PIC MICRO CONTROLLERS

INTRODUCTION

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 μc's 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.

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 is256-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.

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)

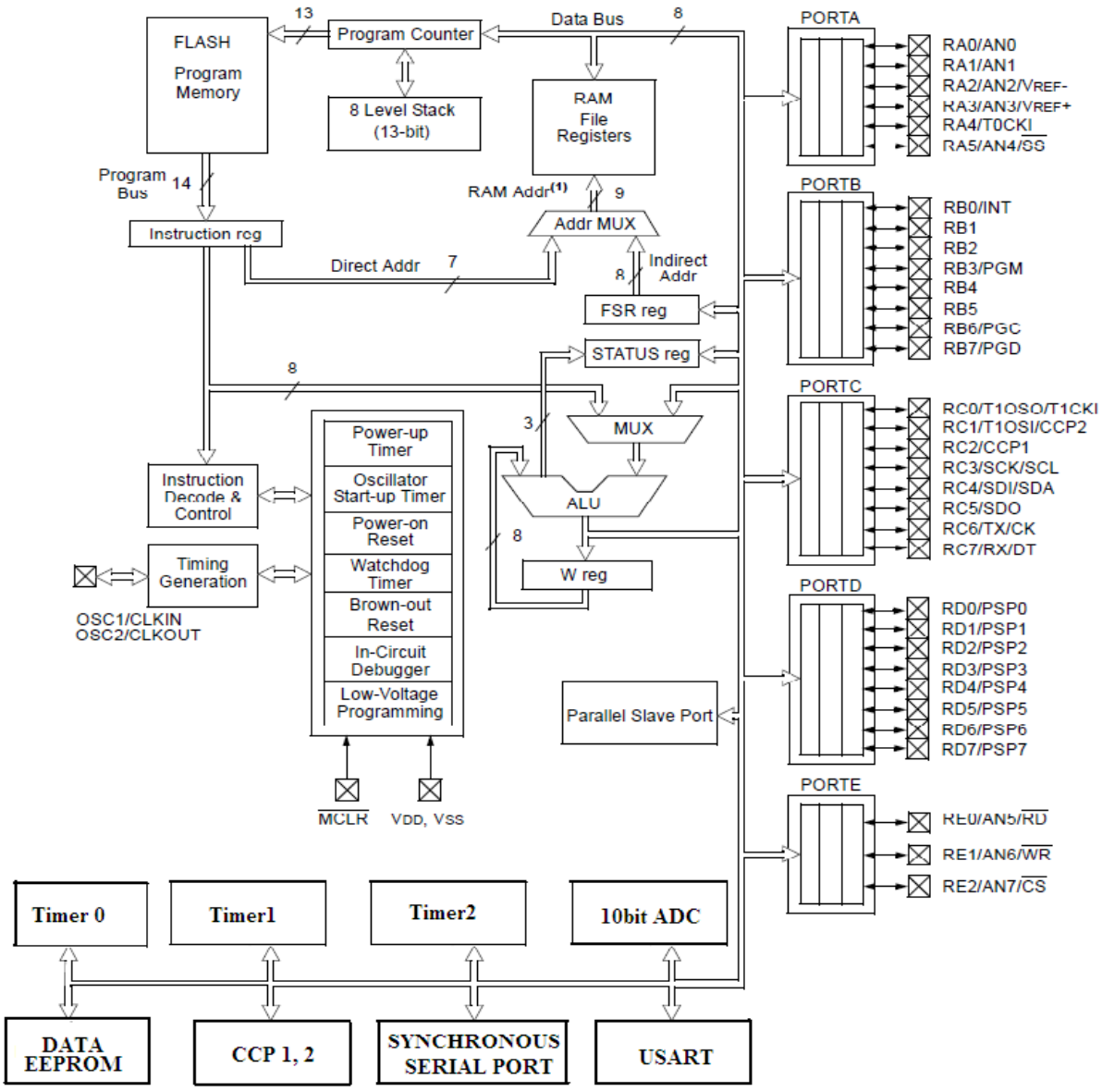
- Three Timers Timer0, Timer 1 and Timer 2.
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- 10-bit multi-channel Analog-to-Digital converter
- Selectable oscillator options
- One USART / SCI port with 9-bit address 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

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.

. 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.



Block diagram of PIC 16F87X Microcontroller

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 PICcontroller has three memory blocks.

