Nutan Maharashtra Vidya Prasarak Mandal's

NUTAN MAHARASHTRA INSTITUTE OF ENGINEERING AND TECHNOLOGY

ESTD : 1906

Department of Electronics & Telecommunication Engineering

LABORATORY MANUAL

SUBJECT: Microcontrollers Lab

[SUBJECT CODE: 304188]

CLASS: T.E. ENTC

ACADEMIC YEAR: 2023-24

PREPARED BY: **APPROVED BY:**

(Prof.Sarika N.Patil) **H.O.D. [E&TC]**

Vision and Mission of the Institute

1. **Vision of the Institute-**To be a notable institution for providing quality technical education, ensuring ethical, moral, and holistic development of students.
2. **Mission of the Institute**- To nurture engineering graduates with highest technical competence, professionalism and problem solving skills to serve the needs of industry and society.

Vision and Mission of the Department

Department Vision:

To be a renowned department of Electronics and Telecommunication engineering for providing quality technical education through holistic development of the students.

Department Mission:

3. To impart quality technical education for students with continuous upgraded teaching learning process.
4. To enhance employability and entrepreneurship through Industry Institute association.
5. To enhance the research competency in students by adapting state of art technology in Electronics and Communication Engineering.
6. To inculcate the needs of profession for the society.

Program Educational Objectives (PEOs):

1. To provide graduates with a professional career in Electronics, Communication, and allied disciplines.
2. To develop managerial and entrepreneurial skills among the students to solve societal problems.
3. To impart analytical skills in order to develop a multidisciplinary approach to relate engineering issues and innovations.
4. To inculcate effective communication skills, teamwork spirit and professional ethics in students to meet employer needs at large and prepare them for higher studies.

Program Outcomes

1. **Engineering knowledge**:

Graduates can apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to Civil Engineering related problems.

2. **Problem analysis**:

An ability to identify, formulate, review research literature, and analyse Civil engineering problems reaching substantiated conclusions using principles of mathematics and engineering sciences.

3. **Design/development of solutions**:

An ability to plan, analyse, design, and implement engineering problems and design system components or processes to meet the specified needs.

4. **Conduct investigations of complex problems**:

An ability to use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**:

An ability to apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**:

An ability to apply contextual knowledge to assess societal, legal issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**:

An ability to understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**:

An ability to apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and teamwork**:

An ability to function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings to accomplish a common goal.

10. **Communication**:

An ability to communicate effectively on engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation and make effective presentations.

11. **Project management and finance**:

Ability to demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**:

An ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEOs)

1. To provide graduates with a professional career in Electronics, Communication and allied disciplines.
2. To develop managerial and entrepreneurial skills among the students to solve societal problems.
3. To impart analytical skills in order to develop a multidisciplinary approach to relate engineering issues and innovations.
4. To inculcate effective communication skills, teamwork spirit and professional ethics in students to meet employer needs at large and prepare them for higher studies

Program Specific Outcomes (PSOs)

PSO1: An ability to apply design and development of complex system in the areas of signal processing, Semiconductor Technologies, Communication, Embedded and Power System.

PSO2: Demonstrate proficiency in the use of Software and Hardware for the need of Industry and society.

Course Outcomes (CO)

EC307.1	Implement fundamental programming concepts of 8 bit microcontrollers.
EC307.2	Interface peripherals to the 8 bit microcontrollers.
EC307.3	Design microcontroller based system using serial protocols.

List of Experiments with Mapping

Sr. No	Name of the Experiment	CO mapping	Level of CO mapping
1	Simple programmes on Memory transfer	**EC307.1**	3
2	Parallel port interacting of LEDS	**EC307.2**	3
3	Interfacing of Stepper motor to 8051- software delay using Timer OR Interfacing of DAC with 8051 to generate waveforms	**EC307.3**	3
4	Interfacing Of 7-Segment Display	**EC307.3**	3
4	Write a program for interfacing button, LED, relay & buzzer with PIC 18FXX.	**EC307.4**	3
5	Interfacing of LCD to PIC 18FXXXX.	**EC307.4**	3
6	Generate square wave using timer with interrupt	**EC307.5**	3
7	Generation of PWM signal for DC Motor control	**EC307.5**	3
8	Interfacing serial port with PC both side communication OR Interface analog Voltage 0-5 V to Internal ADC and display value on LCD	**EC307.6**	3

Rubrics for Evaluation

Sr. No	Evaluation Criteria	Marks for each Criteria	Rubrics
1	Timely submission	5 or 10	Punctuality reflects the work ethics. Students should reflect that work ethics by completing the lab assignments and reports in a timely manner .
2	Journal Presentation	5 or 10	Students are expected to prepare the journal. The journal presentation of the course should be complete, clear, and understandable.
3	Performance	5 or 10	After performance, the students should have good knowledge of the experiment.
4	Understanding	5 or 10	The student should be able to explain methodology used for designing and developing the program/solution. Student should clearly understand the purpose of the assignment and its outcome.
5	Oral	5 or 10	The student should be able to answer the questions related to the lab assignments.

EXPERIMENT NO.1

Aim:

A. To study Memory organization of 8051 microcontroller .
B. Write 8051Program for simple programs on memory transfer.

Objective:

1. To be familiar with data transfer between 8051 microcontroller and internal / external memory

Apparatus: PC, Keil simulator

Theory:

Memory Organization

The 8051 has two types of memory and these are Program Memory and Data Memory. Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller. Depending on the model in use (we are still talking about the 8051 microcontroller family in general) at most a few Kb of ROM and 128 or 256 bytes of RAM is used. However All 8051 microcontrollers have a 16-bit addressing bus and are capable of addressing 64 kb memory. It is neither a mistake nor a big ambition of engineers who were working on basic core development. It is a matter of smart memory organization which makes these microcontrollers a real "programmers' goody".

Program Memory

The first models of the 8051 microcontroller family did not have internal program memory. It was added as an external separate chip. These models are recognizable by their label beginning with 803 (for example 8031 or 8032). All later models have a few Kbyte ROM embedded. Even though such an amount of memory is sufficient for writing most of the programs, there are situations when it is necessary to use additional memory as well. A typical example is so called lookup tables. They are used in cases when equations describing some processes are too complicated or when there is no time for solving them. In such cases all necessary estimates and approximates are executed in advance and the final results are put in the tables (similar to logarithmic tables).

EA=0 In this case, the microcontroller completely ignores internal program memory and executes only the program stored in external memory.

EA=1 In this case, the microcontroller executes first the program from built-in ROM, then the program stored in external memory.

In both cases, P0 and P2 are not available for use since being used for data and address transmission. Besides, the ALE and PSEN pins are also used.

Data Memory

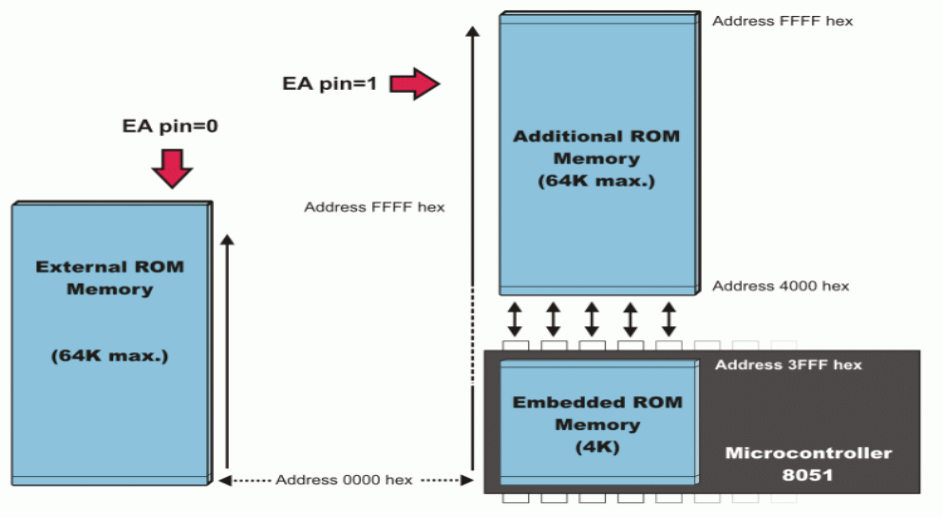
As already mentioned, Data Memory is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller. Besides, RAM memory built in the 8051 family includes many registers such as hardware counters and timers, input/output ports, serial data buffers etc. The previous models had 256 RAM locations, while for the later models this number was incremented by additional 128 registers. However, the first 256 memory locations (addresses 0-FFh) are the heart of memory common to all the models belonging to the 8051 family. Locations available to the user occupy memory space with addresses 0-7Fh, i.e. first 128 registers. This part of RAM is divided in several blocks.

The first block consists of 4 banks each including 8 registers denoted by R0-R7. Prior to accessing any of these registers, it is necessary to select the bank containing it. The next memory block (address 20h-2Fh) is bit- addressable, which means that each bit has its own address (0-7Fh). Since there are 16 such registers, this block contains in total of 128 bits with separate addresses (address of bit 0 of the 20h byte is 0, while address of bit 7 of the 2Fh byte is 7Fh). The third group of registers occupies addresses 2Fh-7Fh, i.e. 80 locations, and does not have any special functions or features.

Additional RAM

In order to satisfy the programmers' constant hunger for Data Memory, the manufacturers decided to embed an additional memory block of 128 locations into the latest versions of the 8051 microcontrollers. However, it's not as simple as it seems to be… The problem is that electronics performing addressing has 1 byte (8 bits) on disposal and is capable of reaching only the first 256 locations, therefore. In order to keep already existing 8-bit architecture and compatibility with other existing models a small trick was done.

What does it mean? It means that additional memory block shares the same addresses with locations intended for the SFRs (80h- FFh). In order to differentiate between these two physically separated memory spaces, different ways of addressing are used. The SFRs memory locations are accessed by direct addressing, while additional RAM memory locations are accessed by indirect addressing.



Bus: Fundamentally Bus is a group of wires which functions as a communication canal or mean for the transfer Data. These buses comprise of 8, 16 or more cables. As a result, a bus can bear 8 bits, 16 bits all together. There are two types of buses:

1. **Address Bus:** Microcontroller 8051 consists of 16 bit address bus. It is brought into play to address memory positions. It is also utilized to transmit the address from Central Processing Unit to Memory.
2. **Data Bus:** Microcontroller 8051 comprise of 8 bits data bus. It is employed to cart data.

There are 4 8-bit ports: P0, P1, P2 and P3.

