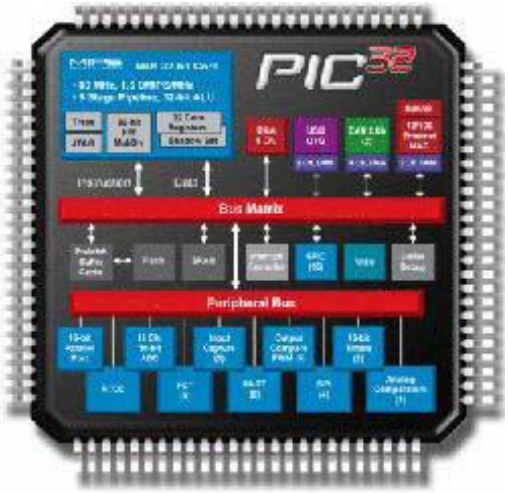
# UNIT-VI: PIC MICRO CONTROLLERS

## Introduction to PIC Microcontroller:




**PIC Microcontroller**

PIC microcontroller was developed in the year 1993 by microchip technology. The term PIC stands for Peripheral Interface Controller. Initially this was developed for supporting PDP computers to control its peripheral devices, and therefore, named as a peripheral interface device. These microcontrollers are very fast and easy to execute a program compared with other microcontrollers. PIC Microcontroller architecture is based on Harvard architecture. PIC microcontrollers are very popular due to their ease of programming, wide availability, easy to interfacing with other peripherals, low cost, large user base and serial programming capability (reprogramming with flash memory), etc.

We know that the microcontroller is an integrated chip which consists of CPU, RAM, ROM, timers, and counters, etc. In the same way, PIC microcontroller architecture consists of RAM, ROM, CPU, timers, counters and supports the protocols such as SPI, CAN, and UART for interfacing with other peripherals. At present PIC microcontrollers are extensively used for industrial purpose due to low power consumption, high performance ability and easy of availability of its supporting hardware and software tools like compilers, debuggers and simulators.

## What is a PIC Microcontroller?

PIC (Programmable Interface Controllers) microcontrollers are the worlds smallest microcontrollers that can be programmed to carry out a huge range of tasks. These microcontrollers are found in many electronic devices such as phones, computer control systems, alarm systems, embedded systems, etc. Various types of microcontrollers exist, even though the

best are found in the GENIE range of programmable microcontrollers. These microcontrollers are programmed and simulated by circuit-wizard software.

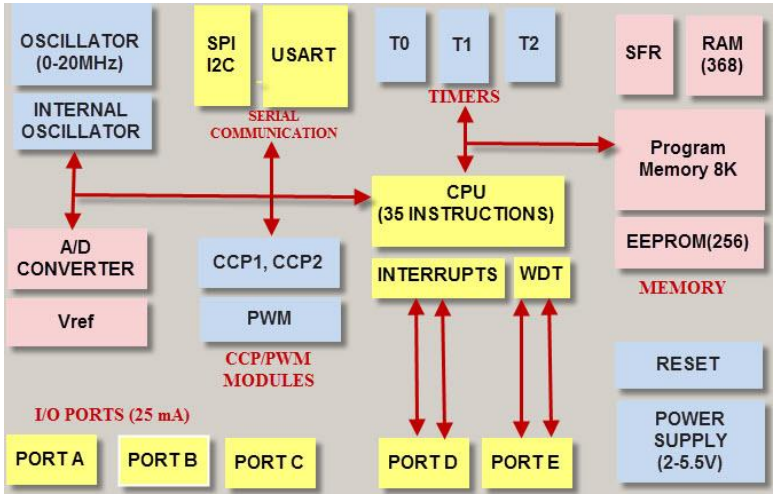
Every PIC microcontroller architecture consists of some registers and stack where registers function as Random Access Memory (RAM) and stack saves the return addresses. The main features of PIC microcontrollers are RAM, flash memory, Timers/Counters, EEPROM, I/O Ports, USART, CCP (Capture/Compare/PWM module), SSP, Comparator, ADC (analog to digital converter), PSP (parallel slave port), LCD and ICSP (in circuit serial programming) The 8-bit PIC microcontroller is classified into four types on the basis of internal architecture such as Base Line PIC, Mid-Range PIC, Enhanced Mid-Range PIC and PIC18

The term PIC stands for Peripheral Interface Controller. It is the brain child of Microchip Technology, USA. Originally this was developed as a supporting device for PDP computers to control its peripheral devices, and therefore named as PIC, Peripheral Interface Controller. They have coined this name to identify their single chip micro controllers. These 8-bit micro controllers have become very important now -a -days in industrial automation and embedded applications etc.

One of the earlier versions of PIC Microcontrollers is PIC16C6x/7x. The 7x family has an enhancement of Analog to Digital converter capability. These µcs are available with a range of capabilities packaged in both dual in-line (DIP) packages and surface-mount packages. These are available in 28 pin DIP, 40 pin DIP, 44 pin surface mount package…etc. Some of PIC controllers contain the letter A in their number. The presence of A indicates the brown-out reset feature, which causes a reset of the PIC when the Power Supply voltage drops below 4.0v.

## Architecture of PIC Microcontroller

The PIC microcontroller architecture comprises of CPU, I/O ports, memory organization, A/D converter, timers/counters, interrupts, serial communication, oscillator and CCP module which are discussed in detailed below.




**Architecture of PIC Microcontroller**

**CPU** (Central Processing Unit)

It is not different from other microcontrollers CPU and the PIC microcontroller CPU consists of the ALU, CU, MU and accumulator, etc. Arithmetic logic unit is mainly used for arithmetic operations and to take logical decisions. Memory is used for storing the instructions after processing. To control the internal and external peripherals, control unit is used which are connected to the CPU and the accumulator is used for storing the results and further process.

**Memory Organization**

The memory module in the PIC microcontroller architecture consists of RAM (Random Access Memory), ROM (Read Only Memory) and STACK.

**Random Access Memory** (RAM)

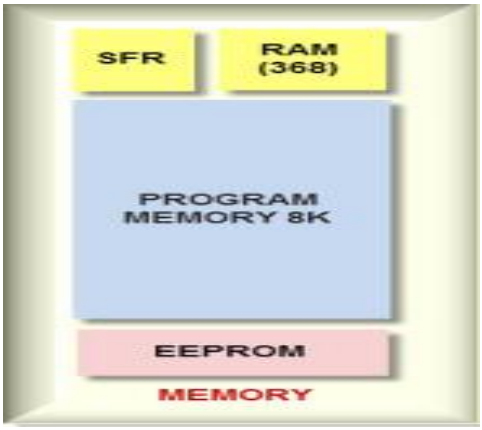
RAM is an unstable memory which is used to store the data temporarily in its registers. The RAM memory is classified into two banks, and each bank consists of so many registers. The RAM registers are classified into two types: Special Function Registers (SFR) and General Purpose Registers (GPR).

- General Purpose Registers (GPR)

  These registers are used for general purpose only as the name implies. For example, if we want to multiply two numbers by using the PIC microcontroller. Generally, we use registers for multiplying and storing the numbers in other registers. So these registers don't have any special function,- CPU can easily access the data in the registers.

- Special Function Registers

  These registers are used for special purposes only as the name SFR implies. These registers will perform according to the functions assigned to them , and they cannot be used as normal registers. For example, if you cannot use the STATUS register for storing the data, these registers are used for showing the operation or status of the program. So, user cannot change the function of the SFR; the function is given by the retailer at the time of manufacturing.




**Memory Organization**

**Read Only Memory** (ROM)

Read only memory is a stable memory which is used to store the data permanently. In PIC microcontroller architecture, the architecture ROM stores the instructions or program, according to the program the microcontroller acts. The ROM is also called as program memory, wherein the user will write the program for microcontroller and saves it permanently, and finally the program is executed by the CPU. The microcontrollers performance depends on the instruction, which is executed by the CPU.

Electrically Erasable Programmable Read Only Memory (**EEPROM**)

In the normal ROM, we can write the program for only once we cannot use again the microcontroller for multiple times. But, in the EEPROM, we can program the ROM multiple times.

**Flash Memory**

Flash memory is also programmable read only memory (PROM) in which we can read, write and erase the program thousands of times. Generally, the PIC microcontroller uses this type of ROM.

**Stack**

When an interrupt occurs, first the PIC microcontroller has to execute the interrupt and the existing process address. Then that is being executed is stored in the stack. After completing the execution of the interrupt, the microcontroller calls the process with the help of address, which is stored in the stack and get executes the process.

**I/O Ports**

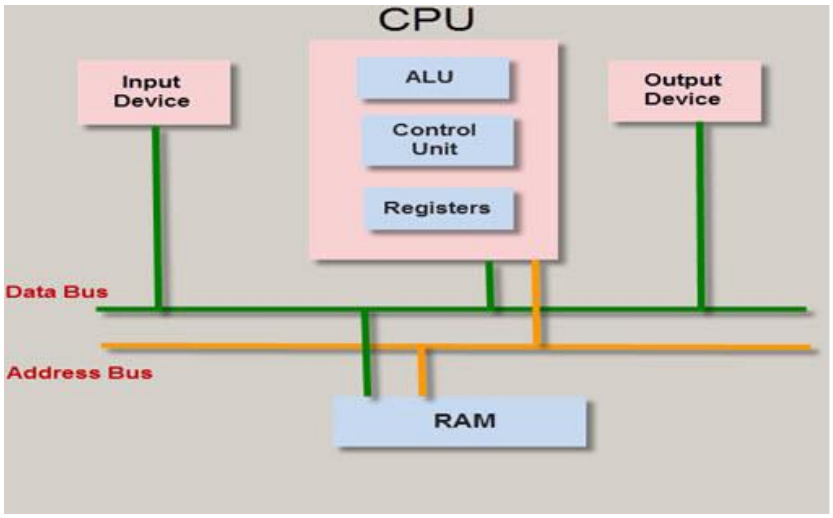
- The series of PIC16 consists of five ports such as Port A, Port B, Port C, Port D & Port E.
- Port A is an 16-bit port that can be used as input or output port based on the status of the TRISA (Tradoc Intelligence Support Activity) register.
- Port B is an 8- bit port that can be used as both input and output port.
- Port C is an 8-bit and the input of output operation is decided by the status of the TRISC register.
- Port D is an 8-bit port acts as a slave port for connection to the microprocessor BUS.
- Port E is a 3-bit port which serves the additional function of the control signals to the analog to digital converter.

**BUS**

BUS is used to transfer and receive the data from one peripheral to another. It is classified into two types such as data bus and address.

**Data Bus:** It is used for only transfer or receive the data.

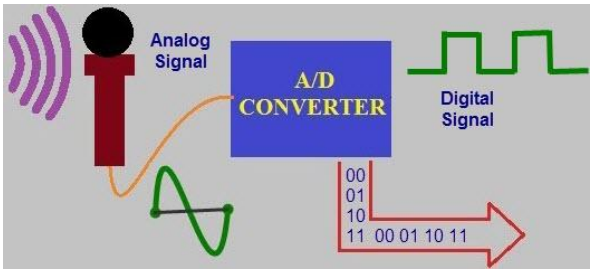
**Address Bus:** Address bus is used to transmit the memory address from the peripherals to the CPU. I/O pins are used to interface the external peripherals; UART and USART both are serial communication protocols which are used for interfacing serial devices like GSM, GPS, Bluetooth, IR , etc.




**BUS**

**A/D converters**

The main intention of this analog to digital converter is to convert analog voltage values to digital voltage values. A/D module of PIC microcontroller consists of 5 inputs for 28 pin devices and 8 inputs for 40 pin devices. The operation of the analog to digital converter is controlled by ADCON0 and ADCON1 special registers. The upper bits of the converter are stored in register ADRESH and lower bits of the converter are stored in register ADRESL. For this operation, it requires 5V of an analog reference voltage.




**A/D CONVERTER**

**Timers/ Counters**

PIC microcontroller has four timer/counters wherein the one 8-bit timer and the remaining timers have the choice to select 8 or 16-bit mode. Timers are used for generating accuracy actions, for example, creating specific time delays between two operations.