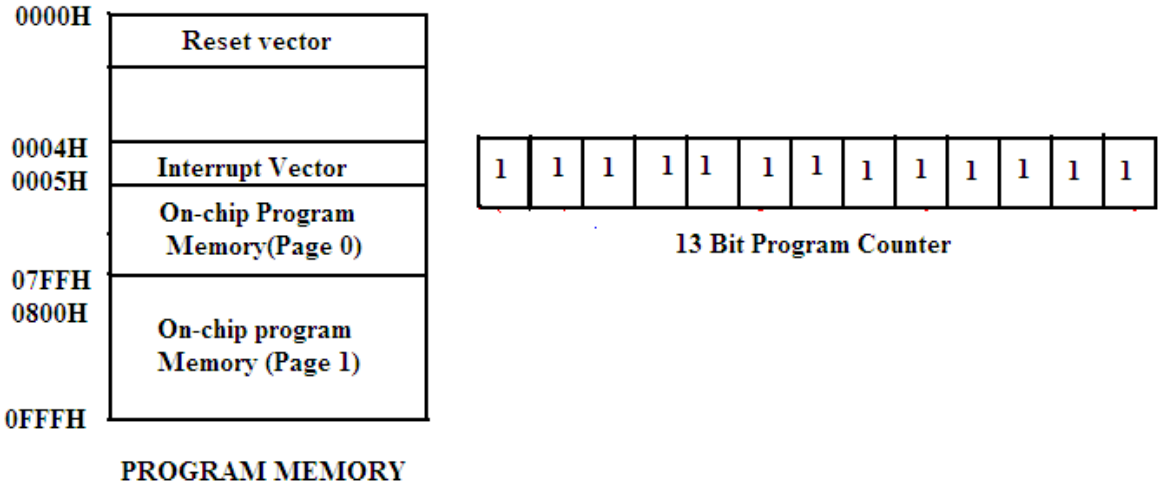
a) Program memory

b) Data memory and

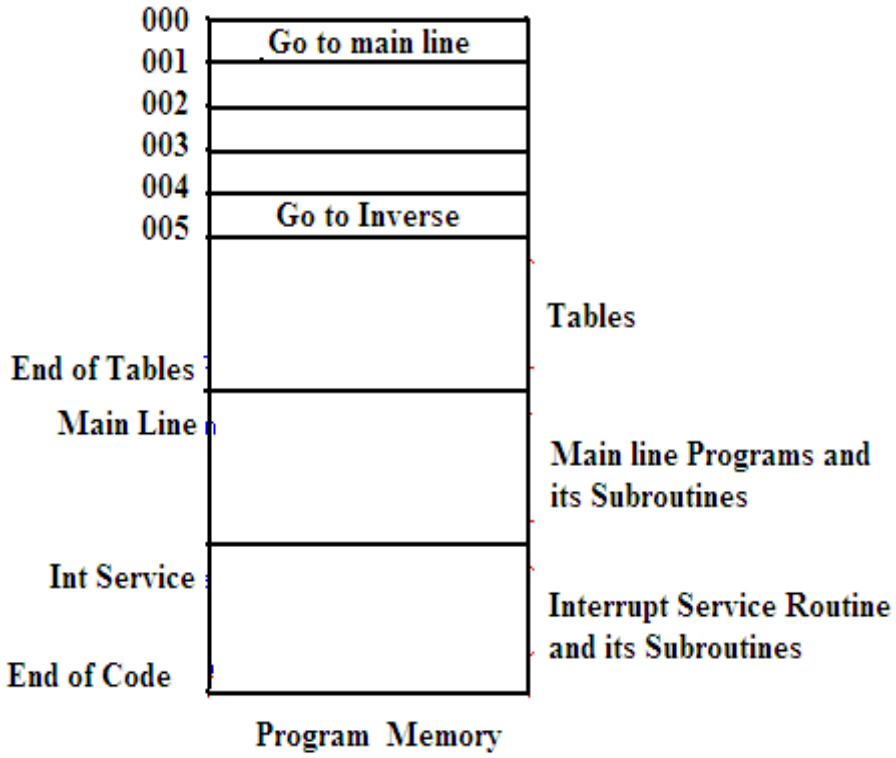
c) Stack

a) Program Memory:

The PIC 16F8XX 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.