PORT P1 (Pins 1 to 8): The port P1 is a general purpose input/output port which can be used for a variety of interfacing tasks. The other ports P0, P2 and P3 have dual roles or additional functions associated with them based upon the context of their usage.The port 1 output buffers can sink/source four TTL inputs. When 1s are written to portn1 pins are pulled high by the internal pull-ups and can be used as inputs.

PORT P3 (Pins 10 to 17): PORT P3 acts as a normal IO port, but Port P3 has additional functions such as, serial transmit and receive pins, 2 external interrupt pins, 2 external counter inputs, read and write pins for memory access.

PORT P2 (pins 21 to 28): PORT P2 can also be used as a general purpose 8 bit port when no external memory is present, but if external memory access is required then PORT P2 will act as an address bus in conjunction with PORT P0 to access external memory. PORT P2 acts as A8-A15.

PORT P0 (pins 32 to 39) PORT P0 can be used as a general purpose 8 bit port when no external memory is present, but if external memory access is required then PORT P0 acts as a multiplexed address and data bus that can be used to access external memory in conjunction with PORT P2. P0 acts as AD0-AD7.

Algorithm:

A) To add Five 8 bit no.s stored in internal RAM and store result in internal RAM

1. Initialize address for internal memory.
2. Initialize counter of 04
3. Take first Byte in Acc.
4. Increment address.
5. Add byte along with carry to Acc.
6. Decrement counter if it is not zero jump to step 4.
7. Store result from acc to internal memory.
8. Stop

B] To Transfer Block of 5 Bytes From internal RAM to Internal RAM.

1. Initialize Source Address in R0
2. Initialize Destination Address in R1
3. Initialize Count of 05H in Register R2
4. Move fist no. in ACC from source Address
5. Move Acc Data to Destination Address
6. Increment source and destination Address
7. Decrement R2 and check if it is Zero
8. If R2 is not zero then Jump to step-4

9.Stop

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 02

Aim:

A. To study Port Structure of 8051 microcontroller
B. To write Embedded C Program for Parallel port interacting of LEDS—Different programs(flashing, Counter, BCD, HEX, Display of Characteristic)

Objective:

1. To be familiar with software delay calculations using 8051 microcontroller
2. To find functions of P0,P1,P2,P3 of 8051 Microcontroller

Apparatus: PC, Keil simulator

Theory: Examining Figure 2.1, note that of the 40 pins, a total of 32 pins are set aside for the four ports P0, P1, P2, and P3 where each port takes 8 pins. The rest of the pins are designated as V_{cc}, GND, XTAL1, XTAL2, RST, EA, PSEN, and ALE**.**

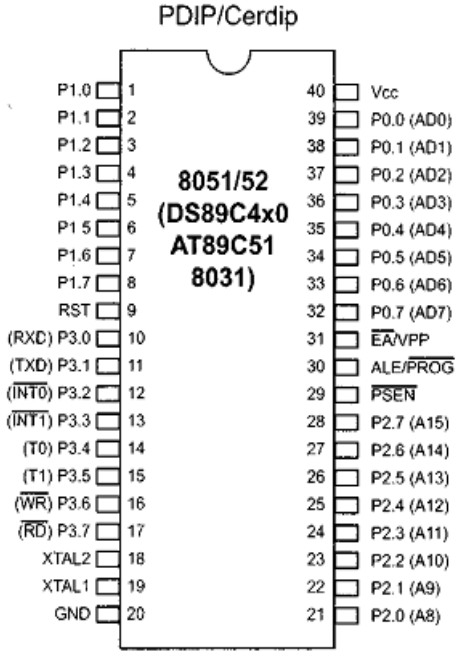


Figure 2.1 Pin Diagram of 8051 Microcontroller

Of these pins, six (V_{cc}, GND, XTAL1, XTAL2, RST, and EA) are used by all members of the 8051 and 8031 families. In other words, they must be connected in order for the system to work, regardless of whether the microcontroller is of the 8051 or 8031 family. The other two pins, PSEN and ALE, are used mainly in 8031-based systems. We first describe the function of each pin. Ports are discussed separately.

***Vcc*-** Pin 40 provides supply voltage to the chip. The voltage source is +5V.

GND-Pin 20 is the ground.

XTAL1 and XTAL2

The 8051 has an on-chip oscillator but requires an external clock to run it. Most often a quartz crystal oscillator is connected to inputs XTAL1 (pin 19) and XTAL2 (pin 18). The quartz crystal oscillator connected to XTAL1 and XTAL2 also needs two capacitors of 30 pF value. One side of each capacitor is connected to the ground

It must be noted that there are various speeds of the 8051 family. Speed refers to the maximum oscillator frequency connected to XTAL. For example, a 12-MHz chip must be connected to a crystal with 12 MHz frequency or less. Likewise, a 20-MHz microcontroller requires a crystal frequency of no more than 20 MHz. When the 8051 is connected to a crystal oscillator and is powered up, we can observe the frequency on the XTAL2 pin using the oscilloscope.

If you decide to use a frequency source other than a crystal oscillator, such as a TTL oscillator, it will be connected to XTAL1; XTAL2 is left unconnected

RST- Pin 9 is the RESET pin. It is an input and is active high (normally low). Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities This is often referred to as a *power-on reset.* Activating a power-on reset will cause all values in the registers to be lost. It will set program counter to all Os. Figures 2-3 (a) and (b) show two ways of connecting the RST pin to the power-on reset circuitry.

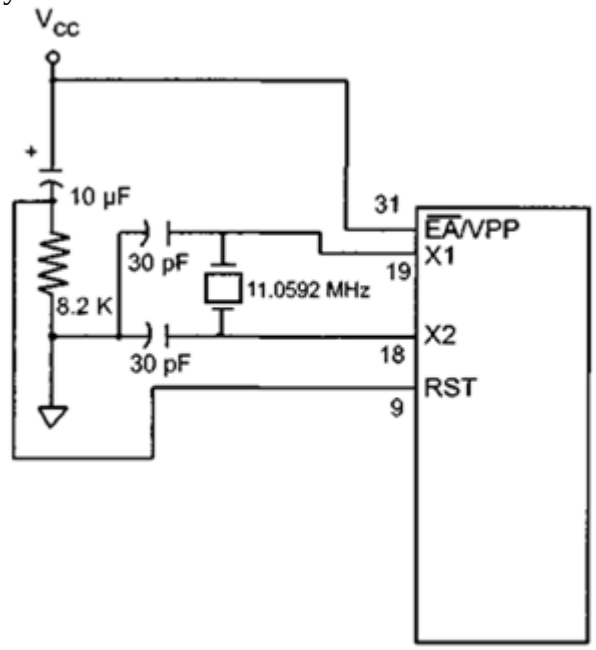


Figure 2-3(a) RST pin to the power-on reset circuitry.

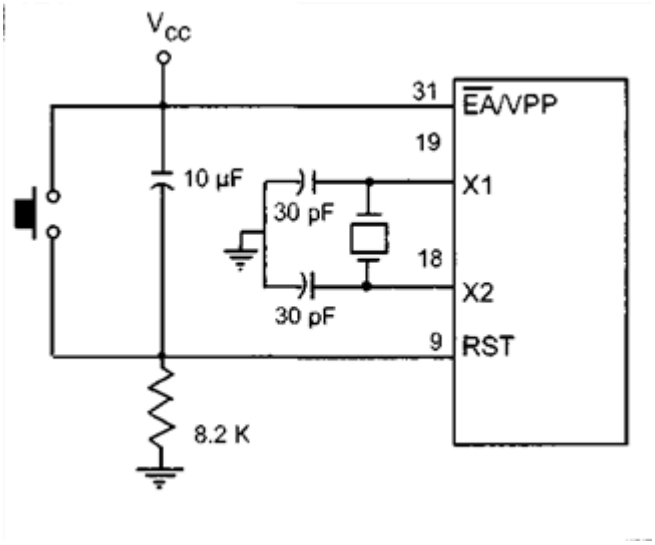


Figure 2-3(b) a momentary switch for reset circuit.

Figure 2-3 (b) uses a momentary switch for reset circuitry.In order for the RESET input to be effective, it must have a minimum duration of two machine cycles. In other words, the high pulse must be high for a minimum of two machine cycles before it is allowed to go low. Here is what the Intel manual says about the Reset circuitry: "When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles." Although, an 8.2K-ohm resistor and a 10-uF capacitor will take care of the vast majority of the cases, you still need to check the data sheet for the 8051 you are using.

EA-The 8051 family members, such as 8751/52, 89C51/52, or DS89C4xO, all come with on-chip ROM to store programs. In such cases, the EA pin is connected to V_{cc}. For family members such as the 8031 and 8032 in which there is no on-chip ROM. code is stored on an external ROM and is fetched by the 8031/32. Therefore, for the 8031 the EA pin must be connected to GND to indicate that the code is stored externally. EA. which stands for "external access," is pin number 31 in the DIP packages. It is an input pin and must be connected to either V_{cc} or GND. In other words, it cannot be left unconnected.

PSEN-This is an output pin. PSEN stands for "program store enable." In an 8031-based system in which an external ROM holds the program code, this pin is connected to the OE pin of the ROM.

ALE-ALE (address latch enable) is an output pin and is active high. When connecting an 8031 to external memory, port 0 provides both address and data. In other words, the 8031 multiplexes address and data through port 0 to save pins. The ALE pin is used for demultiplexing the address and data by connecting to the G pin of the 74LS373 chip.

Ports 0, 1, 2 and 3

As shown in Figure 2-1 the four ports PO, PI, P2, and P3 each use 8 pins, making them 8-bit ports. All the ports upon RESET are configured as input, since PO – P3 have value FFH on them. The following is a summary of features of PO – P3

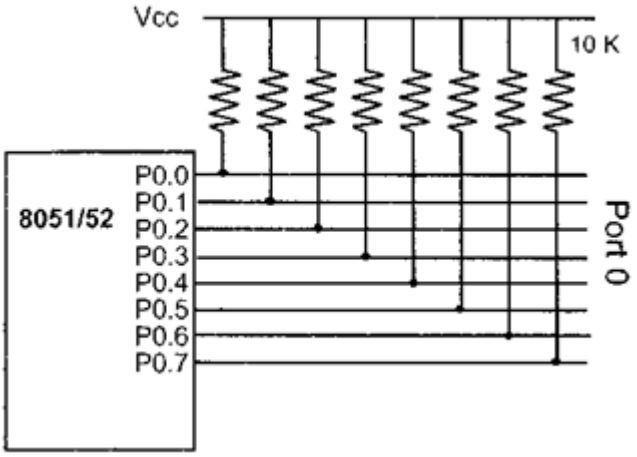


Figure 2-4. Port 0 with Pull-Up Resistors

As shown in Figure 2-1. Port 0 is also designated as ADO – AD7, allowing it to be used for both address and data. When connecting an 8051/31 to an external memory. Port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save pins. ALE indicates if PO has address or data. When ALE = 0. It provides data DO – D7. But when ALE = 1 it has address A0 – A7". Therefore ALE is used for de-multiplexing address and data with the help of a "4LS373 latch,. In the 8051-based systems where there is no external memory connection, the pins of PO must be connected externally to a 1 OK-ohm pull-up resistor. This is due to the fact that PO is an open drain, unlike P1, P2.and P3. *Open drain* is a term used for MOS chips in the same way that *open collector* is used for TTL chips. In many systems using the 8751, 89C51. or DS89C4xO chips, we normally connect PO to pull-up resistors. See Figure 8-4. With external pull-up resistors connected to PO, it can be used as a simple I/O port, just like PI and P2. In contrast to port 0, ports PI, P2, and P3 do not need any pull-up resistors since they already have pull-up resistors internally. Upon reset, ports PI, P2, and P3 are configured as input ports.