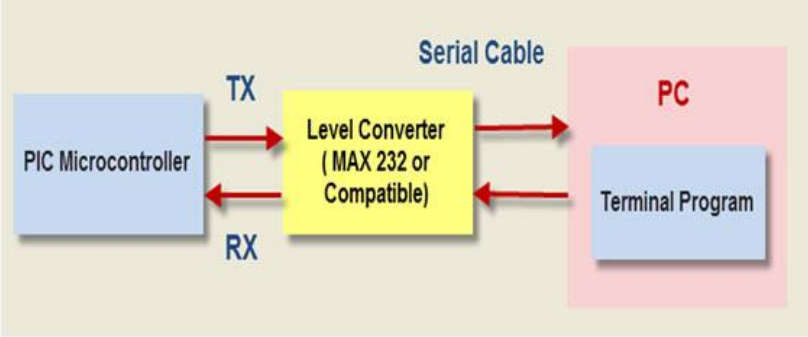
**Interrupts**

PIC microcontroller consists of 20 internal interrupts and three external interrupt sources which are associated with different peripherals like ADC, USART, Timers, and so on.

**Serial Communication**

Serial communication is the method of transferring data one bit at a time sequentially over a communication channel.

- **USART:** The name USART stands for Universal synchronous and Asynchronous Receiver and Transmitter which is a serial communication for two protocols. It is used for transmitting and receiving the data bit by bit over a single wire with respect to clock pulses. The PIC microcontroller has two pins TXD and RXD. These pins are used for transmitting and receiving the data serially.
- **SPI Protocol:** The term SPI stands for Serial Peripheral Interface. This protocol is used to send data between PIC microcontroller and other peripherals such as SD cards, sensors and shift registers. PIC microcontroller support three wire SPI communications between two devices on a common clock source. The data rate of SPI protocol is more than that of the USART.
- **I2C Protocol:** The term I2C stands for Inter Integrated Circuit , and it is a serial protocol which is used to connect low speed devices such as EEPROMS, microcontrollers, A/D converters, etc. PIC microcontroller support two wire Interface or I2C communication between two devices which can work as both Master and Slave device.




**Serial Communication**

**Oscillators**

Oscillators are used for timing generation. Pic microcontroller consists of external oscillators like RC oscillators or crystal oscillators. Where the crystal oscillator is connected between the two oscillator pins. The value of the capacitor is connected to every pin that decides the mode of the operation of the oscillator. The modes are crystal mode, high-speed mode and the low-power mode. In case of RC oscillators, the value of the resistor & capacitor determines the clock frequency and the range of clock frequency is 30KHz to 4MHz.

## CCP module

The name CCP module stands for capture/compare/PWM where it works in three modes such as capture mode, compare mode and PWM mode.

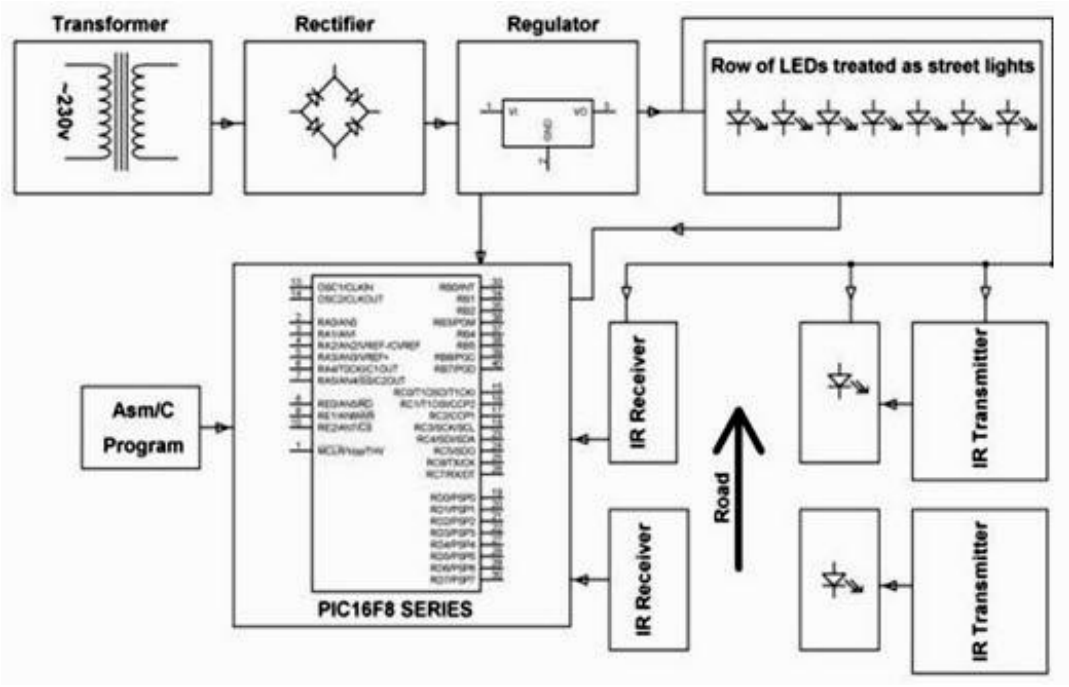
- **Capture Mode:** Capture mode captures the time of arrival of a signal, or in other words, when the CCP pin goes high, it captures the value of the Timer1.
- **Compare Mode:** Compare mode acts as an analog comparator. When the timer1 value reaches a certain reference value, then it generates an output.
- **PWM Mode:** PWM mode provides pulse width modulated output with a 10-bit resolution and programmable duty cycle.

# PIC Microcontroller Applications

The PIC microcontroller projects can be used in different applications, such as peripherals, audio accessories, video games, etc. For better understanding of this PIC microcontroller, the following project demonstrates PIC microcontroller's operations.

**Street Light that Glows on Detecting Vehicle Movement:**

The main intention of this project is to detect the movement of vehicles on highways to switch on a block of street lights ahead of it, and also switch off the trailing lights to conserve energy. In this project, a PIC microcontroller is done by using assembly language or embedded C.




**Street Light that Glows on Detecting Vehicle Movement by Edgefxkits.com**

The power supply gives the power to the total circuit by stepping down, rectifying, filtering and regulating AC mains supply. When there are no vehicles on highway, then all lights will turn OFF so that

the power can be conserved. The IR sensors are placed on the road to sense the vehicle movement. When there are vehicles on highway, then the IR sensor senses the vehicle movement immediately, it sends the commands to the PIC microcontroller to switch ON/OFF the LEDs. A bunch of LEDS will be turned on when a vehicle come near to the sensor and once the vehicle passes away from the sensor the intensity will become lower than the LEDs will turn OFF

**Advantages of PIC Microcontroller:**

- PIC microcontrollers are consistent and faulty of PIC percentage is very less. The performance of the PIC microcontroller is very fast because of using RISC architecture.
- When comparing to other microcontrollers, power consumption is very less and programming is also very easy.
- Interfacing of an analog device is easy without any extra circuitry

**Disadvantages of PIC Microcontroller:**

- The length of the program is high due to using RISC architecture (35 instructions)
- One single accumulator is present and program memory is not accessible

Thus, this is all about PIC microcontroller tutorial which includes PIC microcontroller architecture, advantages, disadvantage and applications. Apart from this, for any help regarding any electronic and electrical projects, you can contact us by commenting in the comment section below.

**The architectures of various PIC microcontrollers can be divided as follows.**

***Low - end PIC Architectures:***

Microchip PIC microcontrollers are available in various types. When PIC microcontroller MCU was first available from General Instruments in early 1980's, the microcontroller consisted of a simple processor executing 12-bit wide instructions with basic I/O functions. These devices are known as low-end architectures. They have limited program memory and are meant for applications requiring simple interface functions and small program & data memories. Some of the low-end device numbers are

12C5XX

16C5X

16C505

***Mid-range PIC Architectures:***

Mid range PIC architectures are built by upgrading low-end architectures with more number of peripherals, more number of registers and more data/program memory. Some of the mid-range devices are

16C6X

16C7X

16F87X

Program memory type is indicated by an alphabet.

C = EPROM

F = Flash

RC = Mask ROM

Popularity of the PIC microcontrollers is due to the following factors (**features**).

1. Speed: Harvard Architecture, RISC architecture, 1 instruction cycle = 4 clock cycles.
2. Instruction set simplicity: The instruction set consists of just 35 instructions (as opposed to 111 instructions for 8051).
3. Power-on-reset and brown-out reset. Brown-out-reset means when the power supply goes below a specified voltage (say 4V), it causes PIC to reset; hence malfunction is avoided.

   A watch dog timer (user programmable) resets the processor if the software/program ever malfunctions and deviates from its normal operation.
4. PIC microcontroller has four optional clock sources.
   - Low power crystal
   - Mid-range crystal
   - High range crystal
   - RC oscillator (low cost).
5. Programmable timers and on-chip ADC.
6. Up to 12 independent interrupt sources.
7. Powerful output pin control (25 mA (max.) current sourcing capability per pin.)
8. EPROM/OTP/ROM/Flash memory option.
9. I/O port expansion capability.
10. Free assembler and simulator support from Microchip at www.microchip.com

**CPU Architecture:**

The CPU uses Harvard architecture with separate Program and Variable (data) memory interface. This facilitates instruction fetch and the operation on data/accessing of variables simultaneously.

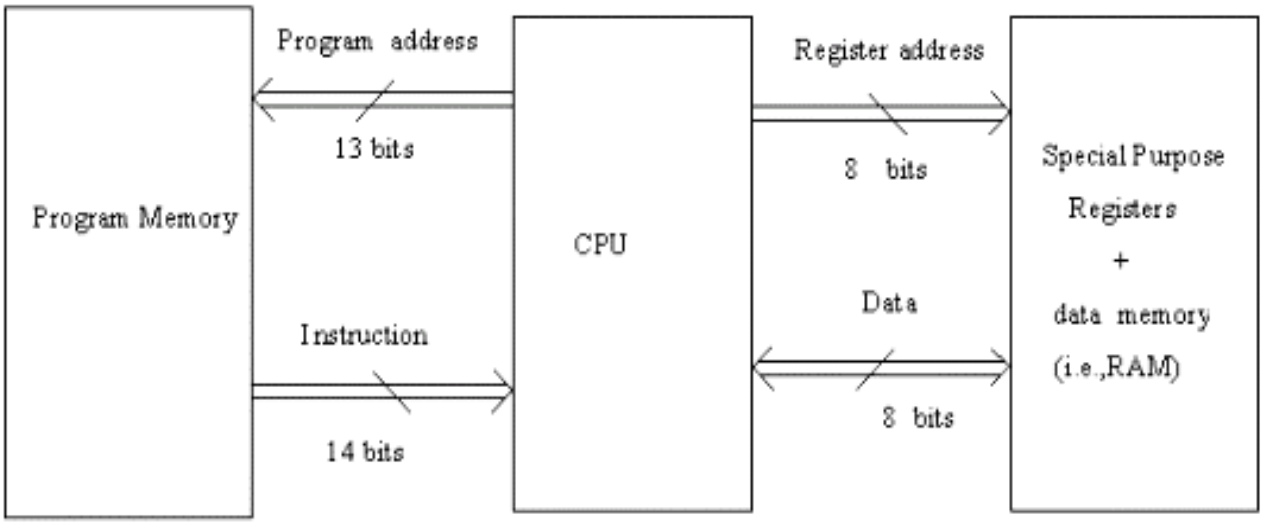



**Fig: CPU Architecture of PIC microcontroller**

**PIC Memory Organization:**

PIC microcontroller has 13 bits of program memory address. Hence it can address up to 8k of program memory. The program counter is 13-bit. PIC 16C6X or 16C7X program memory is 2k or 4k. While addressing 2k of program memory, only 11- bits are required. Hence two most significant bits of the program counter are ignored. Similarly, while addressing 4k of memory, 12 bits are required. Hence the MSB of the program counter is ignored.

