Dr. D. Y. Patil Institute of Engineering, Management & Research, Akurdi , Pune – 411044.

Department of Electronics and Telecommunication Engineering

MICROCONTROLLERS LAB MANUAL
Code: 304188

T.E. E&TC

Academic Year 2024-25

INDEX

Experiment No: 01

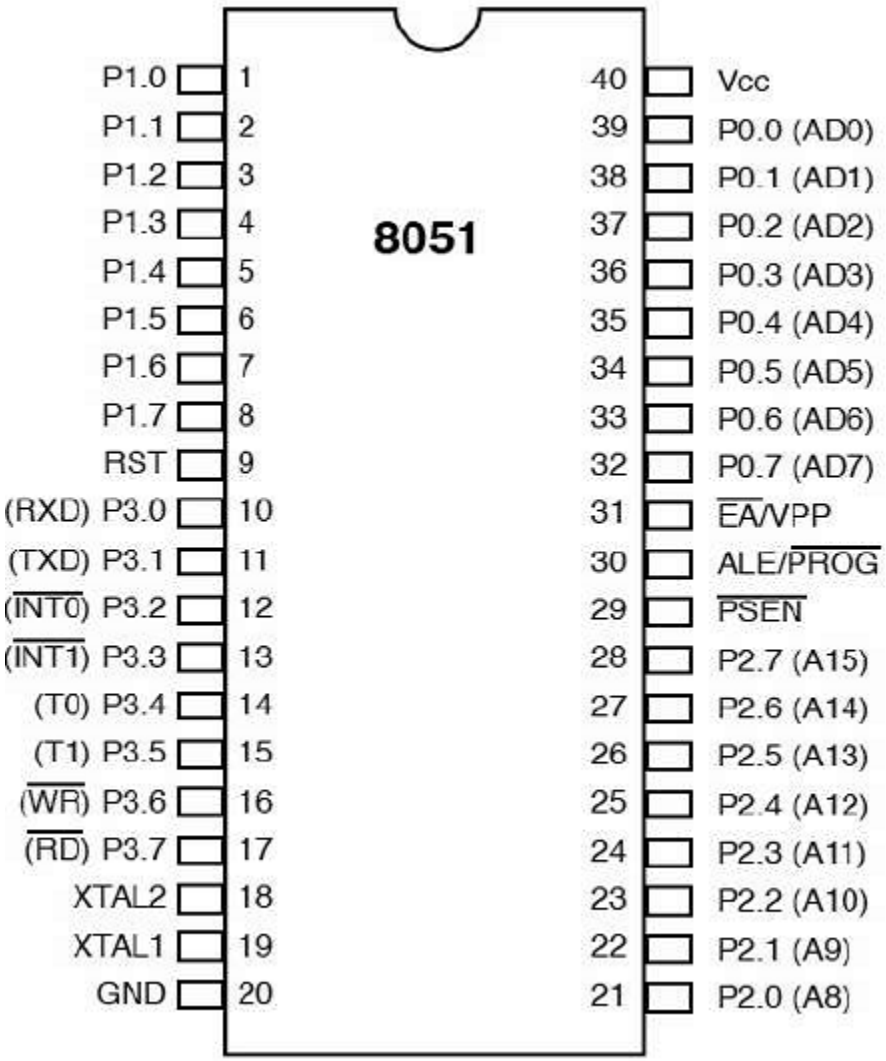
Aim:- To write an Embedded C Program for 8051to interface LEDs

Apparatus Required:

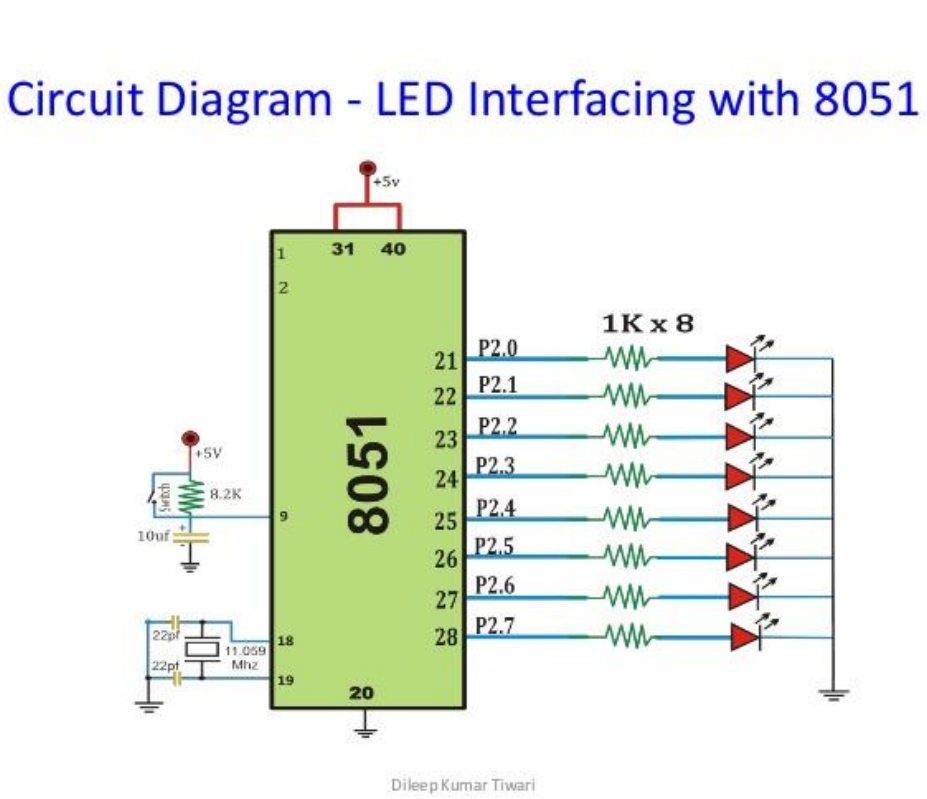
8051 Trainer kit, SSTflashflex51, LED interfacing kit

Theory:

8051 chips are used in a wide variety of control systems, telecom applications, robotics as well as in the automotive industry. By some estimations, 8051 family chips make up over 50% of the embedded chip market.



PIN 9: PIN 9 is the reset pin which is used to reset the microcontroller's internal registers and ports upon starting up. (Pin should be held high for 2 machine cycles.)
PINS 18 & 19: The 8051 has a built-in oscillator amplifier hence we need to only connect a crystal at these pins to provide clock pulses to the circuit.
PIN 40 and 20: Pins 40 and 20 are VCC and ground respectively. The 8051 chip needs +5V 500mA to function properly, although there are lower powered versions like the Atmel 2051 which is a scaled down version of the 8051 which runs on +3V.
PINS 29, 30 & 31: As described in the features of the 8051, this chip contains a built-in flash memory. In order to program this we need to supply a voltage of +12V at pin 31. If external memory is connected then PIN 31, also called EA/VPP, should be connected to ground to indicate the presence of external memory. PIN 30 is called ALE (address latch enable), which is used when multiple memory chips are connected to the controller and only one of them needs to be selected.We will deal with this in depth in the later chapters. PIN 29 is called PSEN. This is "program store enable". In order to use the external memory it is required to provide the low voltage (0) on both PSEN and EA pins.
Pin 29: If we uses an external ROM then it should have a logic 0 which indicates Micro controller to read data from memory.
Pin 30: This Pin is used for ALE that is Address Latch Enable. If we use multiple memory chips then this pin is used to distinguish between them.It is activated periodically with a constant rate of 1/6th of oscillator frequency. This Pin also gives program pulse input during programming of EPROM.
Pin 31: If we have to use multiple memories then by applying logic 1 to this pin instructs Micro controller to read data from both memories first internal and afterwards external.
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, as can be seen from fig 1.1
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.
PIN 10: Serial Asynchronous Communication Input or Serial synchronous communication output.
PIN 11: Serial Asynchronous Communication Output or Serial Synchronous Communication clock Output.



Procedure:

1. Create a project in Keil to toggle a particular port for blinking the LEDs, or move the value of a variable or BCD counter to the port pin.
2. Compile the code and generate the .hex file.
3. Flash the program into the microcontroller using the SST FlashFlex Utility.
4. Connect the LED addon card to the microcontroller board using the FRC cable.
5. Press reset to run the code and observe output on the LEDs.