P1 and P2

In 8051-based systems with no external memory connection, both P1 and P2 are used as simple I/O. However, in 8031/51-based systems with external memory connections, port 2 must be used along with P0 to provide the 16-bit address for the external memory. As shown in Figure 2-1, port 2 is also designated as A8 – A15, indicating its dual function. Since an 8031/51 is capable of accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. While P0 provides the lower 8 bits via A0 – A7, it is the job of P2 to provide bits A8 – A15 of the address. In other words, when the 8031/51 is connected to external memory, P2 is used for the upper 8 bits of the 16-bit address, and it cannot be used for I/O. This is discussed in detail in Chapter 14.

From the discussion so far, we conclude that in systems based on 8051 microcontrollers, we have three ports, P0, P1, and P2, for I/O operations. This should be enough for most microcontroller applications. That leaves port 3 for interrupts as well as other signals.

Port3

Port 3 occupies a total of 8 pins, pins 10 through 17. It can be used as input or output. P3 does not need any pull-up resistors, the same as PI and P2 did not. Although port 3 is configured as an input port upon reset, this is not the way it is most commonly used. Port 3 has the additional function of providing some extremely important signals such as interrupts. Table 2-1 provides these alternate functions of P3. This information applies to both 8051 and 8031 chips.

P3.0 and P3.1 are used for the RxD and TxD serial communications signals. S. Bits P3.2 and P3.3 are set aside for external interrupts. Bits P3.4 and P3.5 are used for Timers 0 and 1, Finally, P3.6 and P3.7 are used to provide the WR and RD signals of external memory connections. In systems based on the 8051, pins 3.6 and 3.7 are used for I/O while the rest of the pins in port 3 are normally used in the alternate function role.

Table 2-1: Port 3 Alternate Functions

P3Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Program counter value upon reset

Activating a power-on reset will cause all values in the registers to be lost. Table 2-2 provides a partial list of 8051 registers and their values after power-on reset. From Table 8-2 we note that the value of the PC (program counter) is 0 upon reset, forcing the CPU to fetch the first opcode from ROM memory location 0000. This means that we must place the first byte of opcode in ROM location 0 because that is where the CPU expects to find the first instruction.

Table 2-2: RESET Value of Some 8051 Registers

Register	Reset Value (hex)
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF

Machine cycle and crystal frequency

The 8051 uses one or more machine cycles to execute an instruction. The period of machine cycle varies among the different versions of 8051 from 12 clocks in the AT89C51 to 1 clock in the DS89C4xO chip. See Table 2-3. The frequency of the crystal oscillator connected to the X – X2 pins dictates the speed of the clock used in the machine cycle. From Table 2-3, we can conclude that using the same crystal frequency of 12 MHz for both the AT89C51 and DS89C4xO chips gives performance almost 12 times better from the DS89C4xO chip.

Table 2-3: Clocks per Machine Cycle (MC) for Various 8051 Versions

Chip (Maker)	Clocks per Machine Cycle
AT89C51/52 (Atmel)	12
P89C54X2 (Phillips)	6
DS5000 (Dallas Semiconductor)	4
DS89C4x0 (Dallas Semiconductor)	1

Calculations of Count of Timer: Delay generation of 5ms .Assume XTAL is 11.0592 MHz

Interfacing Diagram: LED to 8051

Algorithm:

A] Main Program:

1. Initialize port as output
2. Set all port pins to logic high
3. Call Delay
4. Clear all Port pins
5. Call Delay
6. Jump to step 2

B] Delay Subroutine:

1.Initialize Timer 0/1 selecting required Mode
2.Load value of TH0 and TL0
3.Clear TF flag
4. Start timer
5. Check TF flag .If it is not set keep checking.
6.Clear TF and stop timer
7. Return back to Main Program

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 03 A

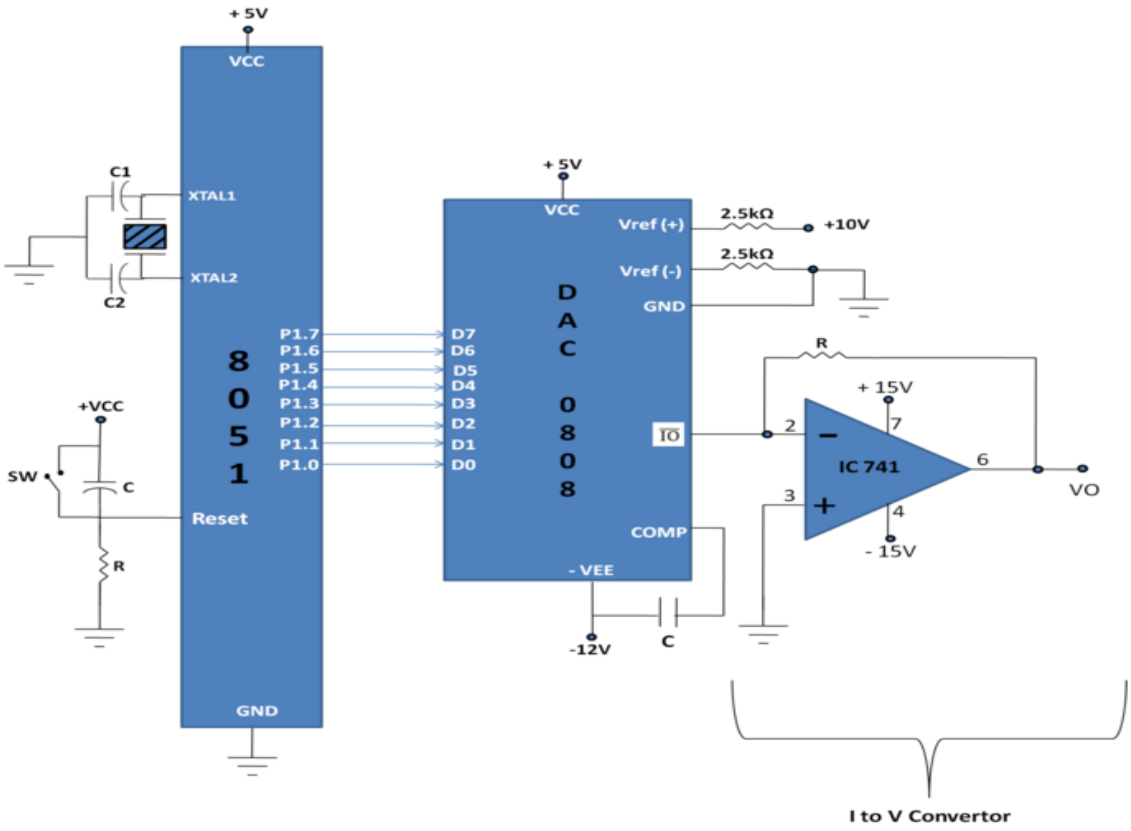
Aim: Waveform Generation using DAC 0808

Objective:

1. To write assembly program to interface DAC 0808 with microcontroller 8051 to generate square wave and triangular wave.

Apparatus: PC, keil simulator, 8051 controller kit, LED Kit, Flash magic software

Theory: Microcontroller are used in wide variety of applications like for measuring and control of physical quantity like temperature, pressure, speed, distance, etc.
In these systems microcontroller generates output which is in digital form but the controlling system requires analog signal as they don't accept digital data thus making it necessary to use DAC which converts digital data into equivalent analog voltage.
In the figure shown, we use 8-bit DAC 0808. This IC converts digital data into equivalent analog Current. Hence we require I to V converter to convert this current into equivalent voltage.



- According to theory of DAC Equivalent analog output is given as:

$$V0=Vref\left[\frac{D0}{2}+\frac{D1}{4}+\frac{D2}{8}+\frac{D3}{16}+\frac{D4}{32}+\frac{D5}{64}+\frac{D6}{128}+\frac{D7}{256}\right]$$

Ex:
1. IF data =00H [00000000], Vref= 10V

$$V0=10\left[\frac{0}{2}+\frac{0}{4}+\frac{0}{8}+\frac{0}{16}+\frac{0}{32}+\frac{0}{64}+\frac{0}{128}+\frac{0}{256}\right]$$

Therefore, V0= 0 Volts.

2. If data is 80H [10000000], Vref= 10V

$$V0=10\left[\frac{1}{2}+\frac{0}{4}+\frac{0}{8}+\frac{0}{16}+\frac{0}{32}+\frac{0}{64}+\frac{0}{128}+\frac{0}{256}\right]$$

Therefore, V0= 5 Volts.

I_{ref} current is generally set to 2.0 mA.

Different Analog output voltages for different Digital signal is given as:

DATA	OUTPUT VOLTAGE
00H	0V
80H	5V
FFH	10V

With the power supply voltage Now assuming that I_{ref} = 2 mA, if all the inputs to the DAC are high, the maximum output current is 1.99 mA

Algorithm:

A]To generate Square Wave

1]Initialize Port as Output
2] Send Data FFH to PORT
3]Call Delay
4] Send Data 00H to PORT
5]Call Delay
6] Jump to Step 2

B] To Generate Triangular Wave

1]Initialize Port as Output
2] Send Data 00H to PORT
3]Increment Data by 1
4] Jump to Step 2

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 03 B

Aim:

A .To study working of stepper motor.

B. Write Embedded C Program to interface 8051 with Stepper motor to rotate motor in clockwise and Anticlockwise direction.

Objective:

1 To be familiar with interfacing of stepper motor with 8051 microcontroller

2 To be familiar with the calculations of different step angles and speeds of stepper motor

Apparatus: PC keil simulator, 8051 controller kit, Flash magic software , Stepper motor and driver kit.

