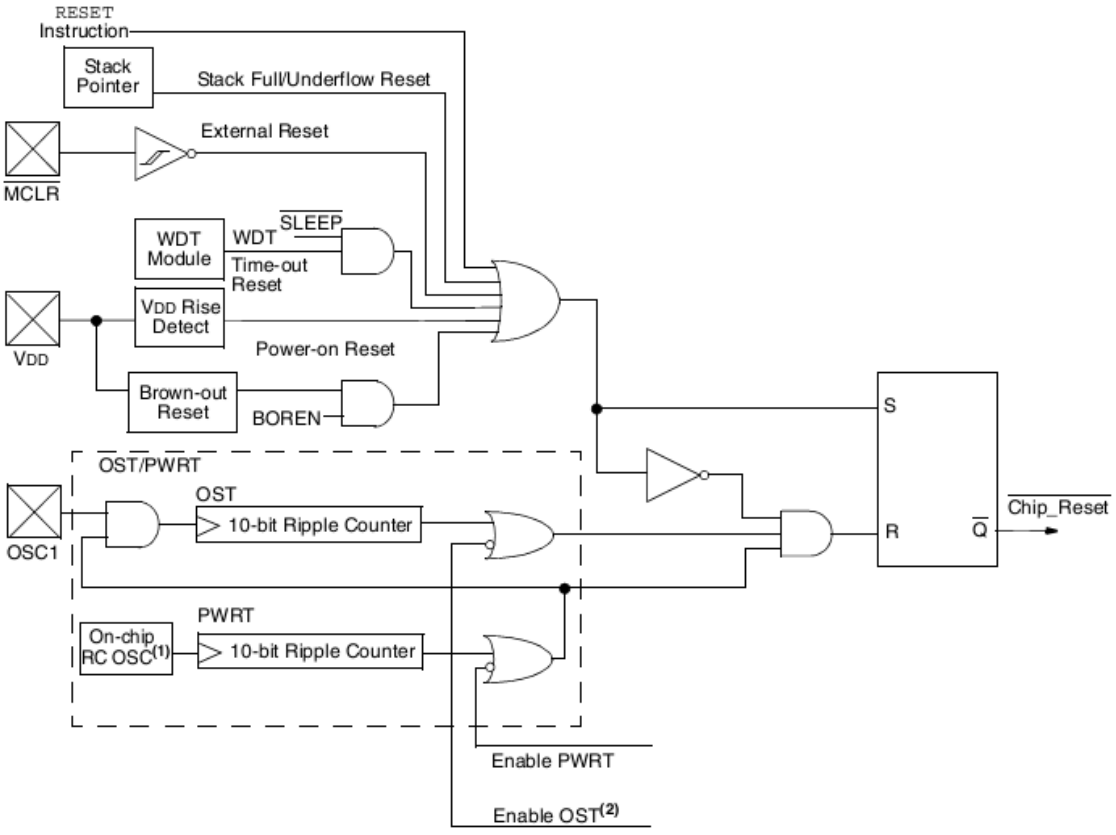
d) Watchdog Timer (WDT) Reset (during normal operation).

e) Programmable Brown-out-Reset (BOR)

f) `RESET` Instruction

g) Stack Full Reset

h) Stack Underflow Reset.



Note **1:** This is a separate oscillator from the RC oscillator of the CLKI pin.
2: See Table 3-1 for time-out situations.

Figure: 2.5: Block diagram of On-Chip Reset Circuit

Lab 3: To Introduce EEDT 6.0 Multi-Microcontroller Kit

Objective:

To introduce EEDT 6.0 multi-microcontroller kit.

Introduction:

The Embedded Engineer's Development Tool 6.0 is multi-microcontroller development board. It includes an Atmel, ATmega32 AVR microcontroller, an P89V51RD2 8051 microcontroller, a Microchip PIC16F873A and ATmega328 with the Arduino boot loader installed.

All-in-one Board Popular Supported Devices:

8051:	AT89S51, AT89C52
AVR:	ATmega8, ATmega16, ATmega328, ATmega32
PIC:	PIC16F873A, **PIC18F452** and pin compatible
ARM:	LPC2138
Arduino:	ATmega328.

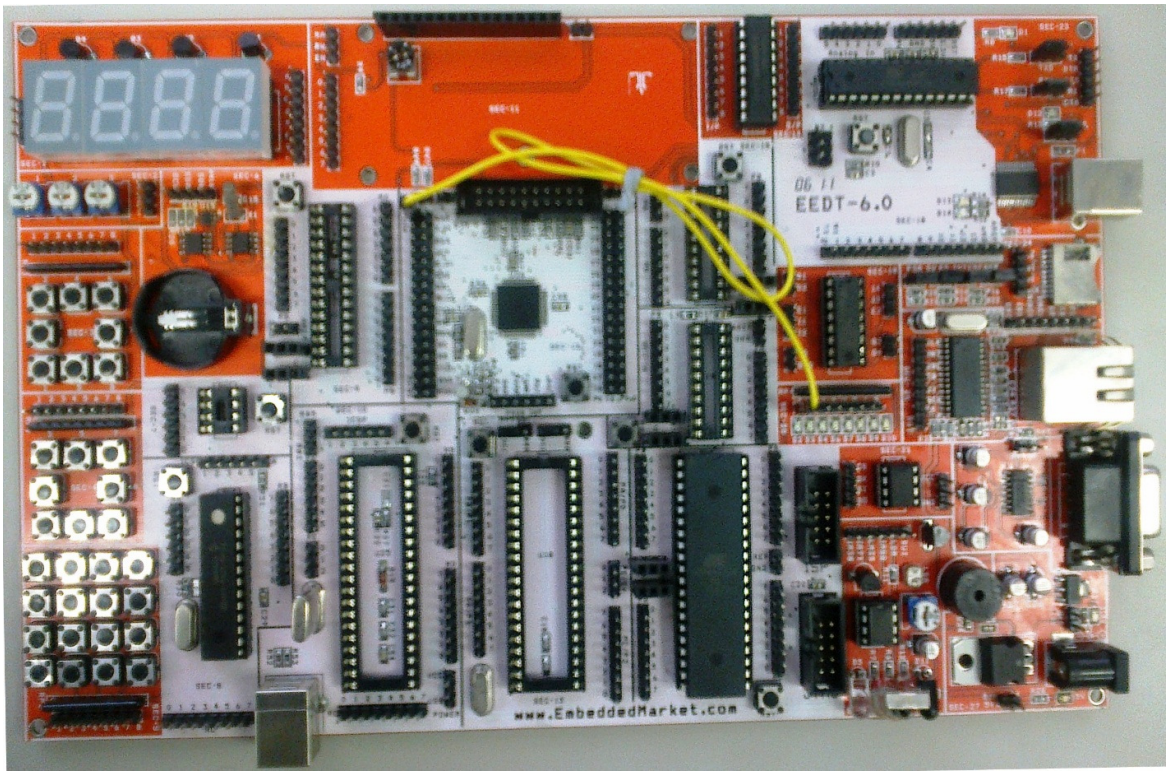


Figure 3.1: EEDT 6.0 All in one Development Board

All in one Development Board Features

The EEDT 6.0 Development board includes the following sections numbered on the PCB:

Sec-1: Seven-Segment Display — Four multiplexed

Sec–2: Variable resistance for Analog Input — Three separate presets

Sec-3: Pulled down Push-to-On Switches — Eight switches

Sec-4: Pulled up Push-to-On Switches — Eight switches

Sec-5: Matrix keypad of 4x4 keys — Total 16 keys

Sec-6: I2C-Based Real-time clock, EEPROM & Digital-to-Analog converter

Sec-7: Socket for ATTiny13 and pin-compatible AVR microcontrollers

Sec-8: Socket for PIC16F873A and pin-compatible PIC microcontrollers

Sec-9: Socket for ATmega8 / ATmega168 and pin-compatible AVR microcontrollers

Sec-10: Socket for PIC18F4550 **/ PIC18F452** and pin-compatible PICs

Sec-11: 16x2 LCD Interface. LCD is mounted on the board

Sec-12: Pre-soldered ARM7 LPC2138 microcontroller

Sec-13: Socket for P89V51RD2 (8051 family) microcontroller or for ATmega8515 AVR and pin-compatible microcontrollers. Set Reset pin Jumper to Vcc when using 8051 family microcontroller; Set it to Gnd for AVR.

Sec-14: High Current Driver based on ULN2803 — Use it to drive stepper motors and seven-segment displays

Sec-15: Socket for ATTiny2313 and pin-compatible AVR microcontrollers

Sec-16: Socket for ATTiny26 and pin-compatible AVR microcontrollers

Sec-17: Socket for ATmega16 / ATmega32 / ATmega8535 and pin-compatible AVR microcontrollers (populated with ATmega32)

Sec-18: Implementation of Arduino Duemilanove platform

Sec-19: DC Motor driver using L293D

Sec-20: Bank of 8 LEDs

Sec-21: SPI EEPROM AT93C46

Sec-22: Collection of sensors and other interfaces — Temperature sensor, Light sensor, Infrared (IR) sensor, 38KHz IR receiver, IR Transmitter, Buzzer

Sec-23: USB-to-TTL converter interface — This is used for Arduino as well as a standalone USB-to-TTL interface. Jumper settings required.

Sec-24: Micro SD Card — Can be interfaced with 3.3V I/Os or with 5V I/Os. Jumper settings required.

Sec-25: ENC28J60-based Ethernet interface. Use it to create web-enabled applications.

Sec-26: RS232-to-TTL converter interface.

Sec-27: Power supply section which includes 5V and 3.3V regulation. Wall-mount DC adapter that can source 500mA or higher at 9V DC is suitable. Exceeding input voltage may heat the onboard Voltage regulator ICs.