Conclusion:

Experiment No: 02

Aim:- To write an Embedded C Program for 8051 to Interface 7-segment display

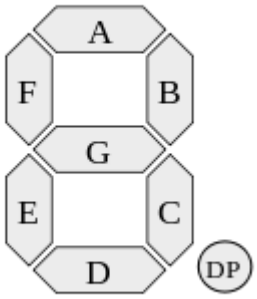
Apparatus Required:

8051 Trainer kit, SSTflashflex51, 7 segment display kit

Theory:

A seven-segment display (SSD), or seven-segment indicator, is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

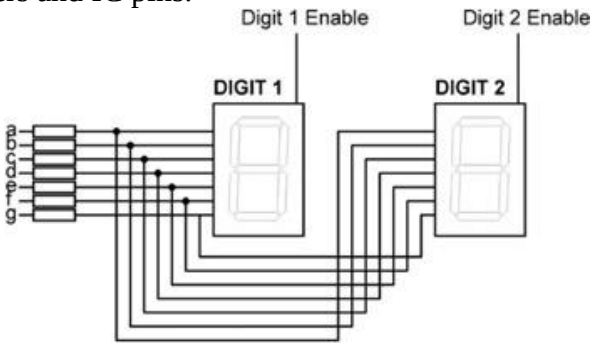
The seven elements of the display can be lit in different combinations to represent the arabic numerals. The segments of a 7-segment display are referred to by the letters A to G, where the optional decimal point (an "eighth segment", referred to as DP) is used for the display of non-integer numbers.



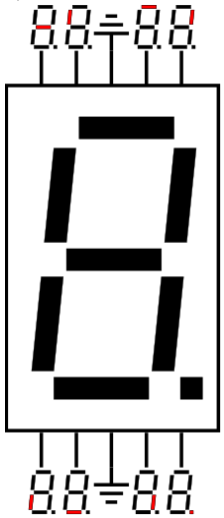
In a simple LED package, typically all of the cathodes (negative terminals) or all of the anodes (positive terminals) of the segment LEDs are connected and brought out to a common pin; this is referred to as a "common cathode" or "common anode" device. Hence a 7 segment plus decimal point package will only require nine pins, though commercial products typically contain more pins, and/or spaces where pins would go, in order to match standard IC sockets.

Multiple-digit LED displays as used in pocket calculators and similar devices used multiplexed displays to reduce the number of I/O pins required to control the display. For example, all the anodes of the A segments of each digit position would be connected together and to a driver circuit pin, while the cathodes of all segments for each digit would be connected. To operate any particular segment of any digit, the controlling integrated circuit would turn on the cathode driver for the selected digit, and the anode drivers for the desired segments; then after a short blanking interval the next digit would be selected and new segments lit, in a sequential fashion. In this manner an eight digit display with seven

segments and a decimal point would require only 8 cathode drivers and 8 anode drivers, instead of sixty-four drivers and IC pins.



A single byte can encode the full state of a 7-segment-display. The most popular bit encodings are gfedcba and abcdefg, where each letter represents a particular segment in the display. In the gfedcba representation, a byte value of 0x06 would (in a common anode circuit) turn on segments 'c' and 'b', which would display a '1'.



Hexadecimal digits can be displayed on seven-segment displays. A combination of uppercase and lowercase letters is used for A–F; this is done to obtain a unique, unambiguous shape for each hexadecimal digit (otherwise, a capital D would look identical to a 0/O and a capital B would look identical to an 8). Also the digit 6 must be displayed with the top bar lit to avoid ambiguity with the letter b.

Digit	gfedcba	abcdefg	a	b	c	d	e	f	g	Dot
0	0×3F	0×7E	**on**	**on**	**on**	**on**	**on**	**on**	off	X
1	0×06	0×30	off	**on**	**on**	off	off	off	off	X
2	0×5B	0×6D	**on**	**on**	off	**on**	**on**	off	**on**	X
3	0×4F	0×79	**on**	**on**	**on**	**on**	off	off	**on**	X
4	0×66	0×33	off	**on**	**on**	off	off	**on**	**on**	X
5	0×6D	0×5B	**on**	off	**on**	**on**	off	**on**	**on**	X
6	0×7D	0×5F	**on**	off	**on**	**on**	**on**	**on**	**on**	X
7	0×07	0×70	**on**	**on**	**on**	off	off	off	off	X
8	0×7F	0×7F	**on**	**on**	**on**	**on**	**on**	**on**	**on**	X
9	0×6F	0×7B	**on**	**on**	**on**	off	off	**on**	**on**	X

A	0×77	0×77	**on**	**on**	**on**	off	**on**	**on**	**on**	X
b	0×7C	0×1F	off	off	**on**	**on**	**on**	**on**	**on**	X
C	0×39	0×4E	**on**	off	off	**on**	**on**	**on**	off	X
d	0×5E	0×3D	off	**on**	**on**	**on**	**on**	off	**on**	X
E	0×79	0×4F	**on**	off	off	**on**	**on**	**on**	**on**	X
F	0×71	0×47	**on**	off	off	off	**on**	**on**	**on**	X

Short messages giving status information (e.g. "no dISC" on a CD player) are also commonly represented on 7-segment displays. In the case of such messages it is not necessary for every letter to be unambiguous, merely for the words as a whole to be readable.

Procedure:

1. Create a project in Keil and write the code in a .asm file.
2. Compile the code and generate the .hex file.
3. Flash the program into the microcontroller using the SST FlashFlex Utility.
4. Connect the Seven Segment addon card to the microcontroller board using the FRC cable.
5. Press reset to run the code and observe output on the Seven Segment Display.

Conclusion:

Experiment No: 03

Aim:- To write an Embedded C Program for waveform generation using DAC

Apparatus Required:

8051 Trainer kit, SSTflashflex51, DAC kit

Theory:

The DAC08 series of 8-bit monolithic digital-to-analog converters provide very high speed performance coupled with low cost and outstanding applications flexibility.

Advanced circuit design achieves 85 ns settling times with very low "glitch" energy and at low power consumption. Monotonic multiplying performance is attained over a wide 20-to-1 reference current range. Matching to within 1 LSB between reference and full-scale currents eliminates the need for full-scale trimming in most applications. Direct interface to all popular logic families with full noise immunity is provided by the high swing, adjustable threshold logic input.

High voltage compliance complementary current outputs are provided, increasing versatility and enabling differential operation to effectively double the peak-to-peak output swing. In many applications, the outputs can be directly converted to voltage without the need for an external op amp. All DAC08 series models guarantee full 8-bit monotonicity, and nonlinearities as tight as ±0.1% over the entire operating temperature range are available. Device performance is essentially unchanged over the ±4.5 V to ±18 V power supply range, with 33 mW power consumption attainable at ±5 V supplies.