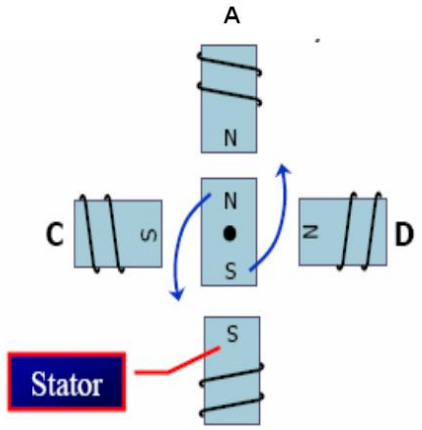
Theory:

Stepper Motor

• A stepper motor is a widely used device that translates electrical pulses into mechanical movement

• The stepper motor is used for position control in applications such as disk drivers, dot matrix printers, and robotics, etc.

• Every stepper motor has a permanent magnet rotor (also called the shaft) surrounded by a stator

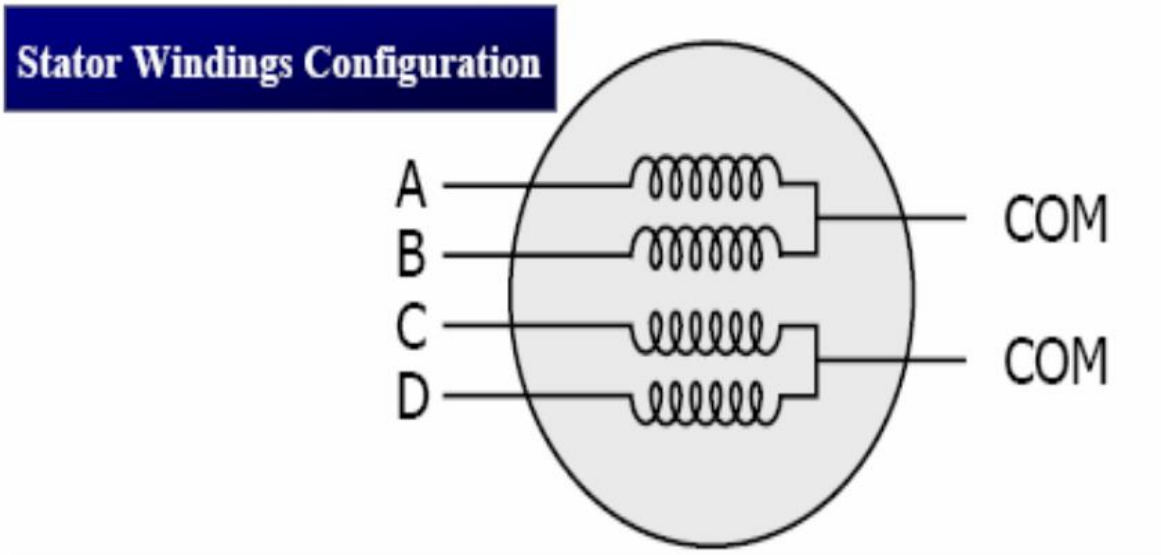


The most common stepper motors have four stator windings that are paired with a center-tapped common.

• This type of stepper motor is commonly referred to as a four-phase stepper motor

• The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator

Stator Windings Configuration



The stepper motor discussed here has a total of 6 leads, 4 leads representing the four stator windings, 2 commons for the center tapped leads.

• As the sequence of power is applied to each stator winding, the rotor will rotate.

Normal 4-Step Sequence

Clockwise

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Counter-Clockwise

The step angle is the minimum degree of rotation associated with a angles for various motors

• The step per revolution is the total number of steps needed to rotate one complete rotation or 360 degrees. To allow for finer resolutions, all stepper motors allow what is called an 8-step switching sequence.

• It's also called half-stepping, since each step is half of the normal step angle

Half-Step 8-Step Sequence

Clockwise

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

Counter-Clockwise

The relation between RPM (revolutions per minute), steps per revolution, and steps per second is as follow

- Steps/Sec = RPM x Steps per revolution / 60
- After completing every four steps, the rotor moves only one tooth pitch
- The smaller the step angle, the more teeth the motor passes
- ex. In a stepper motor with 200 steps per revolution, its rotor has 50 teeth since 4 x 50 = 200 steps are needed.

Interfacing Diagram :

Calculation Of Timer count Value :

Algorithm:

1]Start

2] Initialize PORT as Output

3]Send value of Sequence

4] Call Delay

5] jump to step 3

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 04

Aim: write a program for interfacing button, LED, relay & buzzer.

Objective: To write a embedded C program for interfacing button, LED, relay & buzzer as follows

A. when button 1 is pressed relay and buzzer is turned ON and LED's start chasing from left to right

B. when button 2 is pressed relay and buzzer is turned OFF and Led start chasing from right to left

Apparatus: MPLAB Simulator, C18, PC, PIC18f Development KIT, LED KIT, USB cable, mikrobootloader.

Features of PIC18F4550:

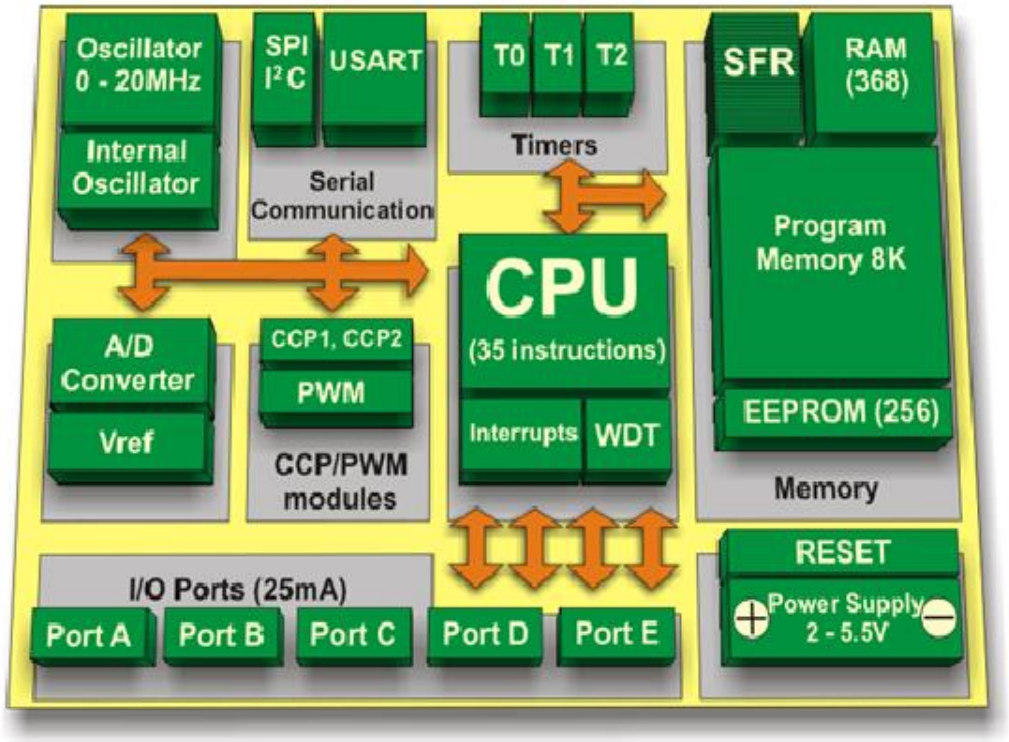
- PIC18F4550 belongs to the PIC18F family; PIC18F4550 is an 8bit microcontroller and uses RISC architecture. PIC18F4550 has 40 pins in PDIP (dual in line package) and 44 pin in TQFP (Quad flat package).
- 32KB flash memory, 2048 bytes of SRAM (synchronous Random Access memory), EEPROM (Electrically Erasable Program Read Only Memory) of 256 bytes are embedded in the PIC18F4550.
- It has 35 I/O pins for interfacing and communication with other peripherals, 13channel of 10bit analog to digital converters which are used for interfacing and communicating the analog peripherals (DC motor, LDR, etc.).
- It has 2 CCP and 1 ECCP module that is enhanced capture and compare module which is mainly used for modulation and waveform generation functions. CCP module is of 16bit register works as 16 capture bit register, 16 compare bit register, and PWM and duty cycle register.
- PIC18F4550 has SPI (serial peripheral interface) and i2c (inter integrated circuit) for master and slave modes. It has SPP (Streaming Parallel Port) for USB streaming transfer.
- PIC18F4550 is embedded with 4 timer modules (timer0 to timer3), 2 comparator modules and 3 external interrupt. It has Dual Oscillator options allow microcontroller and USB module to run at different clock speeds. It can operate in 2.0V to 5.5V

Peripheral Interface Controller (PIC) is microcontroller developed by Microchip, PIC microcontroller is fast and easy to implement program when we compare other microcontrollers like 8051. The ease

of programming and easy to interfacing with other peripherals PIC became successful microcontroller.

We know that microcontroller is an integrated chip which consists of RAM, ROM, CPU, TIMERS, and COUNTERS etc. PIC is a microcontroller which also consists of RAM, ROM, CPU, timers, counter, ADC (analog to digital converters), DAC (digital to analog converter). PIC also supports the protocols like CAN, SPI, UART for interfacing with other peripherals. PIC mainly used modified Harvard architecture and also supports RISC (Reduced Instruction Set Computer) by the above specification RISC and Harvard we can easily that PIC is faster than the 8051 based controller which is made-up of Von-Newman architecture.

PIC Microcontroller Architecture:



CPU: CPU is not different from other microcontrollers CPU. PIC microcontroller CPU consists of Arithmetic logic unit (ALU), memory unit (MU), control unit (CU), Accumulator etc. we know that ALU mainly used for arithmetic operations and taking the logical decisions, memory used for storing the instruction which is to processed and also storing the instructions after processing, Control unit is used for controlling the all the peripherals which are connected to the CPU both internal peripherals and external peripherals. Accumulator is used for storing the results and used for further processing. As I said earlier PIC micro controller supports the RISC architecture that is reduced instruction set

computer, if a computer or controller is said that it supports reduced instruction set you should remember the following points:

1. RISC has very few instructions (approx. ~ 35) which are used in the program.
2. Length of the instruction is small and fixed and takes same amount of time for processing.
3. As the instruction is small it will take less time to process another words CPU will be fast.
4. Compiler need not be complex and debugging will be very easy in the programmer point of view.

Memory: Memory module in the PIC consists of RAM, ROM and STACK

RAM: we know that RAM (Random Access Memory) which is a volatile memory used for storing the data temporarily in its registers. RAM memory is divided in to Banks, in each banks we have number of registers. The RAM registers is divided into 2 types. They are General purpose registers (GPR) and Special purpose registers (SPR).