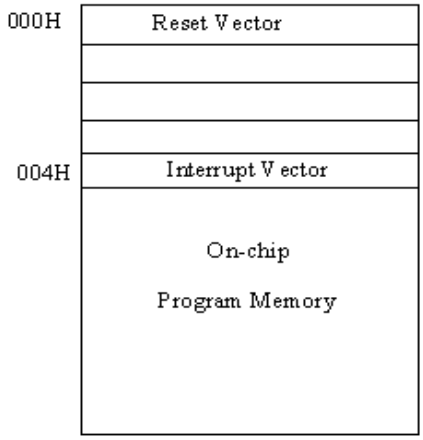



**Fig: Program Memory map**

The program memory map of PIC16C74A is shown in Fig

- ✓ On reset, the program counter is cleared and the program starts at 00H. Here a 'goto' instruction is required that takes the processor to the mainline program.
- ✓ When a peripheral interrupt, that is enabled, is received, the processor goes to 004H. A suitable branching to the interrupt service routine (ISR) is written at 004H.

Data memory (Register Files):

Data Memory is also known as Register File. Register File consists of two components.

1. General purpose register file (same as RAM).
2. Special purpose register file (similar to SFR in 8051).

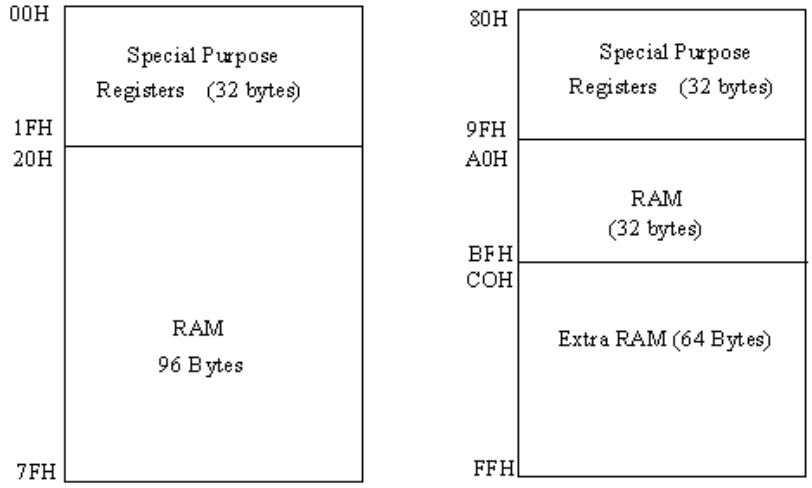



**Fig: Data Memory map**

The special purpose register file consists of input/output ports and control registers. Addressing from 00H to FFH requires 8 bits of address. However, the instructions that use direct addressing modes in PIC to address these register files use 7 bits of instruction only. Therefore the register bank select (RP0) bit in the STATUS register is used to select one of the register banks.

In indirect addressing FSR register is used as a pointer to anywhere from 00H to FFH in the data memory.

**SALIENT FEATURES**

- **Speed:** When operated at its maximum clock rate a PIC executes most of its instructions in 0.2 μs or five instructions per microsecond.
- **Instruction set Simplicity:** The instruction set is so simple that it consists of only just 35 instructions
- **Integration of operational features:** Power-on-Reset (POR) and brown-out protection ensure that the chip operates only when the supply voltage is within specifications. A watch dog timer resets the PIC if the chip malfunctions or deviates from its normal operation at any time.
- **Programmable timer options:** Three timers can characterize inputs, control outputs and provide internal timing for the program execution.

- **Interrupt control:** Up to 12 independent interrupt sources can control when the CPU deal with each sources.
- **Powerful output pin control:** A single instruction can select and drive a single output pin high or low in its 0.2 μs instruction execution time. The PIC can drive a load of up to 25μA.
- **I/O port expansion:** With the help of built in serial peripheral interface the number of I/O ports can be expanded.

EPROM/DIP/ROM options are provided.

- High performance RISC CPU
- Operating speed: DC – 20 MHz clock input DC – 200 ns instruction cycle
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Three Timers Timer0, Timer 1 and Timer 2.
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- 10-bit multi-channel Analog-to-Digital converter
- Selectable oscillator options
- One USART /SCI port with 9-bit address detection.
- Low-power, high-speed CMOS EPROM/ROM technology
- Fully static design
- Wide operating voltage range: 2.5V to 6.0V
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption: <2mA @5V, 4MHz, 15 μA typical @ 3V, 32 kHz, <1 μA typical standby current

# PIC Microcontroller product family:

## 8-bit microcontrollers

- ✓ PIC10
- ✓ PIC12
- ✓ PIC14
- ✓ PIC16
- ✓ PIC17
- ✓ PIC18

## 16-bit microcontrollers

- ✓ PIC24F
- ✓ PIC24H

## 32-bit microcontrollers

- ✓ PIC32

## 16-bit digital signal controllers

- ✓ dsPIC30
- ✓ dsPIC33F

- The F in a name generally indicates the PIC microcontroller uses flash memory and can be erased electronically.
- The C generally means it can only be erased by exposing the die to ultraviolet light (which is only possible if a windowed package style is used). An exception to this rule is the PIC16C84 which uses EEPROM and is therefore electrically erasable.

Some PIC16CXXX and PIC16FXXX family members

| Microcontroller | Program memory | Data RAM | Max speed (MHz) | I/O ports | A/D converter |
|---|---|---|---|---|---|
| PIC17C42 | 2048 × 16 | 232 | 33 | 33 | – |
| 16C554 | 512 × 14 | 80 | 20 | 13 | – |
| 16C64 | 2048 × 14 | 128 | 20 | 33 | – |
| 16C71 | 1024 × 14 | 68 | 20 | 13 | 4 |
| 16F877 | 8192 × 14 | 368 | 20 | 33 | 8 |
| 16F84 | 1024 × 14 | 68 | 10 | 13 | – |

**Why PIC16F877A is very popular?**

This is because PIC16F877A is very cheap. Apart from that it is also very easy to be assembled. An additional component that you need to make this IC work is just a 5V power supply adapter, a 20MHz crystal oscillator and 2 units of 22pF capacitors.

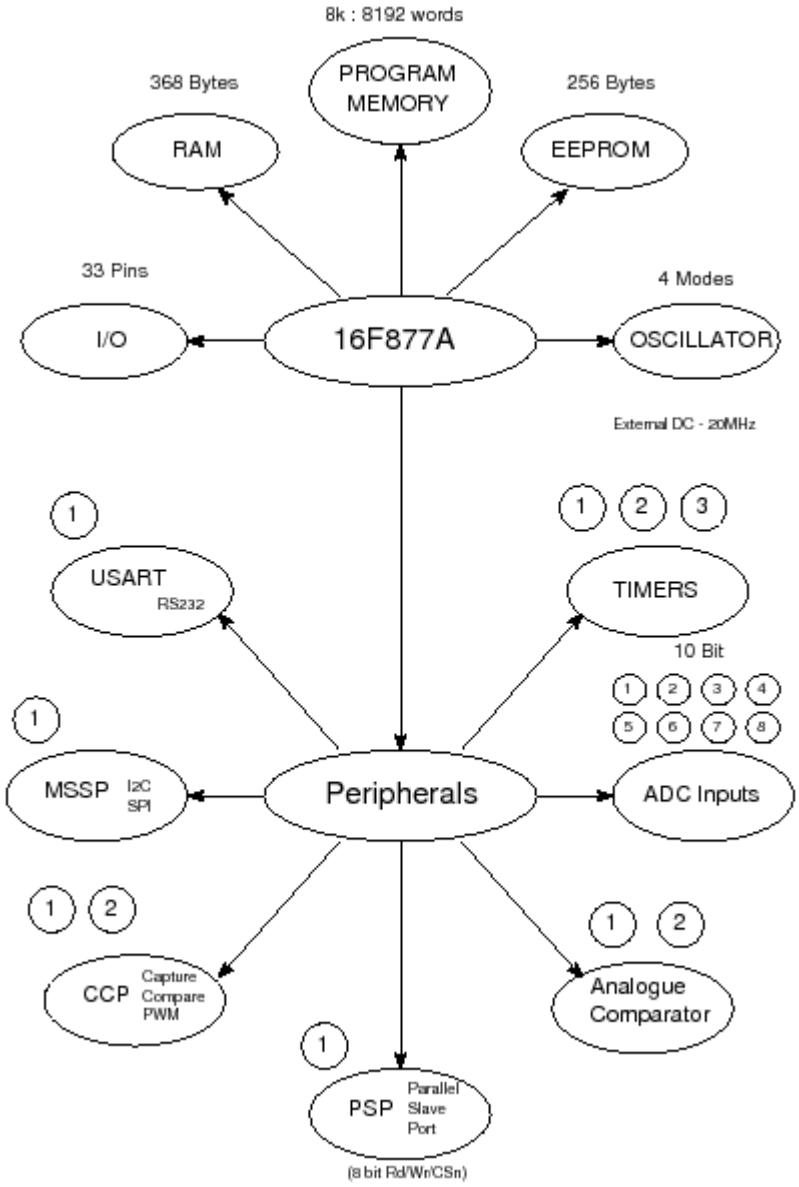
**What is the advantage of PIC16F877A?**

This IC can be reprogrammed and erased up to 10,000 times. Therefore it is very good for new product development phase.

**What is the disadvantage of PIC16F877A?**

•This IC has no internal oscillator so you will need an external crystal of other clock source.