The EEDT 6.0 hardware is divided into two major sections:

1. The onboard Microcontroller Section (This area has White color background on the hardware board).
2. The onboard Interface Circuit Section (This area has Red color background on the hardware board)

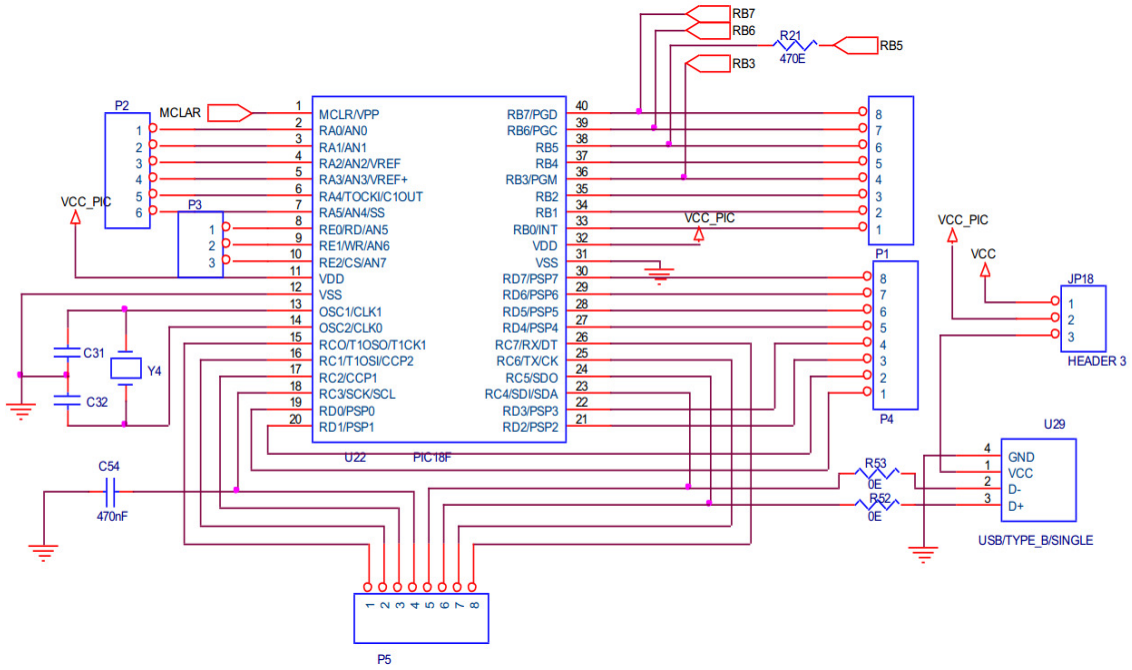


Figure 3.2: PIC 18F452 base on EEDT 6.0 kit

The EEDT6.0 includes the following programmers (all are ISP/ICSP):

USB Programmer for AVR and 8051:

External ISP programmer connects to USB port of your PC and ISP programming header on the EEDT6.0. AVR microcontrollers are programmed and automatically switched to Run mode. Also programs AT89S51 and AT89S52. This programmer uses proprietary HandyProg software (does not use AVR Studio).

USB Programmer for P89V51RD2 and LPC2138 (and compatible from NXP):

External programmer connects to USB port of your PC and programming header on the EEDT6.0. This programmer is based on FlashMagic.

Serial-port Programmer for PIC:

External programmer connects to serial port on a PC or laptop and ICSP header on the EEDT6.0. Your computer must have an RS232 serial port; USB-to-Serial converters will not work. This programmer is based on IC-Prog/PICPgm.

Figure 3.3: EEDT 6.0 PIC Serial Programmer

Lab 4: Introduction to mikroC Pro for PIC microcontroller

Objective:

To introduce mikroCPro Software for PIC microcontroller.

Introduction:

The mikroC PRO for PIC organizes applications into projects, consisting of a single project file (extension .mcppi) and one or more source files (extension .c). mikroC PRO for PIC IDE allows you to manage multiple projects (see Project Manager). Source files can be compiled only if they are part of a project.

The project file contains the following information:

- project name and optional description,
- target device,
- device flags (config word),
- device clock,
- list of the project source files with paths,
- header files (*.h),
- binary files (*.mcl),
- image files,
- other files.

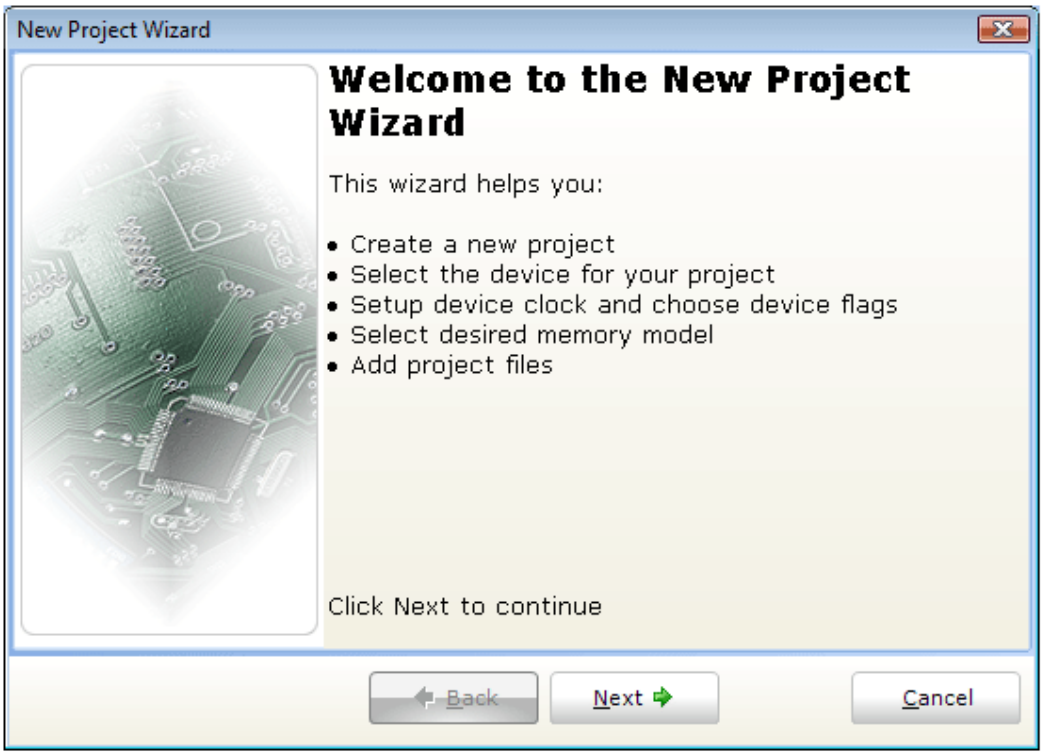
Note that the project does not include files in the same way as preprocessor does, see Add/Remove Files from Project.

New Project

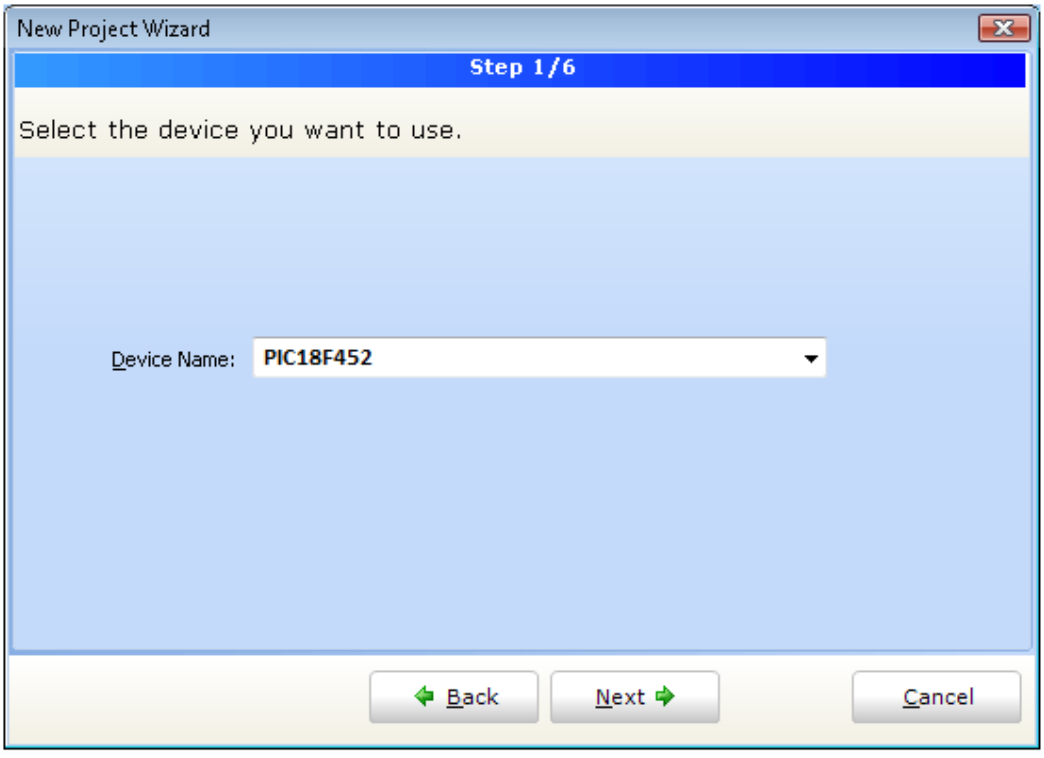
The easiest way to create a project is by means of the New Project Wizard, drop-down menu Project › New Project or by clicking the New Project Icon from Project Toolbar.