1. **GPR:** general purpose registers as the name implies for general usage. For example if we want to multiply any two numbers using PIC we generally take two registers for storing the numbers and multiply the two numbers and store the result in other registers. So general purpose registers will not have any special function or any special permission, CPU can easily access the data in the registers.
2. **SPR:** Special function registers are having the specific functions, when we use this register they will act according to the functions assigned to them. They cannot be used like normal registers. For example you cannot use STATUS register for storing the data, STATUS registers are used for showing the status of the program or operation. User cannot change the function of the Special function register; the function is given by the vendor at the manufacturing time.

ROM: we know that ROM (Read Only memory) is a non volatile memory used for storing the data permanently. In microcontroller ROM will store the complete instructions or program, according the program microcontroller will act. Rom is also called program memory in this memory user will write the program for microcontroller and save it permanently and get executed by the CPU. According to the instruction executed by the CPU the PIC microcontroller will perform the task. In ROM there are different types which are used in different PIC microcontrollers.

- **EEPROM:** In the normal ROM we can write the program for only one time we cannot reuse the Microcontroller for another time where as in the EEPROM (Electrically Erasable Programmable Read Only Memory) we can program the ROM for number of times.
- **Flash Memory:** flash memory is also PROM in which we can read write and erase the program more than 10,000 times. Mostly PIC microcontroller uses this type of ROM.

Stack: when an interrupt occur PIC has to first execute the interrupt and the existing process address which is being executed is stored in the stack. After completing the interrupt execution, PIC will call the process with the help of address which is stored in stack and get executing the process.

Bus: Bus is mainly used for transferring and receiving the data from one peripheral to another. There are two types of buses.

- Data Bus: It is used to transfer/receive only the data.
- Address Bus: is used to transmit the memory address from peripherals to CPU.
- I/O pins are used for interfacing the external peripherals, UART and USART is serial communication protocol which is used for interfacing serial devices like GPS, GSM, IR, Bluetooth etc.

Advantages of PIC Microcontroller:

- They are reliable and malfunctioning of PIC percentage is very less. And performance of the PIC is very fast because of using RISC architecture.
- Power conception is also very less when compared to other micro controllers. When we see in the programmer point of view interfacing is very easy, also we can connect analog devices directly with out any extra circuitry and use them. Programming is also very easy when compared to other microcontrollers.

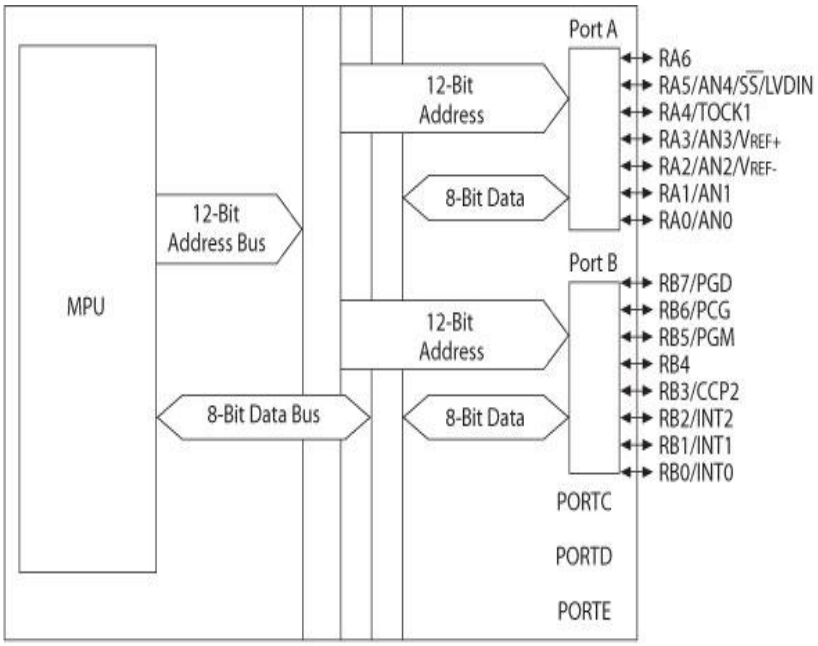
Disadvantages of PIC Microcontroller:

- The length of the program will be big because of using RISC (35 instructions).
- Program memory is not accessible and only one single accumulator is present.

PIC18F452 I/O Ports

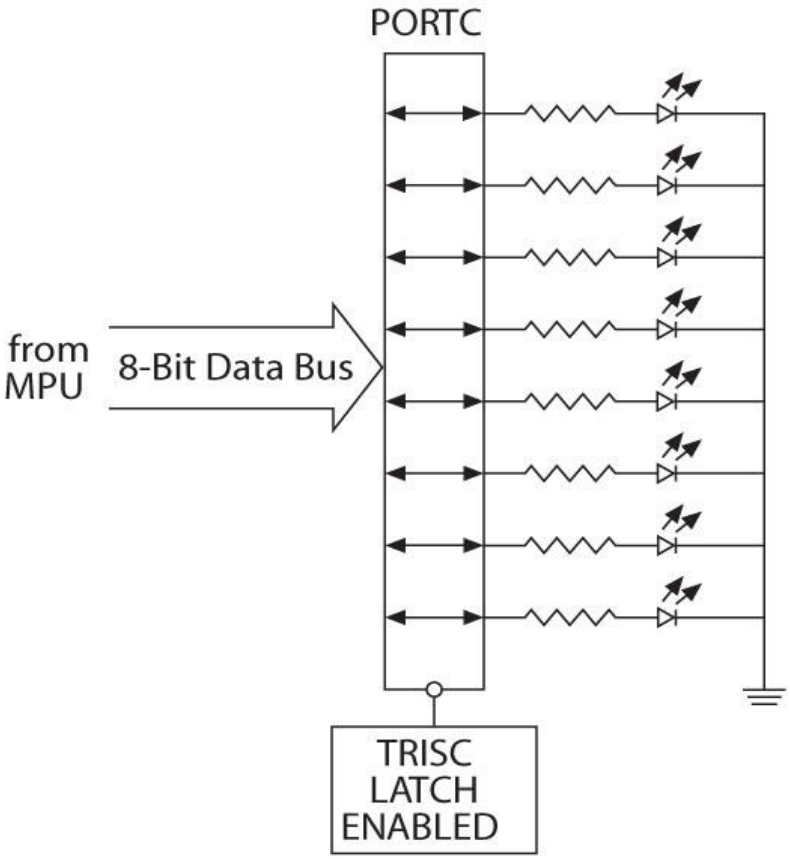
- ☐ Five I/O ports

- PORT A through PORT E
- Most I/O pins are multiplexed
- Generally have eight I/O pins with a few exceptions
- Addresses already assigned to these ports in the design stage
- Each port is identified by its assigned SFR



TRISD must be set to specify signal direction of PORT D.

- ☐ Interfacing LEDs to PORTD
- ☐ Port D is F83H
- ☐ Note that PORT D is set to be an output!
- ☐ Hence, TRISD (address H) has to be set to 0 **TRISD=0**



Interfacing diagram:

Algorithm:

1. Start
2. Initialize port B as input and port D, port A as output (RA2= relay, RB5= buzzer, RB0= row0, RB3= row 3, RB4= column 0, RB8=column 3)
3. Initialize integer as code for checking key status
4. Take data from port B into code variable
5. Check key 1 is pressed or not
6. If key 1 is pressed make relay on, buzzer on and go for LED part 1
7. Check key 2 is pressed or not
8. If key 2 is pressed make buzzer and relay off and go for led part 2
9. End

Algorithm for LED part 1

1. Start
2. Initialize integer i
3. Declare for loop with initial values i=0x1, final value 0x80, shift towards left with each iteration
4. Send 0x0 to port D. Send value of i to port D
5. Call delay
6. Stop

Algorithm for LED part 2

1. Start
2. Initialize integer i
3. Declare for loop with initial values i=0x80, final value 0x1, shift towards right with each iteration
4. Send 0x0 to port D. Send value of i to port D
5. Call delay
6. Stop

Conclusion: In this experiment we studied interfacing of various input device such as buttons and output devices such as LED, relay and buzzer with PIC18f4550 microcontroller and also observe when input button 1 is pressed relay and buzzer is turned on and LED changing from left to right. When input button 2 is pressed relay and buzzer turn off LED changing from right to left

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 05

Aim: write a program for interfacing LCD to PIC18FXX

Objective: To write a embedded C program for interfacing LCD with PIC18FXX
A. Display Message on 1st Line 4Th Position

Apparatus: MPLAB Simulator, , PC, PIC18fxx Development KIT, LCD KIT, USB cable, mikrobootloader.

Theory:

LCD is finding widespread use replacing LEDs

1. The declining prices of LCD
2. The ability to display numbers, characters, and graphics
3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
4. Ease of programming for characters and graphics

Pin Descriptions for LCD

Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

used by the LCD to latch information presented to its data bus

LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

Interfacing Diagram:

Algorithm:

A]Main Program

1] Start
2] Initialize Data PORT and Control PORT as output
3] Initialize LCD in 8 bit mode.
4]Send command 01H.
5]Call Delay
6]Send command 06H.
7]Call Delay
8]Send command 0EH
9]Call Delay
10]Send Data on Port1
11]Call Delay
12]STOP

B]Subroutine 1:

1]Transfer command to Data PORT
2]EN= H to L
3] RS=0
4]R/W=0
5]Return

B]Subroutine 2:

1]Transfer Data to Data PORT
2]EN= H to L
3] RS=1
4]R/W=0
5]Return

B]Subroutine 3:

1] Generate Delay
2]Return

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 06

Aim: write a program for Generate square wave using timer with interrupt.

Objective: To write a embedded C program to Generate square wave on Port pin using timer with interrupt

Theory:

Timers in microcontrollers are used for introducing delay, counting events, generating waveforms, and also for PWM generation.

Delays in a microcontroller can be induced by either of the two ways –

1. Provide delay using software (through code using loops). But, a delay provided in this way forces the microcontroller to put all its resources for the processing of the loops, and thus blocks the code execution.
2. Providing delay using the timer in the microcontroller. In this approach, the delay is provided by loading a count in a timer. So when a timer interrupt is generated current execution will move to the ISR to serve the task. But when there is no timer interrupt it will perform another task. Therefore, we can say it is a kind of non-blocking function.

PIC18F4550 has 4 in-built timers

- Timer0 : 16-bit
- Timer1 : 16-bit
- Timer2 : 8-bit
- Timer3 : 16-bit

Out of the four timers in PIC18F4550, we will be discussing here Timer1. The working of the rest of the timers is the same. To generate a time delay using Timer1, we have to calculate a count of the desired time.

How to calculate count for Timer?

Let's take an example. Suppose we have to generate a delay of 1 ms having 8 MHz oscillator frequency of PIC18F4550. To do this, we need to find the period of the instruction cycle.