# Bubble diagram of PIC16F877



# PIC 16F877

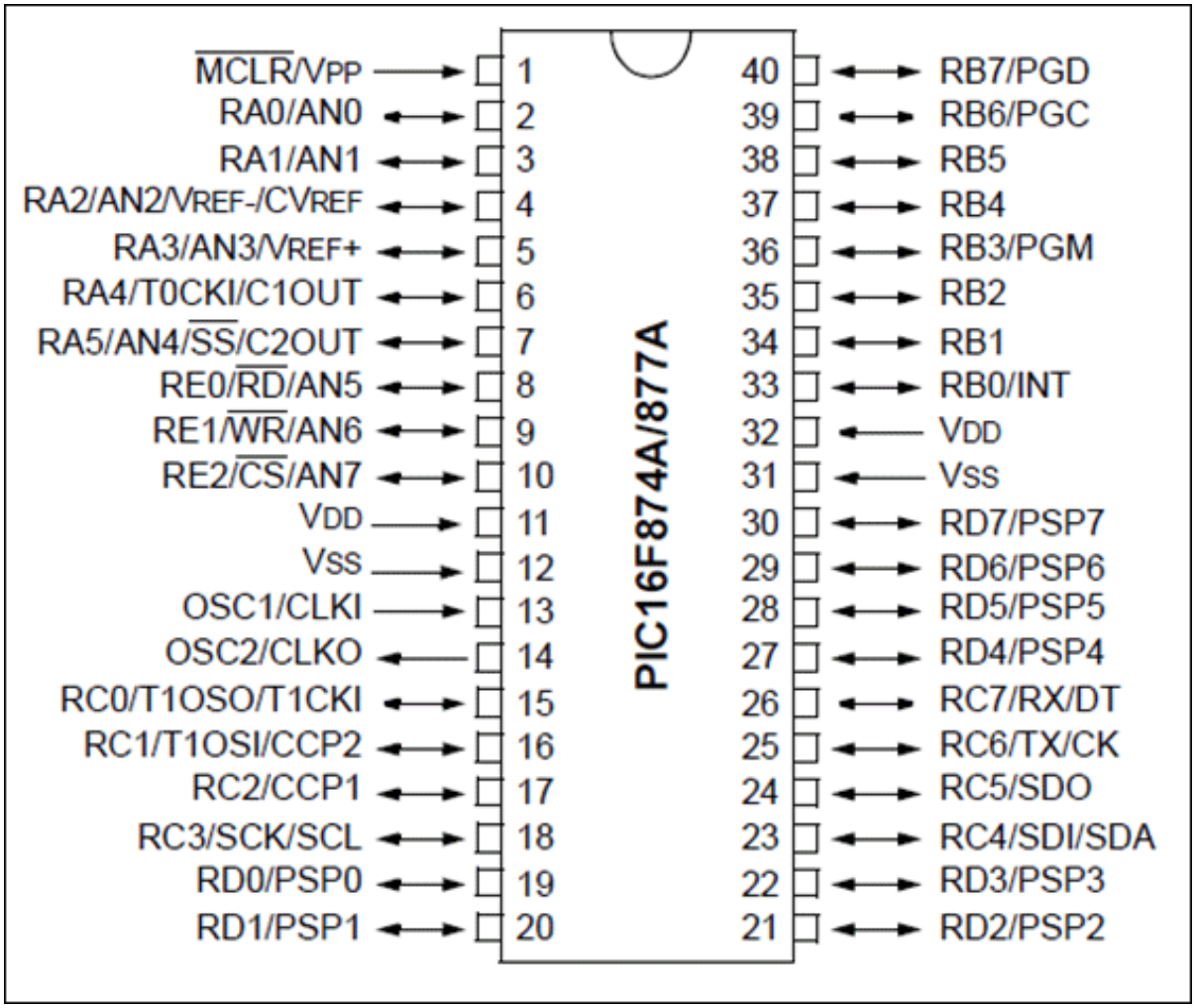
**Overview and Features**

- The PIC 16F8XX Microcontrollers are basically RISC microcontrollers with very small instruction set of only 35 instructions and a two-stage pipeline concept fetch and execution of instructions. As a result, all instructions execute in a single cycle except for program branches. .
- There are four devices in 16F8xx family, PIC16F873, PIC16F874, PIC16F876 and PIC16F877. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.
- PIC 16F877 is a 40-pin 8-Bit CMOS FLASH Microcontroller. The core architecture is high-performance RISC CPU.
- Since it follows the RISC architecture, all single cycle instructions take only one instruction cycle except for program branches which take two cycles.
- 16F877 comes with 3 operating speeds with 4, 8, or 20 MHz clock input. Since each instruction cycle takes four operating clock cycles, each instruction takes 0.2 μs when 20MHz oscillator is used.
- It has two types of internal memories .One is program memory and the other is data memory. Program memory is provided by 8K words (or 8K*14 bits) of FLASH Memory, and data memory has two sources. One type of data memory is a 368-byte RAM (random access memory) and the other is 256-byte EEPROM (Electrically erasable programmable ROM).
- The core features include interrupt up to 14 sources, power saving SLEEP mode, a single 5V supply and In-Circuit Serial Programming (ICSP) capability.
- The sink/source current, which indicates a driving power from I/O port, is high with 25mA. Power consumption is less than 2 mA in 5V operating condition.

PICs have different on-board peripherals some common peripherals are:

- 3 Timers (0 & 2- 8bits, 1-16 bits)
- 2 Compare/Capture/PWM Modules
- Analog to Digital Converters (ADC) (8, 10 and 12bit, 50ksps)
- Serial communications: UART (RS-232C), SPI, $I^2C$, CAN
- Pulse Width Modulation (PWM) (10bit)
- Voltage Comparators
- Voltage Reference Modules
- MSSP – Master Synchronous Serial Port
- I2C (Master and Slave)
- SPI (Master and Slave)
- Watchdog timers, Brown out detect, LCD drivers

**PIN DIAGRAM OF PIC16F877:**



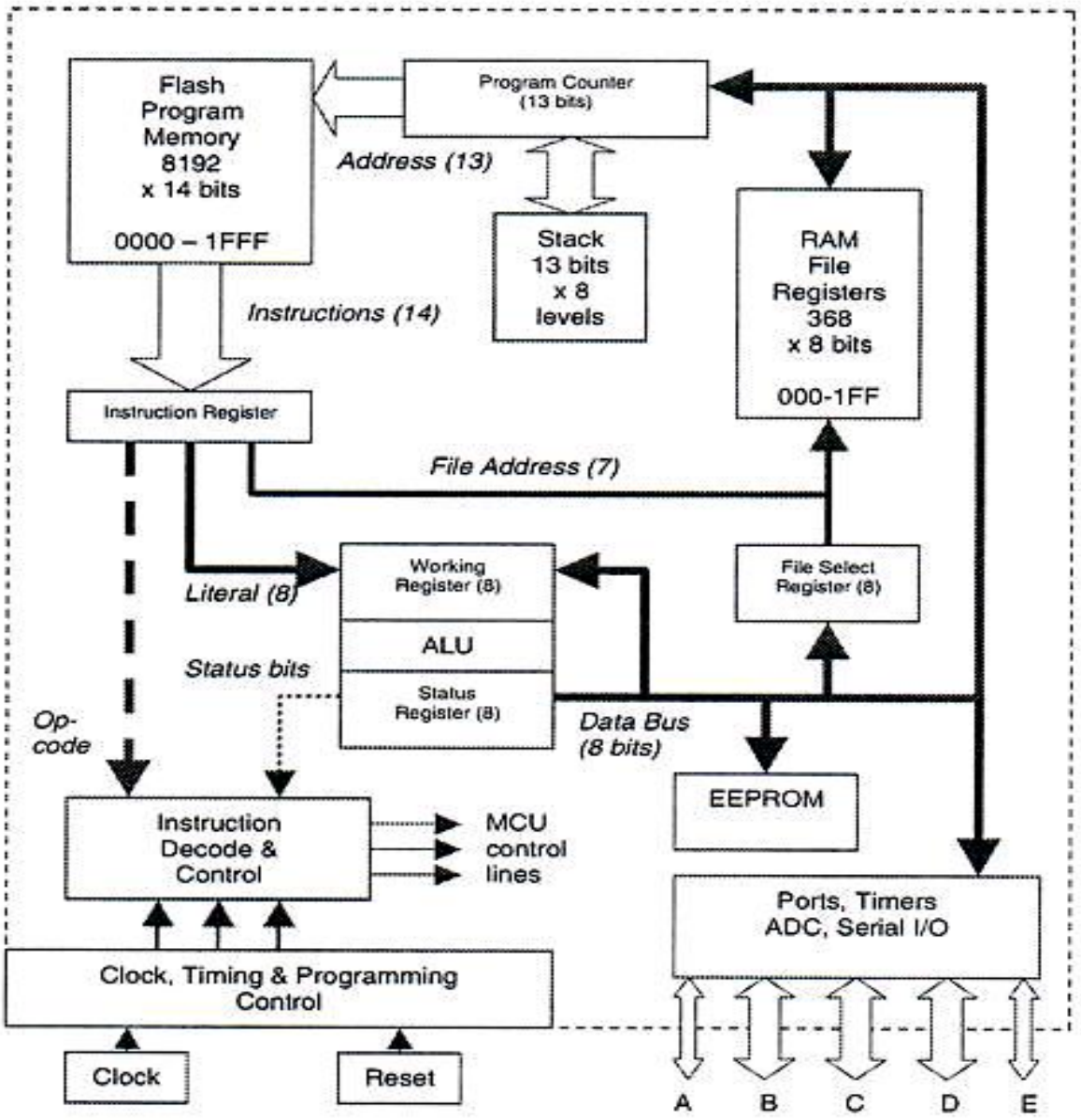

| Pin Number | Pin Name | Description |
|---|---|---|
| 1 | MCLR/Vpp | MCLR is used during programming, mostly **connected to programmer** like PIC Kit |
| 2 | RA0/AN0 | **Analog pin 0** or 0th pin of PORTA |
| 3 | RA1/AN1 | **Analog pin 1** or 1st pin of PORTA |
| 4 | RA2/AN2/Vref- | **Analog pin 2** or 2nd pin of PORTA |
| 5 | RA3/AN3/Vref+ | **Analog pin 3** or 3rd pin of PORTA |
| 6 | RA4/T0CKI/C1out | 4th pin of PORTA |
| 7 | RA5/AN4/SS/C2out | **Analog pin 4** or 5th pin of PORTA |
| 8 | RE0/RD/AN5 | **Analog pin 5** or 0th pin of PORTE |
| 9 | RE1/WR/AN6 | **Analog pin 6** or 1st pin of PORTE |
| 10 | RE2/CS/AN7 | 7th pin of PORTE |

| | | |
|---|---|---|
| 11 | Vdd | **Ground pin** of MCU |
| 12 | Vss | **Positive pin** of MCU (+5V) |
| 13 | OSC1/CLKI | **External Oscillator**/clock input pin |
| 14 | OSC2/CLKO | **External Oscillator**/clock output pin |
| 15 | RC0/T1OSO/T1CKI | $0^{th}$ pin of PORT C |
| 16 | RC1/T1OSI/CCP2 | $1^{st}$ pin of POCTC or **Timer/PWM** pin |
| 17 | RC2/CCP1 | $2^{nd}$ pin of POCTC or **Timer/PWM** pin |
| 18 | RC3/SCK/SCL | $3^{rd}$ pin of POCTC |
| 19 | RD0/PSP0 | $0^{th}$ pin of POCTD |
| 20 | RD1/PSPI | $1^{st}$ pin of POCTD |
| 21 | RD2/PSP2 | $2^{nd}$ pin of POCTD |
| 22 | RD3/PSP3 | $3^{rd}$ pin of POCTD |
| 23 | RC4/SDI/SDA | $4^{th}$ pin of POCTC or Serial Data in pin |
| 24 | RC5/SDO | $5^{th}$ pin of POCTC or Serial Data Out pin |
| 25 | RC6/Tx/CK | $6^{th}$ pin of POCTC or **Transmitter pin** of Microcontroller |
| 26 | RC7/Rx/DT | $7^{th}$ pin of POCTC or **Receiver pin** of Microcontroller |
| 27 | RD4/PSP4 | $4^{th}$ pin of POCTD |
| 28 | RD5/PSP5 | $5^{th}$ pin of POCTD |
| 29 | RD6/PSP6 | $6^{th}$ pin of POCTD |
| 30 | RD7/PSP7 | $7^{th}$ pin of POCTD |
| 31 | Vss | **Positive pin** of MCU (+5V) |
| 32 | Vdd | **Ground pin** of MCU |
| 33 | RB0/INT | $0^{th}$ pin of POCTB or **External Interrupt** pin |
| 34 | RB1 | $1^{st}$ pin of POCTB |
| 35 | RB2 | $2^{nd}$ pin of POCTB |
| 36 | RB3/PGM | $3^{rd}$ pin of POCTB or **connected to programmer** |
| 37 | RB4 | $4^{th}$ pin of POCTB |
| 38 | RB5 | $5^{th}$ pin of POCTB |
| 39 | RB6/PGC | $6^{th}$ pin of POCTB or **connected to programmer** |
| 40 | RB7/PGD | $7^{th}$ pin of POCTB or **connected to programmer** |

**ARCHITECTURE**

The PIC16FXX is a family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC microcontrollers employ an advanced RISC architecture. The PIC16FXX microcontroller family has enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The two-stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available. Also, a large register set helps to achieve a very high performance.