Two addresses in the program memory address space are treated in a special way by the CPU. The first address H' 000' being a go to mainline instruction the second special address, H' 004' 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 16F8XX 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 upto 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

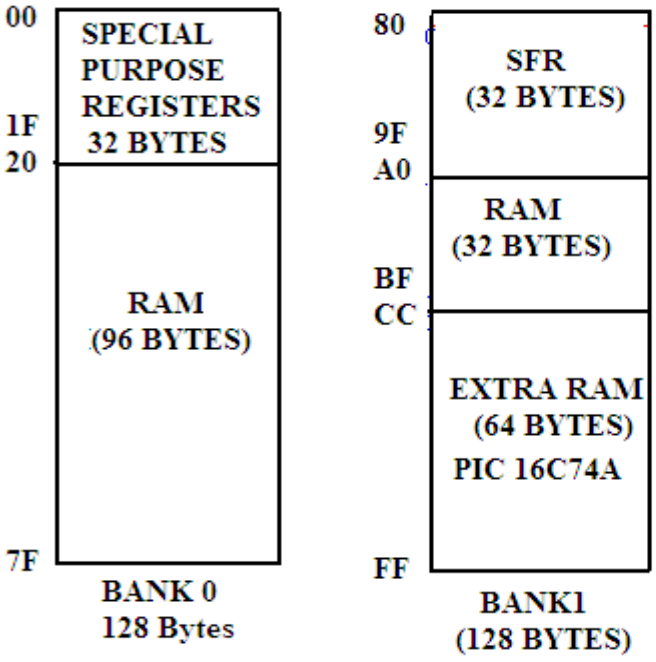
a) General Purpose Register File
b) Special Purpose Register File

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.



RP0	RP1	Bank
0	0	Bank 0
0	1	Bank 1

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.

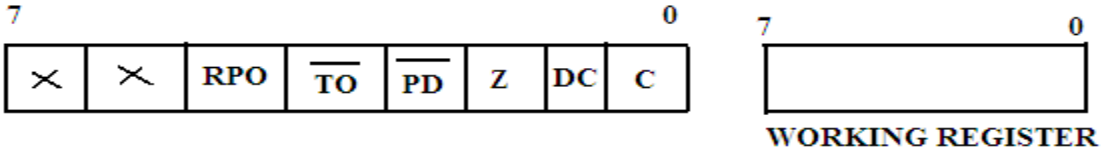
1. Working Register-W (Similar to Accumulator)
2. Status Register
3. FSR – File Select Register (Indirect Data memory address pointer)
4. INDF
5. Program Counter

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:

This is an 8-bit register which denotes the status of ALU after any arithmetic operation and also RESET status and the bank select bits for the data memory.



C: Carry/$\overline{\text{borrow}}$ bit

DC: Digit carry/$\overline{\text{borrow}}$ bit

Z: Zero bit

NOT_PD : Reset Status bit (Power-down mode bit)

NOT_TO : Reset Status bit (time- out bit)

RPO: Register bank Select

The bits 7 and 6 of Status Register are unused by 16c6x/7x. The 'C' bit is set when two 8-bit operands are added together and a 9-bit result occurs. This 9-bit is placed in the carry bit.

The DC or Digit carry bit indicates that a carry from the lower 4 bits occurred during an 8-bit addition.

Example: 0011 1000

0011 1000

0111 0000

Here DC=1 as a result of the carry from the bit 3 to the bit 4 position.

The Z or zero bits is affected by the execution of arithmetic or logic instructions.

The reset status bits NOT_TO and NOT_PD are used in conjunction with PIC's sleep mode. The micro controller can put itself to sleep mode to save power during intervals when it has nothing to do. It can be reset by any of three kinds. Upon reset the CPU can check these two reset status bits to determine which kind of event resettled it and then respond accordingly.

The Register bank select bit RPO is used to select either bank or bank.When RPO=0, select Bank 0, RPO=1, select Bank 1.

```
Example:    bcf STATUS, RPO     ;    Select bank 0
            bsf STATUS, RPO     ;    Select bank 1.
```

3.FSR – (File Select Register):

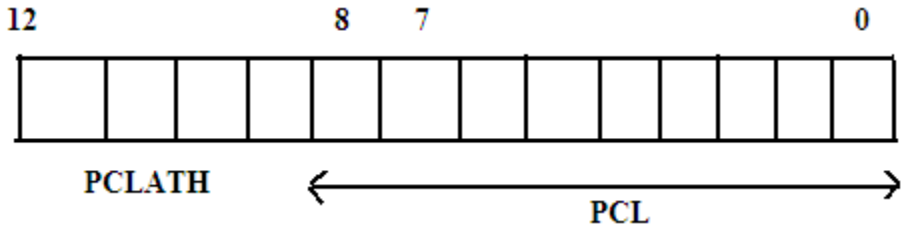
It is the pointer used for indirect addressing. In the indirect addressing mode the 8-bit register file address is first written into FSR. It is a special purpose register that serves as an address pointer to any address through out the entire register file.