New Project Wizard Steps

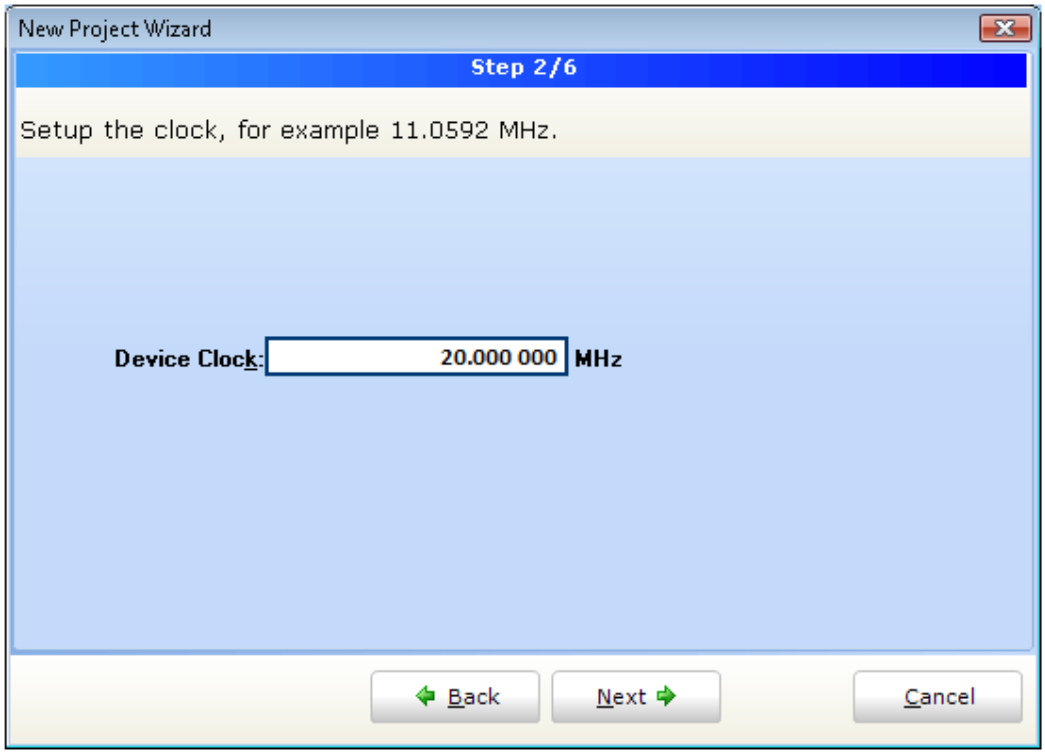
Start creating your New project, by clicking Next button:



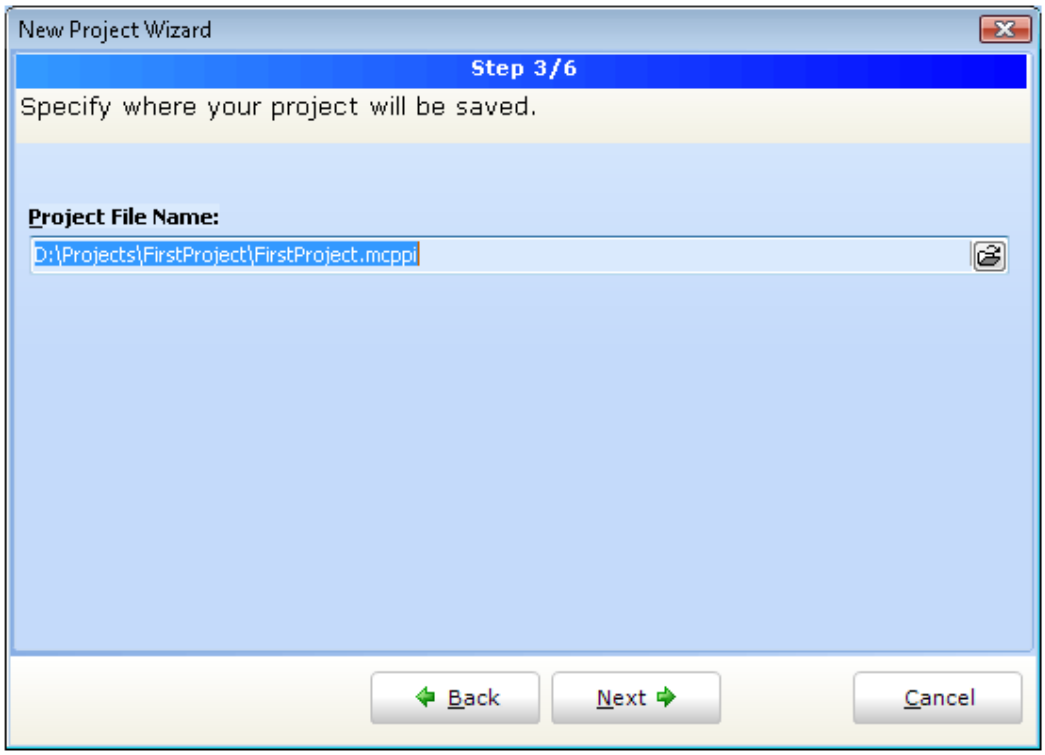
Step One - Select the device from the device drop-down list :



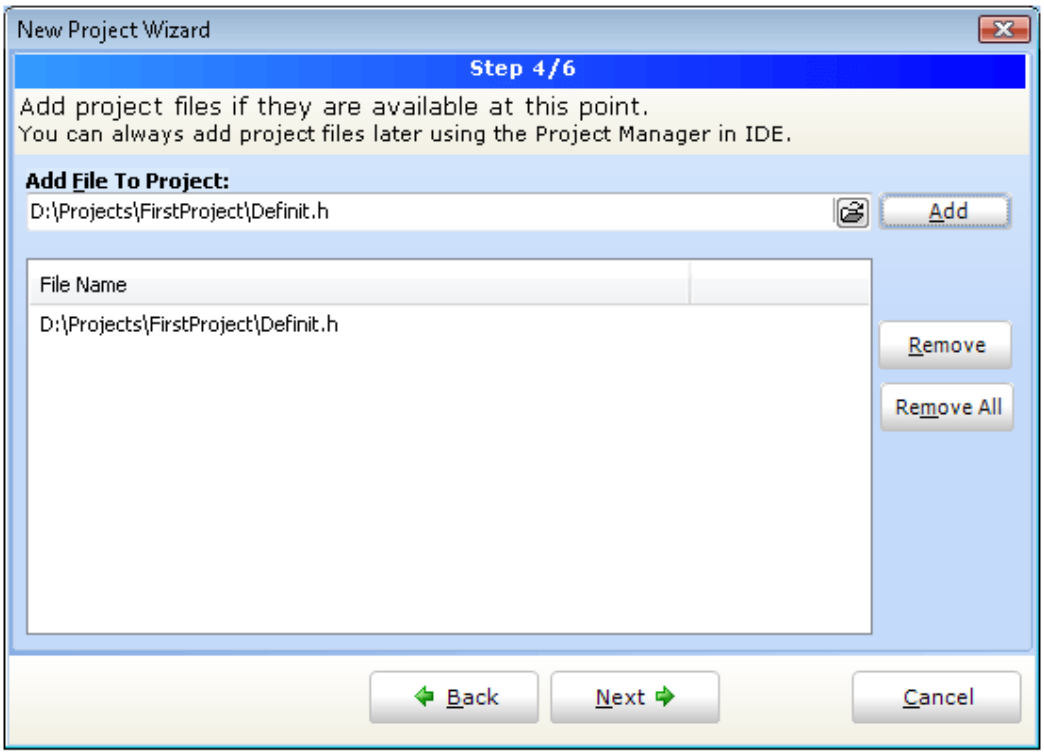
Step Two – Enter the oscillator frequency value:



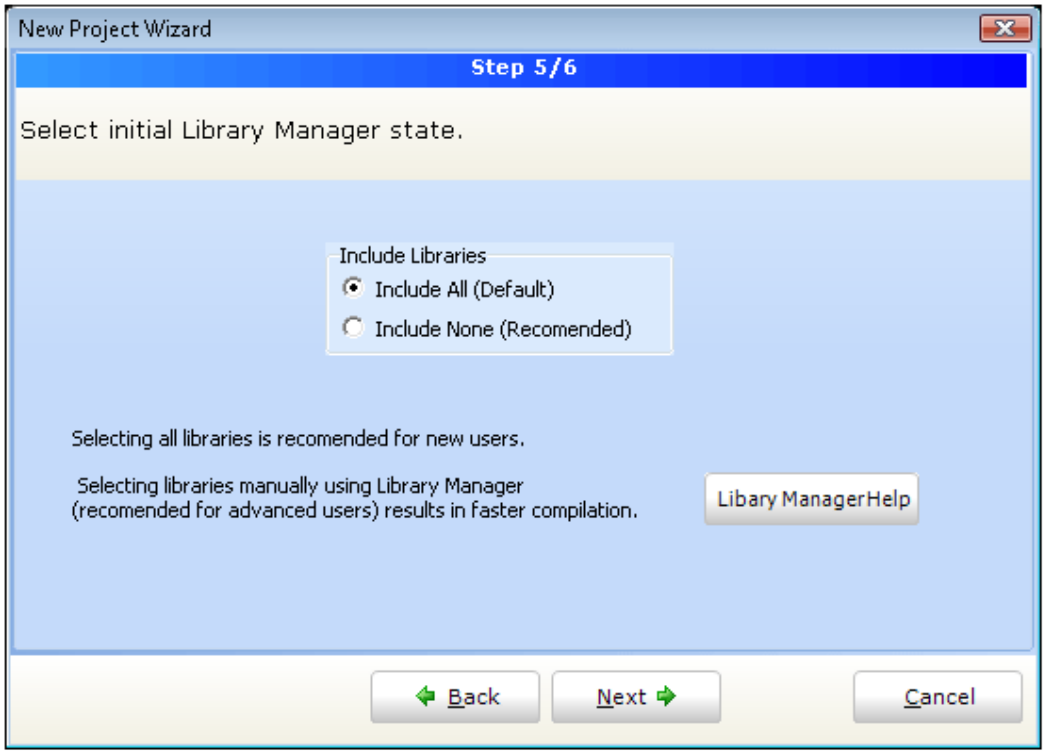
Step Three – Specify the location where your project will be saved:



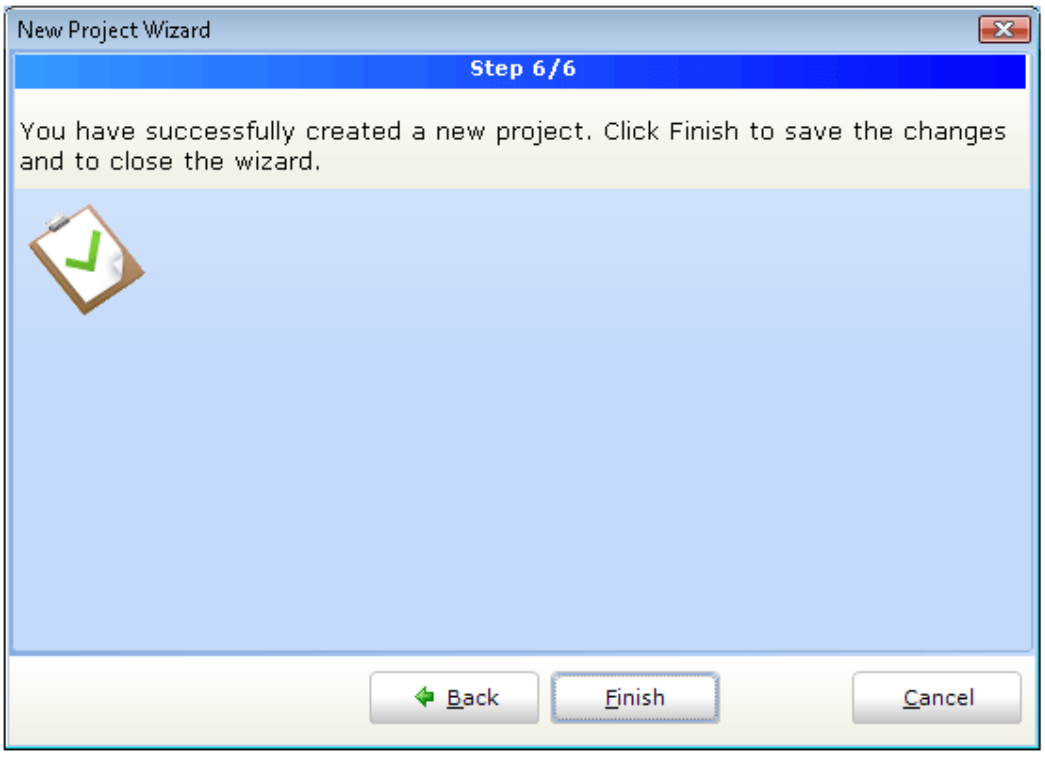
Step Four – Add project file to the project if they are available at this point. You can always add project files later using Project Manager :



Step Five – Select initial Library Manager state :



Step Six – Click Finish button to create your New Project ::



Customizing Projects

You can change basic project settings in the Project Settings window. You can change chip, and oscillator frequency. Any change in the Project Setting Window affects currently active project only, so in case more than one project is open, you have to ensure that exactly the desired project is set as active one in the Project Manager. Also, you can change configuration bits of the selected chip in the Edit Project window.

Managing Project Group

mikroC PRO for PIC IDE provides convenient option which enables several projects to be open simultaneously. If you have several projects being connected in some way, you can create a project group.

The project group may be saved by clicking the Save Project Group Icon from the Project Manager window. The project group may be reopened by clicking the Open Project Group Icon . All relevant data about the project group is stored in the project group file (extension .mcpigroup)

Edit Project

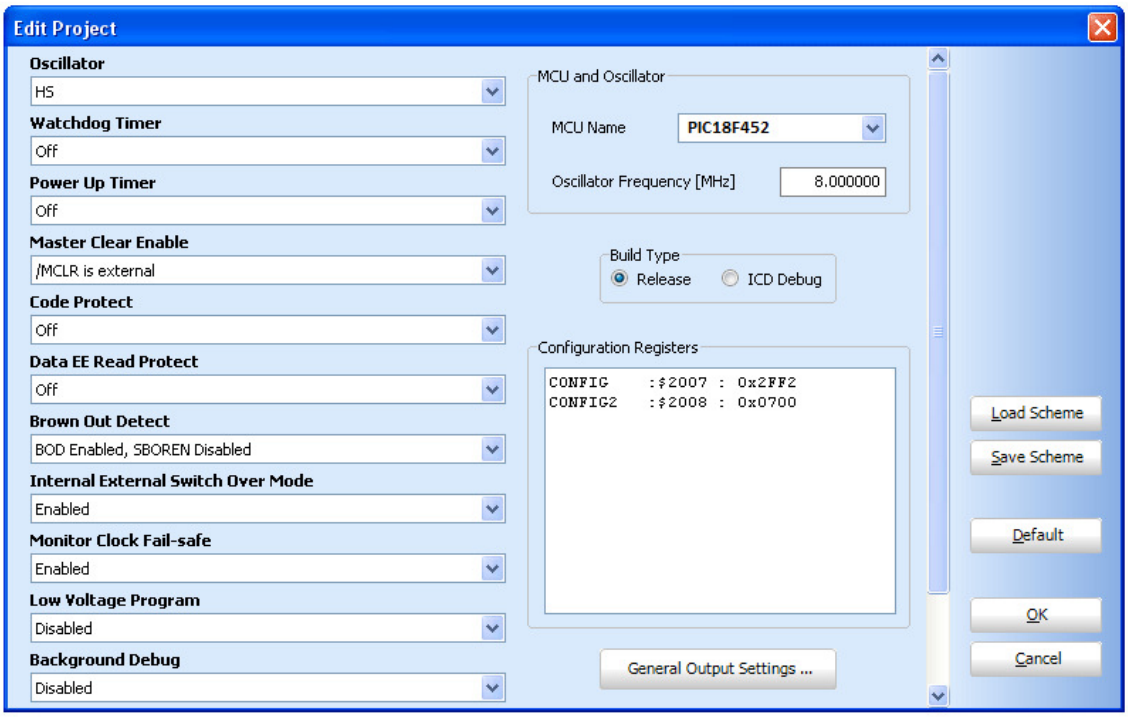
Edit Project gives you option to change MCU you wish to use, change its oscillator frequency and build type. Also, Edit Project enables you to alter specific configuration bits of the selected device.

As you alter these bits, appropriate register values will be updated also. This can be viewed in the Configuration Registers pane.

When you have finished configuring your device, you can save bit configuration as a scheme, using button.

In case you need this scheme in another project, you can load it using button.

There is also a button which lets you select default configuration bit settings for the selected device.


Edit Project
Oscillator
HS
Watchdog Timer
Off
Power Up Timer
Off
Master Clear Enable
/MCLR is external
Code Protect
Off
Data EE Read Protect
Off
Brown Out Detect
BOD Enabled, SBOREN Disabled
Internal External Switch Over Mode
Enabled
Monitor Clock Fail-safe
Enabled
Low Voltage Program
Disabled
Background Debug
Disabled
MCU and Oscillator
MCU Name
PIC18F452
Oscillator Frequency [MHz]
8.000000
Build Type
Release
ICD Debug
Configuration Registers
CONFIG :$2007 : 0x2FF2
CONFIG2 :$2008 : 0x0700
General Output Settings ...
Load Scheme
Save Scheme
Default
OK
Cancel

Lab 5: Program to Turn ON and OFF LEDs

Objective:

The objective of this experiment is to turn ON and OFF LEDs with interval.

Equipment:

1. EEDT 6.0 Microcontroller trainer board
2. Personal computer.
3. PIC Programmer

Programming Steps:

1. Create NEW PROJECT
2. Select Device as PIC18F452 microcontroller
3. Set Clock frequency 20 MHz
4. Select Project file name