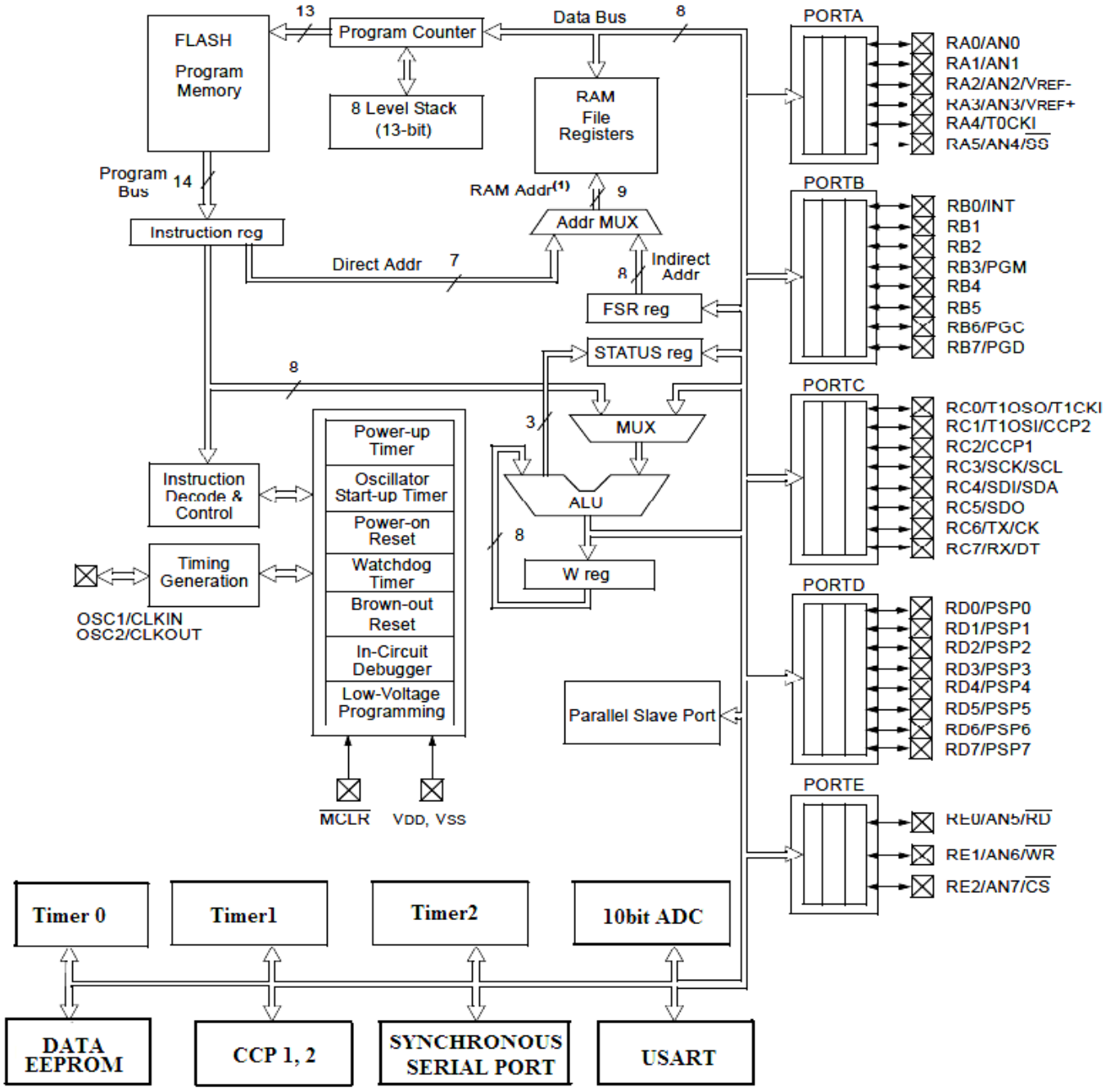



**Fig.: Block diagram of PIC 16F87X Microcontroller**

The PIC 16FXX uses Harvard architecture, in which; program and data are accessed from separate memories using separate buses. This improves bandwidth over traditional Von Neumann architecture where program and data may be fetched from the same memory using the same bus. Separating program and data buses further allows instructions to be sized differently than 8-bit wide data words. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle (200 ns@ 20MHz) except for program branches.

The PIC 16F87X devices have a 13-bit program counter capable of addressing an 8KX14 program memory space. The PIC 16FF876/877 devices have 8Kx 14 words of Flash program memory .The RESET vector is at 0000h and the Interrupt vector is at 0004h.

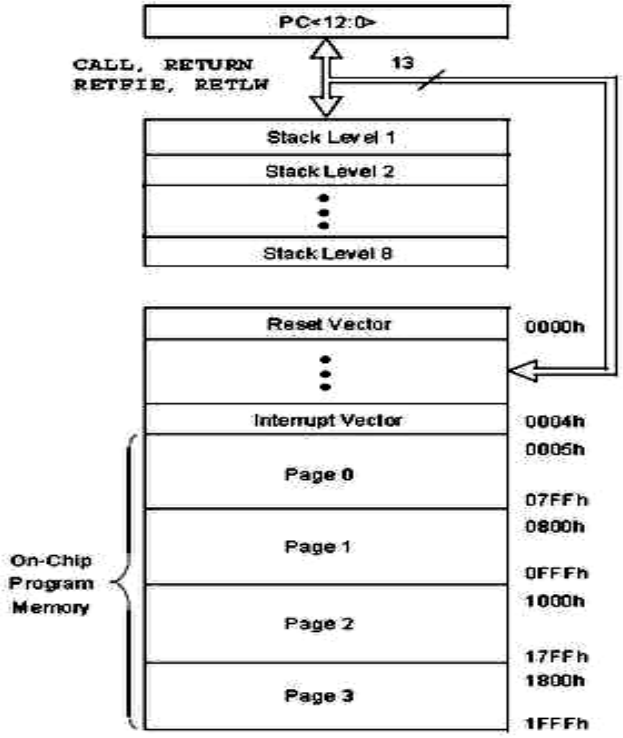
## Memory organization:

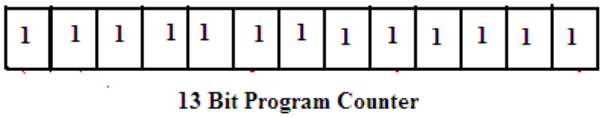
The memory module of the Controller has three memory blocks.

a) Program memory
b) Data memory
c) Stack

**a) Program Memory:**

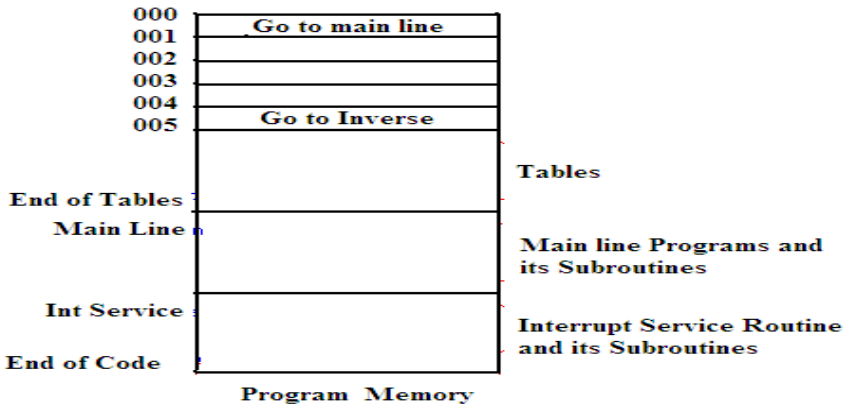
The PIC 16F877 has 4k x14 program memory space (0000H-0FFFH).It has a 13 bit Program counter (PC) to access any address ($2^{13}$=4k). This PIC family uses 13-bit program counter allowing the controllers to an 8k-program memory without changing the CPU structure.



13 Bit Program Counter

Two addresses in the program memory address space are treated in a special way by the CPU. The first address 000H being a go to mainline instruction the second special address, 004H being a 'go to in service' instruction can be assigned to this address to make the CPU to jump to the beginning of the Interrupt Service routine located elsewhere in the memory space.




When we deal with tables, they are assigned to addresses in the range H'005 – H'0FF' because for most of the applications this space is sufficient. The main line program begins after the tables.

## Data Memory:

The data memory of PIC 16F877 is partitioned into multiple banks which contain the general purpose registers and the Special function Registers. (SFRs).The bits RP1 and RP0 bits of the status register are used to select these banks. Each bank extends up to 7FH (128 Bytes).The lower bytes of the each bank are reserved for the Special Function Registers. Above the SFRs are general purpose registers implemented as static RAM.

**Register File Structure:**

In PIC Microcontrollers the Register File consists of two parts namely

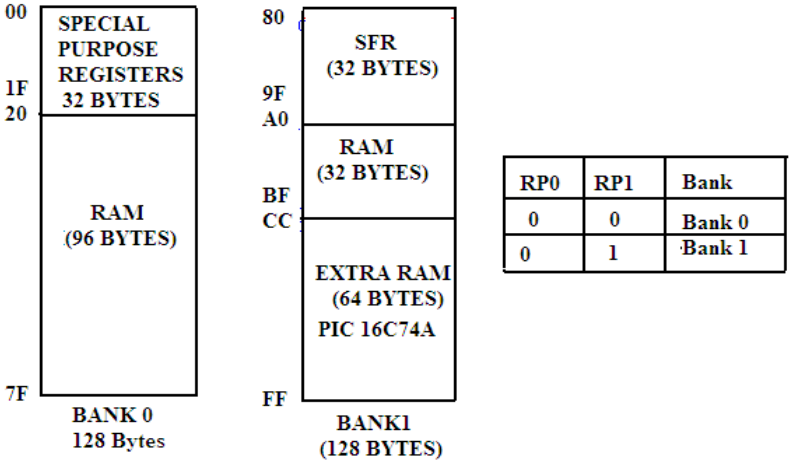
a) General Purpose Register File
b) Special Purpose Register File

**a) General Purpose Register File:**

The general purpose register file is another name for the microcontroller's RAM. Data can be written to each 8-bit location updated and retrieved any number of times.

**b) Special Purpose Register File:**

The special function register file consists of input, output ports and control registers used to configure each 8-bit port either as input or output. It contains registers that provide the data input and data output to a chip resources like Timers, Serial Ports and Analog to Digital converter and also the registers that contains control bits for selecting the mode of operation and also enabling or disabling its operation.




| | File Address | | File Address | | File Address | | File Address |
|---|---|---|---|---|---|---|---|
| Indirect addr.(*) | 00h | Indirect addr.(*) | 80h | Indirect addr.(*) | 100h | Indirect addr.(*) | 180h |
| TMR0 | 01h | OPTION_REG | 81h | TMR0 | 101h | OPTION_REG | 181h |
| PCL | 02h | PCL | 82h | PCL | 102h | PCL | 182h |
| STATUS | 03h | STATUS | 83h | STATUS | 103h | STATUS | 183h |
| FSR | 04h | FSR | 84h | FSR | 104h | FSR | 184h |
| PORTA | 05h | TRISA | 85h | | 105h | | 185h |
| PORTB | 06h | TRISB | 86h | PORTB | 106h | TRISB | 186h |
| PORTC | 07h | TRISC | 87h | | 107h | | 187h |
| PORTD(1) | 08h | TRISD(1) | 88h | | 108h | | 188h |
| PORTE(1) | 09h | TRISE(1) | 89h | | 109h | | 189h |
| PCLATH | 0Ah | PCLATH | 8Ah | PCLATH | 10Ah | PCLATH | 18Ah |
| INTCON | 0Bh | INTCON | 8Bh | INTCON | 10Bh | INTCON | 18Bh |
| PIR1 | 0Ch | PIE1 | 8Ch | EEDATA | 10Ch | EECON1 | 18Ch |
| PIR2 | 0Dh | PIE2 | 8Dh | EEADR | 10Dh | EECON2 | 18Dh |
| TMR1L | 0Eh | PCON | 8Eh | EEDATH | 10Eh | Reserved(2) | 18Eh |
| TMR1H | 0Fh | | 8Fh | EEADRH | 10Fh | Reserved(2) | 18Fh |
| T1CON | 10h | | 90h | | 110h | | 190h |
| TMR2 | 11h | SSPCON2 | 91h | | 111h | | 191h |
| T2CON | 12h | PR2 | 92h | | 112h | | 192h |
| SSPBUF | 13h | SSPADD | 93h | | 113h | | 193h |
| SSPCON | 14h | SSPSTAT | 94h | | 114h | | 194h |
| CCPR1L | 15h | | 95h | | 115h | | 195h |
| CCPR1H | 16h | | 96h | | 116h | | 196h |
| CCP1CON | 17h | | 97h | General Purpose Register 16 Bytes | 117h | General Purpose Register 16 Bytes | 197h |
| RCSTA | 18h | TXSTA | 98h | | 118h | | 198h |
| TXREG | 19h | SPBRG | 99h | | 119h | | 199h |
| RCREG | 1Ah | | 9Ah | | 11Ah | | 19Ah |
| CCPR2L | 1Bh | | 9Bh | | 11Bh | | 19Bh |
| CCPR2H | 1Ch | CMCON | 9Ch | | 11Ch | | 19Ch |
| CCP2CON | 1Dh | CVRCON | 9Dh | | 11Dh | | 19Dh |
| ADRESH | 1Eh | ADRESL | 9Eh | | 11Eh | | 19Eh |
| ADCON0 | 1Fh | ADCON1 | 9Fh | | 11Fh | | 19Fh |
| | 20h | | A0h | | 120h | | 1A0h |
| General Purpose Register 96 Bytes | | General Purpose Register 80 Bytes | | General Purpose Register 80 Bytes | | General Purpose Register 80 Bytes | |
| | 7Fh | | EFh | | 16Fh | | 1EFh |
| | | accesses 70h-7Fh | F0h | accesses 70h-7Fh | 170h | accesses 70h - 7Fh | 1F0h |
| Bank 0 | | Bank 1 | FFh | Bank 2 | 17Fh | Bank 3 | 1FFh |

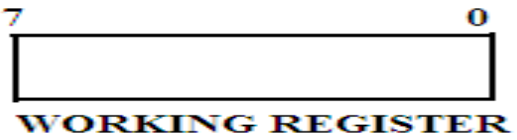
**CPU REGISTERS:**

The CPU registers are used in the execution of the instruction of the PIC microcontroller. The PIC **PIC16F877** Microcontroller has the following registers.