4.INDF – (Indirect File):

It is not a physical register addressing but this INDF will cause indirect addressing. Any instruction using the INDF register 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

Most of the PIC16cx/7x family controllers have 33 I/O lines and five I/O ports They are PORT A, PORT B, PORT C , PORT D and PORT E.

PORT A:

Port A is a 6-bit wide bi-directional port. Its data direction register is TRISA setting TRISA bit to 1 will make the corresponding PORT A Pin an input. Clearing a TRIS a bit will make the corresponding pin as an output.

PORT B:

Port B is an 8-bit wide, bi-directional port. Four of the PORT B pins $RB_7 – RB_4$ have an interrupt-on- change feature. Only the pins configured as inputs can cause this interrupt to occur.

PORT C:

Port C is an 8-bit wide, bidirectional port. Bits of the TRISC Register determine the function of its pins. Similar to other ports, a logic one 1 in the TRISC Register configures the appropriate port pin as an input.

PORT D:

Port D is an 8-bit wide bi-directional port. In addition to I/O port, Port D also works as 8-bit parallel slave port or microprocessor port. When control bit PSPMODE (TRISE:4) is set.

PORT E:

It is a 3-bit bi-directional port. Port E bits are multiplexed with analog inputs of ADC and they serve as control signals ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$) for parallel slave port mode of operation.

TIMER MODULES:

There are three completely independent Timers available in PIC 16F8XX Microcontrollers. They are

- Timer 0
- Timer1 and
- Timer2

Timer 0:

The Timer 0 module is a simple 8-bit overflow counter. The clock source can be either the internal system clock (Fosc/4) or an external clock. When the clock source is an external clock, the Timer0 module can be selected to increment on either the rising or falling edge.

The Timer 0 module also has a programmable prescaler option. This prescaler can be assigned to either the Timer0 module or the Watchdog Timer. Bit PSA assigns the prescaler and bits PS2:PSO determine the prescaler value. TMR0 can increment at the following rates: 1:1 when the prescaler is assigned to Watchdog Timer, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128 and 1:256.

Synchronization of the external clock occurs after the prescaler. When the prescaler is used, the external clock frequency may be higher then the device's frequency. The maximum frequency is 50 MHz, given the high and low time requirements of the clock.

Timer 1

Timer1 is a 16-bit timer/counter. The clock source can be either the internal system clock (Fosc/4), an external clock, or an external crystal. Timer1 can operate as either a timer or a counter. When operating as a counter (external clock source), the counter can either operate synchronized to the device or asynchronously to the device. Asynchronous operation allows Timer1 to operate during sleep, which is useful for applications that require a real-time clock as well as the power savings of SLEEP mode.

Timer 1 also has a prescaler option, which allows TMR1 to increment at the following rates: 1:1, 1:2, 1:4 and 1:8 TMR1 can be used in conjunction with the Capture/Compare/PWM module. When used with a CCP module, Timer1 is the time-base for 16-bit capture or 16-bit compare and must be synchronized to the device.

Timer 2

Timer 2 is an 8-bit timer with a programmable prescaler and a programmable postscaler, as well as an 8-bit Period Register (PR2). Timer 2 can be used with the CCP module (in PWM mode) as well as the Baud Rate Generator for the Synchronous Serial Port (SSP). The prescaler option allows Timer2 to increment at the following rates: 1:1, 1:4 and 1:16.

The post scaler allows TMR2 register to match the period register (PR2) a programmable number of times before generating an interrupt. The postscaler can be programmed from 1:1 to 1:16 (inclusive).

CCP (Capture-Compare –PWM)

The CCP module(s) can operate in one of three modes 16-bit capture, 16-bit compare, or up to 10-bit Pulse Width Modulation (PWM)

Capture mode captures the 16-bit value of TMR1 into the CCPRxH:CCPRxL register pair. The capture event can be programmed for either the falling edge, rising edge, fourth rising edge, or sixteenth rising edge of the CCPx pin.