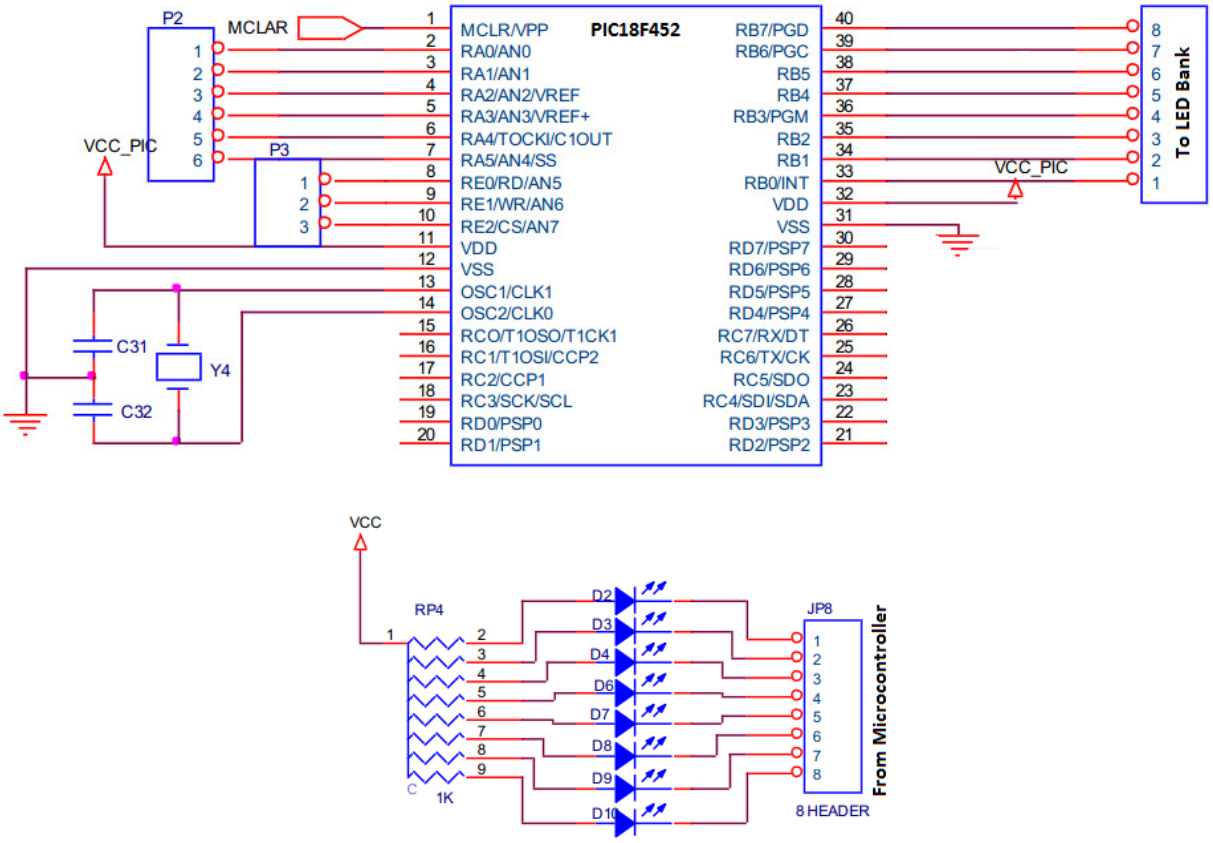


Figure: 5.1: PIC18F452 microcontroller and Bank of LEDs.

```
/************************************************************/
/* Program to Turn ON and OFF LEDs with 500ms interval      */
/************************************************************/

void main (void){
TRISB = 0;                    // Set Port B as Output
PORTB = 0;                    // Initialize Port B
while (1){
   PORTB = 0x55;              // Send the value to Port B
   Delay_ms (500);            // Delay of 0.5 seconds
   PORTB = 0xAA;              // Send the value to Port B
   Delay_ms (500);            // Delay of 0.5 seconds
      }
   }
```

Exercise: Modify the code for Port D

```
/************************************************************/
/* Turn ON and OFF LEDs with 1000 ms interval using PORT“D” */
/* Student Name:                          Student ID:       */
/************************************************************/
```

Lab 6: Program to make Binary Counter on LEDs

Objective:

The objective of this experiment is to make binary counter and show the output on LEDs.

Equipment:

4. EEDT 6.0 Microcontroller trainer board
5. Personal computer.
6. PIC Programmer

Programming Steps:

5. Create NEW PROJECT
6. Select Device as PIC18F452 microcontroller
7. Set Clock frequency 20 MHz
8. Select Project file name

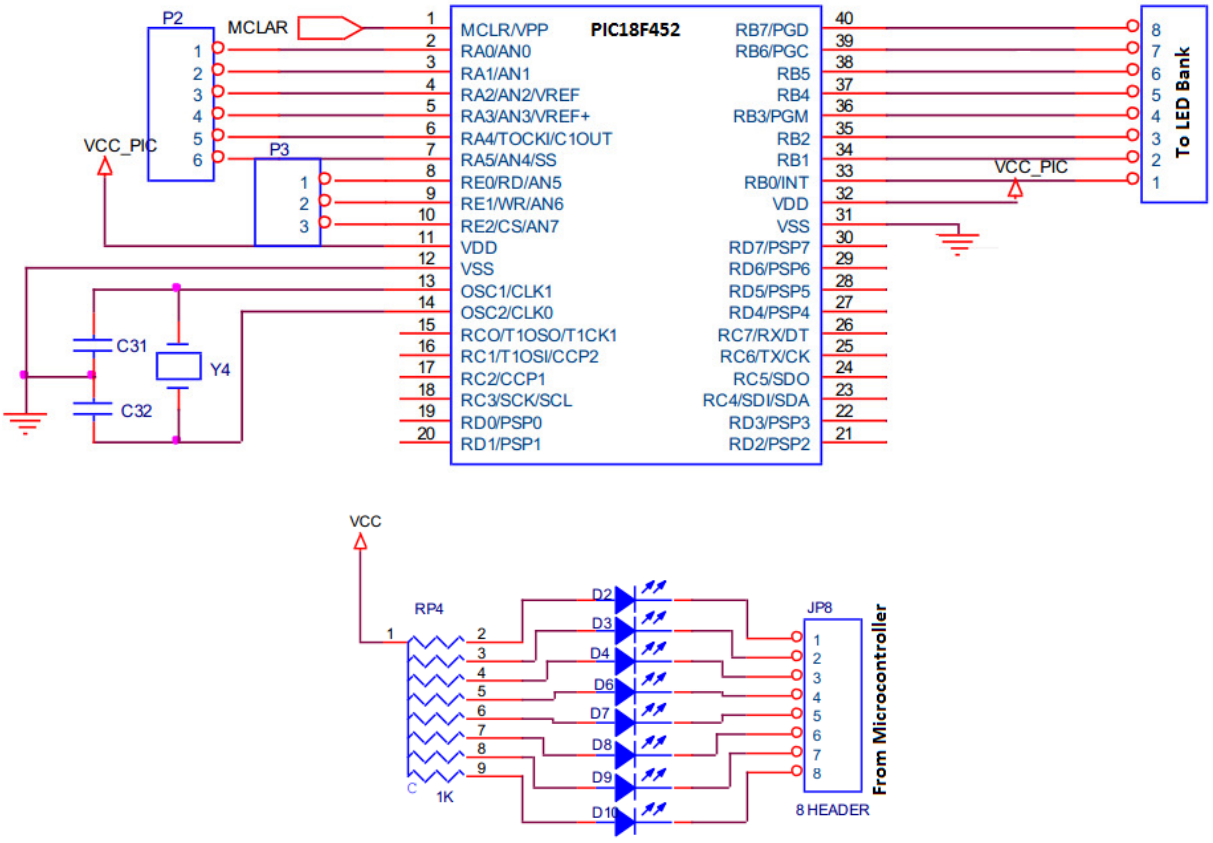


Figure: 5.1: PIC18F452 microcontroller and Bank of LEDs.

```
/*************************************************************/
/* Program to make 4-bit binary counter, showing output on
LEDs    */
/*************************************************************/

void main()
{
unsigned char count=0;

    TRISB.F0 = 1;                   // set RB0 pin as input
    TRISC = 0x00;              // Configure PORTC as output
    PORTC = 0x00;         // Initial PORTC value
do {
if (Button(&PORTB, 0, 1, 0))  // Detect one-to-zero transition
{
PORTC=count;
if(count==16)
count=0;
count++;
Delay_ms(500);

}
} while(1);                 // Endless loop
}

/*************************************************************/
/* Home assignment:  Make a knight rider program*/
/*************************************************************/
```

Lab 7: Program to Read from Button

Objective:

The objective of this experiment is to read from button.

Equipments:

1. EEDT 6.0 Microcontroller trainer board
2. Personal computer.
3. Programmer

Programming Steps:

1. Create NEW PROJECT
2. Select Device as PIC18F452 microcontroller
3. Set Clock frequency 20 MHz
4. Select Project file name

NOTE: The Button Library provides routines for detecting button presses and denouncing (eliminating the influence of contact flickering upon pressing a button).

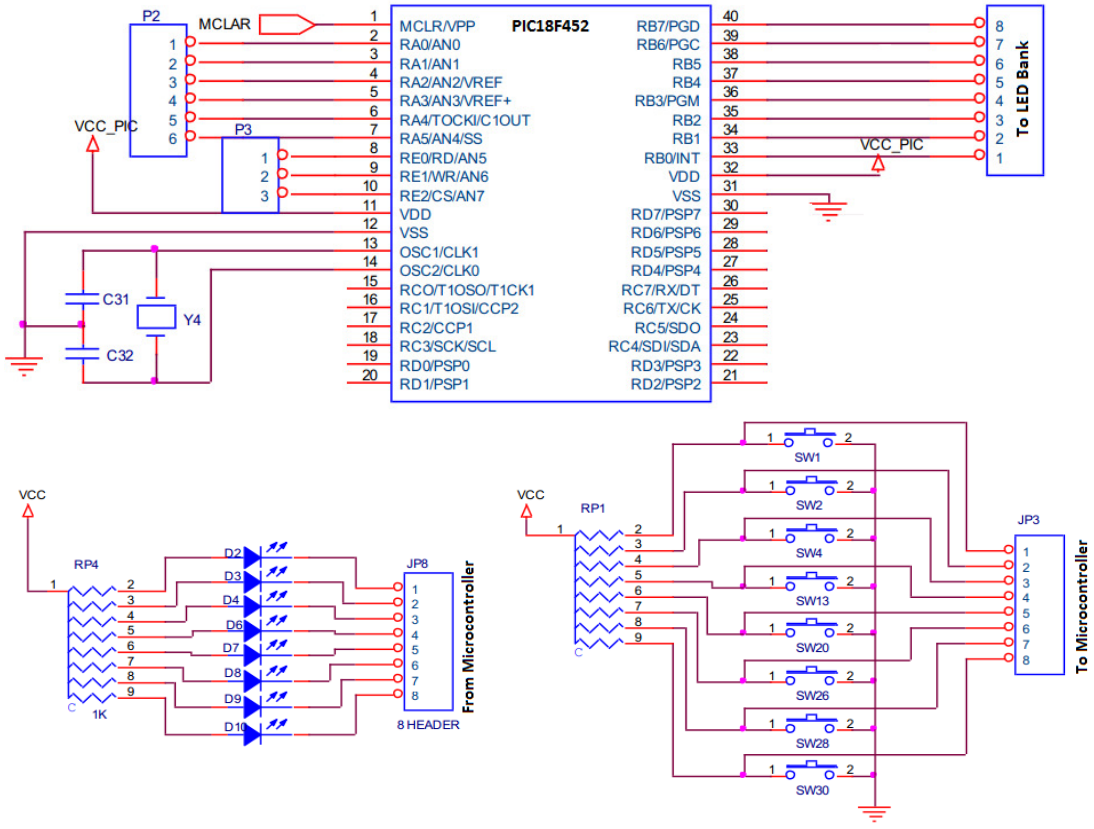


Figure: 6.1: PIC18F452 microcontroller and Bank of LEDs.

```
/*************************************************************/
/* Program reads RB0, to which button is connected           */
/* On transition from 1-0 (button release),Invert Port D     */
/*************************************************************/

bit oldstate;                  // Old state flag
void main() {
TRISB0_bit = 1;                // set RB0 pin as input
TRISC = 0x00;                  // Configure PORTC as output
PORTC = 0xAA;                 // Initial PORTC value
oldstate = 0;
     do {
     if (Button(&PORTB, 0, 1, 1)) {     // Detect logical one
     oldstate = 1;                      // Update flag
        }
    if (oldstate && Button(&PORTB, 0, 1, 0)) {
                               // Detect one-to-zero transition
      PORTC = ~PORTC;                   // Invert PORTC
      oldstate = 0;                     // Update flag
          }
        }
     while(1);                          // Endless loop
     }
```

Lab 8: Interfacing 7 Segment Display with PIC18f452

Objective:

This project describes the design of a 7-segment LED-based counter which counts from 0 to 9 continuously with a one-second delay between counts. The project shows how a 7-segment LED can be interfaced and used in a PIC microcontroller project. 7-segment displays are used frequently in electronic circuits to show numeric or alphanumeric values. As shown in Figure a 7-segment display consists basically of 7 LEDs connected such that the numbers from 0 to 9 and some letters can be displayed. Segments are identified by the letters from a to g, and Figure 7.2 shows the segment names of a typical 7-segment display.