1. Working Register-W (Similar to Accumulator)
2. Status Register
3. FSR – File Select Register (Indirect Data memory address pointer)
4. INDF
5. Program Counter

**1. Working Register:**

Working Register is used by many instructions as the source of an operand. It also serves as the destination for the result of instruction execution and it is similar to accumulator in other μcs and μps.




**2. Status register:**

In most cases, this register is used to switch between the banks (Register Bank Select), but also has other capabilities.

**STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)**

| R/W-0 | R/W-0 | R/W-0 | R-1 | R-1 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| IRP | RP1 | RP0 | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |
| bit 7 | | | | | | | bit 0 |

With the help of three left bits (IRP, RP1, and RP0) one can control the transition between the banks:

- IRP - Register Bank Select bit, used for indirect addressing method.
- RP1:RP0: - Register Bank Select bits, used for direct addressing method.

To distinguish between the two methods, at this point, the will use the definition of fundamental concepts. Later on, the two methods will be studied in detail.

When the IRP Equal to 0, the program will work with banks 0, 1.

When the IRP Equal to 1, the program will work with banks 2, 3.

The following table demonstrates, which of the Banks the program is working with, based on the selection of the RP0 and RP1 bits:

| RP1:RP0 | BANK |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 2 |
| 11 | 3 |

An example of using STATUS register and Register Bank Select bit:

```
1.  bsf STATUS, 5       ; Change to Bank 1
2.  clrf TRISB          ; Set PORTB as output
3.  bcf STATUS, 5       ; Change to Bank
```

In the first line, we are in changing/setting the 5th bit, RP0, in the STATUS register to 1, and thus, base on the table we are switching/selecting Bank 1. After PortB was set as output in the second line, we switched back to Bank 0 by in changing/setting the 5th bit, RP0, in the STATUS register to 0, in the third line.

**C: Carry/borrow bit (ADDWF, ADDLW,SUBLW,SUBWF instructions)**

1 = A carry-out from the Most Significant bit of the result occurred

0 = No carry-out from the Most Significant bit of the result occurred

An example of using STATUS register and Carry/borrow bit:

1. Movlw 200
2. Addwf 100, 0

In this example, we are assigning value of 200 to the W (working) register. Then, we are adding the value of 100 and the W register together. The result is stored in W register and should be 300 (200+100). However, the maximum value is 256, resulting in carry out. The C (bit 0) of the STATUS register becomes 1 (C = 1). Register W will contain the reminder: 44.

**DC: Digit carry/borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)** (for borrow, the polarity is reversed)

1 = A carry-out from the 4th low order bit of the result occurred

0 = No carry-out from the 4th low order bit of the result

**Z: Zero bit**

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

The bits 3 and 4 are used with WDT - Watchdog Timer.

**PD: Power-down bit**

1 = After power-up or by the CLRWDT instruction

0 = By execution of the SLEEP instruction

**TO: Time-out bit**

1 = After power-up, CLRWDT instruction or SLEEP instruction

0 = A WDT time-out occurred

**3. FSR – (File Select Register):**

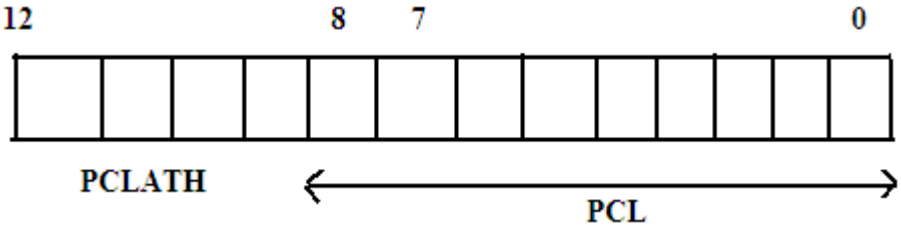
It is the pointer used for indirect addressing. In the indirect addressing mode the 8-bit register file address is first written into FSR. It is a special purpose register that serves as an address pointer to any address through out the entire register file.

**4. INDF – (Indirect File):**

It is not a physical register addressing but this INDF will cause indirect addressing. Any instruction using the INDF registers actually access the register pointed to by the FSR.

**5. PROGRAM COUNTER**

PIC PIC16F877A has a 13 bit program counter in which PCL is the lower 8-bits of the PC and PCLATH is the write buffer for the upper 5 bits of the PC.




PCLATH (program counter Latch can be read or from or written to without affecting the Program Counter (PC).The upper 3 bits of PCLATH remain zero. It is only when PCL is written to that PCLATH is automatically written into the PC at the same time.

**PARALLEL I/O Ports**

## PORT register:

The role of the PORT register is to receive the information from an external source (e.g. sensor) or to send information to the external elements (e.g. LCD). The 28-pin devices have 3 I/O ports, while the 40/44-pin devices, like PIC16F877, have 5 I/O ports located in the BANK 0.

**PORTA** is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA.
Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input. Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output.

**PORTB** is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB.

Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input. Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output.

**PORTC** is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC.

Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input. Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output.

**PORTD** is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**PORTE** has three pins (RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

## Timer modules in PIC 16F877:

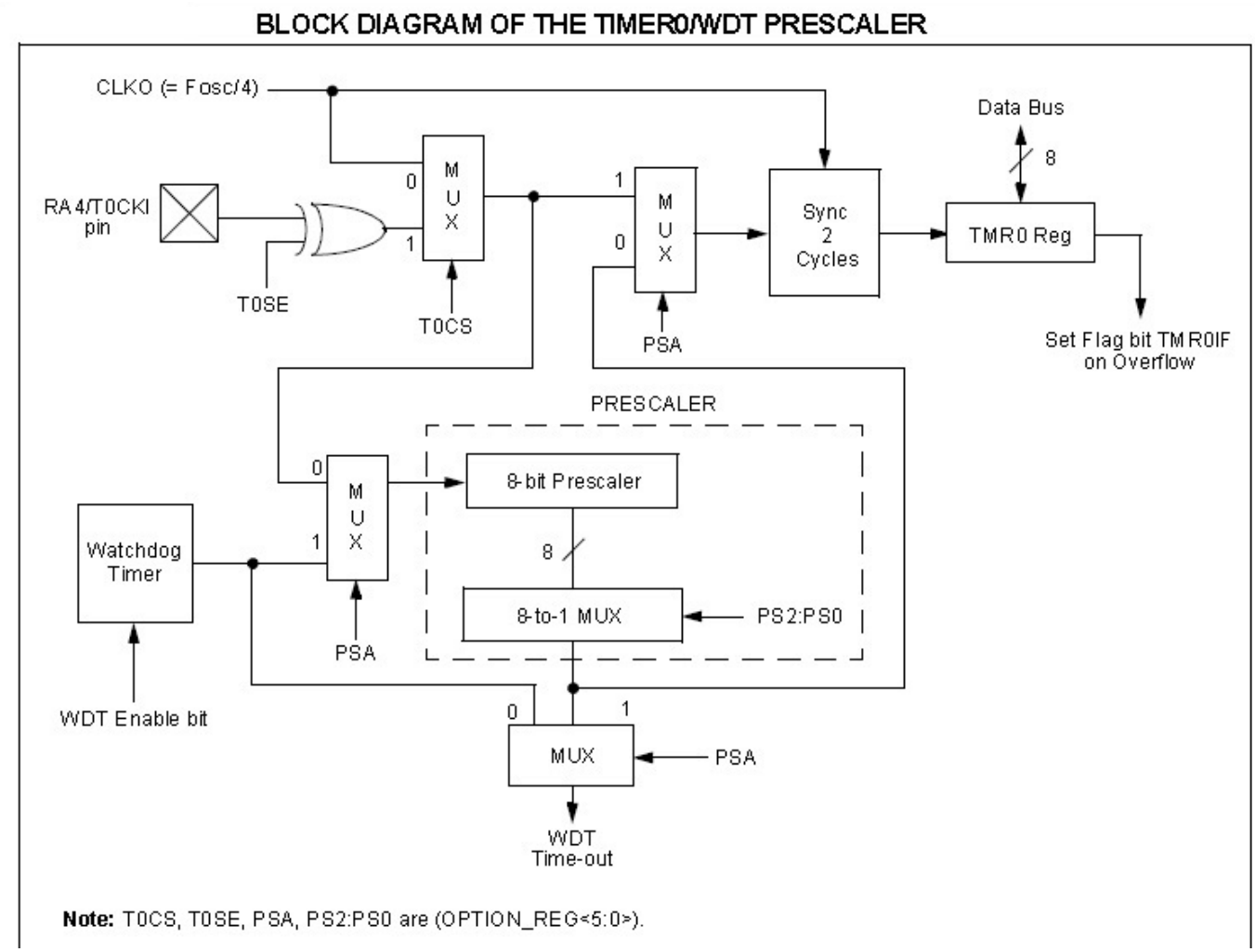
The PIC 16F877 basically has three timer modules. These timer module terminals are also multiplexed with other functions for handling alternate functions. These timer modules are usually denoted by the symbols TIMER-0, TIMER-1, and TIMER-2.These modules help to perform various timing and counting functions inside the chip.

### TIMER-0 module:

The main timing/counting features of Timer-0 module are given below.

- ✓ Timer-0 module has built in 8 bit timer/counter
- ✓ It can be easily readable/writable
- ✓ Built in 8 bit software programmable pre-scalar functions
- ✓ Easily select internal/external clock pulses
- ✓ Interrupt with overflow from the value FFh to 00h
- ✓ Edge selection for external clock pulse

The block diagram of timer-0 module is given in the figure below. The timer mode is normally selected by clearing the T0CS bit in the register. In Timer mode, when the Timer 0 Module increases with every instruction cycle, the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register. Counter mode is selected by setting bit T0CS in Counter mode. Timer 0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer 0 Source Edge Select bit, T0SE. Clearing bit T0SE selects the rising edge. The pre-scalar is mutually exclusively shared between the Timer0 module and the Watchdog Timer.




## TIMER-1 module:

Timer 1 module is a 16 bit timer/counter unit. That is, it consists of two 8 bit (8+8) registers (TMR1H, TMR1L) which read and write easily. TMR1 register is a pair of TMR1H and TMR1L and also its value increment its value from 0000h to FFFFh and rolls over to 0000h.