Compare mode compares the TMR1H:TMR1L register pair to the CCPRxH:CCPRxL register pair. When a match occurs, an interrupt can be generated and the output pin CCPx can be forced to a given state (High or Low) and Timer1 can be reset. This depends on control bits CCPxM3:CCPxM0.

PWM mode compares the TMR2 register to a 10-bit duty cycle register (CCPRxH:CCPRxl<5:4>) as well as to an 8-bit period register (PR2). When the TMR2 register=Duty Cycle register, the CCPx pin will be forced low. When TMR2=PR2, TMR2 is cleared to 00h, an interrupt can be generated, and the CCPx pin (if an output) will be forced high.

INTERRUPTS :

The PIC16F8XX 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.

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 prescaler). 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 prescaler 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 scaler, 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 prescaler) it may take several seconds before a WDT time-out occurs.

ADDRESSING MODES.

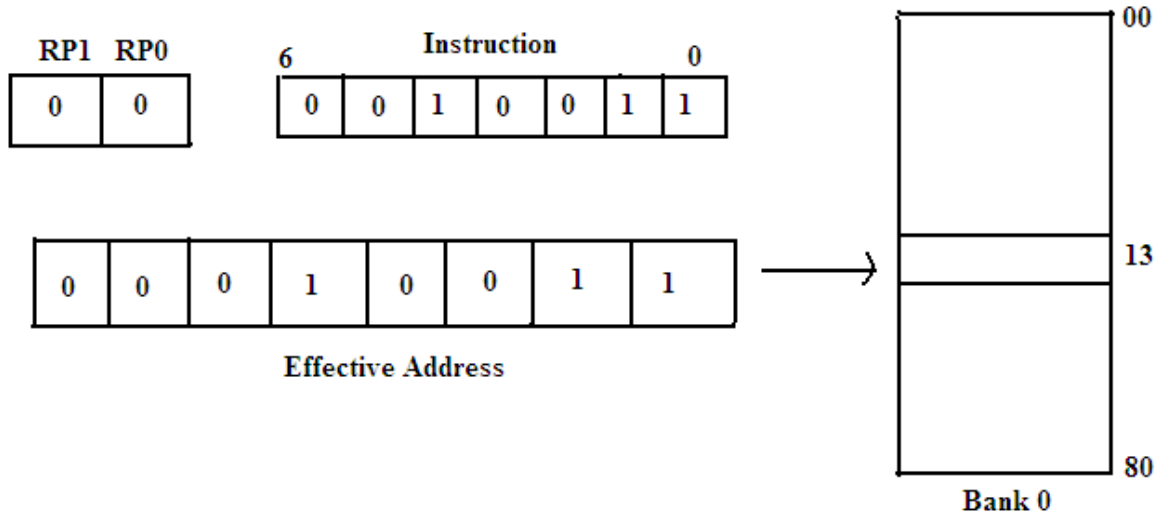
The PIC microcontrollers support only TWO addressing modes .They are

(i) Direct Addressing Mode

(ii) Indirect Addressing mode

Direct Addressing Mode :

In direct addressing mode 7 bits (0-6) of the instruction identify the register file address and the 8 th bit of the register file address register bank select bit(RP0).



The above diagram explains the method of accessing register file address 13H by direct addressing method.

Indirect Addressing Mode

In the indirect addressing mode the 8-bit register file address is first written into a Special Function Register(SFR) which acts as a pointer to any address location in the register file.A subsequent direct access of INDF will actually access the register file using the content of FSR as a pointer to the desired location of the operand.

INSTRUCTION SET

While writing the instructions the following guidelines are followed.

a) Write the instructions mnemonics in lower case (example: xorwf)
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, LoopTime..)

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 Instrucrtions

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 litweral 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.(i)Conditional Branch and (ii) Un conditional Branch instructions**.**