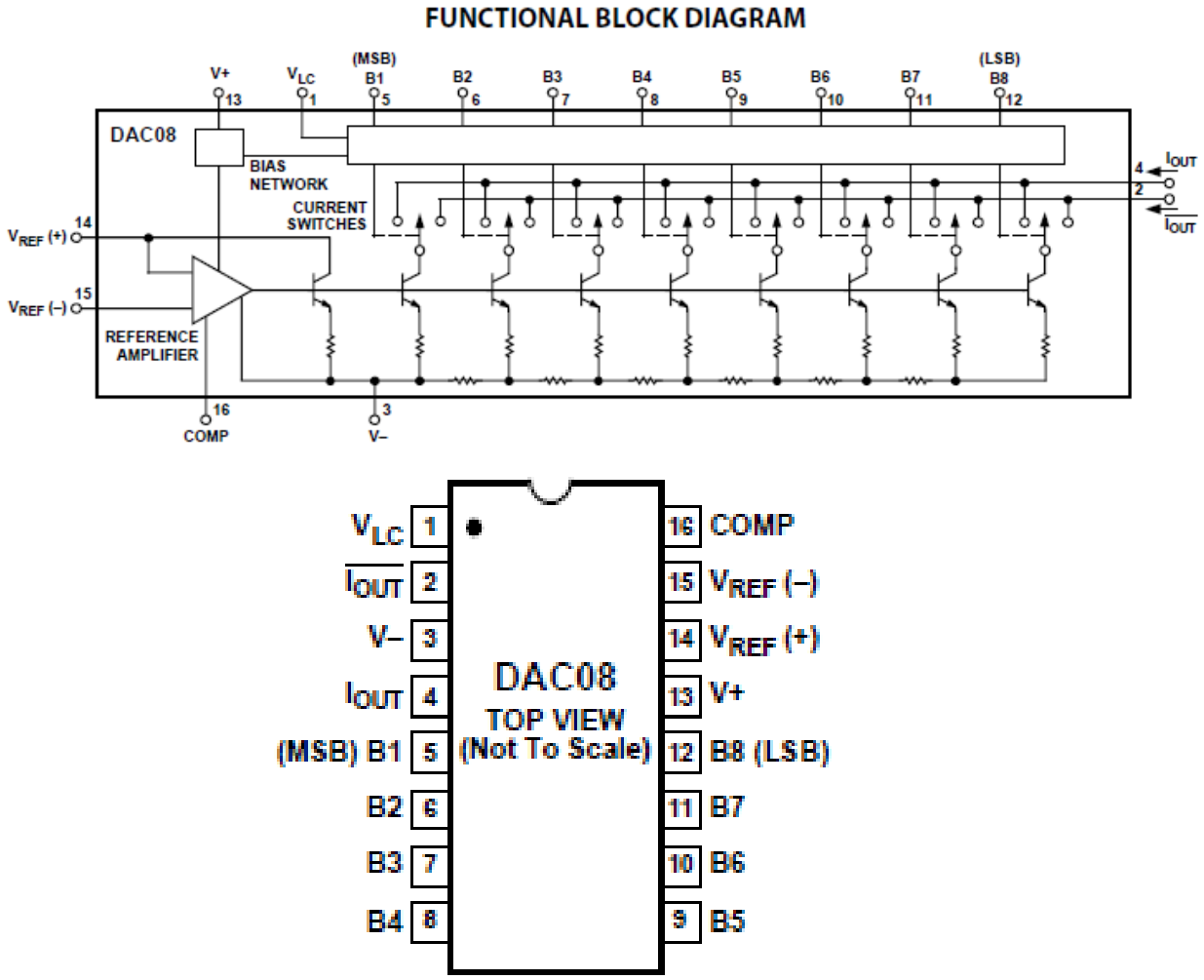
The compact size and low power consumption make the DAC08 attractive for portable and military/aerospace applications; devices processed to MIL-STD-883, Level B are available.

DAC08 applications include 8-bit, 1 μs A/D converters, servo motor and pen drivers, waveform generators, audio encoders and attenuators, analog meter drivers, programmable power supplies, LCD display drivers, high speed modems, and other applications where low cost, high speed, and complete input/output versatility are required.

Features of DAC08:

1. Fast settling output current: 85 ns
2. Full-scale current prematched to ±1 LSB
3. Direct interface to TTL, CMOS, ECL, HTL, PMOS
4. Nonlinearity to 0.1% maximum over temperature range
5. High output impedance and compliance: −10 V to +18 V
6. Complementary current outputs
7. Wide range multiplying capability: 1 MHz bandwidth
8. Low FS current drift: ±10 ppm/°C
9. Wide power supply range: ±4.5 V to ±18 V
10. Low power consumption: 33 mW @ ±5 V

11. Low cost



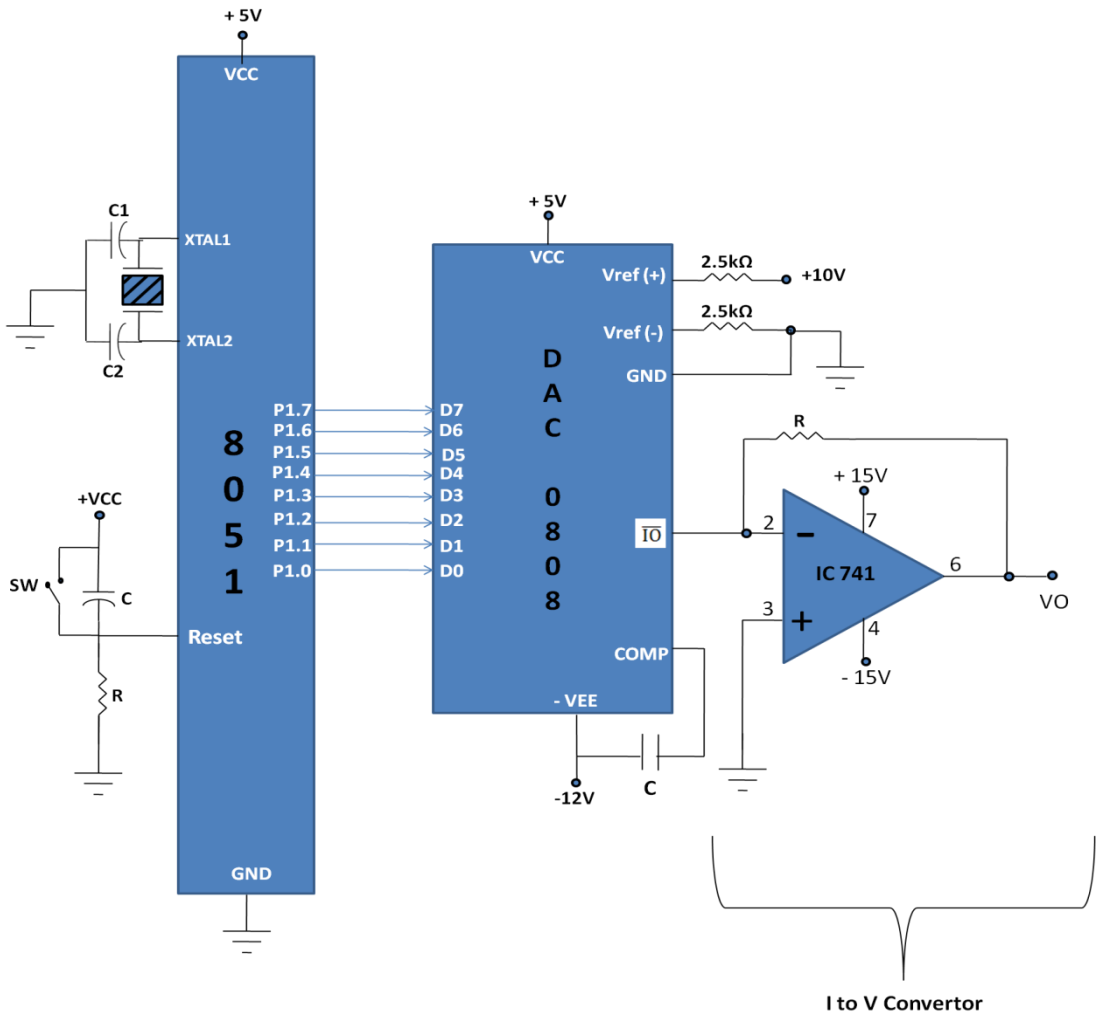
The DAC08 operates over a wide range of power supply voltages from a total supply of 9 V to 36 V. When operating at supplies of ±5 V or lower, IREF ≤ 1 mA is recommended.Operation from lower supplies is possible; however, at least 8 V total must be applied to ensure turn-on of the internal bias network.Symmetrical supplies are not required, as the DAC08 is quite insensitive to variations in supply voltage. Battery operation is feasible because no ground connection is required: however, an artificial ground may be used to ensure logic swings, etc., remain between acceptable limits.

The DAC08 design incorporates a unique logic input circuit that enables direct interface to all popular logic families and provides maximum noise immunity. This feature is made possible by the large input swing capability, 2 µA logic input current, and completely adjustable logic threshold voltage. For V− = −15 V, the logic inputs may swing between −10 V and +18 V. This enables direct interface with 15 V CMOS logic, even when the DAC08 is powered from a 5 V supply.