Timer 1 module basically operates in two different modes. They are

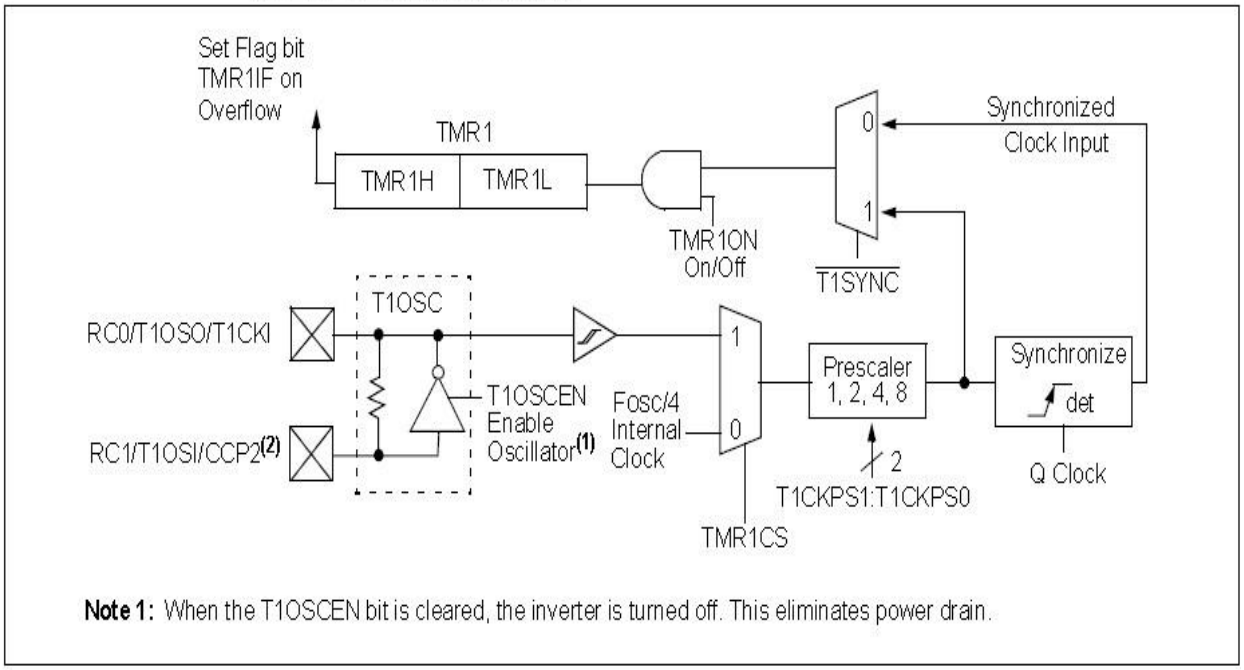
1) Timer mode

2) Counter mode

The operating mode of timer 1 module is selected by using the clock select bit (TMR1CS), in timer mode. The timer 1 increases on every instruction cycle. But in counter mode, it increases on every rising edge of the external clock input. Timer 1 pin can be enabled/disabled easily by setting/clearing the control bit (TMR1ON). This timer1 pin also has an internal reset input function. It can be generated by either of the two CCP modules.

The block diagram of timer1 module I given in the image below.

**TIMER1 BLOCK DIAGRAM**




**Note 1:** When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

**T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| bit 7 | | | | | | | bit 0 |

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

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Control bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

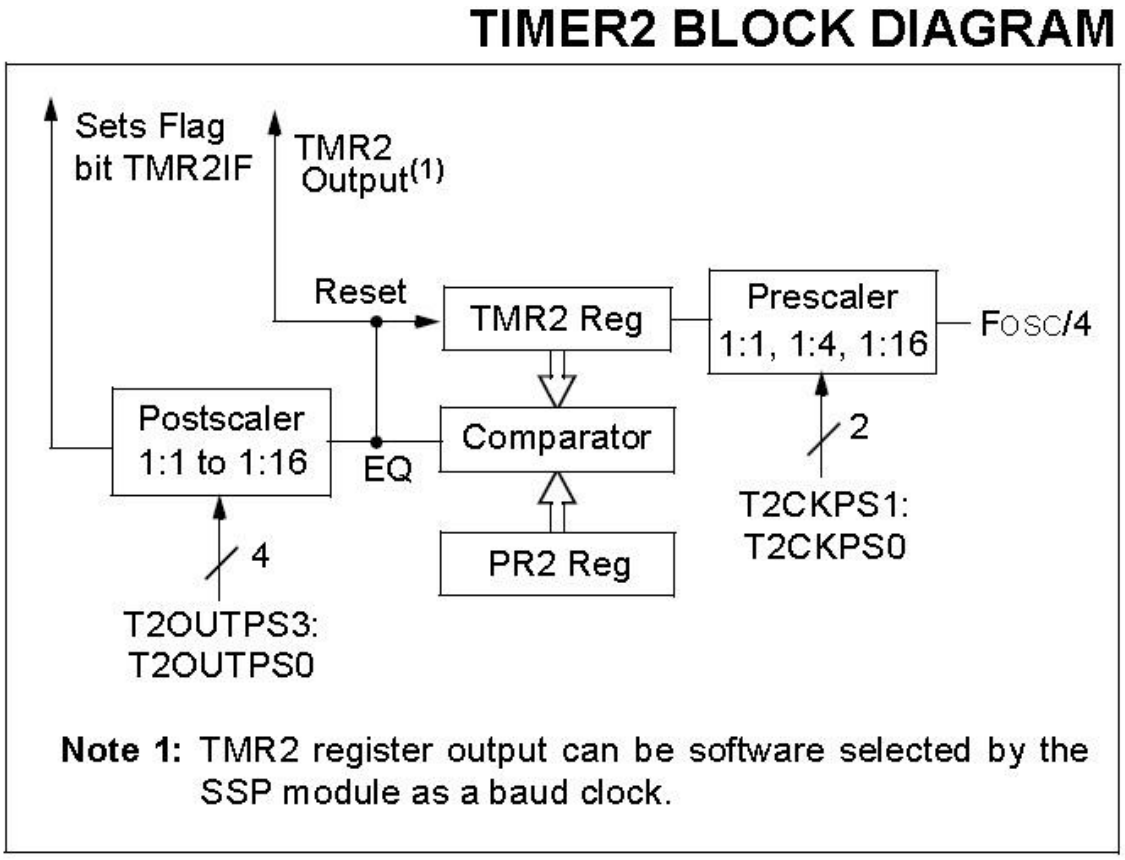
bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

| 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 |

## TIMER 2 Module:

Timer 2 is an 8-bit timer with a pre-scalar and a post-scalar. It can be used as the PWM (pulse width modulation) time base for the PWM mode of the CCP module(s). The block diagram of timer 2 module is given in the figure below.




The TMR2 register is readable and writable and is cleared on any device Reset. The input clock (FOSC/4) has a pre-scale option of 1:1, 1:4 or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>). The Timer 2 module has an 8-bit period register, PR2. The value of Timer 2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon Reset. The match output of TMR2 goes through a 4-bit post-scalar (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit, TMR2IF (PIR1<1>)). Timer 2 can be shut-off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

## CAPTURE/COMPARE/PWM (CCP) Modules in PIC 16F877

Capture-Compare-Pulse-Width-Module (CCP) is a special module designs for modulation and waveform generation applications. This module basically works on three different modes (capture/compare and PWM odes). The PIC 16F877 chip contains two CCP ports (CCP1 and CCP2). Each of this CCP module contains 16 bit registers which works as

- ✓ 16-bit Capture register
- ✓ 16-bit Compare register
- ✓ PWM Master/Slave Duty Cycle registers

The CCP1 and CCP2 modules are identical in its operation except in its special event trigger operation. In each CCP modules, the capture, compare and PWM modes using different timer resources. The table below shows the different CCP modes and its timer resources. The detailed explanations and functions of CCP module is given below.

**INTERRUPTS:**

The PIC16F877 family has up to 11 sources of interrupt. The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

Global interrupt enable bit, GIE enables all un-masked interrupts or disables all interrupts. When bit GIE is enabled, and an interrupt flag bit and mask bit are set, the interrupt will vector immediately. Individual interrupts can be disabled through their corresponding enable bits in the INTCON register. GIE is cleared on reset.

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

The RBO/INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flag bits are contained in the INTCON register.

The peripheral interrupt flag bits are contained in special function registers PIR1 and PIR2. The corresponding interrupt enable bits are contained in special function registers PIE1 and PIE2 and the peripheral interrupt enable bit is contained in special function register INTCON.

When an interrupt is responded to, bit GIE is cleared to disable any further interrupts, the return address is pushed onto the stack and the PC is loaded with 0004h. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

For external interrupt events, such as the RB0/INT pin or RB port change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs. The latency is the same for one or two cycle instructions. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid infinite interrupt requests. Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

**INTCON**

| 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 | EEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF |
| bit7 | | | | | | | bit0 |

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **GIE**: Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

**Note:** For the operation of the interrupt structure, please refer to Section 8.5.

bit 6: **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 5: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE**: RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT 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: **T0IF**: TMR0 overflow interrupt flag bit
1 = TMR0 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF**: RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = When 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

**INT INTERRUPT**:

External interrupt on RB0/INT pin is edge triggered: either rising if edge select bit INTEDG is set, or falling, if bit INTEDG is clear. When a valid edge appears on the RB0/INT pin, flag bit INTF is set. This interrupt can be disabled by clearing enable bit INTE. The INTF bit must be cleared in software in the interrupt service routine before re-enabling this interrupt. The INT interrupt can wake the processor from SLEEP, if enable bit INTE was set prior to going into SLEEP. The status of global enable bit GIE decides whether or not the processor branches to the interrupt vector following wake-up. See for details on SLEEP mode.

**TMR0 INTERRUPT**:

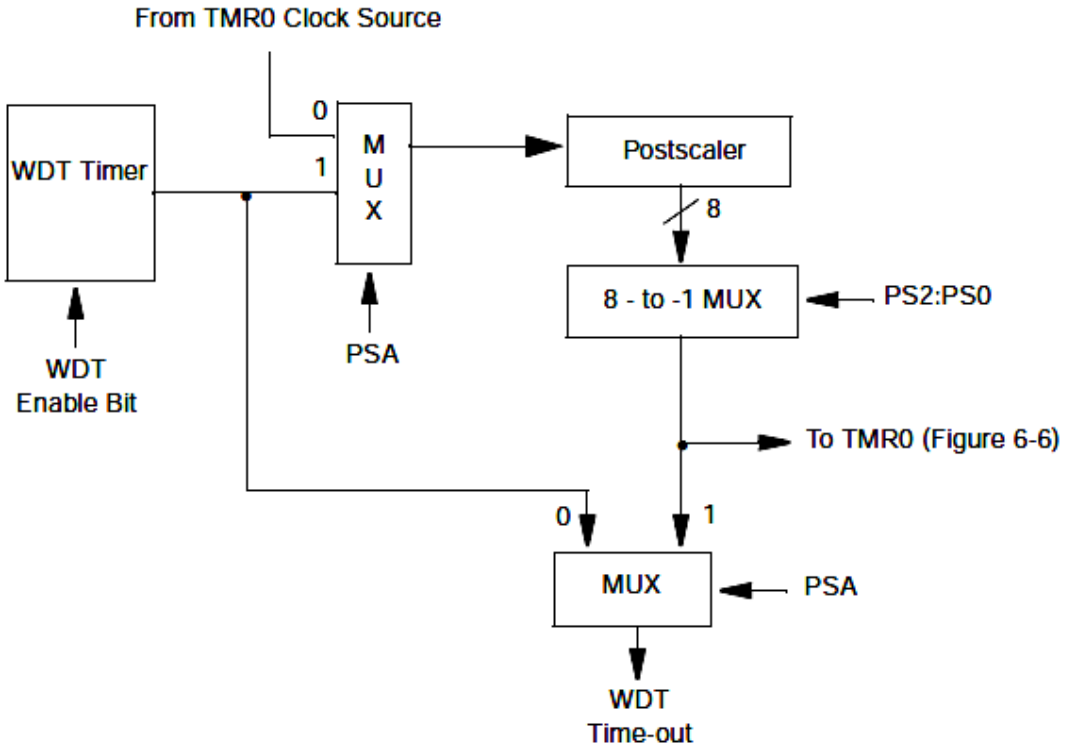
An overflow in the TMR0 register will set flag bit T0IF. The interrupt can be enabled/disabled by setting/clearing enable bit T0IE.

**PORTB INTERRUPT ON CHANGE**:

An input change on PORTB sets flag bit RBIF. The interrupt can be enabled/disabled by setting/clearing enable bit RBIE.

**WATCH DOG TIMER (WDT):**

The Watchdog Timer is a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run, even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT time-out generates a device reset. If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by clearing configuration bit WDTE.




**WDT PERIOD:**

The WDT has a nominal time-out period of 18 ms, (with no pre-scalar). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, a pre-scalar with a division ratio of up to can be assigned to the WDT under software control by writing to the OPTION register. Thus, time-out periods up to seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and the post-scalar, if assigned to the WDT, and prevent it from timing out and generating a device RESET condition.

The $\overline{TO}$ bit in the STATUS register will be cleared upon a WDT time-out.

WDT PROGRAMMING CONSIDERATIONS**:**

It should also be taken in account that under worst case conditions ($V_{DD}$ = Min., Temperature = Max., max WDT pre-scalar) it may take several seconds before a WDT time-out occurs.