(i) 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

(ii) 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 stand by mode

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

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

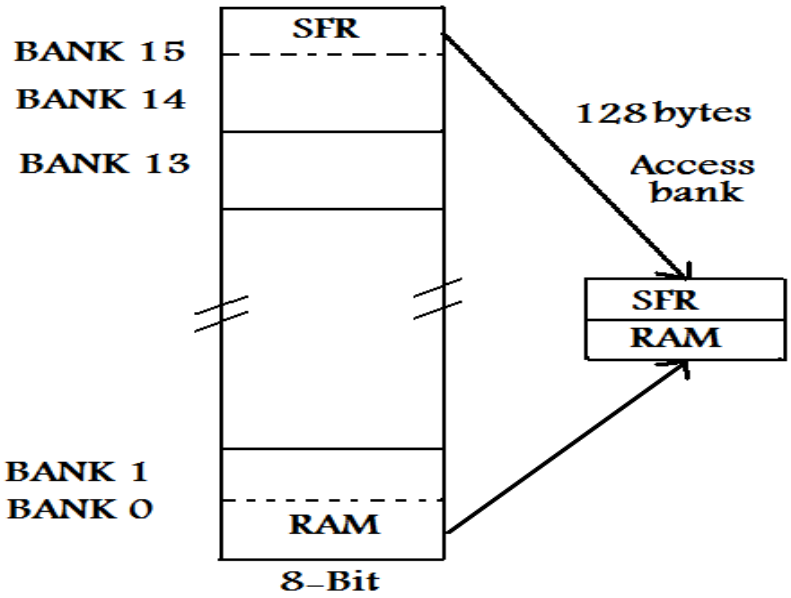
Single-bit manipulation		Operation
bcf	PORTB, 0	;Clear bit 0 of PORTB
bsf	STATUS, C	;Set the carry bit
Clear/move		
Clrw		;Clear the working register, W
clrf	TEMP1	;Clear temporary variable TEMP1
movlw	5	;Load 5 into W
movlw	10	;Load D ‘10’ or H ‘10’ into W

		;depending upon default representation
movwf	TEMP1	;Move W into TEMP1
movwf	TEMP1, F	;Incorrect syntax
movf	TEMP1, W	;Move TEMP1 into W
swapf	TEMP1, F	;Swap 4-bit nibbles of TEMP1
swapf	TEMP1, W	;Move TEMP1 to W, swapping nibbles
		;and leave TEMP1 unchanged
Increment/decrement/complement		
incf	TEMP1, F	;Increment TEMP1
incf	TEMP1, W	;W < - TEMP1 + 1; TEMP1 unchanged
decf	TEMP1, F	;Decrement TEMP1
comf	TEMP1, F	;Change 0s to 1s and 1s to 0s
Multiple-bit manipulation		
andlw	B'00000111'	;Force upper 5 bits of W to zero
andwf	TEMP1, F	;TEMP1 < - TEMP1 and W
andwf	TEMP1, W	;W < - TEMP1 AND W
iorlw	B'00000111'	;Force lower 3 bits of W to one
iorwf	TEMP1, F	;TEMP1 < - TEMP1 or W
xorlw	B'00000111'	;Complement lower 3 bits of W
xorwf	TEMP1, W	;W < - TEMP1 XOR W
Addition/Subtraction		
addlw	5	;Add 5 to W
addwf	TEMP1, F	;TEMP1 < - TEMP1 + W
sublw	5	;W < - 5 – W (not W < - W – 5!)
subwf	TEMP1, F	;TEMP1 < - TEMP1 – W
Rotate		
rlf	TEMP1, F	;Nine-bit left rotate through C
		;(C < - TEMP1, 7; TEMP1, I+1 < - TEMP1, I
		; TEMP1, 0 < - C)
rrf	TEMP1, W	; Leave TEMP1 unchanged
		;copy to W and rotate W right through C

Conditional branch	
btfsc TEMP1, 0 btfss STATUS, C decfsz TEMP1 , F incfsz TEMP1, W	;Skip the next instruction if bit 0 of ;TEMP1 equals zero ;Skip if C = 1 ;Decrement TEMP1; skip if zero ;Leave TEMP1 unchanged; skip if ;TEMP1 = H'FF'; W< - TEMP1 + 1
Goto/call/return/return from interrupt	
goto There call Task1 return retlw 5 retfie	;Next instruction to be executed is ; labeled "There" ;Pushed return address; next instruction ;to be executed is labeled "Task1" ;Pop return address off of stack ;Pop return address; W < -5 ;Pop return address; reenable interrupts
Miscellaneous	
Clrwdt sleep nop	;If watchdog timer is enabled, this; instruction will reset it (before it,;resets the CPU) ;Stop clock; reduce power; wait,;for watchdog timer or external signal;to begin program execution again ; Do nothing; wait one clock cycles

Bank Switching : In PIC 18F family of microcontrollers **t**he data memory 4096 bytes is partitioned into multiple banks. These banks contain the general purpose registers and the Special function Registers(SFRs).The minimum bank that every PIC has is known as Access bank.The access bank consists of 128 bytes of lower addresses and 128 bytes of higher addresses.The lower 128 bytes of address space 000 – 07fh is used for general purpose RAM and the higher 128 bytes F80 – FFF H are dedicated to SFRs.Bank switching is used to take the advantage of the entire RAM. Space. By default the access bank is selected.If we want to change the bank ,the bit in the bank selection register A must be changed. If A =1 , the instruction **MOVWF filereg A**, will use the bank selector register (BSR) to select the desired bank.