Fosc = 8 MHz

PIC18F4550 timer uses frequency as follows:

$$FTIMER = \frac{Fosc}{4}$$

FTIMER = 8 MHz / 4 = 2 MHz

This 2 MHz frequency is fed to the PIC timer.

Then, to find the period which will be taken by the timer to increment the count,

$$Period = \frac{1}{FTIMER}$$

Period = 1 / 2 MHz = 0.5 us

Therefore, after each **0.5 us** time elapses, the timer value will get incremented by 1.

To find the count needed for a delay of 1 ms,

Count = Desired Delay / Timer Period

$$Count = \frac{DesiredDelay}{TimerPeriod}$$

Count = 1 ms / 0.5 us = 2000

Before we load this count in the timer, we first need to subtract it from 65536. The reason we need to subtract it is that the timer is 16-bit. Therefore,

65536 – count = 65536 – 2000 = 63536

Converting this count into hex we get **0xF830**. Load this hex value in the Timer1 i.e. TMR1 register. **TMR1** is a 16-bit register that is used to load the count for Timer1.

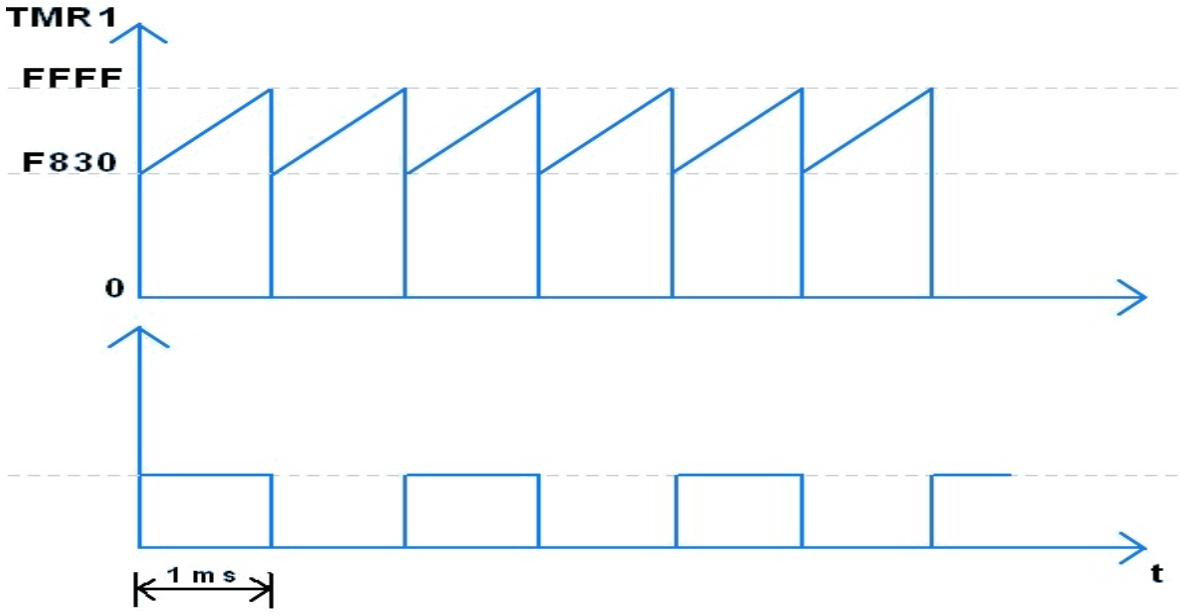
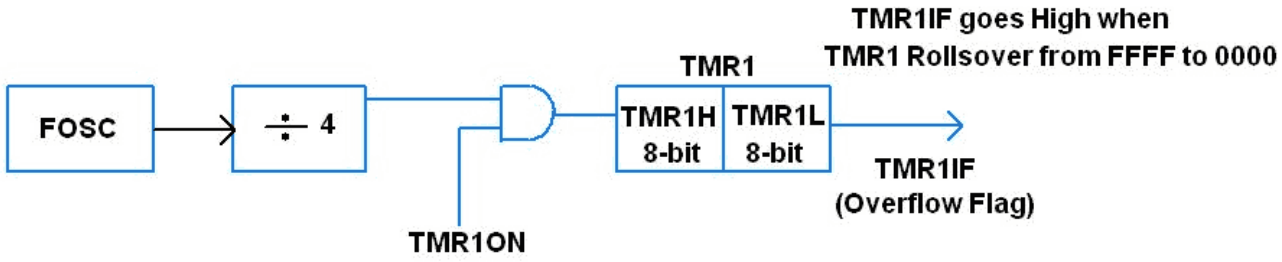


Fig.PIC18F4550 Waveform Generation using Timer



TMR1H and **TMR1L** registers can also be individually used to load the count in Timer1. These are both 8-bit registers.

To use Timer1 in PIC18F4550, Control and Status registers are used. Let us understand more about these registers of Timer1 in PIC18F4550.

T1CON: Timer1 Control Register

- It is used to configure Timer1 and also to start the Timer1.

7	6	5	4	3	2	1	0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON

RD16: 16-bit read/write mode enable bit

1 = Enable read/write of Timer1 in one 16-bit register which is TMR1.

0 = Enable read/write of Timer1 in two 8-bit registers which are TMR1H and TMR1L

T1RUN: Timer1 system clock status bit

1 = Device clock is derived from the Timer1 oscillator

0 = Device clock is derived from another source; it may be an internal oscillator or external crystal oscillator.

T1CKPS1 : T1CKPS0: Timer1 input clock pre-scale bits

These bits divide the clock by a value before it is applied to the Timer1 as a clock.

11 = 1 : 8 pre-scale value

10 = 1 : 4 pre-scale value

01 = 1 : 2 pre-scale value

00 = 1 : 1 pre-scale value

T1OSCEN: Timer1 oscillator enable bit

1 = Enable Timer1 oscillator.

0 = Disable Timer1 oscillator.

T1SYNC: Timer1 external clock input synchronization bit

When **TMR1CS = 0**

Timer1 uses an internal clock oscillator. So this bit is ignored.

When **TMR1CS = 1**

1 = Do not synchronize external clock input.

0 = Synchronize external clock input.

TMR1CS: Timer1 clock source select bit

1 = External clock from RC0 / T1OSO / T13CKl

0 = Internal clock (Fosc / 4)

This bit is generally used for counting events. To use Timer1 as counter, set TMR1CS = 1.

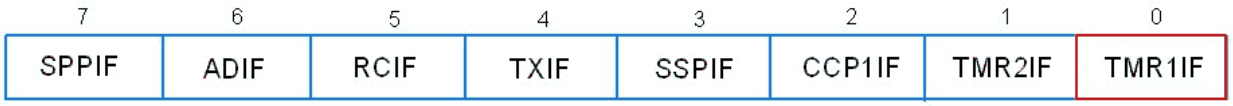
TMR1ON: Timer1 ON bit

1 = Start Timer1.

0 = Stop Timer1.

After Timer1 initializes, it continuously monitors the TMR1IF interrupt flag which gets set when the Timer1 overflows. TMR1IF is located in the PIR1 register as shown below.

PIR1: Peripheral Interrupt Register



PIR Register

TMR1IF: Timer1 Overflow Interrupt Flag

0 = Timer1 register overflow not occurred

1 = Timer1 register overflow occurred

TMR2IF: Timer2 Overflow Interrupt Flag

CCP1IF: Capture / Compare Interrupt Flag

SSPIF: Master Synchronous Serial Port Interrupt Flag

RCIF and TXIF: These are Serial Communication Interrupt Flag

ADIF: A / D Converter Interrupt Flag

SPPIF: Streaming Parallel Port Read/Write Interrupt Flag

Steps for Programming PIC18F4550 Timer

1. Configure the T1CON register.
2. Clear TMR1IF Timer1 interrupt flag.
3. Load the count in TMR1 16-bit or TMR1H (higher byte) and TMR1L (lower byte).
4. Set TMR1ON to start the Timer1 operation.
5. Wait for TMR1IF = 1. Setting this bit to '1' shows that Timer1 count overflows from 0xFFFF to 0x0000.

Timer Using Interrupt

Generate Delay of 1 ms Using an Interrupt

Using Timer1 by calling an Interrupt Service Routine (ISR) eliminates the continuous polling of the TMR1IF flag. Also, the microcontroller can perform other tasks until an interrupt occurs. To use ISR for generating delay using Timer1, three bits need to be enabled – two in the INTCON register and one in the PIE register.

INTCON: Interrupt Control Register

INTCON Register

The two bits – GIE and PEIE – in the INTCON register, as shown above, need to be enabled.

GIE: Global Interrupt Enable

Enable Global Interrupt for ISR to generate a delay.

PEIE: Peripheral Interrupt Enable

For some peripheral interrupts like TMR1IF, TMR2IF, and TXIF, we have to enable PEIE bit in addition to the GIE bit.

The third bit i.e. the **TMR1IE** in the **PIE1** register can be used to enable the Timer1 Interrupt Flag (TMR1IF).

PIE1: Peripheral Interrupt Enable

PIE Register

TMR1IE: Timer1 Overflow Interrupt Enable Bit

1 = Enable the TMR1 overflow interrupt

0 = Disable the TMR1 overflow interrupt

Note: While generating a delay using ISR, the program takes more time to switch to ISR. Therefore, to avoid this we have to increase the count to compensate for the delayed time.

Steps for Programming PIC18F4550 Timer using Interrupt

1. Enable GIE, PEIE, TMR1IE.
2. Configure the T1CON register.
3. Clear TMR1IF Timer1 interrupt flag.
4. Load the count in TMR1 16-bit or TMR1H (higher byte) and TMR1L (lower byte).
5. Set TMR1ON to start the Timer1 operation.
6. When TMR1IF = 1, code will jump to ISR to execute it, and after execution control returns to the main program.

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 7

AIM: Generation of PWM signal for DC motor control

Objective: 1.To write embedded "C" program to generation of PWM signal .
2.To observe PWM signal on CRO

Apparatus: MPLAB Simulator, C18, PC, PIC18f Development KIT, USB cable, CRO

Theory: PIC18F2455/2550/4455/4550 devices all have two CCP (Capture/Compare/PWM) modules. Each module contains a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

In 28-pin devices, the two standard CCP modules (CCP1 and CCP2) operate as described in this chapter. In 40/44-pin devices, CCP1 is implemented as an Enhanced CCP module, with standard Capture and Compare modes and Enhanced PWM modes. The ECCP is "Enhanced Capture/Compare/PWM (ECCP) Module".

The Capture and Compare operations described in this chapter apply to all standard and Enhanced CCP modules.