Both true and complemented output sink currents are provided where IO + IO = IFS. Current appears at the true (IO) output when a 1 (logic high) is applied to each logic

input. As the binary count increases, the sink current at Pin 4 increases proportionally, in the fashion of a positive logic DAC. When a 0 is applied to any input bit, that current is turned off at Pin 4 and turned on at Pin 2. A decreasing logic count increases IO as in a negative or inverted logic DAC. Both outputs may be used simultaneously. If one of the outputs is not required, it must be connected to ground or to a point capable of sourcing IFS; do not leave an unused output pin open.The dual outputs enable double the usual peak-to-peak load swing when driving loads in quasi-differential fashion. This feature is especially useful in cable driving, CRT deflection and in other balanced applications such as driving center-tapped coils and transformers.

	B1	B2	B3	B4	B5	B6	B7	B8	E_O
POS. FULL RANGE	1	1	1	1	1	1	1	1	+4.960
ZERO SCALE	1	0	0	0	1	0	0	0	0.000
NEG. FULL SCALE +1LSB	0	0	0	0	0	0	0	1	–4.960
NEG. FULL SCALE	0	0	0	0	0	0	0	0	–5.000



Procedure:

1. Create a project in Keil and write a code to generate waveforms such as square wave, triangle wave, trapezoidal wave, sine wave, etc.
2. Compile the code and generate the .hex file.
3. Burn the .hex file into the microcontroller using the SST FlashFlex utility.
4. Connect the DAC addon card to the port specified in the program.
5. Connect a CRO or a DSO to the DAC out terminal (green block connector) of the DAC addon card.
6. Press reset and run the code. View the output on the CRO/DSO screen.

Conclusion:

INTRODUCTION

Overview

PIC is a family of modified Harvard architecture microcontrollers made by Microchip Technology. The PIC microcontroller is the most widely used controller in the automation industry today. It is the most preferred choice of engineers working in the field of industrial electronics solutions. The Micro-PIC18F kit has interfaced the PIC18F series microcontroller, to various peripherals on the board. Devices like LED's and LCD, I2C interface based memory devices, RTC, SD/MMC Card interface, Matrix Keypad, ADC, DAC, graphical LCD, Stepper motor, etc are also interfaced on the board. Different peripherals can be selected by using selection switches so multiplexing of devices has been possible.

Features

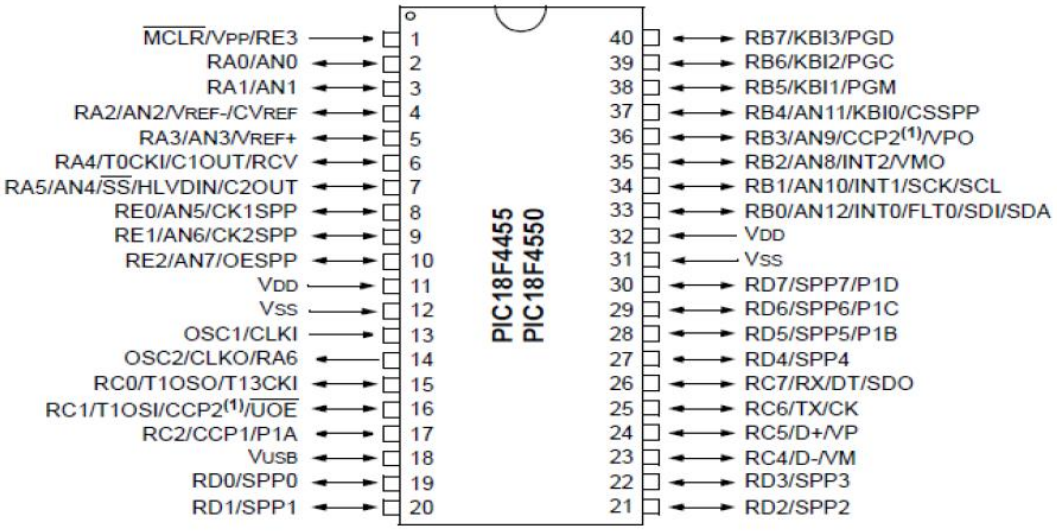
The Micro-PIC18F microcontroller board has been specifically designed keeping in mind the needs of students for learning the PIC architecture. The board gives a complete overview for interfacing various peripheral devices which are used in the industry and consumer devices alike. A hands-on with the board will develop in the student the experience to design and implement various devices and products based on the PIC Microcontroller.

Following are the features of the PIC18F Family of microcontrollers:

- High Performance RISC CPU.
- 77 instructions
- C Language friendly.
- Linear program and data memory space.
- Up to 48 MHz and 10 MIPS operation.
- Priority levels for interrupts.
- 16 bit wide instructions and 8 bit wide data path.
- 8x8 hardware multiplier.
- Peripheral Features.
- Up to 25 ma current source and sink.
- Up to 4 external interrupt pins.
- Up to 3 16 bit timer counters.
- Timer, counter and PWM operations.
- MSSP ports with support for SPI and I2C protocols.
- Addressable USART modules.
- Up to 16 channel 10 bit ADC with high sampling rate.
- Comparators.
- Low Voltage and Brown out detector.
- Watchdog timer and In-System programming via two pins.

Power Managed Features.

o Dynamic switching to low power oscillator.
o SLEEP Mode.
o 6 power managed modes.



Hardware: Functional Description and Interfacing.

The Micro-PIC18F microcontroller board has been specifically designed keeping in mind the needs of students for learning the PIC architecture. The board gives a complete overview for interfacing various peripheral devices which are used in the industry and consumer devices alike. A hands-on with the board will develop in the student the experience to design and implement various devices and products based on the PIC Microcontroller.

Following are the features of the Micro-PIC18F Microcontroller board.

• PIC18F microcontroller from Microchip running at maximum 48 MHz.
• UART with USB to Serial driver for PC Connectivity.
• On-Board 16x2 character LCD module with provision for 128x64 Graphical LCD.
• On-Board 16 Key Matrix Keypad.
• 8 general purpose LED's.
• Two Multiplexed Common Anode Seven Segment display.
• DS1307 I2C RTC with power backup.
• EEPROM with I2C interface.
• SD/MMC card on SPI interface.
• Two ADC channels of 10 bit on Chip ADC interfaced to External voltage source.
• Stepper Motor and DC Motor Driver with L293 Chip.
• Peripheral selection using switches.

Experiment No: 04

Aim:- To write an 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 Required:

PIC 18F4550 Trainer kit, MPLAB X IDE, XC8 Compiler, PIC Loader

Theory:

I/O Ports:

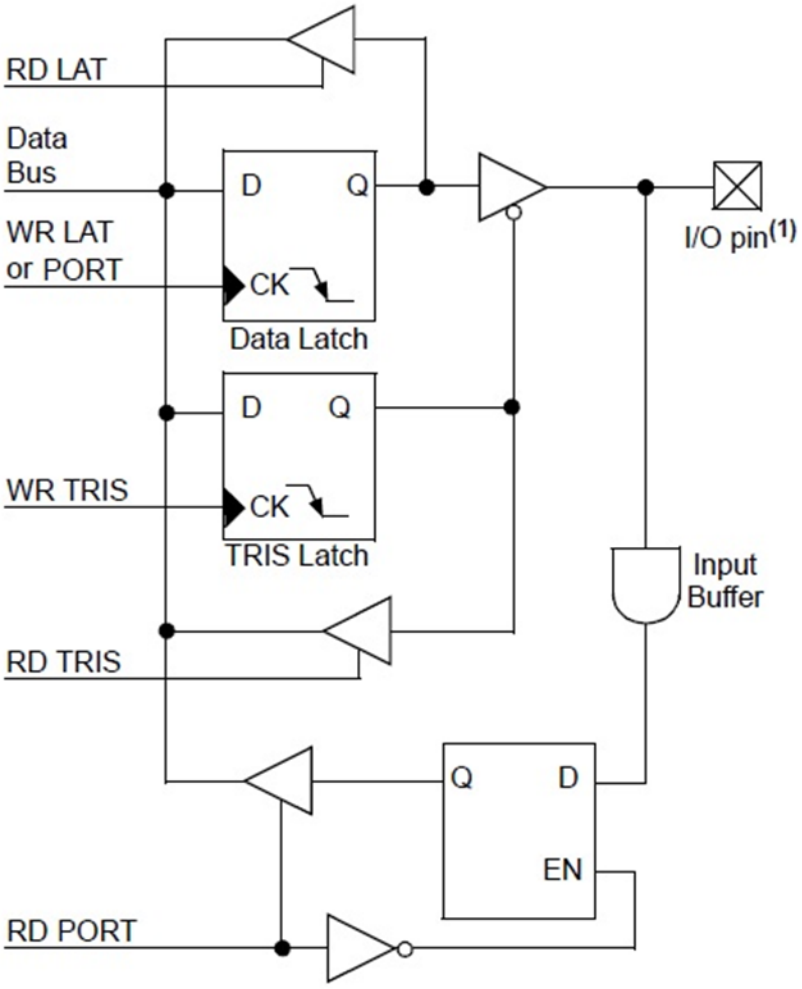
Depending on the device selected and features enabled, there are up to five ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.
Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The Data Latch register (LATA) is useful for read - modify - write operations on the value driven by the I/O pins.