The BSR is a part of SFRs and it is an 8- bit register .Among the 8-bits only the least 4-bits are used in bank switching and the upper 4-bits are set to zero and ignored. Using these 4-bits , we can select any of the 16 banks namely Bank 0 to Bank F, which covers the entire 4096 bytes of the RAM area. If the BSR is equal to 1 ,it will select Bank 1 and if BSR is 2 ,Bank2 is selected .The diagram below shows the Data RAM registers.



Let us consider an example to implement the Bank switching.

```
MOVLB   0X2             ;Load 2 into BSR ,i.e select Bank 2
MOVLW  0                ; wreg = 0
MOVWF  MYREG ,1         ; A = 1
INCF MYREG F ,1         ; A =1
```

In the above examples ,the bit A = 1 ,so the BSR can be used to switch to banks other than default access bank.

Table processing: PIC 18F microcontrollers have a ROM area equal to 2MB,which is also known as program memory. This program memory is under the direct control of program counter register. So, to fetch the data from this ROM we need a special function register and hence this method is widely known as register indirect addressing mode.This process of accessing the Code in the ROM is also called Table processing.

Certain instructions are provided for accessing the tables. To read the fixed data type ,an address pointer is used ,which points to to the byte to be fetched. The TBLPTR is a 21 –bit register used to point to the byte to be fetched. With this 21-bit register ,the entire 2MB of

program ROM space can be covered. But the problem is ,there is no instruction to load the 21 –bit address into TBLPTR. So,the TBLPTR is divided into three 8-bit registers called TBLPTRL(low),TBLPTRH(high) and TBLPTRU(upper).All the three are part of the SFRs

The other SFR register used for table processing is TAB-LAT(Table Latch) which is used to keep the fetched byte into the CPU.The instruction INCF TBLPTRL is used to increment the pointer. There are also instructions like TBLRD*+ ,which read the table and increment and TBLRD*D, which read the table and decrement.

The following table gives the table read instructions and their description.

S.No	Instruction	Function	Description
1	TBLRD*	Table Read	No change in the Table Pointer
2	TBLRD*+	Table read with post-Increment	Table reads and increment
3	TBLRD*-	Table read with post-Decrement	Table reads and decrement
4	TBLRD+*	Table read with pre--Increment	First increment ,then table read

Macros and Modules : In assembly language programming certain group of instructions may have to be called repeatedly in the program. It will be highly time consuming to write the code every time we use it. Instead of writing the code every time we use the concept of writing the code once and invoking it when ever it is required..This concept is called Macros. These Macros will allow to call and execute the program written already .

To create a Macro ,first it must be defined. Every Macro definition has three parts as shown below.

```
Name        MACRO       dummy1, dummy2, dummy3, ……..dummyN
            ---------
            ----------
            ENDM
```

Here the MACRO is a directive which indicates the beginning and the ENDM directive indicates the End of the Macro. The code between MACRO and ENDM denotes the body of the macro. The dummies are the names or parameters or even registers used in the body of the macro.Once the macro is written ,it can be called by its name and suitable values can be

added to the dummy parameters. The normally used Macro service is moving the literal data in to RAM.This can be done as shown below.

```
MOVLF    MACRO K,  MYJOB
         MOVLW  K
         MOVWF    MYJOB
         ENDM.
```

Example 1: MOVLF 0X55 ,0X20 ;send the value 55H to the location 20H

```
Example 2:  VAL_1     EQU  0X55
            RAM_LOC   EQU 0X20
            MOVLF      VAL_1, RAM_LOC
```

Example 3: MOVLF 0X55 ,PORT B ; Send thevalue to Port B

The instruction MOVLF 0X55 ,0X20 invokes the macro.

The directive **LOCAL** is used to indicate the labels field body of the macro.

INCLUDE is another directive used to write the macros and save them in a file and later bring them into any program file.

Differences between Macros and Subroutines: Macros increase the code size every time they are invoked.For example if we call a macro of 5 instructions for 5 times ,the code size is increased by 25 instructions.This is not the same in the case of a subroutine .In a subroutine the code size remains same.The limitation of the subroutine is ,that it uses the stack space when called, which gives problems during the nested callsdue to stack overflow.