REGISTER 15-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—(1)	—(1)	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 Unimplemented: Read as '0'

bit 5-4 DCxB1:DCxB0: PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module

Capture mode: Unused.

Compare mode:Unused.

PWM mode:These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 CCPxM3:CCPxM0: CCPx Module Mode Select bits

0000= Capture/Compare/PWM disabled (resets CCPx module)

0001= Reserved

0010= Compare mode: toggle output on match (CCPxIF bit is set)

0011= Reserved

0100= Capture mode: every falling edge

0101= Capture mode: every rising edge

0110= Capture mode: every 4th rising edge

0111= Capture mode: every 16th rising edge

1000= Compare mode: initialize CCPx pin low; on compare match, force CCPx pin high (CCPxIF bit is set)

1001= Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set)

1010= Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state)

1011= Compare mode: trigger special event, reset timer, start A/D conversion on CCPx match (CCPxIF bit is set)

11xx=PWM mode

CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (generically, CCPxCON) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte). All registers are both readable and writable.

CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available to modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode.

The assignment of a particular timer to a module is determined by the Timer to CCP enable bits in the T3CON register (Register 14-1). Both modules may be active at any given time and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time. The interactions between the two modules are summarized in Figure 15-2. In Timer1 in Asynchronous Counter mode, the capture operation will not work.

CCP2 PIN ASSIGNMENT

The pin assignment for CCP2 (capture input, compare and PWM output) can change, based on device configuration. The CCP2MX Configuration bit determines which pin CCP2 is multiplexed to. By default, it is assigned to RC1 (CCP2MX = 1). If the Configuration bit is cleared, CCP2 is multiplexed

with RB3.Changing the pin assignment of CCP2 does not automatically change any requirements for configuring the port pin. Users must always verify that the appropriate TRIS register is configured correctly for CCP2 operation, regardless of where it is located.

INTERACTIONS BETWEEN CCP1AND CCP2 FOR TIMER RESOURCES

CCP/ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
Timer 2	PWM

Algorithm:-

1) Start
2) Initialize code ,duty =0
3) Select PWM mode, configure CCPON register.
4) Set PWM period in PR2 Register (PR2=FF)
5) Declare port c as a output port.
6) Start timer 2 & load T2ON register with 0*7E
7) Declare While loop.
8) Take the code value from port B.
9) If the key 1 is pressed increases the duty cycle.
10) Load the new value of CCP1L & CCP1CON Register.
11) If key 2 is pressed decreases the duty cycle.
12) Load the new value & CCP1L & CCP1CON register.
13) Call Delay.
14) Initialize for Loop With (i=0;i<1000;i++)
15) Initialize for loop with (j=0;j<1000;j++)
16) STOP.

Conclusion:

In this experiment for generation of PWM Signal for DC Motor Control we use CCP Module in PWM Mode. we observed when button 1 is pressed then increase the duty cycle & vice versa condition when button 2 is pressed like that we can control speed of DC Motor.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 08

Aim: Interfacing serial port with PC for both side communication.

Objective: To write embedded "c" program to to transmit and receive data serially through USART

Apparatus: MPLAB Simulator, C18, PC, PIC18f Development KIT, UART converter cable, mikrobootloader.

Several devices such as GPS, GSM, RFID, sensors, etc need to communicate with the PIC microcontroller for transmitting or receiving information. To communicate with the PIC microcontroller, several communication protocols are used such as RS232, SPI, I2C, CAN, etc. Basically, a protocol is a set of rules agreed by both, the sender and the receiver, on -

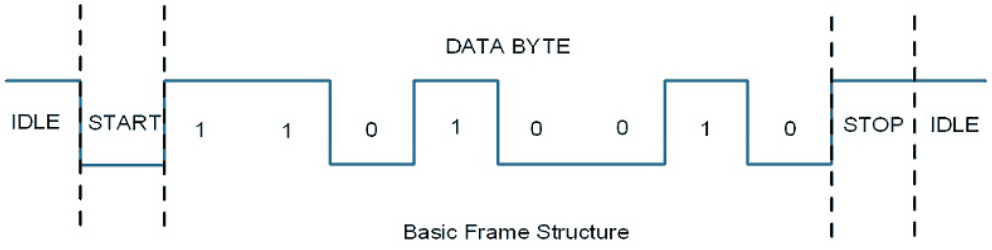
- How the data is packed?
- How many bits constitute a character?
- When the data begins and ends?

PIC18F4550 has an in-built USART module which is useful for serial communication. With the help of USART, we can send/receive data to a computer or other devices. USART is also used in interfacing PIC with various modules like Wi-Fi (ESP8266), Bluetooth, GPS, GSM, etc.

We will see how the communication is established between PIC microcontroller and PC through USART using RS232 protocol. We will also see how to communicate with laptops, which do not have an RS232 DB9 port, and instead use a USB port.

Let us start with the serial communication using PIC18F4550.

Asynchronous Communication: PIC18F4550 has a built-in asynchronous receiver-transmitter. Asynchronous means each character (data byte) is placed in between the start and stop bits. The start bit is always 0 (low) and the stop bit is always 1 (high).



Basic Frame Structure

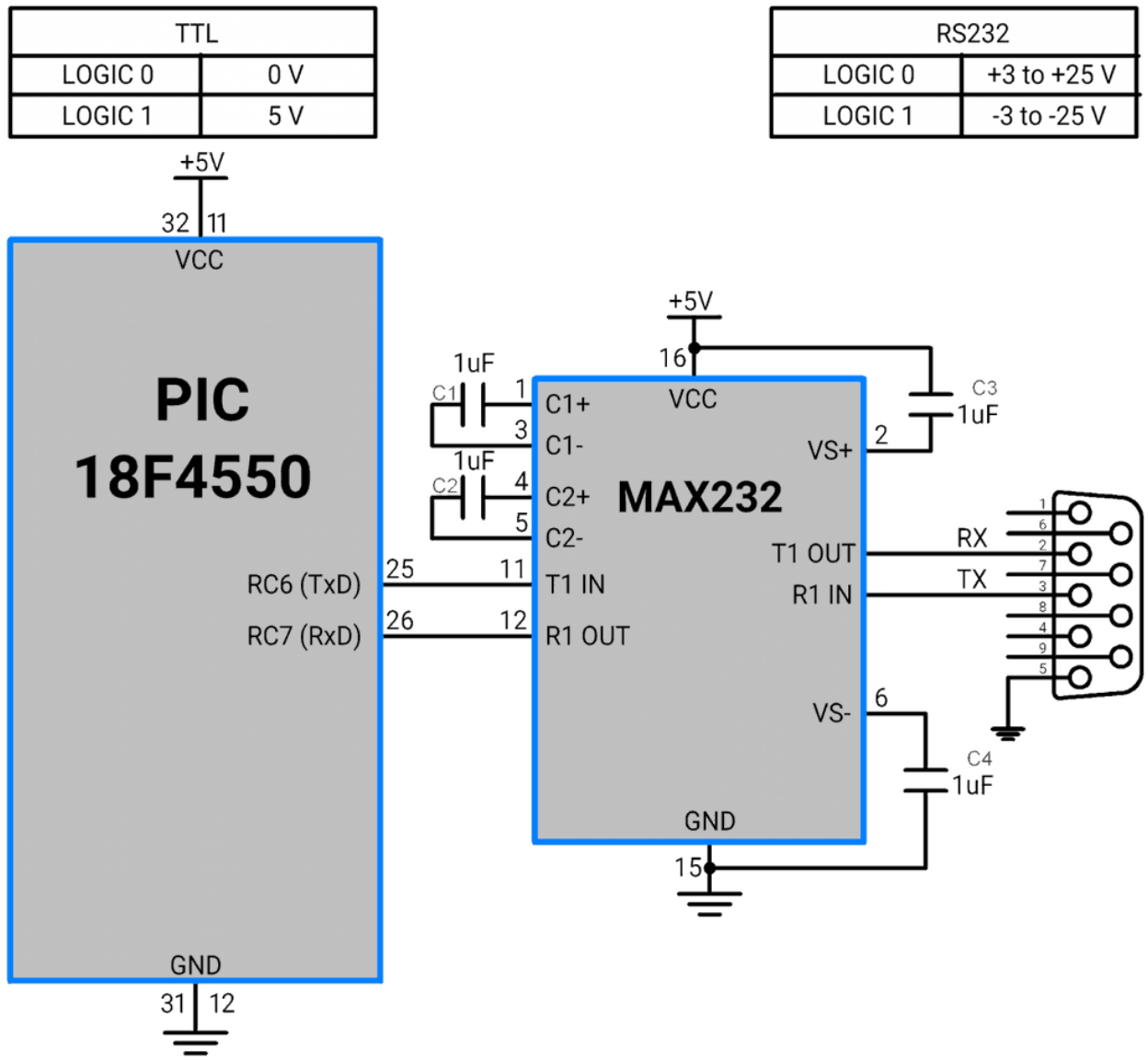
Bit Rate & Baud Rate: The rate of data transfer in serial data communication is stated in bps (bits per second). Another widely used terminology for bps is baud rate; means, a number of changes in signal per second. Here the signal is in bits, therefore bit rate = baud rate.

Interface: Although there are many pins in the DB9 connector, we do not need all. We only use pins RX, TX, and GND.

Level Conversion

Convert PIC18F TTL levels to RS232 levels and vice-versa

PC has RS232 levels whereas the PIC microcontroller has TTL levels. The RS232 has different voltage levels for logic 0 and 1. To make it compatible with the PIC TTL voltage levels, we have to use a MAX232 IC.



PIC18F TTL to RS232 Level conversion

Baud Rate Calculation:

How to Calculate the Baud Rate in PIC18F4550?

$$Desired\ Baud\ Rate = \frac{Fosc}{64 * (X + 1)}$$

a value that will be loaded into the SPBRG register (16-bit) of the PIC18F4550 to get the desired baud rate. The value of SPBRG for the desired baud rate is calculated as,

$$SPBRG = \frac{Fosc}{64 * Desired\ Baud\ Rate} - 1$$

E.g.

Suppose, Fosc = 8 MHz and Baud Rate = 9600 bps

Then, **SPBRG=((8 MHz)/(64 ×9600))-1**

SPBRG = (8 MHz / 64 × 9600) - 1

Therefore, SPBRG = 12

The above formula depends on the **BRGH** bit in the **TXSTA** register.

Let's see the TXSTA register, which is used for transmission setting, in detail as follows

TXSTA: Transmit Status and Control Register

7	6	5	4	3	2	1	0
CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D

TXSTA Register

TXEN: Transmit Enable Bit

1 = Enable the transmission

0 = Disable the transmission

BRGH: High Baud Rate select bit

0 = Low speed

$$SPBRG = \frac{Fosc}{64 * Desired\ Baud\ Rate} - 1$$

1 = High speed

$$SPBRG = \frac{Fosc}{16 * Desired\ Baud\ Rate} - 1$$

Load the calculated value directly to the SPBRG register.