Equipment:

- EEDT 6.0 Microcontroller trainer board
- Personal computer.
- Programmer

Programming Steps:

- Create NEW PROJECT
- Select Device as PIC18F452 microcontroller
- Set Clock frequency 20 MHz
- Select Project file name

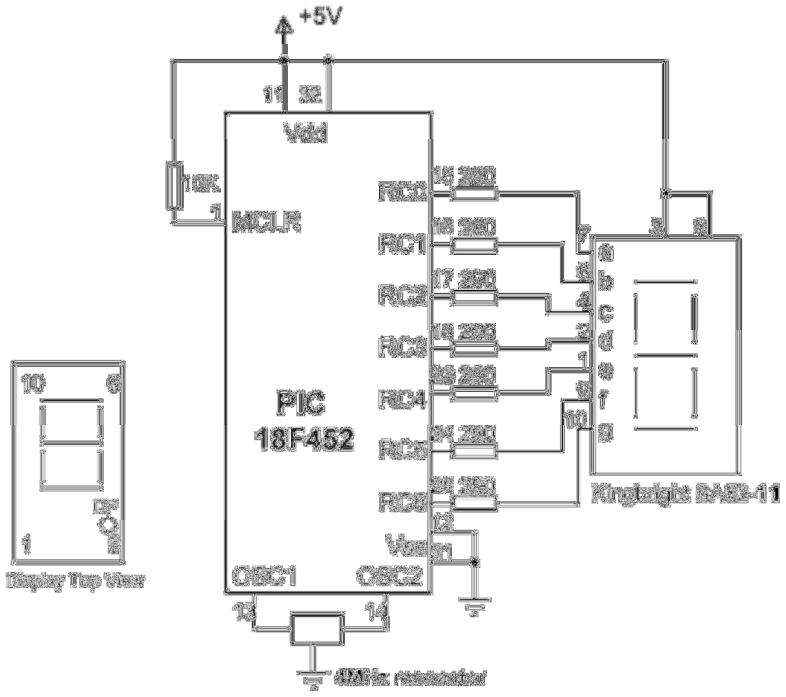


Figure 8.1 Interfacing Seven Segment Display with PIC 18f452

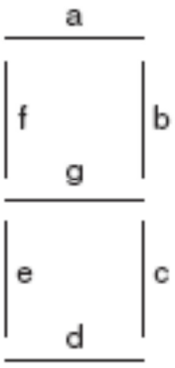


Figure 8.2 Segment name of seven segment display

Number	x g f e d c b a	PORTC Data
0	00111111	0x3F
1	00000110	0x06
2	01011011	0x5B
3	01001111	0x4F
4	01100110	0x66
5	01101101	0x6D
6	01111101	0x7D
7	00000111	0x07
8	01111111	0x7F
9	01101111	0x6F

x is not used, taken as 0.

Figure 8.3 Displayed number and data sent to port c

```
/************************************************************/
/* Program for interfacing Seven segment display with PIC
18f452   */
/************************************************************/

void main()
{
unsigned char Pattern, Cnt = 0;
unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,
0x7D,0x07,0x7F,0x6F};
TRISC = 0; // PORTC are outputs
for(;;) // Endless loop
{
Pattern = SEGMENT[Cnt]; // Number to send to PORTC
Pattern = ~Pattern; // Invert bit pattern
PORTC = Pattern; // Send to PORTC
Cnt++;
if(Cnt == 10) Cnt = 0; // Cnt is between 0 and 9
Delay_ms(1000); // 1 second delay
}
```

Lab 9: Interfacing character LCD with PIC18f452

Objective:

This project describes the design of interfacing character LCD with PIC18f452 in 8-bit mode..

Equipment:

- EEDT 6.0 Microcontroller trainer board
- Personal computer.
- Programmer

Programming Steps:

- Create NEW PROJECT
- Select Device as PIC18F452 microcontroller
- Set Clock frequency 20 MHz
- Select Project file name

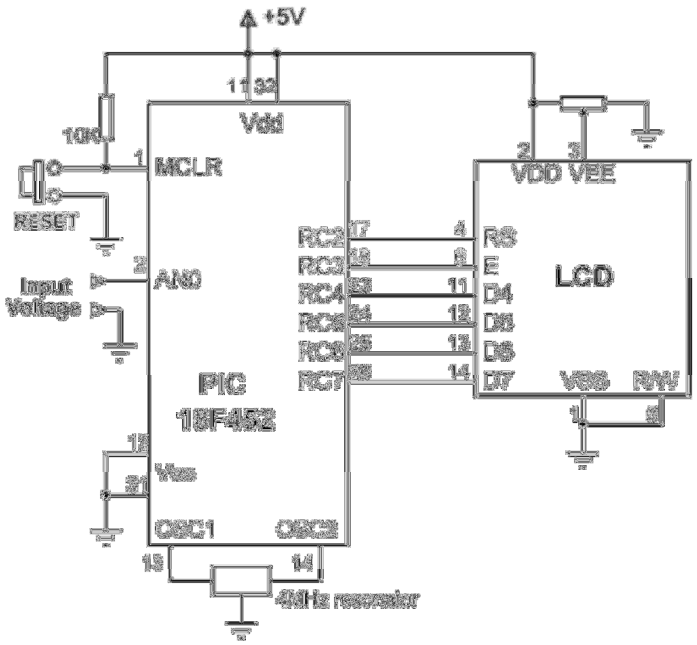


Figure 9.1. Interfacing Character LCD with PIC 18f452

```
/*
 * Project name:
     Lcd8_Test (Demonstration of the Lcd8 Library routines
*/

// Lcd8 module connections
sbit LCD8_RS at RC2_bit;
sbit LCD8_RW at RC3_bit;
sbit LCD8_EN at RC4_bit;
sbit LCD8_D7 at RD7_bit;
sbit LCD8_D6 at RD6_bit;
sbit LCD8_D5 at RD5_bit;
sbit LCD8_D4 at RD4_bit;
sbit LCD8_D3 at RD3_bit;
sbit LCD8_D2 at RD2_bit;
sbit LCD8_D1 at RD1_bit;
sbit LCD8_D0 at RD0_bit;

sbit LCD8_RS_Direction at TRISC2_bit;
sbit LCD8_RW_Direction at TRISC3_bit;
sbit LCD8_EN_Direction at TRISC4_bit;
sbit LCD8_D7_Direction at TRISD7_bit;
sbit LCD8_D6_Direction at TRISD6_bit;
sbit LCD8_D5_Direction at TRISD5_bit;
sbit LCD8_D4_Direction at TRISD4_bit;
sbit LCD8_D3_Direction at TRISD3_bit;
sbit LCD8_D2_Direction at TRISD2_bit;
sbit LCD8_D1_Direction at TRISD1_bit;
sbit LCD8_D0_Direction at TRISD0_bit;
// End Lcd8 module connections

char txt1[] = "Umm Al-Qura University";
char txt2[] = "Microcomputer Systems Design";
char txt3[] = "8bit Works";
char txt4[] = "----------";

char i;                          // Loop variable
void Move_Delay() {             // Function used for text moving
Delay_ms(500);          // You can change the moving speed here
}

void main(){

 ADCON1 |= 0x0F;                // Configure AN pins as digital
 CMCON  |= 7;                        // Disable comparators

  Lcd8_Init();                        // Initialize Lcd8
  Lcd8_Cmd(_LCD_CLEAR);               // Clear display
  Lcd8_Cmd(_LCD_CURSOR_OFF);          // Cursor off

  Lcd8_Out(1,6,txt3);               // Write text in first row
```

```
  Lcd8_Out(2,6,txt4);               // Write text in second row
  Delay_ms(2000);
  Lcd8_Cmd(_LCD_CLEAR);               // Clear display

  Lcd8_Out(1,1,txt1);              // Write text in first row
  Lcd8_Out(2,5,txt2);              // Write text in second row
  Delay_ms(2000);

  // Moving text
  for(i=0; i<5; i++) {       // Move text to the right 4 times
    Lcd8_Cmd(_LCD_SHIFT_RIGHT);
    Move_Delay();
  }

  while(1) {                           // Endless loop
    for(i=0; i<9; i++) {      // Move text to the left 7 times
      Lcd8_Cmd(_LCD_SHIFT_LEFT);
      Move_Delay();
    }

    for(i=0; i<9; i++) {     // Move text to the right 7 times
      Lcd8_Cmd(_LCD_SHIFT_RIGHT);
      Move_Delay();
    }

  }
}
```

Lab 10: Program to Run Stepper Motor using PIC Microcontroller

Objective:

The objective of this experiment is to run stepper motor using PIC 18 Microcontroller.