Reading the PORTA register reads the status of the pins; writing to it will write to the port latch.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure.



Note 1: I/O pins have diode protection to VDD and VSS.

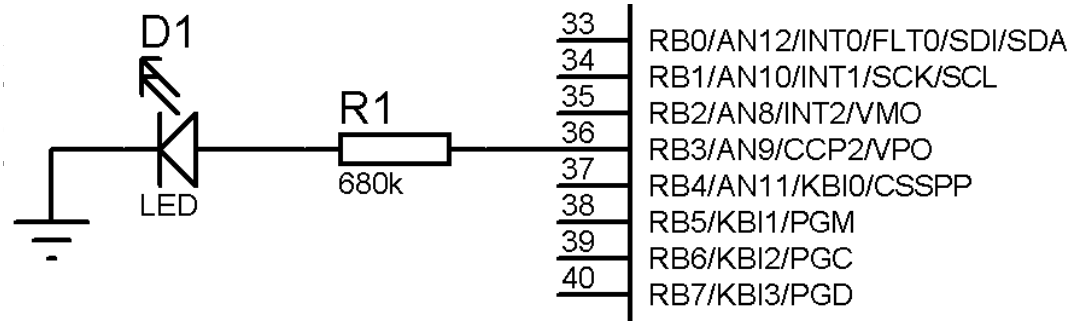
Writing a '1' to the TRIS register configures the pin as input; whereas writing a '0' configures the pin as output.

When configured as output, data can be written on the I/O pin using the LAT or the PORT register.

When configured as input, the status of the I/O pin can be read using the PORT register.

Interfacing LED with microcontroller:

1.



a. Configuring entire PORTB as output:
   TRISB = 0b00000000; or 0x00
b. Configuring only the RB3 pin as output:
   TRISBbits.TRISB3 = 0; other pins remain unaffected.

2. Writing the data on the PORT pin to drive the LED ON or OFF. To turn the LED ON, we can follow one of these steps:
   a. LATB = 0b00001000; or 0x08 or
   b. PORTB= 0b00001000; or 0x08.
   c. LATBbits.LATB3 = 1; or
   d. PORTBbits.RB3 = 1;
3. To turn the LED OFF, we can follow one of these steps:
   a. LATB = 0b00000000; or 0x00
   b. PORTB=0b00000000; or 0x00
   c. LATBbits.LATB3 = 0; or
   d. PORTBbits.RB3 = 0;

Interfacing Relay and Buzzer with microcontroller:

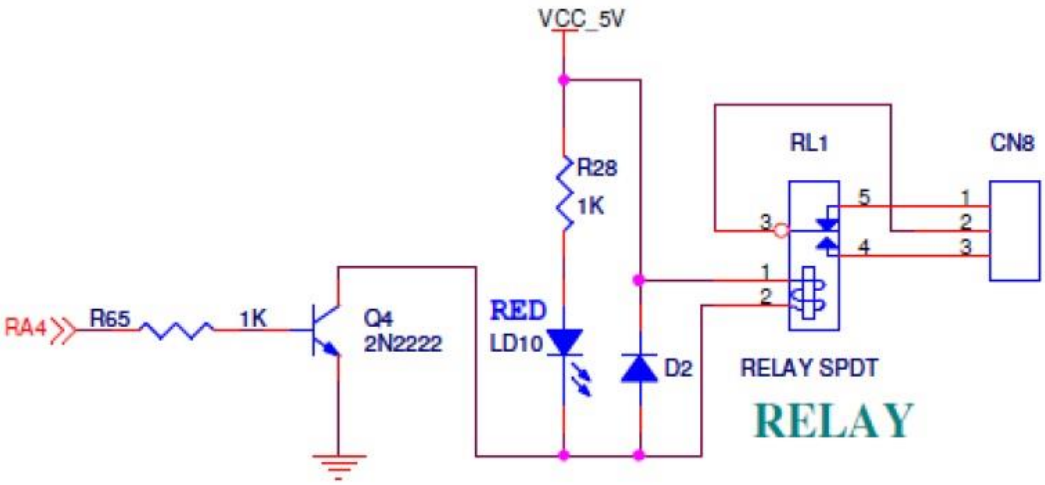
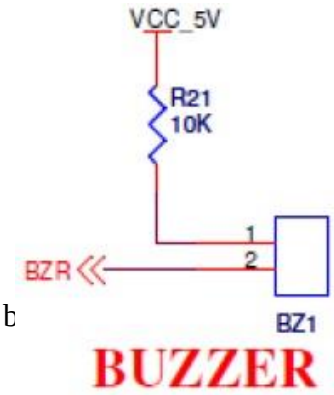


Fig.Buzzer Interfacing



To turn the relay and b w the same steps as that of LED.

However, from the figure we can see that the logic of the buzzer is inverted i.e. when we write '0' to the PORT pin connected to the buzzer, it is turned ON; whereas on writing a '1' turns the buzzer OFF.

Interfacing a BUTTON with microcontroller:

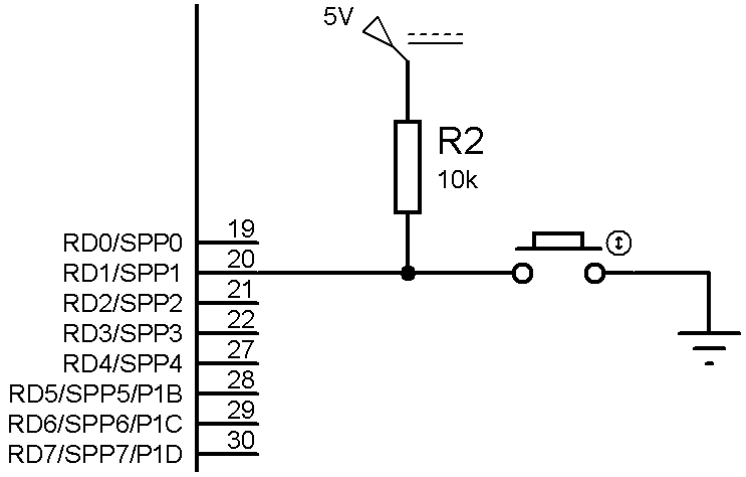
To interface a push-button with a pin of the microcontroller, one end of the button is grounded, while the other end is connected to the pin of the microcontroller.

The microcontroller pin is pulled HIGH with the help of a Pull-Up resistor or by setting the LAT register bit HIGH.

When the button is not pressed i.e. when it is open, the PORT pin receives logic '1' through the Pull-Up resistor or through the LAT register.

When the button is pressed, the PORT pin receives logic '0' since it is connected to the ground directly via the closed push-button.

The status or the logic received by the pin is read with the help of the PORT register.



Algorithm:

Step 1: Configure port B as output port
Step 2: Set D0 and D1 as input port
Step 3: Set buzzer and relay pin D2 and A4 as output port
Step 4: If button 1 is pressed, then relay off, buzzer off and chase LED from right to left
Step 5: If button 2 is pressed, then relay on, buzzer on and chase LED from left to right
Step 6: End

Procedure:

1. Open MPLABX IDE and create a new project.
   File-> New Project->Microchip Embedded -> Standalone Project.
   Select device PIC18F4550.
   Select compiler XC8.
   Project name -> Finish

2. Create a C source file: Right click on the Source Files
   New -> C source file
   Give a name to the file.
3. Copy or type the source program in the C file created.
4. Save the file.
5. Right click on the project name, and select "Clean and Build".
6. Now, open the PicLoader.
7. Settings -> Select COM port.
8. Assert break (pause) and press the reset button on the kit.
9. Click on the Bootloader Mode. If error appears "No Bootloader Found", repeat step 7,8,9.
10. Browse the HEX file and click on Write Device.