MODULES : Generally ,while developing big software programs ,it is customary to divide the work into small packages and distribute the task of writing these parts among the different programmers. These small packages are called the modules and this method is called modular programming. This modular programming has the following advantages.

(i).Each module can be developed ,debugged and tested individually

(ii).The error in one module do not affect the entire program.

(iii) The task of debugging is easy and takes less time.

(iv).One can use the modules to link with high level languages like C.

(v).Due to the parallel development ,the total time of development decreases.

(vi).The overall project will be more manageable by this modular programming.

After writing each module in a program it is tested and all the modules are linked to make the complete program.To enable these modules to be linked ,certain assembly language

directives are used. The two widely used directives are EXTERN (external) and GLOBAL. This GLOBAL is same as PUBLIC in other assembly language programs.

The EXTERN directive is used to notify the assembler and linker that certain names and variables that are not defined in the present module are defined externally .In the absence of the EXTERN directive ,the assembler would show an error, because it cannot find the where the names are defined.

PIC I/O programming (Programming the Ports) : The PIC 16F family of microcontrollers have a total of 33 pins arranged into 5 ports. PortA , Port B, Port C , Port D and Port E. In order to use them as I/O ports, they must be properly programmed. In addition to acting as I/O ports , they also have certain additional functions like ADC, Timers ,Interrupts and serial communication pins etc.

PORT A: Port A is a 6-bit wide bi-directional port. Its data direction register is TRISA setting TRISA bit to 1 will make the corresponding PORT A Pin an input. Clearing a TRIS a bit will make the corresponding pin as an output.

PORT B:

Port B is an 8-bit wide, bi-directional port. Four of the PORT B pins $RB_7 – RB_4$ have an interrupt-on- change feature. Only the pins configured as inputs can cause this interrupt to occur.

PORT C:

Port C is an 8-bit wide, bidirectional port. Bits of the TRISC Register determine the function of its pins. Similar to other ports, a logic one 1 in the TRISC Register configures the appropriate port pin as an input.

PORT D:

Port D is an 8-bit wide bi-directional port. In addition to I/O port, Port D also works as 8-bit parallel slave port or microprocessor port. When control bit PSPMODE (TRISE:4) is set.

PORT E:

It is a 3-bit bi-directional port. Port E bits are multiplexed with analog inputs of ADC and they serve as control signals ($\overline{RD}$, $\overline{WR}$, $\overline{CS}$) for parallel slave port mode of operation.

The Ports of PIC controller are made either inpur or output ports by using the TRISx register ,which is a SFR.To make the Port an output,0 must be written to the TRISx register and to use the pPort as input ,a 1 must be put in to the TRISx register. Fors example to make the PortB as input port , the bits of TRISB are made 1(High).

The following example explains the I/O port programming.

Example 1:

```
        MOVLW 0x0
        MOVWF TRISB   ; make the Port B an output port.
L1 :    MOVLW  0x55  ; WREG = 55
        MOVWF PORTB ; Move 55 into portB.
        CALL DELAY
        MOVLW   0X AA
        MOVWF PORTB ; Move AA into portB
        CALL DELAY
        GO TO L1
```

This program alternately loads Port B with 55 and AA.

Example 2: In this example Port B and PORT C are used to transfer the data continuously.

```
        MOVLW B '00000000' ; WREG = 00000000(Binary)
        MOVWF TRISB        ; Port B an out port
        MOVLW B '11111111' ;WREG = 11111111 (Binary)
        MOVWF TRISC        ; Port C input Port
L2      MOVF PORTC , W  ;Move data from Port C to WREG.
        Addlw  5           ; Add literal 5 to it.
        MOVWF  PORTB       ; send it to Port B
        GOTO L2            ; Continue the loop.
```

Example 3 :

```
        CLRF TRISB      ; Clear TrisB( Port B is made outport)
        SETF TRISC      ;  Set TRISC, (Port C is made Input port)
L2  :   MOVF   PORT C ,W  ; Get data from Port C.
        ADDLW 5           ;  Add  literal 5
        MOVWF  PORT B     ; Send it to Port B
        BRA L2            ;  Branch to Loop L2
```

So , it is clear that unless the TRIS bits are activated by putting a 1 ,the data will not be transferred to WREG from the port pins of PORT C

----------BRP----------