Equipment:

1. EEDT 6.0 Microcontroller trainer board
2. Personal computer.
3. PIC Programmer

Programming Steps:

1. Create NEW PROJECT
2. Select Device as PIC18F452 microcontroller
3. Set Clock frequency 20 MHz
4. Select Project file name.

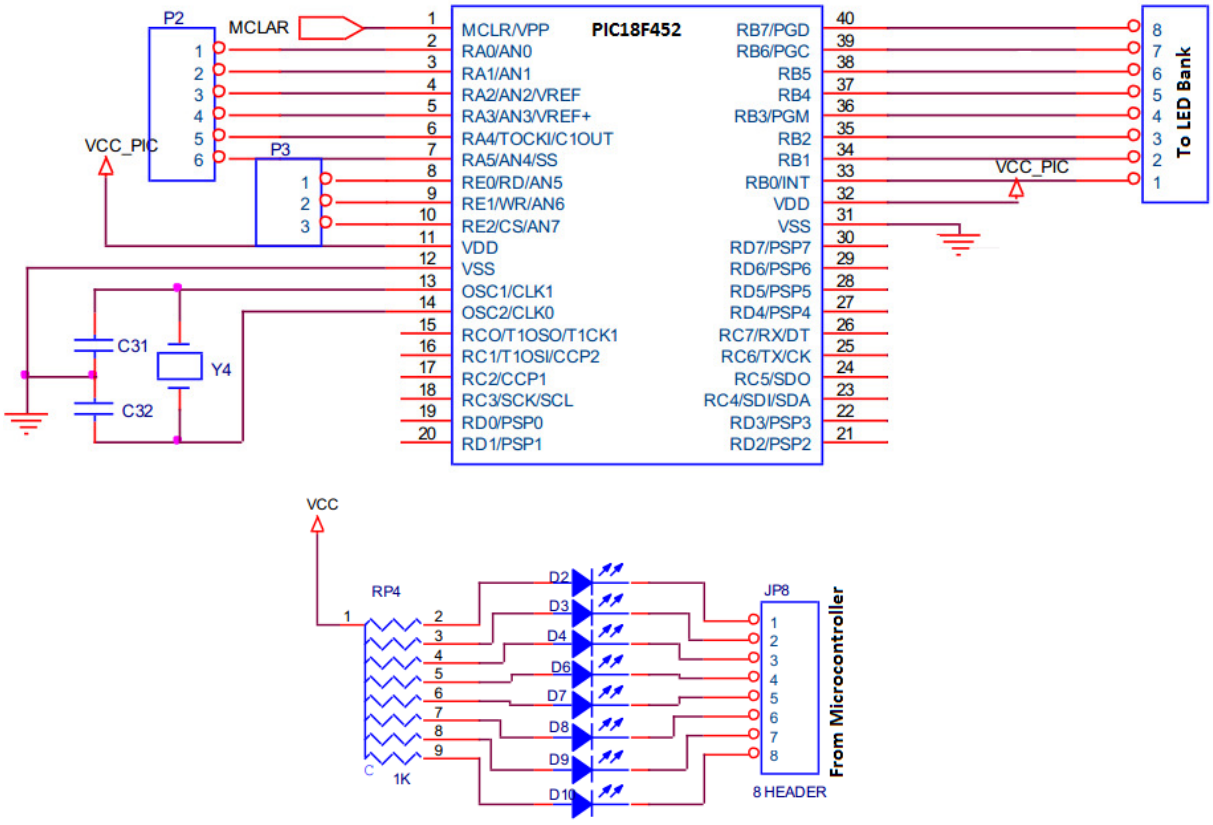


Figure: 10.1: PIC18F452 microcontroller and Bank of LEDs.

```
/************************************************************/
// Program to Run Stepper Motor using PIC18452               */
// Drive a stepper motor connected to port B                 */
//       RB1: Coil 1
//       RB2: Coil 2
//       RB3: Coil 3
//       RB4: Coil 4
/************************************************************/
char step[] = {5, 9, 10, 6};
void main(void) {
  char i;
        TRISB = 0;                /* PORT B is all output */
      PORTB = 0x0F;               /* Set Port B as FFH */
      i = 0;
      while(1) {
        PORTB = (step[i]);
        delay_ms(150);
        i++;
        if(i == 4)
          i = 0;
      }
    }
```

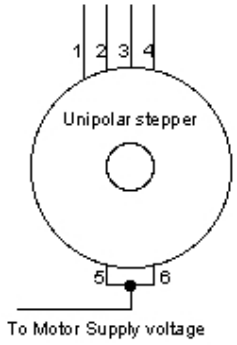


Figure: 10.1: Stepper Motor Wiring.

Lab 11: A-D Converter of PIC18F452 Microcontroller

Objective:

The objective of this experiment is to use Analog to Digital converter of microcontroller.

Equipment:

1. EEDT 6.0 Microcontroller trainer board
2. Personal computer.
3. Programmer

Programming Steps:

1. Create NEW PROJECT
2. Select Device as PIC18F452 microcontroller
3. Set Clock frequency 20 MHz
4. Select Project file name.

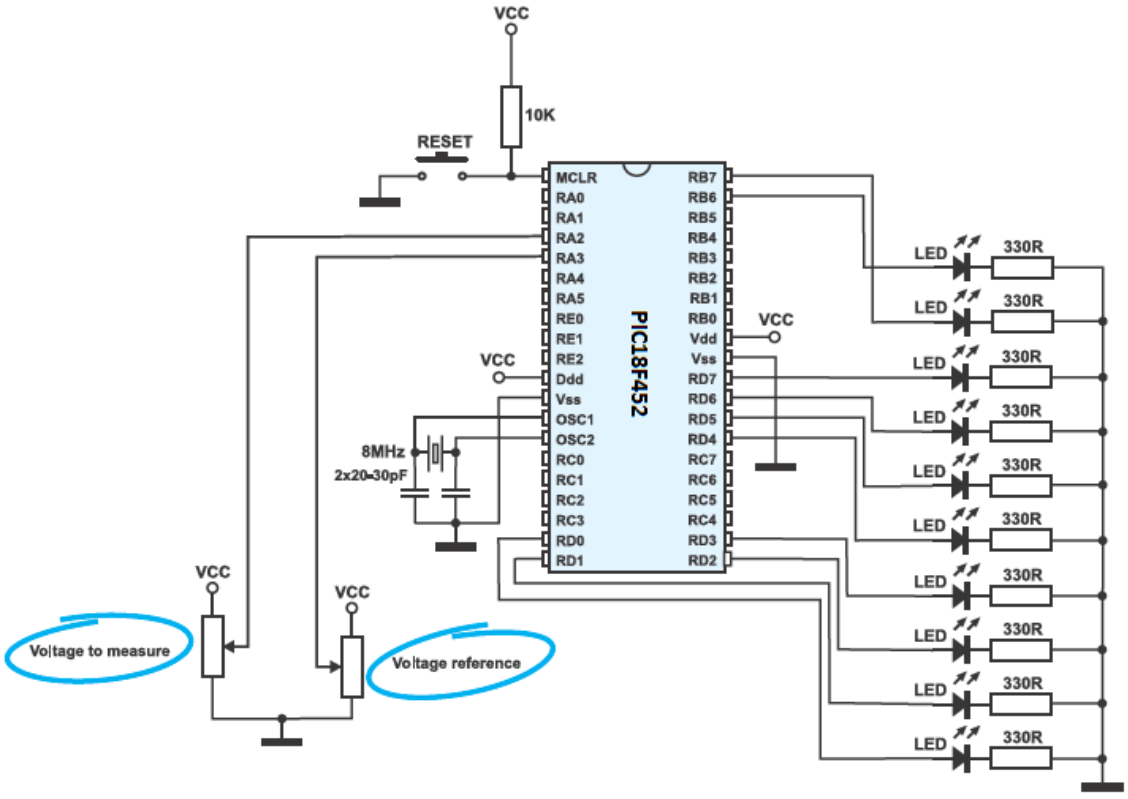


Figure: 11.1: PIC18F452 microcontroller ADC and Bank of LEDs.

In this experiment the PIC18F452 A/D converter is used. A variable analog signal is applied to the AN2 pin, while the 10-bit result of conversion is shown on ports B and D (8 LSBs on port D and 2 MSBs on port B). GND is used as negative voltage reference Vref-, while positive voltage reference is applied to the AN3 pin. It enables voltage measurement scale to 'stretch and shrink'.

In other words, the A/D converter always generates a 10-bit binary result, which means that it detects a total of 1024 voltage levels (210=1024). The difference between two voltage levels is not always the same. The less the difference between Vref+ and Vref-, the less the difference between two of 1024 levels. As seen, the A/D converter is able to detect slight changes in voltage.

```
/*********************************************************/
// Program to use ADC of PIC 18F452 microcontroller
/*********************************************************/

unsigned int temp_res;

void main() {
    ANSEL = 0x0C;              // Pins AN2 and AN3 as analog
    TRISA = 0xFF;
                               // All port A pins as inputs
    ANSELH = 0;
                               // Rest of pins is as digital
    TRISB = 0x3F;
                               // Port B pins RB7 and RB6
                               // configured as o/p
    TRISD = 0;
                               // Port D as outputs
    ADCON1.F4 = 1 ;
                               // Vref at RA3 pin.

    do {
        temp_res = ADC_Read(2);
                               // Result in temp_res
        PORTD = temp_res;
                               // 8 LSBs to Port D
        PORTB = temp_res >> 2;
                               // 2 MSBs at bits RB6 and RB7
    } while(1);
                               // Endless loop
}
```

Lab 12: Serial Communication using RS232

Objective:

The objective of this send and receive data from PC to microcontroller and vice versa.

Equipment:

4. EEDT 6.0 Microcontroller trainer board
5. Personal computer.
6. Programmer

Programming Steps:

5. Create NEW PROJECT
6. Select Device as PIC18F452 microcontroller
7. Set Clock frequency 20 MHz
8. Select Project file name.