After writing the program onto the microcontroller is complete, follow these steps on the hardware:

1. Press the Reset button.
2. Move SW22 switch towards DIP_ON to select SW27 and SW28.
3. Move SW21 switch away from LCD_ON.
4. Move SW20 switch towards LED_ON.

The LEDs will appear to chase in one direction. When SW27 is pressed, the direction of the chasing of LEDs is reversed and the buzzer and relay are turned ON.
When SW28 is pressed, the direction of chasing of LEDs is changed again and the buzzer and relay is turned OFF.
This process continues in an infinite loop.

Conclusion:

Experiment No: 05

Aim:- To write an Embedded C Program for Interfacing of LCD to PIC18F4550 microcontroller

Apparatus Required:

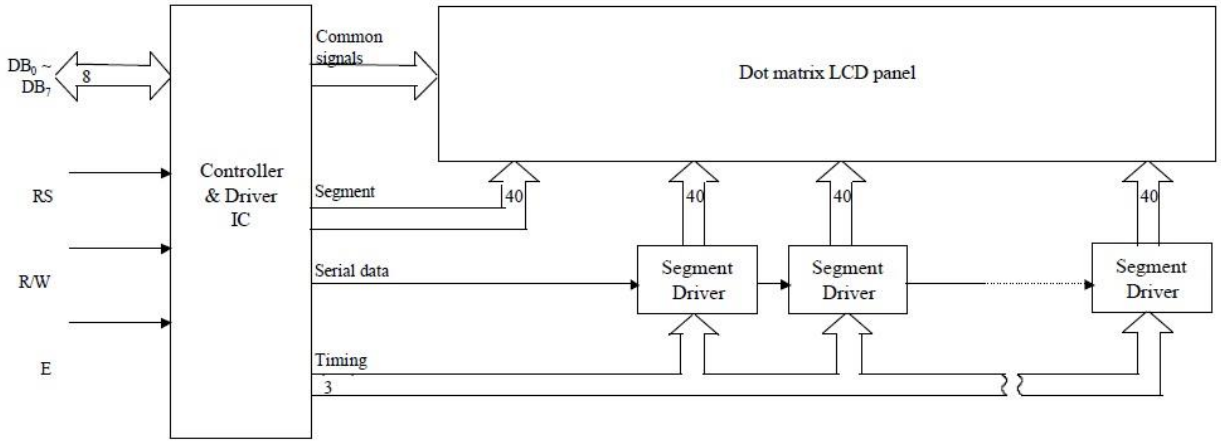
PIC 18F4550 Trainer kit, MPLAB X IDE, XC8 Compiler, PIC Loader

Theory:

The Liquid Crystal Display (LCD) module used in the kit is16x2 JHD 162A which supports HD44780 LCD controller. The controller is equipped with an internal character generator ROM, RAM and RAM for display data.

Characteristics:

- 5 x 7 dots plus cursor
- 4 bit or 8 bit interface with microcontroller is possible.
- Display data RAM 80 x bit (max. 80 characters).
- Character generator ROM 160 5 x 7 Character fonts.
- Custom ROM codes available.
- Character generator RAM Program write (64 x 8 bit).
- Wide variety of operating instructions: Display clear, Cursor home, Display ON/OFF, Display cursor blink, Cursor shift, Display shift.
- Internal automatic reset circuit upon power up.
- Internal oscillator circuit.



Signal Name	No. of Lines	Input/Output	Description
Vcc	1		Power Supply
Vss	1		Power Supply
Vee	1		Contrast Adjustment
RS	1	Input	Register Select: '0' – Instruction '1' – Data
RW	1	Input	'0' – Write '1' – Read
E	1	Input	Enable: data / instruction sampled at falling edge of Enable pulse.
D0-D3	4	Input / Output	Low order data bus. Not used in 4-bit operation.
D4-D7	4	Input/Output	High order data bus.DB7 can be used as a busy flag. Used in 4 bit operation.
LED+/-	2		Power Supply for LCD backlight.

Pin Description:

Vee pin is connected to a potentiometer which forms a voltage divider bias. By varying the potentiometer, we can change its output to set the contrast of the LCD to the required value.

Software: In the program for LCD, we require 3 basic functions:

1. SendData( ) ; This function is required to send the data that needs to be displayed on the LCD.
2. SendInstruction( ) ; This function is required to send an instruction to the LCD.
3. InitLCD( ); This function is required to initialize the LCD module.

The initialization program configures the port pins as output and defines the behavior of the LCD module. For example, whether the LCD should work in 4-bit mode or 8-bit mode, whether the cursor should be ON or OFF, if ON, whether it should be steady or blinking, etc.

List Of Instructions:

Instruction	Code										Description	Execution time (max.) when fcp or fosc is 250 kHz
	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	15.2ms
Return Home	0	0	0	0	0	0	0	0	1	x	Sets DD RAM address 0 in address counter. Also returns shifted display to original position. DD RAM contents remain unchanged.	15.2ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift or display. These operations are performed during data write and read.	40µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Moves cursor and shifts display without changing DD RAM contents.	40µs
Function Set	0	0	0	0	1	DL	N	F	x	x	Sets interface data length (DL), number of display lines (N) and character font (F).	40µs
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address. CG RAM data is sent and received after this setting.	40µs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. DD RAM data is sent and received after this setting.	40µs
Read Busy Flag & Address	0	1	BF	AC							Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	40µs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40µs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40µs
	I/D=1 : Increment I/D=0 : Decrement S=1 : Accompanies display shift S/C=1 : Display shift S/C=0 : Cursor move R/L=1 : Shift to the right										DD RAM : Display Data RAM CG RAM : Character Generator RAM ACG : CG RAM address ADD : DD RAM address. Corresponds to cursor address. AC : Address counter used for both	Execution time changes when frequency changes. Example: When fcp or fosc is 270kHz:
	R/L=0 : Shifts to the left DL=1 : 8 bits, DL=0 : 4 bits N=1 : 2 lines, N=0 : 1 line F=1 : 5x10 dots, F=0 : 5x7 dots BF=1 : Internally operating BF=0 : Can accept instruction										DD and CG RAM address.	40µs x 250/270 = 37 µs

1. 0x20: 4-bit, 1 line.
2. 0x28: 4-bit, 2 lines.
3. 0x30: 8-bit, 1 line.
4. 0x38: 8-bit, 2 lines.
5. 0x06: Entry mode (Increment cursor).
6. 0x04: Decrement cursor.
7. 0x08: Display OFF, Cursor OFF. (DDRAM content not affected)
8. 0x0C: Display ON, Cursor OFF.
9. 0x0E: Display ON, Cursor ON.
10. 0x0F: Display ON, Cursor Blinking.
11. 0x80: Cursor to 1st line, 1st position.
12. 0xC0: Cursor to 2nd line, 1st position.
13. 0x01: Clear Display.

Conclusion:

Experiment No : 6

Aim: To write an Embedded C program to generate square wave using Timer and Interrupt.

Apparatus Required:

PIC 18F4550 Trainer kit, MPLABX IDE, XC8 Compiler, PicLoader

Theory:

The PIC 18 has two to five timers depending on the family members. They are referred to as Timers 0,1,2,3 and 4. They can be used either as timers to generate a time delay or as counters to count events happening outside the microcontroller.

Every timer needs a clock pulse to tick. The source can be internal or external. If we use the internal clock source , then $1/4^{th}$ of the frequency of the crystal oscillator on the OSC1 and OSC2 pins (Fosc/4) is fed into the timer. Therefore, it is used for time delay generation and for that reason is called as timer.

Each 16 bit timer is accessed as two separate registers of low byte (TMRxL) and high byte (TMRxH) . Each timer also has the TCON (timer control) register for setting modes of operation.