**ADDRESSING MODES:**

The PIC microcontrollers support only TWO addressing modes .They are

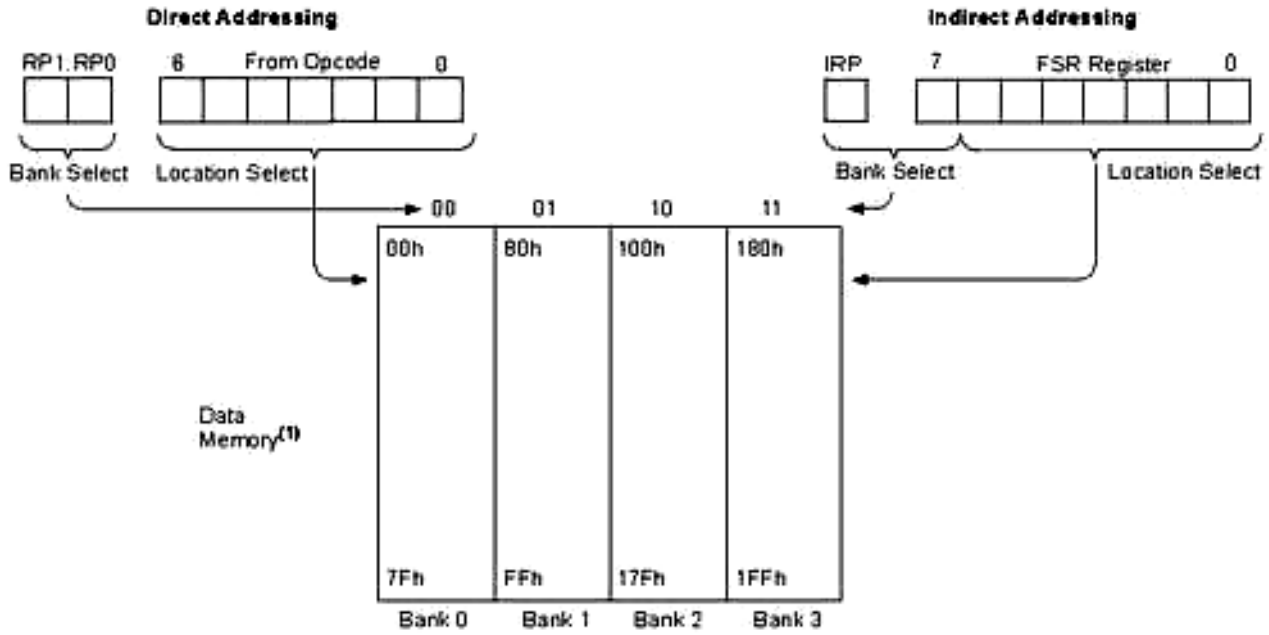
(i) Direct Addressing Mode

(ii) Indirect Addressing mode

**Direct Addressing:** Using this method we are accessing the registers directly by detecting location inside Data Memory from op-code and by selecting the bank using bits RP1 and RP0 of the STATUS register.

**Indirect Addressing:** To implement indirect addressing, a File Select Register (FSR) and indirect register (INDF) are used. In addition, when using this method we choose bank using bit IRP of the STATUS register. Indirect addressing treated like a stack pointer, allowing much more efficient work with a number of variables. INDF register is not an actual register (it is a virtual register that is not found in any bank).

Don't be confused! There is SFR (Special Function Register) - special registers of RAM, and there is FSR (File Select Register). The following figure shows the two addressing methods:



**INSTRUCTION SET:**

While writing the instructions, the following guidelines are followed.

a) Write the instructions mnemonics in lower case (example: x or wf)

b) Write special Register names, RAM variable names and bit names in upper case (example: STATUS, RPO….)

c) Write instruction and subroutine labels in mixed case (example: Mainline, Loop Time..)

**The instruction set of PIC is divided into three basic categories**. They are

(a) Byte oriented Instructions

(b) Bit oriented Instructions

**(c)** Literal and Control Instructions

**Byte Oriented Instructions**

In a byte oriented Instructions **f** represents a file register and d represents destination register. The destination specifies where the result of operation is to be placed. If d = 0 the result is placed in W register (Accumulator) and if d = 1 , the result is placed in the file register specified in the instruction.

```
ADDWF f, d          ; Add  W and  f
CLRF  f             ; Clear f
MOVWF f ,d          ; Move f
NOP                 ; No operation
SUBWF  f ,d         ; Subtract W from f
```

**Bit Oriented Instruction**

In bit oriented instructions, b represents a bit field designator which selects the number of the bit affected by the operation and f represents the number of the file in which the bit is located**.**

```
BCF f, b        ; Bit clear f
BSF   f, b      ; Bit set f
BTFSC   f, b    ; Bit test f, skip if set
```

**Literal and Control Instructions**

In literal and control instructions K represents an 8 or 11 bit constant or literal value.

```
ADDLW   k       ; Add literal and W
ANDLW   k       ; AND   literal with W
CALL k          ; Call subroutine
MOVLW k         ; Move literal to W
```

SUBLW k ; Subtract W from literal

Based on the type of operation PIC supports various Instructions. They are explained below.

**CLASSIFICATION OF INSTRUCTIONS:**

All the instructions of the PIC microcontroller are classified into nearly 9 groups. They are given below with examples.

**(i). Arithmetic Operations:**

ADDLW k ; Add literal value k to W

ADDWF f, d ; the contents of the W register are added with the register f.

SUBWF f, d ; the contents of W register are subtracted from register f

**(ii). Logical Instructions:**

ANDLW k ; the contents of W register are ANDED with the 8-bit literal k .The result is stored in the W register.

IORLW k ; Inclusive OR the literal value into W register

XORWF f, d ; the contents of W register are XORed with register f and the result is stored in W or f.

COMF f, d ; Complement f.

**(iii). Increment/Decrement Instructions**

INCF f, d ; Increment contents of f register by 1

DECF f, d ; Decrement f by 1

**(iv). Data Transfer instructions:**

MOVF f, d ; Move f to W i.e. The contents of register f is moved to a destination depending on d

MOVLW k ; Move literal k to W

MOVWF f ; Move W to f

**(v). Clear Instructions**

CLRF ; Clear file f

CLR W ; Clear the contents of W register and zero bit is set

CLRWDT ; Clear Watch dog timer

BCF ; Clear bit b of register f.

**(vi). Rotate Instructions**

RLF ; Rotate Left f through carry

RRF ; Rotate Right f through carry

**(vii). Branch Instructions:** There are two types of Branch instructions.

a) Conditional Branch and

b) Un conditional Branch instructions**.**

**(a) Conditional Branch Instructions**

BTFSC f, b ; Bit Test skip if clear

BTFSS f, b ; Bit test f , skip if set

If bit B in register f is zero, then the next instruction is executed, otherwise next instruction is Discarded and a NOP is executed.

DECFSZ f, d ; Decrement f, skip if zero.

INCFSZ f, d ; Increment f, skip if zero

**(b) Unconditional Instructions**

CALL k ; Call the subroutine k unconditionally

GOTO k ; Unconditional k branch

RETURN ; Return from subroutine

REETLW k ; Return with literal in W register.

**(viii). Miscellaneous**

BSF f, b ; Set bit b of register f

SLEEP ; Go into standby mode

NOP ; No operation i.e Do nothing, wait one clock cycle.

The various instructions used in PIC are presented in the Table below.

Legend

f - Any memory location (register);

W - Working register (accumulator);

b - Bit address within an 8-bit register;

d - Destination bit; [label] - Set of 8 characters indicating start of particular address in the program;

TOS - Top of stack; [] - Option; <> - bit field in register (several bit addresses);

C - Carry/Borrow bit of the STATUS register;

DC - Digit Carry bit of the STATUS register; and

Z - Zero bit of the STATUS register.

Ex: ADDLW - Add literal and W Syntax: [label] ADDLW k Description:

The content of the register W is added to the 8-bit literal k. The result is stored in the W register. Operation: (W) + k -> W Operand: $0 \leq k \leq 255$ Status affected: C, DC, Z Number of cycles:

| INSTRUCTION | DESCRIPTION | OPERATION | FLAG | CLK | * |
|---|---|---|---|---|---|
| **Data Transfer Instructions** | | | | | |
| MOVLW k | Move constant to W | k -> w | | 1 | |
| MOVWF f | Move W to f | W -> f | | 1 | |
| MOVF f,d | Move f to d | f -> d | Z | 1 | 1, 2 |
| CLRW | Clear W | 0 -> W | Z | 1 | |
| CLRF f | Clear f | 0 -> f | Z | 1 | 2 |
| SWAPF f,d | Swap nibbles in f | f(7:4),(3:0) -> f(3:0),(7:4) | | 1 | 1, 2 |
| **Arithmetic-logic Instructions** | | | | | |
| ADDLW k | Add W and constant | W+k -> W | C, DC, Z | 1 | |
| ADDWF f,d | Add W and f | W+f -> d | C, DC ,Z | 1 | 1, 2 |
| SUBLW k | Subtract W from constant | k-W -> W | C, DC, Z | 1 | |
| SUBWF f,d | Subtract W from f | f-W -> d | C, DC, Z | 1 | 1, 2 |
| ANDLW k | Logical AND with W with constant | W AND k -> W | Z | 1 | |
| ANDWF f,d | Logical AND with W with f | W AND f -> d | Z | 1 | 1, 2 |
| ANDWF f,d | Logical AND with W with f | W AND f -> d | Z | 1 | 1, 2 |
| IORLW k | Logical OR with W with constant | W OR k -> W | Z | 1 | |
| IORWF f,d | Logical OR with W with f | W OR f -> d | Z | 1 | 1, 2 |
| XORLW k | Logical exclusive OR with W with constant | W XOR k -> W | Z | 1 | 1, 2 |

| | | | | | |
|---|---|---|---|---|---|
| XORWF f,d | Logical exclusive OR with W with f | W XOR f -> d | Z | 1 | |
| INCF f,d | Increment f by 1 | f+1 -> f | Z | 1 | 1, 2 |
| DECF f,d | Decrement f by 1 | f-1 -> f | Z | 1 | 1, 2 |
| RLF f,d | Rotate left f through CARRY bit | | C | 1 | 1, 2 |
| RRF f,d | Rotate right f through CARRY bit | | C | 1 | 1, 2 |
| COMF f,d | Complement f | f -> d | Z | 1 | 1, 2 |
| **Bit-oriented Instructions** | | | | | |
| BCF f,b | Clear bit b in f | 0 -> f(b) | | 1 | 1,2 |
| BSF f,b | Set bit b in f | 1 -> f(b) | | 1 | 1,2 |
| **Program Control Instructions** | | | | | |
| BTFSC f,b | Test bit b of f. Skip the following instruction if clear. | Skip if f(b) = 0 | | 1 (2) | 3 |
| BTFSS f,b | Test bit b of f. Skip the following instruction if set. | Skip if f(b) = 1 | | 1 (2) | 3 |
| DECFSZ f,d | Decrement f. Skip the following instruction if clear. | f-1 -> d skip if Z = 1 | | 1 (2) | 1, 2, 3 |
| INCFSZ f,d | Increment f. Skip the following instruction if set. | f+1 -> d skip if Z = 0 | | 1 (2) | 1, 2, 3 |
| GOTO k | Go to address | k -> PC | | 2 | |
| CALL k | Call subroutine | PC -> TOS, k -> PC | | 2 | |
| RETURN | Return from subroutine | TOS -> PC | | 2 | |
| RETLW k | Return with constant in W | k -> W, TOS -> PC | | 2 | |

www.kvrsoftwares.com/

| | | | | | |
|---|---|---|---|---|---|
| RETFIE | Return from interrupt | TOS -> PC, 1 -> GIE | | 2 | |
| **Other instructions** | | | | | |
| NOP | No operation | TOS -> PC, 1 -> GIE | | 1 | |
| CLRWDT | Clear watchdog timer | 0 -> WDT, 1 -> TO, 1 -> PD | TO, PD | 1 | |
| SLEEP | Go into sleep mode | 0 -> WDT, 1 -> TO, 0 -> PD | TO, PD | 1 | |