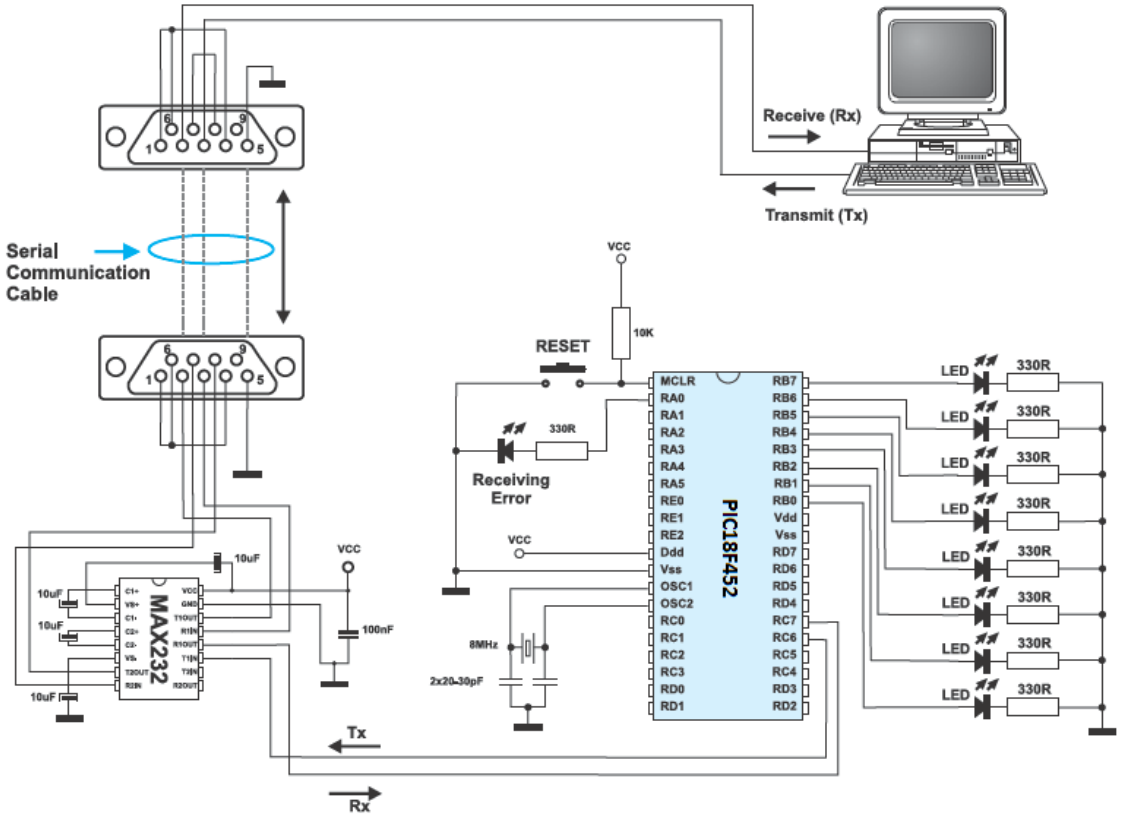


Figure: 12.1: PIC18F452 microcontroller serial TX and RX.

```
/************************************************************/
// Program to send and receive data b/w PC and microcontroller
/************************************************************/
unsigned short i;

void main() {
    UART1_Init(19200);              // Initialize USART module
                                    // (8 bit, 19200 baud rate,
no parity bit...)
    while (1) {
        if (UART1_Data_Ready()) {  // If data has been received
            i = UART1_Read();      // read it
            UART1_Write(i);        // and send it back
        }
    }
}
```

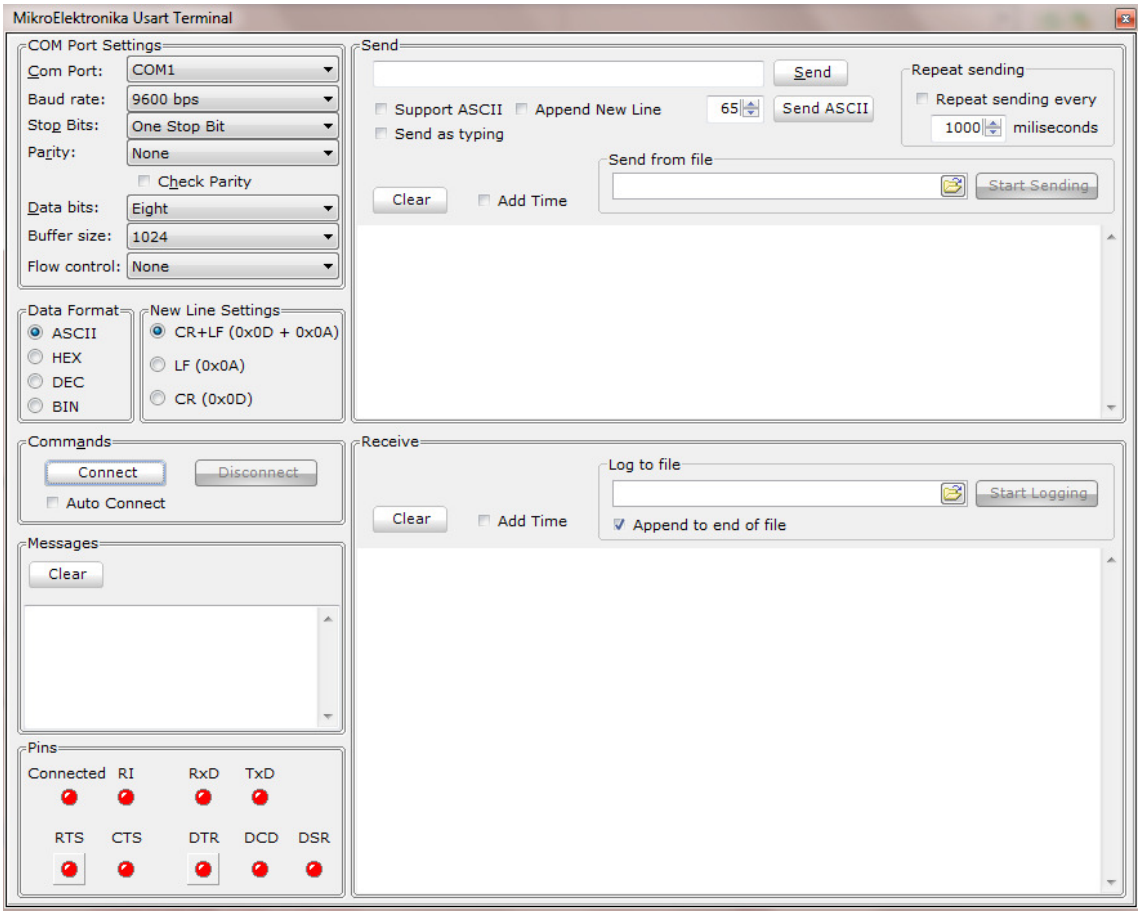


Figure: 12.2: USART Terminal of mikroC with PIC18F452 microcontroller.

Semester Project

Objective:

The objective of this project is to evaluate the students

Requirement:

In this project a PC is connected to a PIC18F452 microcontroller.

The project is a simple integer calculator.

User enters the numbers using the PC keyboard.

Results are displayed on the PC monitor.

The following operations can be performed: (+ − ∗ /)

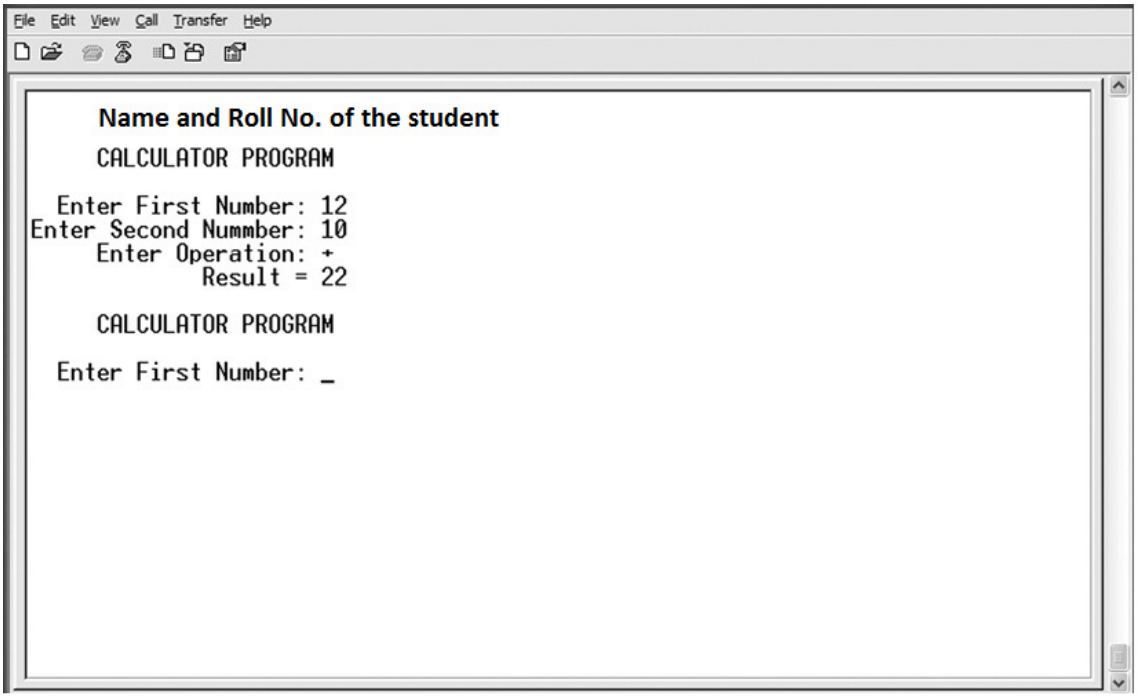


Figure: 13.1: Hyper Terminal Screen of the Project

```
;*****************************************************************
; Student Name:                              Student ID:
;*****************************************************************
```