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt on overflow

T0CON (Timer 0 control) register:

Each timer has a control register , called TCON , to set the various timer operation modes. T0CON is an 8-bit register used for control of Timer0. The bits for T0CON are shown in figure:

R 10-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

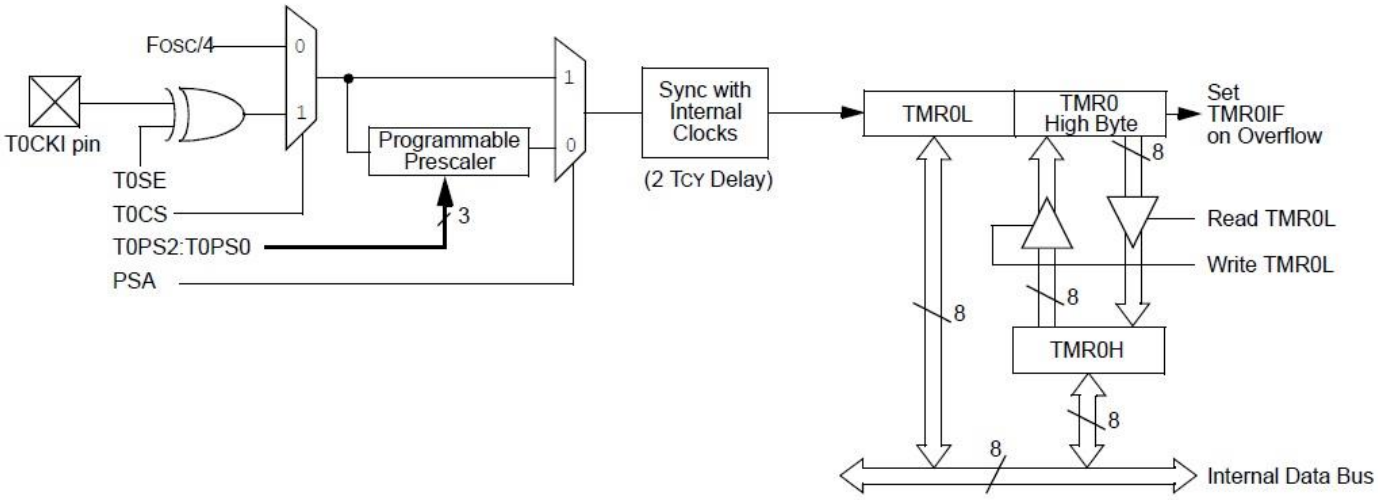
bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value



The TMR0IF bit (Timer 0 interrupt flag) is part of the INTCON (interrupt control) register. When the timer reaches its maximum value of FFFFH , it rolls over to 0000, and TMR0IF is set to 1.

The PIC18F4550 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low-priority level.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON, INTCON2, INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

When an interrupt is responded to, the global interrupt enable (GIE) bit is cleared to disable further interrupts. High-priority interrupt sources can interrupt a low priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address. Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The “return from interrupt” instruction, RETFIE, exits the interrupt routine and sets the GIE bit which re-enables interrupts.

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
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 **GIE/GIEH**: Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high-priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL**: Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low-priority peripheral interrupts (if GIE/GIEH = 1)
0 = Disables all low-priority peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE**: INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF**: INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit(1)
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

RCON Register

The RCON register contains flag bits which are used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the IPEN bit which enables interrupt priorities.

REGISTER 9-10: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
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 **IPEN**: Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6 **SBOREN**: BOR Software Enable bit(1)
For details of bit operation, see Register 4-1.

bit 5 **Unimplemented**: Read as '0'

bit 4 **$\overline{RI}$**: RESET Instruction Flag bit
For details of bit operation, see Register 4-1.

bit 3 **$\overline{TO}$**: Watchdog Time-out Flag bit
For details of bit operation, see Register 4-1.

bit 2 **$\overline{PD}$**: Power-Down Detection Flag bit
For details of bit operation, see Register 4-1.

bit 1 **$\overline{POR}$**: Power-on Reset Status bit(2)
For details of bit operation, see Register 4-1.

bit 0 **$\overline{BOR}$**: Brown-out Reset Status bit
For details of bit operation, see Register 4-1.

To generate a square wave using timer interrupt, follow these steps:

1. Configure port pin as output.
2. Configure Timer0 in 16 bit mode and 1:256 prescalevalue.
3. Set Timer0 Interrupt Enable bit and clear the Timer0 Interrupt Flag.
4. Enable Global Interrupt register (GIE).
5. Load the Timer0 16 bit register in TMR0L and TMR0H.
6. Enable Timer0.

Interrupt Service Routine: When the Timer0 register overflows from 0xFFFF to 0x0000, the Timer0 flag (TMR0IF) is set and the Interrupt Service Routine is invoked.

1. Check if the Timer0 Interrupt flag is set.
2. If the flag is set, clear the flag and stop the timer.
3. Toggle the port pin.
4. Reload the Timer0 values and start the timer.
5. Return to the main function.

Algorithm:

1) Make PortB as output port for LEDs
2) Load the value into T0CON register indicating which mode (16 bit mode with prescaler 1:256)

3) Load register TMR0H followed by TMR0L with initial count values.
4) Start the timer
5) Enable global and Timer0 interrupts; Clear Timer0 interrupt flag
6) If timer0 interrupt flag is set, Stop the timer
7) Reload Timer0
8) Toggle PORTB
9) Repeat step 4

Procedure:

After compiling and writing the program through the PicLoader onto the microcontroller is complete, follow these steps on the hardware:

5. Move SW21 away from LCD_ON.
6. Move SW20 switch towards LED_ON.
7. Press the Reset button.

You can observe the LEDs blink with a period of 200ms i.e. 5 times per second.

Conclusion:

Experiment No : 7

Aim: To write an Embedded C program to interface serial port with PC

Apparatus Required:

PIC 18F4550 Trainer kit, MPLABX IDE, XC8 Compiler, Pic Loader

Theory:

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of the two serial I/O modules. (Generically, the USART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

REGISTER 20-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7: CSRC: Clock Source Select bit
Asynchronous mode: Don't care.
bit 6: TX9: 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission
bit 5: TXEN: Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled
bit 4 SYNC: EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode
bit 3: SENDB: Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)

0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2: BRGH: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1: TRMT: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0: TX9D: 9th bit of Transmit Data
Can be address/data bit or a parity bit.

REGISTER 20-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7: SPEN: Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6: RX9: 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5: SREN: Single Receive Enable bit
Asynchronous mode: Don't care.

bit 4: CREN: Continuous Receive Enable bit
1 = Enables receiver
0 = Disables receiver

bit 3: ADDEN: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Don't care.

bit 2: FERR: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1: OERR: Overrun Error bit

1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0: RX9D: 9th bit of Received Data
This can be address/data bit or a parity bit.

REGISTER 20-3: BAUDCON: BAUD RATE CONTROL REGISTER

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

bit 7: ABDOVF: Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6: RCIDL: Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

bit 5: RXDTP: Received Data Polarity Select bit
1 = RX data is inverted
0 = RX data received is not inverted

bit 4: TXCKP: Clock and Data Polarity Select bit
1 = TX data is inverted
0 = TX data is not inverted

bit 3: BRG16: 16-Bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only

bit 2: Unimplemented: Read as '0'

bit 1: WUE: Wake-up Enable bit
1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected

bit 0 ABDEN: Auto-Baud Detect Enable bit
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed

SPBRG: Baud Rate Generator

The BRG is a dedicated 8-bit, or 16-bit, generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit (BAUDCON<3>) selects 16-bit mode.

Given the desired baud rate n and Fosc, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table:

Configuration Bits		SPBRG Mode	Baud Rate Formula
BRG16	BRGH		
0	0	8-bit	Fosc / [64(n+1)]
0	1	8-bit	Fosc / [16(n+1)]
1	0	16-bit	
1	1	16-bit	Fosc / [4(n+1)]

Algorithm:

Step 1: Initialize UART for baud rate 9600
Step 2: Set the transmit pin as output and receiver pin as input
Step 3: Set TXSTA as Asynchronous 8-bit, transmit enabled and low speed baud rate select
Step 4: Set RCSTA as serial port enabled, 8-bit data and single receive enable
Step 5: Wait till receive buffer becomes full, return the received data to transmit
Step 6: Wait till the transmit register is empty
Step 7: Transmit the received data to the terminal.
Step 8: Goto step 5

Procedure: To initialize the microcontroller for serial communication at 9600 baud rate,

- calculate the SPBRG value.
  BRG16 = 0 : 8-bit baud rate generator.
  BRGH = 0: low baud rate.
  Therefore, n = Fosc / [64 * (SPBRG + 1)]
  i.e. SPBRG = [Fosc / (64 * n)] – 1
  forFosc = 48MHz, n = 9600, we get SPBRG = 77
- Set RC6/TX pin as output and RC7/RX pin as input
- Enable serial port with SPEN = 1
- To transmit a character, wait till TXREG becomes empty, then send the character via TXREG.
- To receive a character, check RCIF bit in the PIR interrupt register and wait till the receive buffer becomes full. Once the RCIF bit is set, read the data from RCREG.