CSRC bit is not used for asynchronous communication.

TX9 : 9th Transmit Enable Bit

0 = Select 8-bit transmission

1 = Select 9-bit transmission

SYNC : USART Mode Select Bit

0 = Asynchronous mode

1 = Synchronous mode

SENDB : Send Break Character bit

1 = Send Sync Break on next transmission (cleared by hardware upon completion)

0 = Sync Break transmission completed

TRMT : Transmit Shift Register Status Bit

0 = TSR full

1 = TSR empty

TX9D: 9th bit of transmitting data

Can be Address / Data bit or a parity bit.

In PIC18F4550, the RCSTA register is used for serial data receive settings.

RCSTA: Receive Control and Status Register

7	6	5	4	3	2	1	0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

RCSTA Register

SPEN: Serial Port Enable

1= Enable Serial Port for communication

0 = Disable Serial Port for communication

RX9 : 9-bit Receive Enable bit

1 = Enable 9-bit reception

0 = Enable 8-bit reception

Generally, we use 8-bit reception

SREN: Single Receive Enable bit

Not used

CREN: Continuous Receive Enable bit

1 = Enable receiver for continuous reception of data byte

0 = Disable receiver

ADDEN: Address Detect Enable bit

Asynchronous mode 9-bit (RX9 = 1)

1 = Enable address detection, enable interrupt, and load the receive buffer when the RSR bit is set.

0 = Disable address detection, all bytes are received and the ninth bit can be used as a parity bit.

Asynchronous mode 9-bit (RX9 = 0)

Don’t care (any 0 or 1)

FERR: Framing Error bit

1 = Framing error (can be updated by reading the RCREG register and receiving the next valid byte)

0 = No framing error

OERR: Overrun Error bit

1 = Overrun error can be cleared by clearing bit CREN.

0 = No overrun error

RX9D: 9th bit of the Receiving Data

This can be address/data bit or a parity bit and must be calculated by user firmware.

Data Buffer and Interrupt Flag for Serial Communication

For transmitting data and reception of data **TXREG** and **RCREG** 8-bit data registers are allocated in PIC18F4550 respectively.

- When we have to transmit data, we directly copy that data to the TXREG register. After completing the transmission of 8-bit data, the TXIF interrupt flag is generated.
- This TXIF (transmit interrupt flag) is located in the PIR1 register. TXIF flag is set when the 8-bit data is transmitted. Then, the buffer is ready to receive another data for transmission.
- Also, RCIF (receive interrupt flag) is located in the PIR1 register. When this flag is set, it indicates that the complete data byte is received by the RCREG register. Read the RCREG register immediately. Now, the RCREG register is ready to receive another data.
- When the RCIF flag is not set, the PIC microcontroller has to wait for the reception of the complete data byte.

Algorithm Steps for Programming PIC18F4550 USART:

Initialization

1. Initialize the Baud Rate by loading a value into the SPBRG register.

2. Then set bit SPEN in the RCSTA for enabling Serial Port.

3. Then set bit BRGH in the TXSTA for low or high speed.

4. Also clear bit SYNC in the TXSTA register for asynchronous communication.

5. Set bit TXEN in the TXSTA register to enable transmission.

6. Set bit CREN in the RCSTA register to enable reception.

Transmit mode

1. Copy the data which we want to transmit into the TXREG register.

2. Monitor the flag TXIF which is set when the transmission is completed.

Receive mode

1. Monitor the flag RCIF until it is set to 1, which indicates a complete 1 byte is received in the RCREG register.

2. Also, check for OERR bit. If it is set then disable and enable CREN.

3. Then, read the RCREG register immediately to avoid overflow.

Conclusion:

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					

EXPERIMENT NO. 08

Aim: Interface analog Voltage 0-5 V to Internal ADC and display value on LCD.

Objective: To write embedded "C" program to interface analog input and display digital 10-bit value on LCD (use inbuilt ADC)

Apparatus: MPLAB Simulator, C18, PC, PIC18f Development KIT, USB cable, LCD,mikrobootloader.

Theory: ADC Conversion

Theory: ADC Conversion

- An **analog-to-digital converter** is a device that converts a continuous physical quantity (usually voltage) to a digital number that represents the quantity's amplitude.
- The conversion involves quantization of the input, so it necessarily introduces a small amount of error. Instead of doing a single conversion, an ADC often performs the conversions periodically. The result is a sequence of digital values that have been converted from a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal.

ADC types

1. A direct-conversion ADC or flash ADC has a bank of comparators sampling the input signal in parallel, each firing for their decoded voltage range. The comparator bank feeds a logic circuit that generates a code for each voltage range. Direct conversion is very fast, capable of gigahertz sampling rates, but usually has only 8 bits of resolution or fewer, since the number of comparators needed, $2^N - 1$, doubles with each additional bit, requiring a large, expensive circuit. ADCs of this type have a large die size, a high input capacitance, high power dissipation, and are prone to produce glitches at the output (by outputting an out-of-sequence code). Scaling to newer submicrometre technologies does not help as the device mismatch is the dominant design limitation. They are often used for video, wideband communications or other fast signals in optical storage.

2. A **successive-approximation ADC** uses a comparator to successively narrow a range that contains the input voltage. At each successive step, the converter compares the input voltage

to the output of an internal digital to analog converter which might represent the midpoint of a selected voltage range. At each step in this process, the approximation is stored in a successive approximation register (SAR). For example, consider an input voltage of 6.3 V and the initial range is 0 to 16 V. For the first step, the input 6.3 V is compared to 8 V (the midpoint of the 0–16 V range). The comparator reports that the input voltage is less than 8 V, so the SAR is updated to narrow the range to 0–8 V. For the second step, the input voltage is compared to 4 V (midpoint of 0–8). The comparator reports the input voltage is above 4 V, so the SAR is updated to reflect the input voltage is in the range 4–8 V. For the third step, the input voltage is compared with 6 V (halfway between 4 V and 8 V); the comparator reports the input voltage is greater than 6 volts, and search range becomes 6–8 V. The steps are continued until the desired resolution is reached.

3. A ramp-compare ADC produces a saw-tooth signal that ramps up or down then quickly returns to zero. When the ramp starts, a timer starts counting. When the ramp voltage matches the input, a comparator fires, and the timer's value is recorded. Timed ramp converters require the least number of transistors. The ramp time is sensitive to temperature because the circuit generating the ramp is often a simple oscillator. There are two solutions: use a clocked counter driving a DAC and then use the comparator to preserve the counter's value, or calibrate the timed ramp. A special advantage of the ramp-compare system is that comparing a second signal just requires another comparator, and another register to store the voltage value. A very simple (non-linear) ramp-converter can be implemented with a microcontroller and one resistor and capacitor. Vice versa, a filled capacitor can be taken from an integrator, time-to-amplitude converter, phase detector, sample and hold circuit, or peak and hold circuit and discharged. This has the advantage that a slow comparator cannot be disturbed by fast input changes.

4. An integrating ADC (also dual-slope or multi-slope ADC) applies the unknown input voltage to the input of an integrator and allows the voltage to ramp for a fixed time period (the run-up period). Then a known reference voltage of opposite polarity is applied to the integrator and is allowed to ramp until the integrator output returns to zero (the run-down period). The input voltage is computed as a function of the reference voltage, the constant run-up time period, and the measured run-down time period. The run-down time measurement is usually made in units of the converter's clock, so longer integration times allow for higher resolutions. Likewise, the speed of the converter can be improved by sacrificing resolution. Converters of

this type (or variations on the concept) are used in most digital voltmeters for their linearity and flexibility.

PIC18F2455/2550/4455/4550

10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

ANALOG-TO-DIGITAL CONVERTER STEPS

1. Configure the A/D module:

- Configure analog pins, voltage reference and digital I/O (ADCON1)
- Select A/D input channel (ADCON0)
- Select A/D acquisition time (ADCON2)
- Select A/D conversion clock (ADCON2)
- Turn on A/D module (ADCON0)

2. Configure A/D interrupt (if desired):

- Clear ADIF bit
- Set ADIE bit
- Set GIE bit

3. Wait the required acquisition time (if required).

4. Start conversion:

- Set GO/DONE bit (ADCON0 register)

5. Wait for A/D conversion to complete, by either:

- Polling for the GO/DONE bit to be cleared

  OR

- Waiting for the A/D interrupt

6. Read A/D Result registers (ADRESH:ADRESL);

clear bit ADIF, if required.

7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 3 TAD is required before the next acquisition starts.

ADCON0: A/D control registers 0:

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

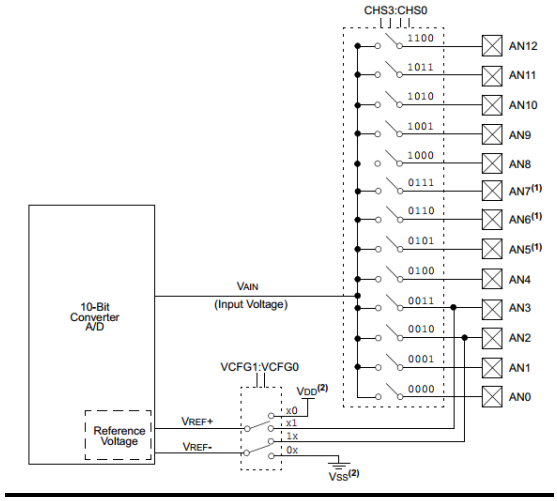
bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit
1 = A/D converter module is enabled
0 = A/D converter module is disabled

A/D BLOCK DIAGRAM :



Algorithm:

1. Start
2. Initialize I, h, u , a, t code as integer
3. Initialize char message for display “ADC data”
4. Declare port D, A as output port
5. Select ADCON0=0x15 to configure analog pins and reference voltage
6. Select ADC input channel AN5
7. ADCON1 and ADCON2 register loaded with 0x09 and 0x45 respectively (for right justified and CH5)
8. Send command to LCD for displaying data on LCD
9. Declare while loop
10. Start analog to digital conversion
11. Declare while loop to check flag for ADC conversion complete
12. After end of conversion, convert result of ADC (which is in hexadecimal form) into decimal format
13. Convert decimal value in ASCII and send it to LCD for display (digital value of analog input will be displayed on LCD in form of decimal numbers)
14. Stop

Conclusion:

For analog to digital conversion in built ADC is used. Analog input is converted to channel 5(AN5). Reference voltage=VCC is selected, converted. Digital value is displayed on LCD in decimal number format.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
Sub Teacher Sign:					