After writing the program through the PicLoader onto the microcontroller is complete, follow these steps on the hardware:

1. Click the “RUN” (play) button in the picloader firmware or press F2.
2. Press the Reset button.
3. Press any key on the PC and the key will be echoed back by the microcontroller on the HyperTerminal screen of the picloader.

Conclusion:

Experiment No : 8

Aim: To write an Embedded C program to interface analog voltage 0-5V to internal ADC and display value on LCD

Apparatus Required:

PIC 18F4550 Trainer kit, MPLAB X IDE, PIC loader, XC8 complier

Theory:

Analog to digital converters are among the most widely used devices for data acquisition. Digital computers use binary values, but in the physical world everything is analog. Temperature, Pressure, humidity and velocity are a few examples of physical quantities that we deal with every day. A physical quantity is converted to electrical signals using device called a transducer. Transducers are also referred to as sensors. Sensors for temperature, velocity, pressure , light , and many other natural quantities produce an output that is voltage or current. Therefore, we need an analog to digital converter to translate the analog signals to digital numbers so that the microcontroller can read and process them.

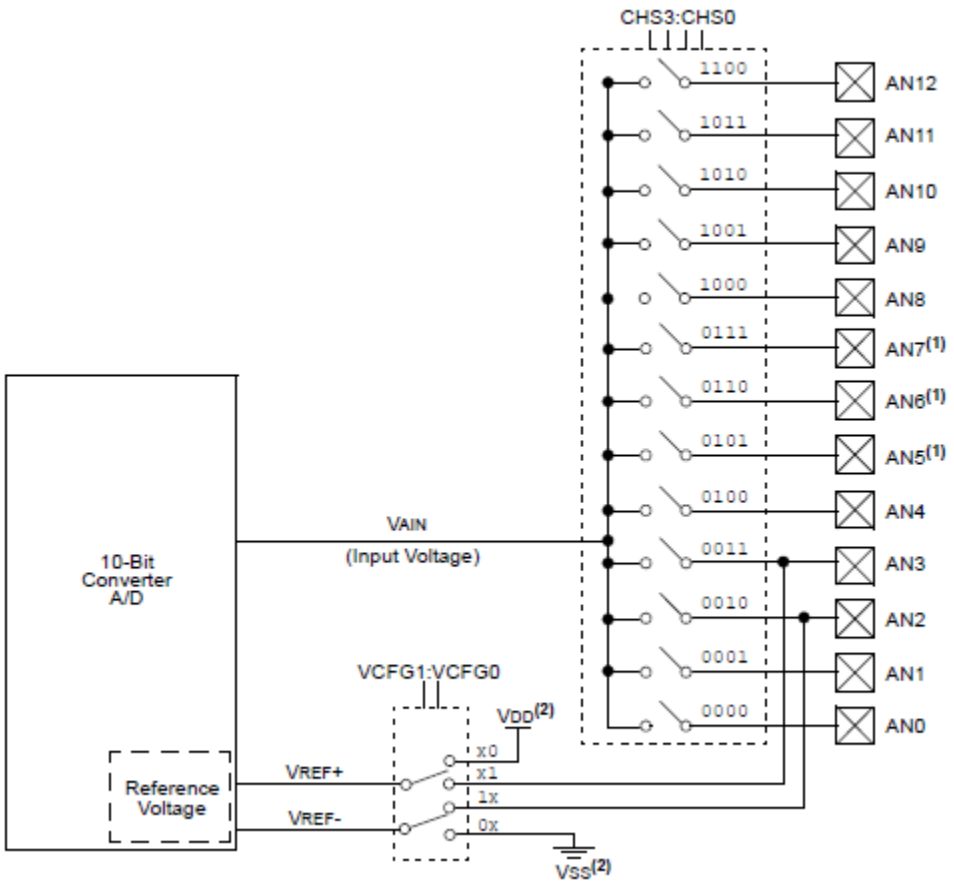
Because the ADC is widely used in data acquisition, in recent years an increasing number of microcontrollers have an on-chip ADC peripheral, just like timers and USART. An on-chip ADC eliminates the need for an external ADC connection, which leaves more pins or other I/O activities. The vast majority of the PIC 18 chips come with 8 channels of ADC and some PIC 18s have as many as 16 channels of ADCs.

The ADC peripheral of the PIC 18 has the following characteristics:

(a) It is a 10-bit ADC
(b) It can have 5 to 15 channels of analog input channels, depending on the family members. Pins RA0-RA7 of PORTA are used for the 8 analog channels.
(c) The converted output binary data is held by two special function registers called ADRESL and ADRESH.
(d) Because the ADRESH:ADRESL registers give us 16 bits and ADC data out is only 10 bits wide, 6 bits of the 16 bit are used.
(e) We have option of using Vdd(Vcc), the voltage source of PIC 18 chip itself, as the Vref or connecting it to an external voltage source for the Vref
(f) The conversion tie is dictated by the Fosc of crystal frequency connected to the OSC pins.

Many of the above features can be programmed by way of ADCON0 and ADCON1.

Block Diagram of A/ D Converter



ADCON0:

The ADCON0 register is used to set the conversion time and select the analog input channel among other things. In order to reduce the power consumption of the PIC18 , the ADC feature is turned off when the microcontroller is powered up. We turn on the ADC with the ADON bit of the ADCON0 register. The other important bit is GO/DONE bit. We use this bit to start conversion and monitor it to see if conversion has ended. The conversion time is set with the ADCS bits

ADCON0

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

.

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

ADCON1 Register

It configures the functions of port pins

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

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

bit 5 **VCFG1:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

ADCON2 Register:

It configures the A/D clock source, programmed acquisition time and justification

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7	**ADFM:** A/D Result Format Select bit 1 = Right justified 0 = Left justified
bit 6	**Unimplemented:** Read as '0'
bit 5-3	**ACQT2:ACQT0:** A/D Acquisition Time Select bits 111 = 20 TAD 110 = 16 TAD 101 = 12 TAD 100 = 8 TAD 011 = 6 TAD 010 = 4 TAD 001 = 2 TAD 000 = 0 TAD(1)
bit 2-0	**ADCS2:ADCS0:** A/D Conversion Clock Select bits 111 = FRC (clock derived from A/D RC oscillator)(1) 110 = FOSC/64 101 = FOSC/16 100 = FOSC/4 011 = FRC (clock derived from A/D RC oscillator)(1) 010 = FOSC/32 001 = FOSC/8 000 = FOSC/2

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set. It also gives users the option to use an automatically determined acquisition time. Acquisition time may be set with the ACQT2:ACQT0 bits (ADCON2<5:3>) which provide a range of 2 to 20 TAD. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Manual acquisition is selected when ACQT2:ACQT0 = 000. When the GO/DONE bit is set, sampling is stopped and a conversion begins.

By using ADCS (A/D clock source) bits , we can set the A/D conversion time.

The A/D conversion time per bit is defined as TAD. The source of the A/D conversion clock is softwareselectable. There are seven possible options for TAD:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC Oscillator

The following steps should be followed to perform an A/D conversion:

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.

Algorithm:

Step 1: Make the pin for the selected ADC channel an input pin.
Step 2: Initialize the port to which LCD is connected as output
Step 3: Initialize ADC and LCD
Step 4: Read the Analog input channel
Step 5: Start ADC , select the channel
Step 6: Start the conversion
Step 7: Wait till A/D conversion is complete
Step 8: Display the converter digital output on LCD

Conclusion: