PIC Microcontroller

PIC Microcontroller (MCU)

Widely used device from Microchip Technology

- Sold > 10 billion PIC controllers
- Several device families
  - Many devices per family
  - Common development environment
- Widely available
  - Large user base
  - Extensive application notes
- Low cost
- Free / low cost development tools

Basic architectural features

- Pipelined RISC microprocessor core
- Accumulator execution model
- Data width — 8 / 16 / 32 bits
- Instruction width — 12 / 14 / 16 / 32 bits

PIC Families

Architecture	Family		Data Width	Instruction Width
8-bit MCU	PIC10 / PIC12 / PIC16	Baseline	8 bits	12-bits
		Mid-Range		14-bits
	PIC18			16-bits
16-bit MCU	PIC24		16 bits	16 bits
	dsPIC30	Integrated DSP		
32-bit MCU			32 bits	32 bits

Data width

8 / 16/ 32 bits

Wider integer $\Rightarrow$ higher precision arithmetic

Instruction width

12 / 14 / 16 / 32 bits

Wider instruction $\Rightarrow$ more complex instructions + higher precision arithmetic

Typical Applications

Baseline

- Replace discrete logic functions
  - Gates, simple state machines, encoders/decoders, etc.
- Disposable electronics
  - Drug / pregnancy testers, dialysis monitor, etc

Mid-Range

- Digital sensors, displays, controllers, telecom equipment
- Glucose / blood pressure set

PIC18

- Integration with peripherals + networks
  - USB, Ethernet, MCU-to-MCU, etc
- Higher level analog peripherals, industrial control, major appliances

PIC24 / dsPIC30

- 16-bit ALU with integrated DSP
  - Portable EGK

PIC32

- General purpose RISC microprocessor + controller
  - MRI

Learning PIC Architecture

Some general observations

Variety

Hundreds of PIC devices in 3 families and several sub-families

Updates

Microchip Technology upgrades devices frequently

Familiar devices replaced with new model

Instruction Set Architecture

8 and 16 bit devices share approximately uniform instruction set

PIC32 implements MIPS ISA

Caveats

Course takes general pedagogical approach to PIC as typical MCU

- Focus on 8-bit devices — Mid-Range + PIC18

Many books + websites on PIC with general-sounding titles

Each device is unique

Few statements are precisely true about each device

8-Bit PIC MCUs

Data memory	Organized as 8-bit registers Some devices also store data on EEPROM 16 B to 4 KB
Program memory	Addressable unit = instruction word = 12 / 14 / 16 bits Smallest: 2 Kword (3 KB of 12-bit instructions) Largest: 64 Kword (128 KB of 16-bit instructions)
Architecture	Pipelined RISC 33 to 77 instructions
Stack	Stores 0 (no stack) to 31 instruction addresses Used for function calls
I/O devices	8-bit parallel ports Synchronous / asynchronous serial ports Timers + watchdog timer A/D + D/A converters Pulse width modulators

8-Bit PIC Operation Model

ALU sources

Special **WORKING** register W

Data register or immediate

ALU destination

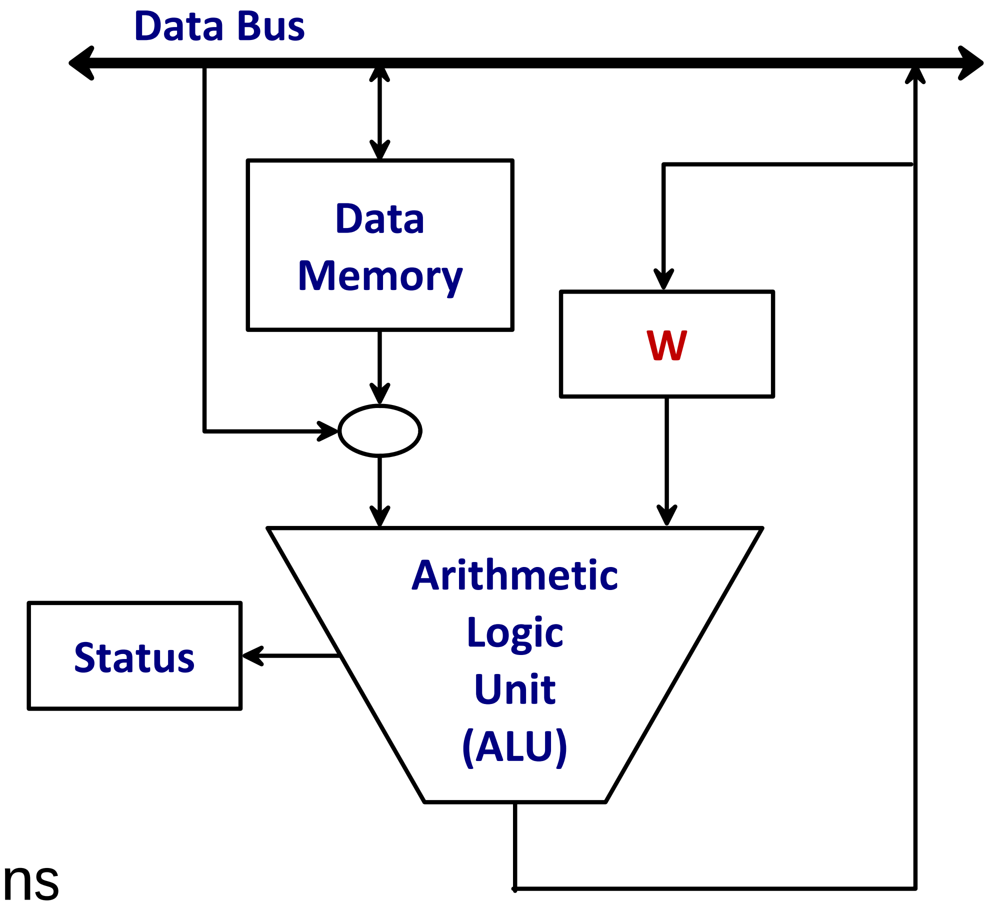
Data register or W

Transfer operations

Data register $\leftrightarrow$ W

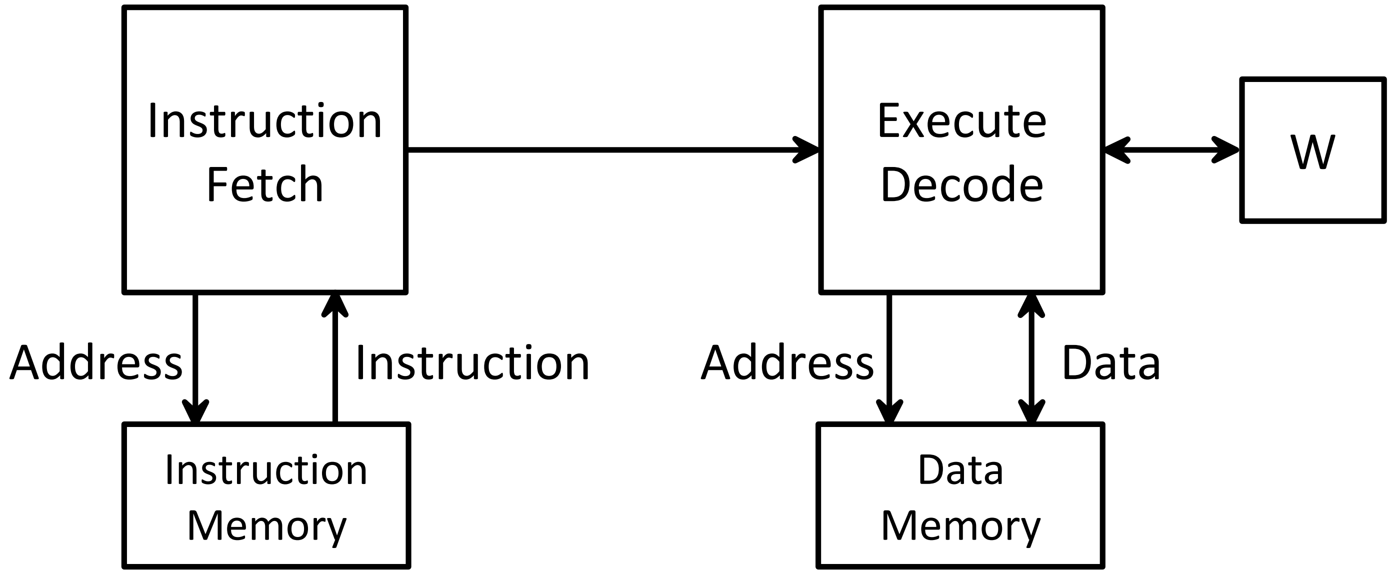
Status register

Flags produced by ALU operations



Pipeline Operation

Instruction cycles (CY)



Instruction Cycles →

	1	2	3	4	5
I_1	fetch	execute			
I_2		fetch	execute		
I_3			fetch	execute	
I_4				fetch	execute

Branch instructions require 2 instruction cycles

Pipeline Operation

Clock cycles (OSC)

IR — instruction register
PC — program counter

Instruction Cycle

4 cycles of external clock (oscillator)

$CY = Q1 \rightarrow Q2 \rightarrow Q3 \rightarrow Q4$

Instruction fetch

Q1	Update Program Pointer	**PC ← PC + 1**
Q2 – Q3		
Q4	Fetch	**IR ← [PC]**

Execution

Q1 – Q4	Decode and Execute	Operation dependent

Instruction Cycles

	1	2	3	4	5
	Q1Q2Q3Q4	Q1Q2Q3Q4	Q1Q2Q3Q4	Q1Q2Q3Q4	Q1Q2Q3Q4
I_1	fetch	execute			
I_2		fetch	execute		
I_3			fetch	execute	
I_4				fetch	execute

Clock Types

RC oscillator

Least expensive

Can be used for non-critical frequency accuracy and stability

Some devices have internal RC oscillator at 4 MHz

Crystal oscillator

Most stable

External clock

Provided by external digital system

Specific modes

LP mode — frequencies between 32 kHz and 200 kHz

XT mode — frequencies between 100 kHz and 4 MHz

HS mode — frequencies between 8 MHz and 20 MHz

Sleep Mode

Low-power mode

Main oscillator stopped

Most MCU functions stopped

Watchdog time continues

Power consumed < 1 mA for some models

Instruction SLEEP

MCU $\rightarrow$ sleep mode

Data register values stable

Pipeline locked

Sleep instruction executes $\Rightarrow$ next instruction already fetched

On wake up

Next instruction executes

Recommendation — instruction after sleep = NOP

Watchdog timer counter reset

Wake Up Events

Reset

Fetch instruction from address 0

Watchdog timer overflow

Normal execution of instruction following sleep

Interrupt

Interrupt not enabled $\Rightarrow$ ignore interrupt

Enabled

Normal execution of instruction following sleep

PC jumps to address 4 in program memory

Finds interrupt routine

Watchdog Timer

WDT oscillator (clock)

- Independent from main clock
- Continues in low power mode
- May be disabled

WDT timeout

- Timeout = 18 ms
- Non-sleep mode
  - MCU resets
- Sleep mode
  - MCU wakes up $\rightarrow$ executes instruction following sleep

Reset WDT

- CLRWDT resets timeout = 18 ms

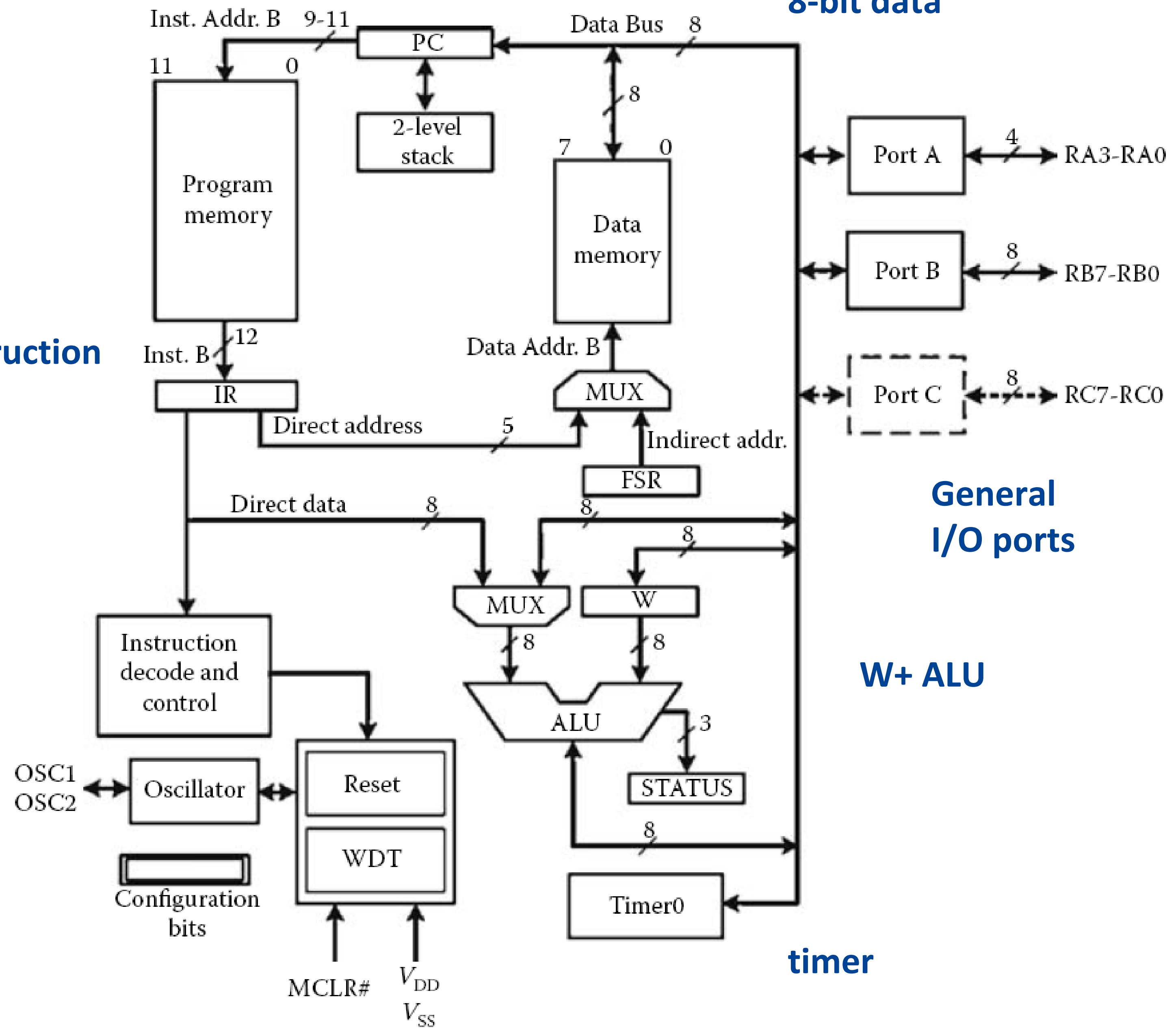
Prescaler

- Divide time-base by 2^k, $k = 0, \dots, 7$
- Extend timeout up to 2300 ms

Some Typical 8-bit PIC Device Families

	PIC10F2xx	PIC12F5xx	PIC16F5xx	PIC10F3xx PIC12F6xx PIC16F6xx	PIC18F5xx
Instruction word	12 bits	12 bits	12 bits	14 bits	16 bits
Instructions	33	33	33	35	83
Program memory	256 – 512 words	512 – 1024 words	1024 – 2048 words	256 – 8192 words	2 Kwords – 64 Kwords
ROM	Flash	Flash	Flash	Flash	Flash
Data memory (bytes)	16 – 24	25 – 41	25 – 134	56 – 368	256 – 4K
Interrupts	0	0	0	int / ext	int / ext
Pins	6	8	14 – 40	6 – 64	18 – 100
I/O pins	4	6	12 – 32	4 – 54	16 – 70
Stack	2 levels	2 levels	2 levels	8 levels	31 levels
Timers	1	1	1	2 – 3	2 – 5
Bulk price	$0.35	$0.50	$0.50 – $0.85	$0.35 – $2.50	$1.20 - $8.50

Typical Baseline MCU —PIC16X5xx Family



8-bit data

12-bit instruction

General I/O ports

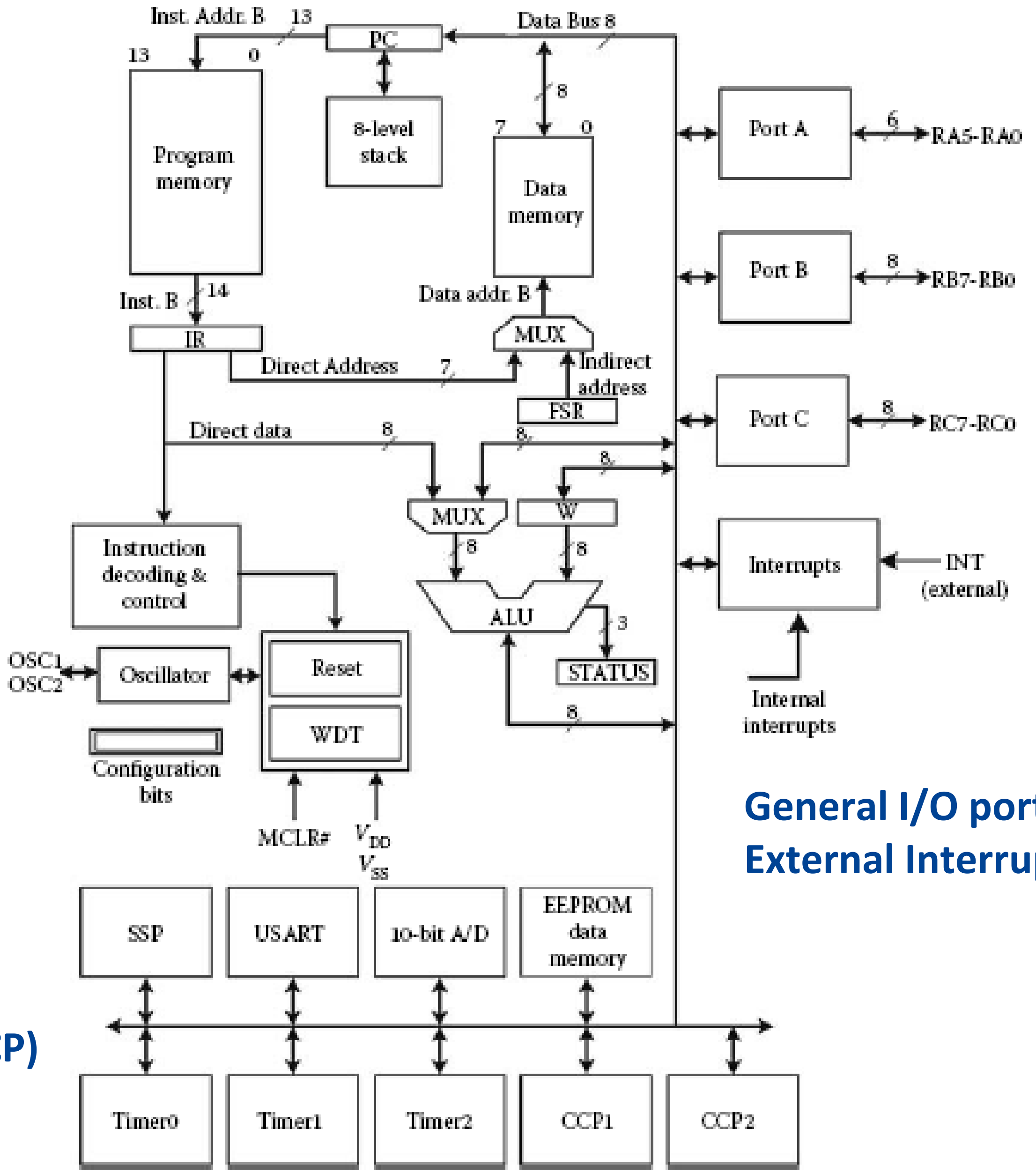
W+ ALU

timer

Typical Mid-Range MCU — PIC16F873

8-bit data

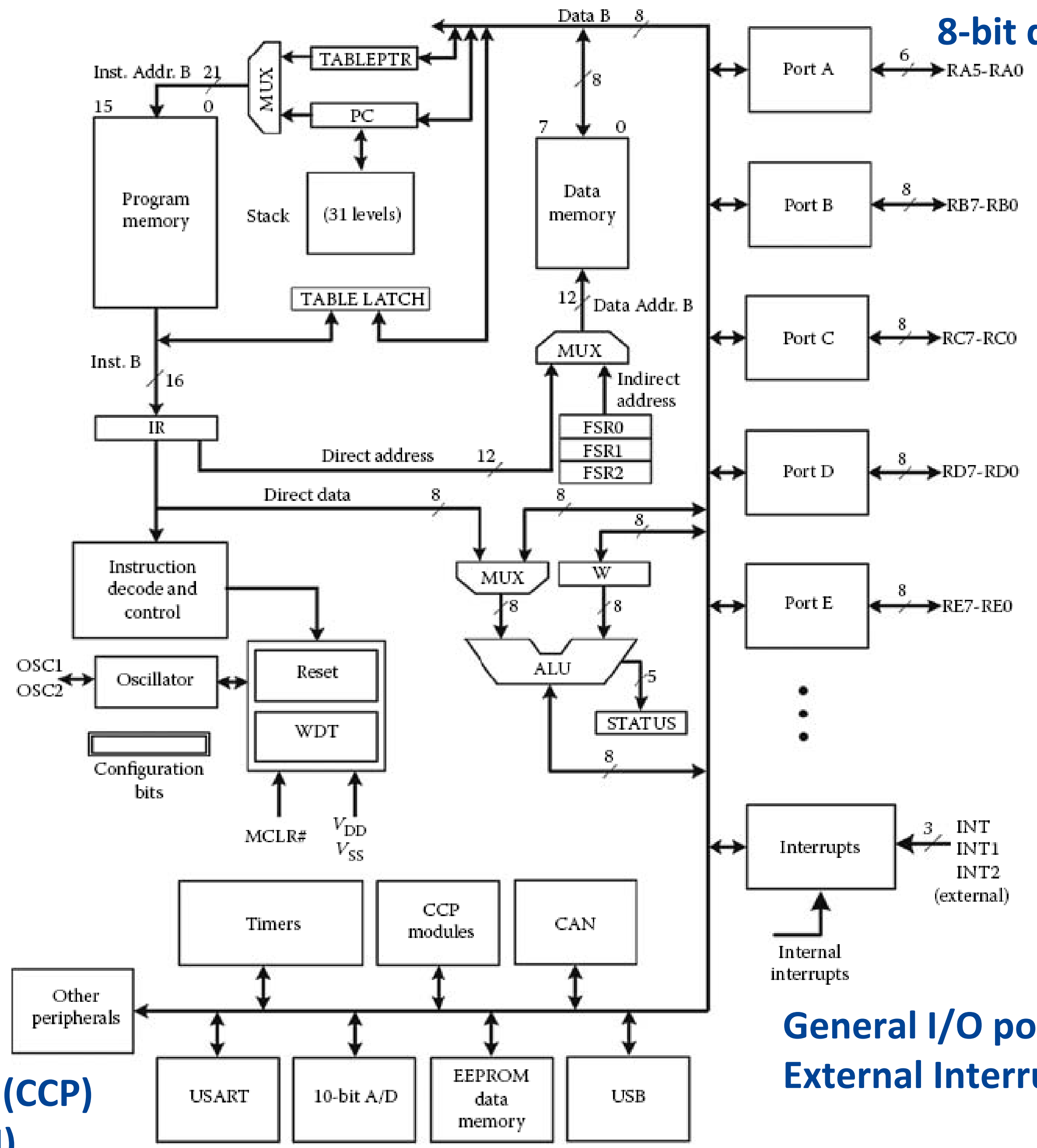
14-bit instruction



General I/O ports
External Interrupt

Timers
A/D
UART
Compare-Capture-Pulsewidth (CCP)
Synchronous Serial Port (SSP)

Typical PIC18 MCU



8-bit data

16-bit instruction

Timers
A/D
UART
USB
Compare-Capture-Pulsewidth (CCP)
Controller Area Network (CAN)

General I/O ports
External Interrupts

Mid-Range PIC MCUs

Data Memory / Registers

Register

Addressable location in data memory

8-bit word (byte)

Data address space

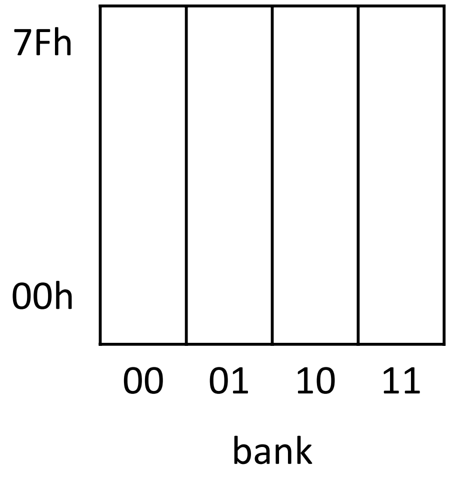
9 bit address $\Rightarrow$ memory $\leq 2^9 = 512$ bytes = 0.5 KB

Memory partitioned into banks

Bank = $2^7 = 128$ = 80h registers (1/8 KB)

7 bit file address

Displacement in bank = 00h ... 7Fh



Banks in address space

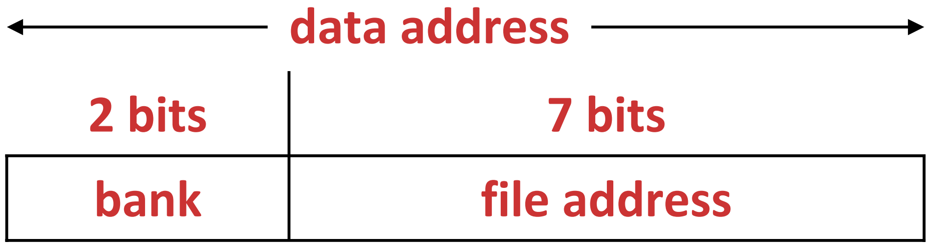
$\leq 2^{9-7} = 4$ banks

2 to 4 banks implemented in device

Unimplemented banks

Read as 0

Write as NOP



Special / General Registers

GPR

General Purpose Registers

User program data

SFR

Special Function Registers

Reserved for

- Control / configuration
- Peripheral access
- Indirect addressing
- Program counter

Core SFRs

Appear in every bank at same file address

Typical SFRs	
STATUS	Status word + flags
OPTION	Timer options
PCLATH PCL	Components of program counter (PC)
FSR	File Select for indirect data addressing
INTCON, PIR1, PIE1, PIR2, PIE2	Components of interrupt handling
PORTA, TRISA, …	Access to parallel ports
TMR0, OPTION, INTCON, …	Timer0
TXREG, TXSTA, RCREG, RCSTA, …	Access to serial port
ADRESH, ADRESL, ADCON0, …	Access to A/D converter
EEADRH, EEDATA, …	Access to EEPROM and Flash memory

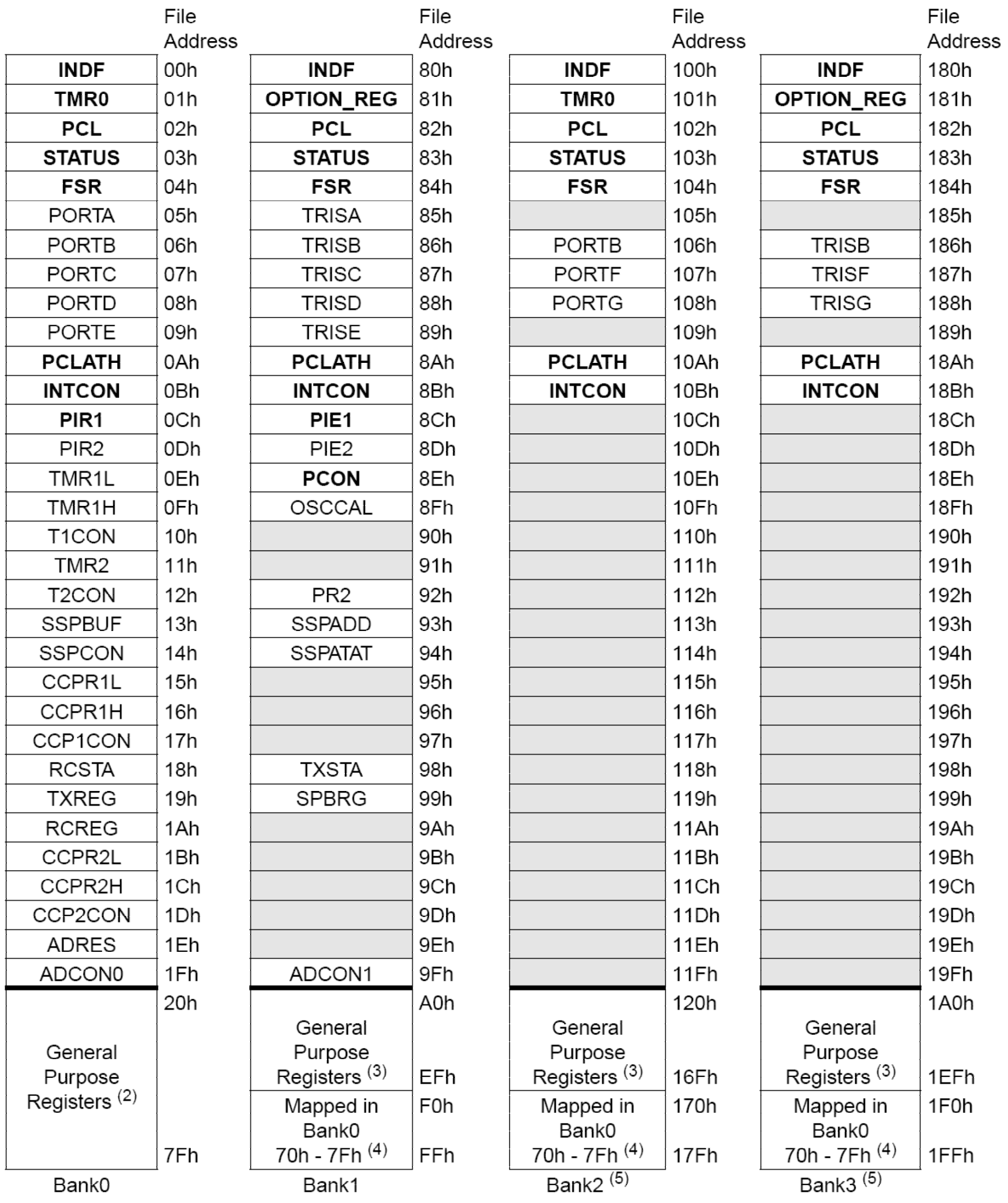
Data Memory Map

Notes

(2,3) Not all locations implemented on all devices

(4) Common RAM — accessible in all banks (on applicable devices)

(5) Not implemented on smaller devices



	File Address		File Address		File Address		File Address
INDF	00h	**INDF**	80h	**INDF**	100h	**INDF**	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	PORTF	107h	TRISF	187h
PORTD	08h	TRISD	88h	PORTG	108h	TRISG	188h
PORTE	09h	TRISE	89h		109h		189h
PCLATH	0Ah	**PCLATH**	8Ah	**PCLATH**	10Ah	**PCLATH**	18Ah
INTCON	0Bh	**INTCON**	8Bh	**INTCON**	10Bh	**INTCON**	18Bh
PIR1	0Ch	**PIE1**	8Ch		10Ch		18Ch
PIR2	0Dh	PIE2	8Dh		10Dh		18Dh
TMR1L	0Eh	**PCON**	8Eh		10Eh		18Eh
TMR1H	0Fh	OSCCAL	8Fh		10Fh		18Fh
T1CON	10h		90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPATAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		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		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRES	1Eh		9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Registers (2)	20h	General Purpose Registers (3)	A0h	General Purpose Registers (3)	120h	General Purpose Registers (3)	1A0h
			EFh		16Fh		1EFh
		Mapped in Bank0 70h - 7Fh (4)	F0h	Mapped in Bank0 70h - 7Fh (4)	170h	Mapped in Bank0 70h - 7Fh (4)	1F0h
	7Fh		FFh		17Fh		1FFh
Bank0		Bank1		Bank2 (5)		Bank3 (5)	

Status Register

Core SFR accessible at file address **03h** in every bank

	7	6	5	4	3	2	1	0
Name	**IRP**	**RP1**	**RP0**	**TO#**	**PD#**	**Z**	**DC**	**C**
Writable	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W
Reset value	0	0	0	1	1	x	x	x

IRP	Bank Select	Indirect Register Pointer
RP1, RP0		Direct Register Pointer
TO#	State of WDT	**TO#** ← **0** on WDT overflow **TO#** ← **1** on power-on reset, CLRWDT, SLEEP
PD#	Low-power	**PD#** ← **0** on SLEEP **PD#** ← **1** on CLRWDT and power-on reset
Z	Zero flag	**Z** ← **1** on ALU zero **Z** ← **0** on non-zero
DC	Half-byte carry (bits 3,4)	**DC** ← **1** on carry (Addition) **DC** ← **0** on borrow (Subtraction)
C	Carry out	**C** ← **1** on carry (Addition) **C** ← **0** on borrow (Subtraction)

Addressing Data Memory

Notation	
REG<b>	Bit **b** in register **REG**
REG<a:b>	Bits **a** to **b** in register **REG**
A.B	Concatenation of **A** and **B** (**A** bits followed by **B** bits)

Direct addressing

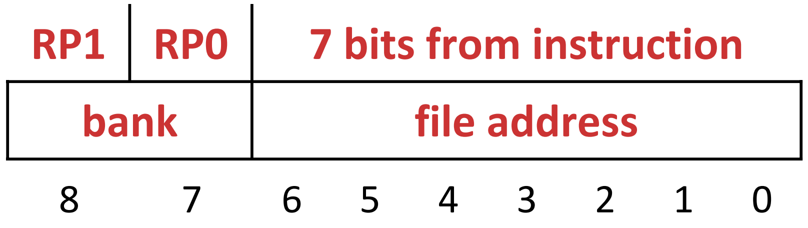
Program specifies data address

Bank selection

- **STATUS** bits **RP1** and **RP0**
- On reset
  - **RP1 = RP0 = 0** $\Rightarrow$ bank **0** selected
- Bank switching
  - Write to **STATUS<6:5>**

File Address

- Literal field in instruction



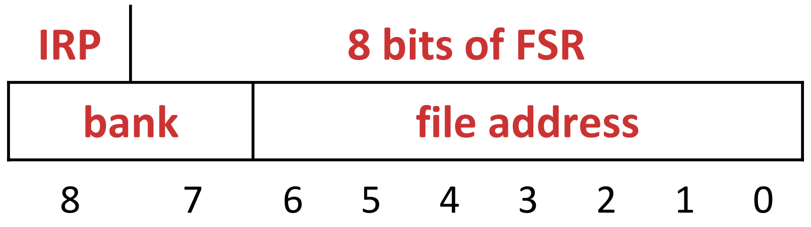
Addressing Data Memory

Indirect addressing

- Program writes to Special Function Registers (SFRs)
  - Address formed from SFRs
- Instructions can increment/decrement SFR values
  - Similar to pointer arithmetic

File Select Register (FSR)

- Core SFR accessible at file address **08h** in all banks
- File Address
  - **FSR<6:0>**
- Bank
  - **IRP.FSR<7>**
  - **STATUS** bit **IRP** (Indirect Register Pointer)



On small devices

- 1 or 2 banks = 128 or 256 bytes of data memory
- 8 bit **FSR** address covers 2 banks
- **IRP** not implemented (read **0** / write = **NOP**)

INDF Register

INDF

Core SFR accessible at file address **00h** in all banks

Virtual pointer — not physical register

Tracks contents of **FSR**

Simplifies pointer arithmetic

Example

In register file,

```
    [05] = 10h
    [06] = 0Ah
Load FSR ← 05  ; FSR points to file address 05
[INDF] = 10h   ; INDF points to file address 05
FSR++          ; increment FSR ⇒ FSR = 06
[INDF] = 0Ah   ; INDF points to file address 06
```

Instruction Memory Space

All 8-bit MCUs

Instruction address

n bit location address

Location = instruction

$\leq$ **2^n** instructions

Instruction width

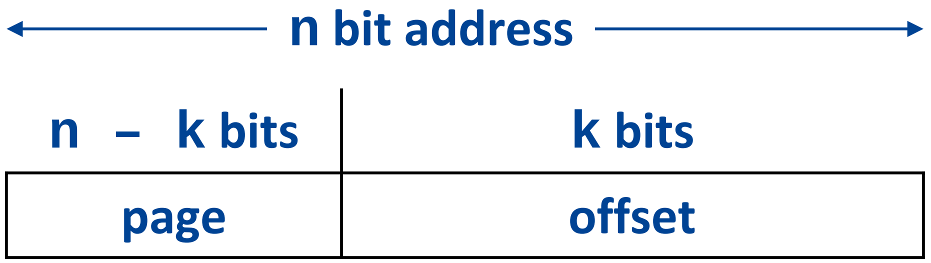
12 / 14 / 16 bits

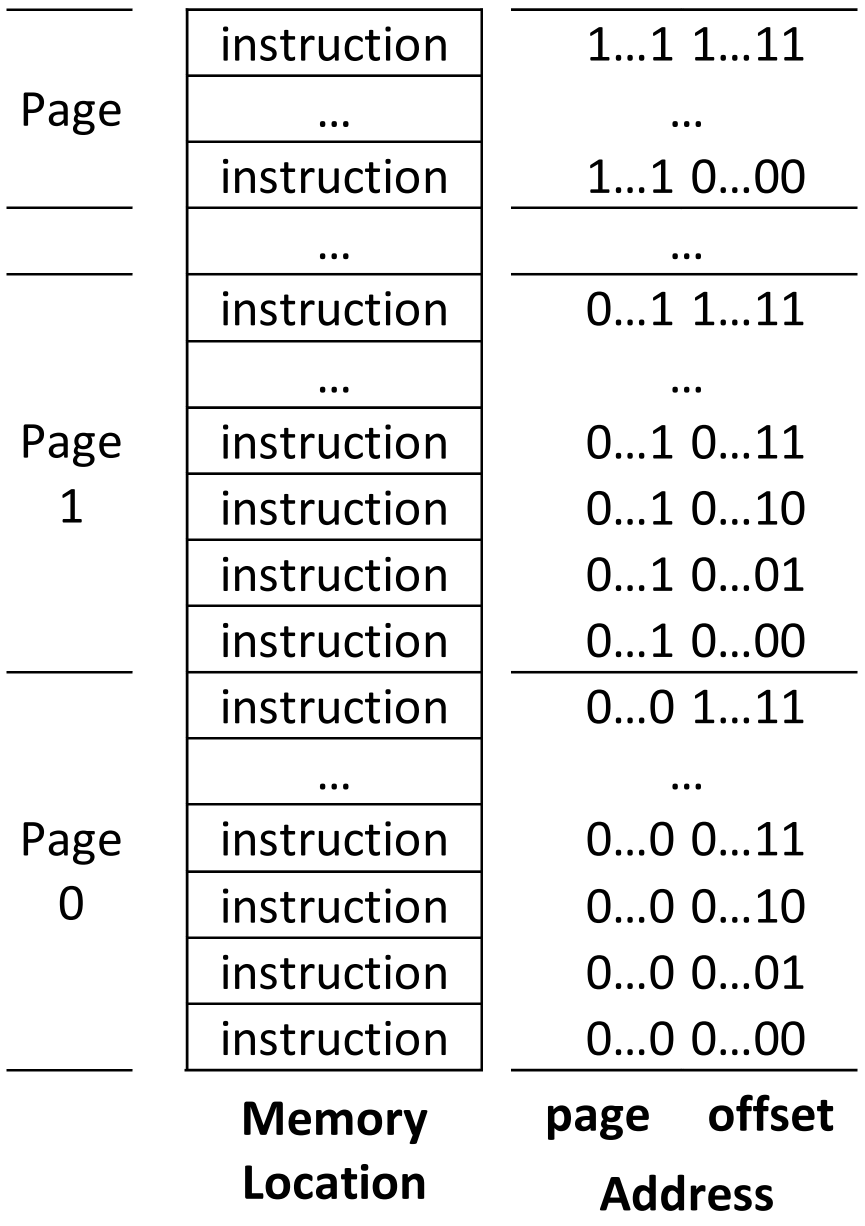
Page

Partition of instruction memory space

2^k instructions / page

k bit offset





Instruction Memory Space

Mid-Range instruction memory

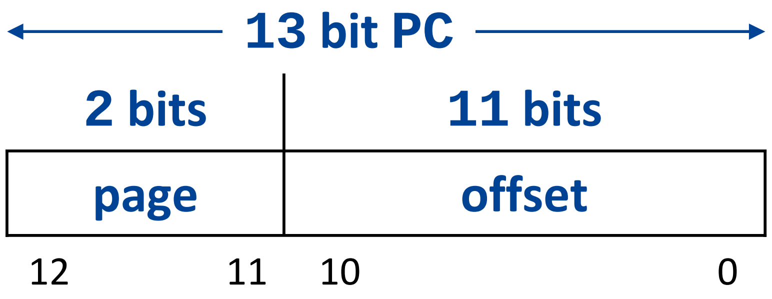
14-bit instruction word

`n = 13`

2^{13} = 8192 instruction words

`k = 11`

Page = 2^{11} = 2048 = 800h words



Program counter (PC)

`PC<12:11>` = page number ⇒ 4 pages

`PC<10:0>` = offset

Reserved addresses

Address **`0h`**

Reset vector — pointer to reset routine

Address **`4h`**

Interrupt vector — pointer to interrupt service routine

PC Access

PC register details

`PC low (PCL) = PC<7:0>`

Accessible by instruction reads/writes

`PC high (PCH) = PC<12:8>`

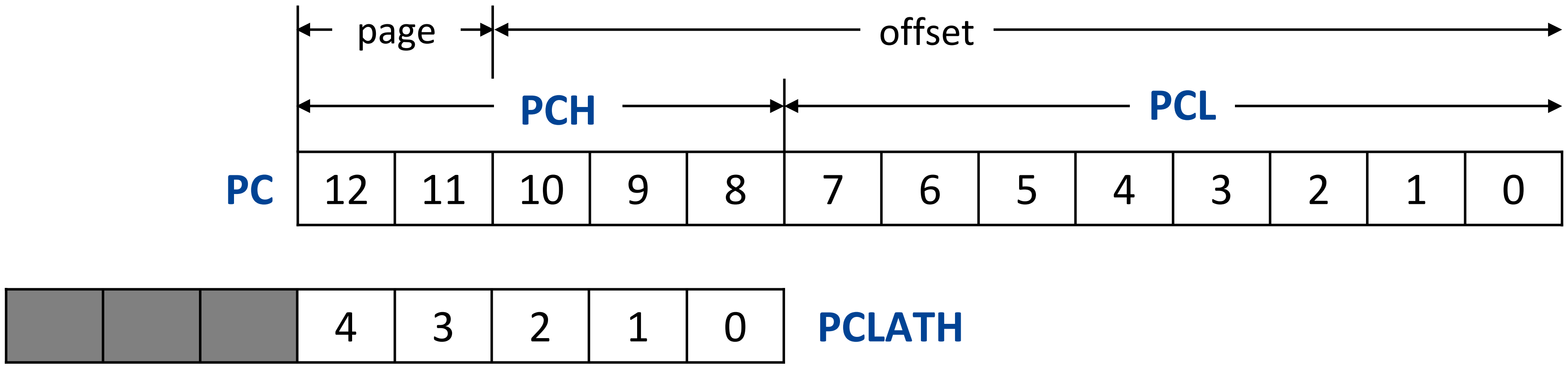
Not directly accessible to instructions

PC latch high (PCLATH)

Core SFR accessible at file address **`0Ah`** in all banks

`PCH = PC<12:8> = PCLATH<4:0>`

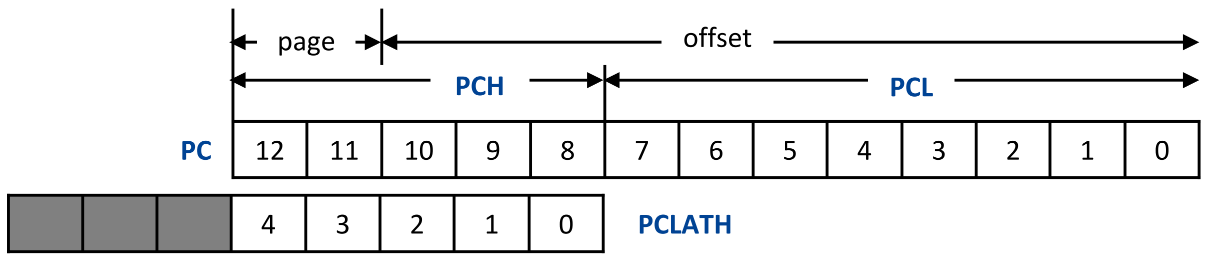
`PCLATH<7:5>` not implemented



PC Updates

Reset

PC ← 0



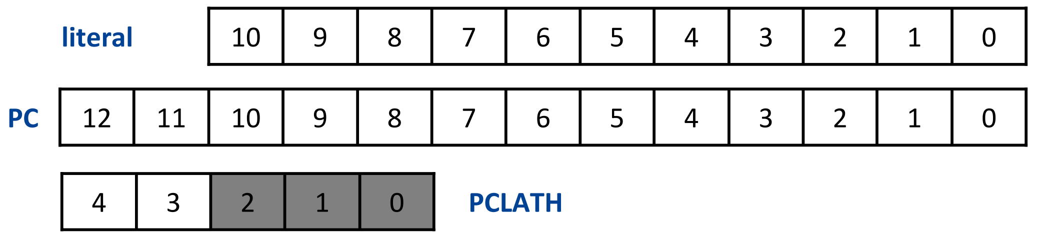
Non-branch instruction

PC ← PC + 1

Branch types

Direct branch

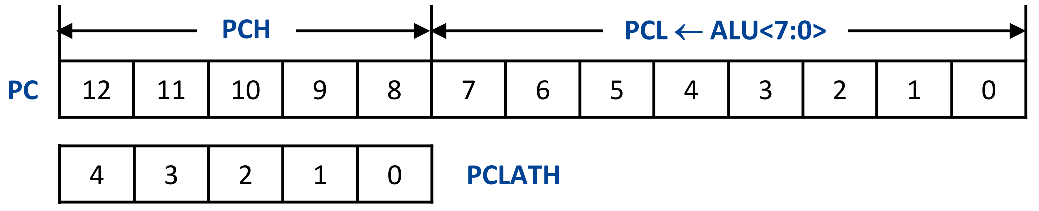
GOTO instruction



PCH<12:11> ← PCLATH<4:3>

Offset = PC<10:0> ← literal<10:0> from instruction

Indirect branch



Computed GOTO

Write to **PCL** as register

Copies **PCL ← ALU result**

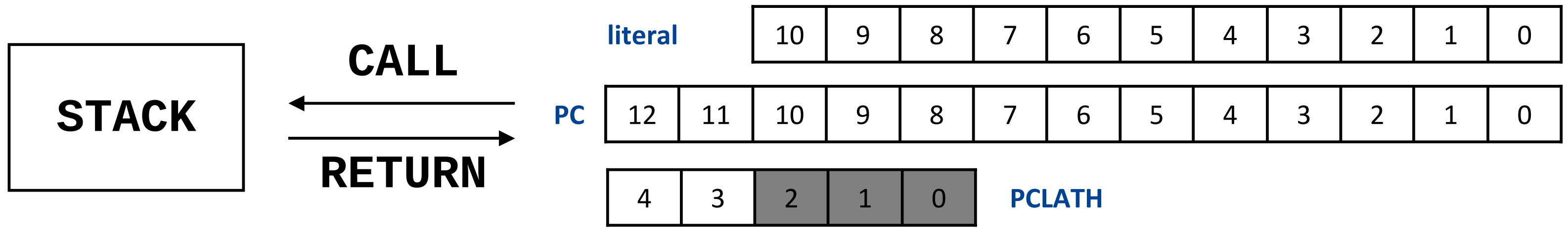
Forces **PCH ← PCLATH<4:0>**

Call / Return

Stack

8 level FILO buffer

Holds 13 bit instruction addresses on **`CALL/RETURN`**



Function entry

`CALL` instruction

`STACK` $\leftarrow$ **`PC<12:0>`**

`PCL` $\leftarrow$ **`literal<10:0>`** from instruction

`PCH<12:11>` $\leftarrow$ **`PCLATH<4:3>`**

Function exit

`RETURN` instruction

`PC<12:0>` $\leftarrow$ **`STACK`**

`PCLATH` **not updated**

May be different from **`PCH`** after **`RETURN`**

Instruction Format

13 8	7	6 0
opcode	**d**	**f**

Byte oriented

d = 0 ⇒ destination = W

d = 1 ⇒ destination = f

f = 7 bit file address

13 10	9 7	6 0
opcode	**b**	**f**

Bit oriented

b = bit position in register

f = 7 bit file address

13 8	7 0
opcode	**k**

General literal

k = 8 bit literal (immediate)

13 11	10 0
opcode	**k**

CALL / GOTO

k = 11 bit literal (immediate)

Instruction Set

Data transfer

Mnemonic	Operation	Comment	Flags
MOVF f, d	$d \leftarrow f$	Move f to d	Z
MOVF f, 0 MOVF f, W	$W \leftarrow f$	d = W	Z
MOVF f, 1 MOVF f, f MOVF f	$f \leftarrow f$	d = f	Z
MOVWF f	$f \leftarrow W$	Move W to f	–
MOVLW k	$W \leftarrow k$	Move literal to W	–
CLRF f	$f \leftarrow 0$	Clear f	Z
CLRW	$W \leftarrow 0$	Clear W	Z

Operands	
f	name / address of register
d	destination
k	literal

$$\text{destination} = \begin{cases} W & , d = 0 \\ f & , d = 1 \end{cases}$$

Instruction Set

Arithmetic and Logic — 1

Mnemonic	Operation	Comment	Flags
ADDWF f, d	$d \leftarrow f + W$	Add w to f	C, DC, Z
ADDLW k	$W \leftarrow k + W$	Add k to W	C, DC, Z
SUBWF f, d	$d \leftarrow f - W$	Sub W from f	C, DC, Z
SUBLW k	$W \leftarrow k - W$	Sub W from k	C, DC, Z
INCF f, d	$d \leftarrow f + 1$	Inc f to d	Z
DECF f, d	$d \leftarrow f - 1$	Dec f to d	Z
ANDWF f, d	$d \leftarrow f \text{ and } W$	And w with f	Z
ANDLW k	$W \leftarrow k \text{ and } W$	And k with W	Z

Operands	
f	name / address of register
d	destination
k	literal

$$\text{destination} = \begin{cases} W & , d = 0 \\ f & , d = 1 \end{cases}$$

Instruction Set

Arithmetic and Logic — 2

Mnemonic	Operation	Comment	Flags
IORWF f, d	$d \leftarrow$ f or W	OR w with f	Z
IORLW k	$W \leftarrow$ k or W	OR k with W	Z
XORWF f, d	$d \leftarrow$ f xor W	XOR W with f	Z
XORLW k	$W \leftarrow$ k xor W	XOR W with k	Z
RLF f, d	$d \leftarrow$ left rotate f,C		C
RRF f, d	$d \leftarrow$ right rotate f,C		C
COMF f, d	$d \leftarrow$ #f (not f)	compliment f to d	C
SWAPF f, d	$d \leftarrow f_L \leftrightarrow f_H$	nibble swap (nibble = half byte)	—

Operands	
f	name / address of register
d	destination
k	literal

$$\text{destination} = \begin{cases} W & , d = 0 \\ f & , d = 1 \end{cases}$$

Instruction Set

Control

Mnemonic	Operation
GOTO a	branch to address
BTFSC f, b	skip one instruction if f<b> = 0
BTFSS f, b	skip one instruction if f<b> = 1
INCFSZ f, d	d ← f + 1, skip one if result = 0
DECFSZ f, d	d ← f - 1, skip one if result = 0
CALL a	call subroutine in address a
RETURN	subroutine return
RETFIE	interrupt return
RETLW k	return from subroutine with k in W

Operands			
f	name / address of register	a	11 bit address
d	destination	b	bit location
k	literal		

Instruction Set

Other

Mnemonic	Operation	Flags
BCF f, b	f<b> ← 0	–
BSF f, b	f<b> ← 1	–
NOP	no operation	–
CLRWDT	WDT ← 0	TO#, PD#
SLEEP	go to low power consumption	TO#, PD#

Operands	
f	name / address of register
b	bit location

Sample Program Fragments
RAM Initialization

```
    CLRF STATUS       ; STATUS ← 0
    MOVLW 0x20        ; W ← 1st address in GPR bank 0
    MOVWF FSR         ; Indirect address register ← W
Bank0_LP
    CLRF INDF0        ; address in GPR ← 0
    INCF FSR          ; FSR++ (next GPR address)
    BTFSS FSR, 7      ; skip if (FSR<7> == 1) ⇒ FSR = 80h
    GOTO Bank0_LP     ; continue
; ** IF DEVICE HAS BANK1 **
    MOVLW 0xA0        ; W ← 1st address in GPR bank 1
    MOVWF FSR         ; Indirect address register ← W
Bank1_LP
    CLRF INDF0        ; address in GPR ← 0
    INCF FSR          ; FSR++ (next GPR address)
    BTFSS STATUS, C   ; skip if (STATUS<0> == 1) ⇒ FSR = 00h
    GOTO Bank1_LP     ; continue
```

Sample Program Fragments

Branch to address in new page

```
Prog:
        movlw HIGH Prog10    ; W ← Prog10<15:8>
          ; operator HIGH reads bits <15:8> of pointer
        movwf PCLATH         ; PCLATH ← W
        goto Prog10          ; PC<10:0> ← Prog10<7:0>
                             ; PC<12:11> ← PCLATH<4:3>


Prog10:
;
; Prog10 labels some address in program memory
;
```

Sample Program Fragments

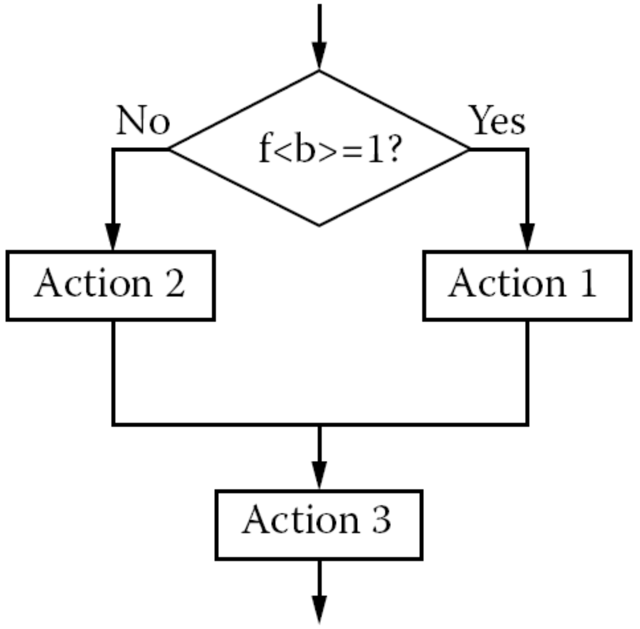
Computed goto

```
movlw HIGH Prog20          ; W ← Prog10<15:8>
movwf PCLATH               ; PCLATH ← W
movlw LOW Prog20           ; W ← Prog10<7:0>
movwf PCL                  ; PCL ← Prog10<7:0>
                           ; PCH ← PCLATH<4:0>
Prog20:
;
; Prog10 labels some address in program memory
;
```

Sample Programs Fragments

if-else branch

```
btfss f,b                ; skip one instruction if
                         ; bit b in register f = 1
goto Action2
Action1:
; instructions for Action1
goto Action3
Action2:
; instructions for Action2
Action3:
; instructions for Action3
```



Sample Programs Fragments

Static loop

```
     movlw times            ; W ← times
     movwf COUNTER          ; COUNTER ← W (times)
Loop:
     ;
     ; loop instructions
     ;
     decfsz COUNTER, f      ; COUNTER--
                            ; COUNTER = 0 ⇒ skip next
                              instruction
     goto Loop              ; next iteration
End:
;
;
```

Sample Programs Fragments

Data table in instruction memory

```
; Function call returns data at Table.INDEX
movlw HIGH Table            ; W ← Table<15:8>
movwf PCLATH                ; PCLATH ← W
movf INDEX, W               ; W ← INDEX
call Table                  ; Call to subroutine table
;
Table:
addwf PCL, f                ; PCL ← PCL + W = PCL + INDEX
                            ; computed goto
retlw 'A'                   ; return with W ← 'A'
retlw 'B'
retlw 'C'
retlw 'D'
retlw 'E'
```

PIC MCU and the Outside World

Oscillator

Generates device clock

Four device clock periods per instruction cycle

Ports

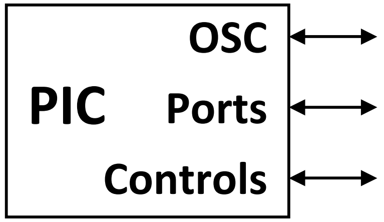
Data I/O pins

Electrical connections to external circuits

Configured to function as

Digital I/O

Analog inputs to A/D converter



Controls

Device dependent

Power + ground

Interrupt (INT)

External clock

Peripheral Modules

Share data pins with general ports

Timer	Counts clock cycles → interrupt on preset count
A/D	Samples analog level → converts to digital representation
Comparator	Samples 2 analog levels → outputs bit **`(A1 > A2)`**
USART	Bit-parallel ↔ bit-serial converter for communications
CCP	Capture/Compare/PWM (Pulse Width Modulation)

Configurable Oscillator Modes

Quartz crystal time base

Crystal connected between PIC pins **OSC1** and **OSC2**

Vibrates in electric field $\rightarrow$ piezoelectric resonance in voltage

Modes

- LP Low Frequency / Low Power Crystal — 32 kHz to 200 kHz
- XT Crystal/Resonator — 100 kHz to 4 MHz
- HS High Speed Crystal/Resonator — 8 MHz to 20 MHz

Resistor / Capacitor time base

Capacitor discharges through resistor in time = $2\pi RC$

Oscillator frequency $f = 1 / (2\pi RC)$

Modes

- EXTRC — External RC connected between PIC pin **OSC1** and ground
- INTRC — Internal 4 MHz RC
- CLKOUT
  - EXTRC or INTRC with instruction clock (= f/4) output on **OSC2**

Interrupts

Interrupt

- Instruction at current PC executes
- Instruction at current PC+1 fetched
- Stack $\leftarrow$ PC+2
- PC $\leftarrow$ interrupt pointer
- On return from interrupt PC $\leftarrow$ stack

Interrupt sources

- External interrupt pin
  - Pin RB0 on some PIC devices (separate pin on other devices)
- Peripheral modules
  - General internal interrupt
  - A/D
  - Timer
  - Comparator
  - USART
  - CCP

Interrupt Control Register

Interrupt Control Register (INTCON)

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
7	6	5	4	3	2	1	0

GIE	**Global Interrupt Enable**	1 = Enables all un-masked interrupts 0 = Disables all interrupts
PEIE	**Peripheral Interrupt Enable**	1 = Enables all un-masked peripheral interrupts 0 = Disables all peripheral interrupts
T0IE	**Overflow Interrupt Enable**	1 = Enables TMR0 overflow interrupt 0 = Disables TMR0 overflow interrupt
INTE	**External Interrupt Enable**	1 = Enables INT external interrupt 0 = Disables INT external interrupt
RBIE	**RB Port Change Interrupt Enable**	1 = Enables RB port change interrupt 0 = Disables RB port change interrupt
T0IF	**Overflow Interrupt Flag**	1 = TMR0 register has overflowed 0 = TMR0 register did not overflow
INTF	**External Interrupt Flag**	1 = INT external interrupt occurred 0 = INT external interrupt did not occur
RBIF	**RB Port Change Interrupt Flag**	1 = At least one of RB7:RB4 pins changed state 0 = None of RB7:RB4 pins have changed state

Peripheral Interrupt Enable (PIE)

Number of PIE registers device dependent

TMR1IE	TMR1 Overflow
TMR2IE	TMR2 to PR2 Match
CCP1IE	CCP1
CCP2IE	CCP2
SSPIE	Synchronous Serial Port
RCIE	USART Receive
TXIE	USART Transmit
ADIE	A/D Converter
ADCIE	Slope A/D Converter Comparator Trip
OVFIE	Slope A/D TMR Overflow
PSPIE	Parallel Slave Port Read/Write
EEIE	EE Write Complete
LCDIE	LCD
CMIE	Comparator

1 = enable device interrupt
0 = disable device interrupt

Peripheral Interrupt Register (PIR) — 1

Number of PIR registers device dependent

TMR1IE	1 = **TMR1** register overflowed 0 = **TMR1** register did not overflow
TMR2IE	Same as **TMR1IE**
CCP1IE	CCP1 Interrupt Flag bit Capture Mode 1 = **TMR1** register capture occurred 0 = No **TMR1** register capture occurred Compare Mode 1 = A **TMR1** register compare match occurred 0 = No **TMR1** register compare match occurred PWM Mode Unused in this mode
CCP2IE	Same as **CCP1IE**
SSPIE	1 = Transmission/reception complete 0 = Waiting to transmit/receive
RCIE	1 = USART receive buffer **RCREG** full 0 = USART receive buffer is empty

Peripheral Interrupt Register (PIR) — 2

Number of PIR registers device dependent

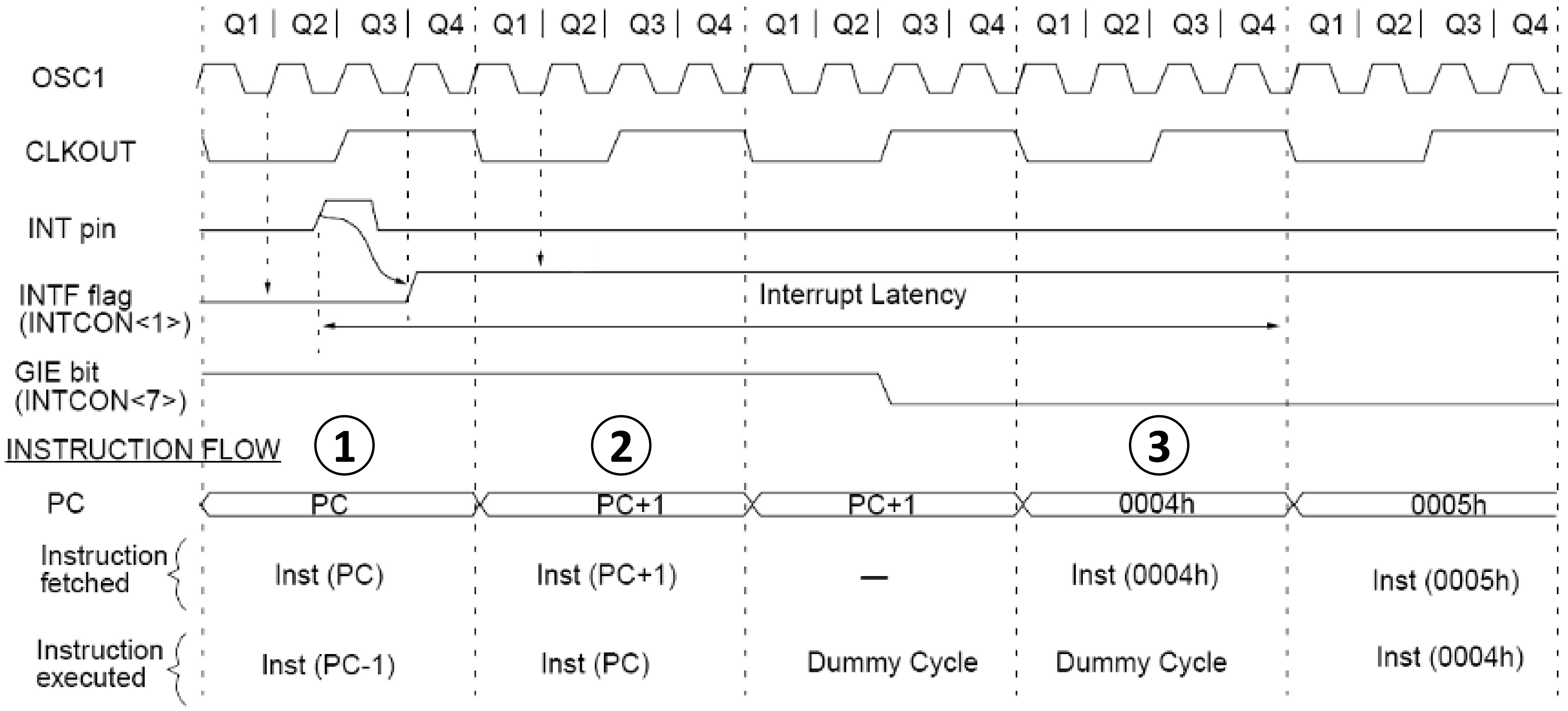
TXIE	1 = USART transmit buffer TXREG empty 0 = USART transmit buffer is full
ADIE	1 = A/D conversion complete 0 = A/D conversion not complete
ADCIE	1 = A/D conversion complete 0 = A/D conversion not complete
OVFIE	1 = Slope A/D TMR overflow 0 = Slope A/D TMR did not overflow
PSPIE	1 = Read or write operation occurred 0 = Read or write did not occur
EEIE	1 = Data EEPROM write operation complete 0 = Data EEPROM write operation not complete
LCDIE	1 = LCD interrupt occurred 0 = LCD interrupt did not occur
CMIE	1 = Comparator input changed 0 = Comparator input not changed

Interrupt Latency

On interrupt

1. Current instruction execution completes
2. Current instruction fetch completes
3. PC $\leftarrow$ interrupt pointer

Latency ~ 3 to 4 instruction cycles



Interrupt Initialization + Enabling

```
PIE1_MASK1 EQU B‘01101010’    ; Interrupt Enable
                              ; Register mask (device
                              ; dependent)
;
;
CLRF STATUS                ; Bank0
CLRF INTCON                ; Disable interrupts during
                           ; configuration
CLRF PIR1                  ; Clear flags
BSF STATUS, RP0            ; Bank1
MOVLW PIE1_MASK1           ; set PIE1 via W
MOVWF PIE1
BCF STATUS, RP0            ; Bank0
BSF INTCON, GIE            ; Enable Interrupts
```

Macros for Register Save / Restore

```
PUSH_MACRO MACRO            ; Save register contents
  MOVWF W_TEMP              ; Temporary register ← W
  SWAPF STATUS,W            ; W ← swap STATUS nibbles
  MOVWF STATUS_TEMP         ; Temporary register ← STATUS
ENDM                        ; End this Macro


POP_MACRO MACRO             ; Restore register contents
SWAPF STATUS_TEMP,W         ; W ← swap STATUS
MOVWF STATUS                ; STATUS  ← W
SWAPF W_TEMP,F              ; W_Temp ← swap W_Temp
SWAPF W_TEMP,W              ; W ← swap W_Temp s
                            ; no affect on STATUS
ENDM                        ; End this Macro
```

Typical Interrupt Service Routine (ISR) — 1

```
org ISR_ADDR           ; store at ISR address
PUSH_MACRO             ; save context registers W, STATUS
CLRF STATUS            ; Bank0
     ; switch implementation in PIC assembly language
BTFSC PIR1, TMR1IF     ; skip next if (PIR1<TMR1IF> == 1)
GOTO T1_INT            ; go to Timer1 ISR
BTFSC PIR1, ADIF       ; skip next if (PIR1<ADIF> == 1)
GOTO AD_INT            ; go to A/D ISR
BTFSC PIR1, LCDIF      ; skip next if (PIR1<LCDIF> == 1)
GOTO LCD_INT           ; go to LCD ISR
BTFSC INTCON, RBIF     ; skip next if (PIR1<RBIF> == 1)
GOTO PORTB_INT         ; go to PortB ISR
GOTO INT_ERROR_LP1     ; default ISR
```

Typical Interrupt Service Routine (ISR) — 2

```
T1_INT                  ; Timer1 overflow routine
   :
   BCF PIR1, TMR1IF     ; Clear Timer1 overflow interrupt flag
   GOTO END_ISR         ; Leave ISR
AD_INT                  ; Routine when A/D completes
   :
   BCF PIR1, ADIF       ; Clear A/D interrupt flag
   GOTO END_ISR         ; Leave ISR
LCD_INT                 ; LCD Frame routine
   :
   BCF PIR1, LCDIF      ; Clear LCD interrupt flag
   GOTO END_ISR         ; Leave ISR
PORTB_INT               ; PortB change routine
   :
END_ISR                 ; Leave ISR
   POP_MACRO            ; Restore registers
   RETFIE               ; Return and enable interrupts
```

Timers

Watchdog timer (WDT)

- Normal program resets timer before timeout (18 ms)
- Timeout
  - Non-sleep mode — MCU resets
  - Sleep mode — MCU wakes up $\rightarrow$ executes instruction following sleep
- Configured in **`OPTION_REG`** SFR

Timer0

- Generic programmable 8-bit timer/counter
- Shares prescaler (divide by 2^k, $k = 0,\ldots,8$) with **`WDT`**
- Configured in **`OPTION_REG`** SFR

Timer1

- Generic programmable 16-bit timer/counter
- Read / write two 8-bit registers

Timer2

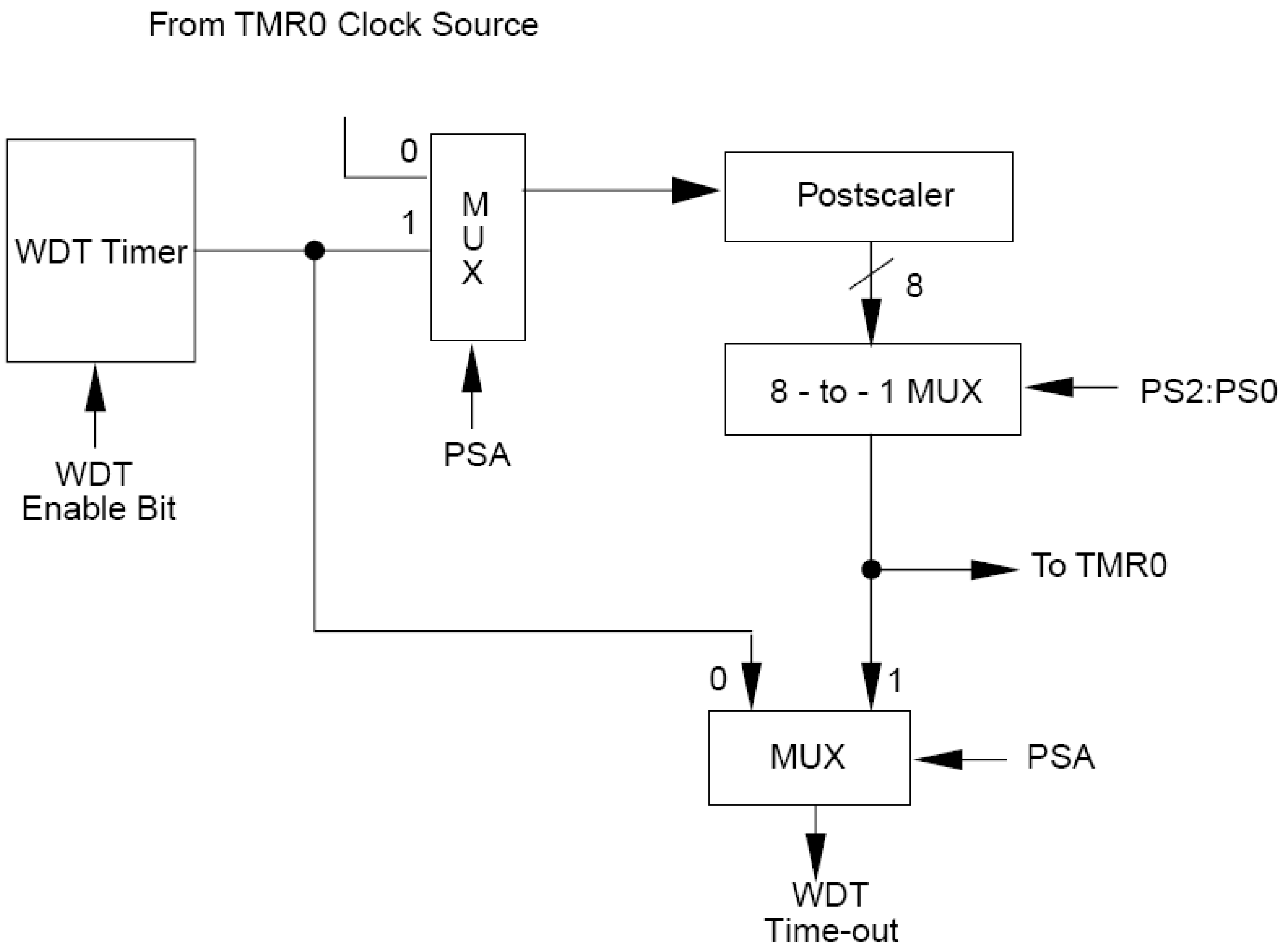
- Generic programmable 8-bit timer/counter
- Time base for PWM mode

Watchdog Timer (WDT)

Time base

Internal RC oscillator

Timer0 clock source



OPTION_REG SFR (File Address 081h)

RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
7	6	5	4	3	2	1	0

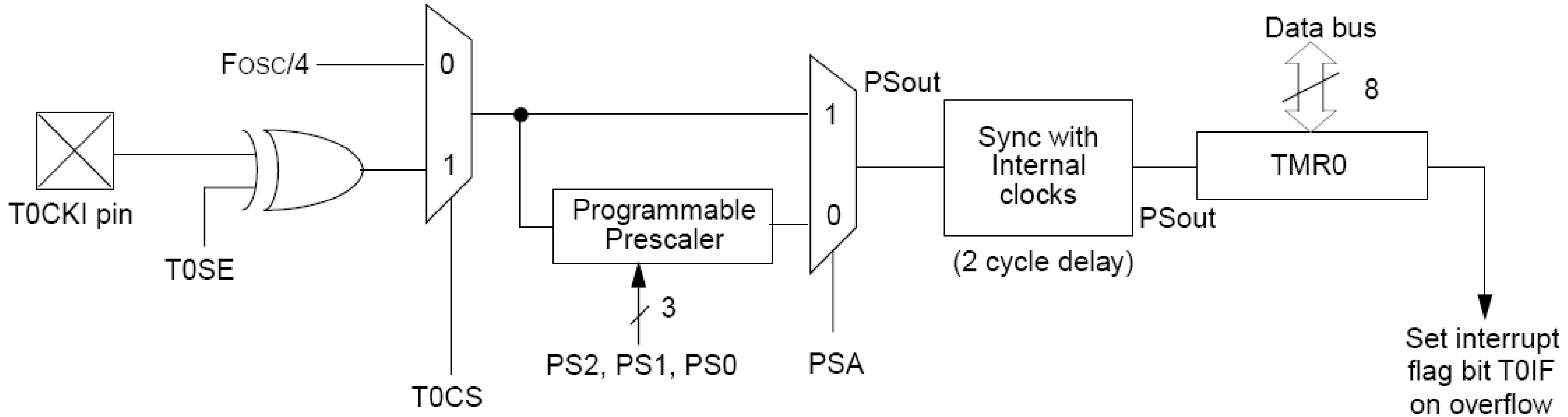
RBPU	**Weak Pull-up Enable**	1 = Weak pull-ups are disabled 0 = Weak pull-ups are enabled by port latch values **Underline** — active = 0 / inactive = 1
INTEDG	**Interrupt Edge Select**	1 = Interrupt on rising edge of INT pin 0 = Interrupt on falling edge of INT pin
T0CS	**TMR0 Clock Source Select**	1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKOUT)
T0SE	**TMR0 Source Edge Select**	1 = Increment on high-to-low transition on T0CKI pin 0 = Increment on low-to-high transition on T0CKI pin
PSA	**Prescaler Assignment**	1 = Prescaler is assigned to the WDT (watchdog) 0 = Prescaler is assigned to the Timer0 module
PS2:PS0	**Prescaler Rate Select**	see below

PS2:PS0	TMR0	WDT	PS2:PS0	TMR0	WDT
000	1:2	1:1	100	1:32	1:16
001	1:4	1:2	101	1:64	1:32
010	1:8	1:4	110	1:128	1:64
011	1:16	1:8	111	1:256	1:128

Timer0

8-bit timer / counter

- Readable / writable at **TMR0** SFR (File Address **081h**)
- 8-bit software programmable prescaler
  - Divide input pulse train (slows time scale)
  - Scale by 1:1 , 1:2, 1:4, ... , 1:128
- Selectable clock source
  - External / internal
- Interrupt on overflow **FFh** → **00h**
- Edge select (phase synchronization with external clock)



Timer0 Operation

Timer mode

`T0CS = 0`

`TMR0++` on every instruction cycle (without prescaler)

Write to **`TMR0`** register ⇒ no increment for two instruction cycles

Counter mode

`T0CS = 1`

`TMR0++` on every rising or falling edge of **`T0CKI`** (external clock)

- Edge determined by **`T0SE`** bit

Prescaler

Set by PSA control bits

TMR0 Interrupt

Generated on overflow **`FFh`** $\rightarrow$ **`00h`**

Sets bit **`T0IF (INTCON<2>)`**

Timer0 interrupt service routine

- Clear **`T0IF`**
- Re-enable interrupt

Initialize Timer0 with Internal Clock Source

```
   CLRF TMR0              ; Clear Timer0 register
   CLRF INTCON            ; Disable interrupts and clear T0IF
   BSF STATUS, RP0        ; Bank1
   MOVLW 0xC3             ; Disable PortB pull-ups
                          ; C3 = 11000011
   MOVWF OPTION_REG       ; Interrupt on rising edge of RB0
                          ; Timer0 increment from internal clock
                          ; Prescale = 1:16.
   BCF STATUS, RP0        ; Bank0
 T0_OVFL_WAIT
   BTFSS INTCON, T0IF            ; poll overflow bit
   GOTO T0_OVFL_WAIT             ; on timer overflow
```

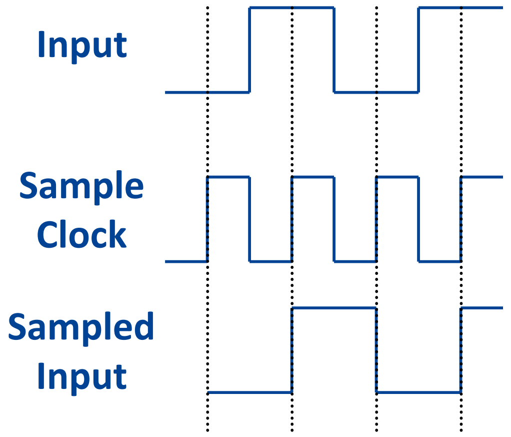
Timer1

16-bit timer/counter

- **`TMR1`** pair **`TMR1H:TMR1L`**
  - Readable and writable 8-bit registers
- Counter **`0000h`** to **`FFFFh`** with rollover to **`0000h`**
  - Generate Timer1 interrupt on rollover (if enabled)

Modes

- Synchronous timer
  - **`TMR1++`** on every instruction cycle (F_{OSC} / 4)
- Asynchronous counter
  - **`TMR1++`** on rising edge of input pin
- Synchronous counter
  - **`TMR1++`** on rising edge of sampled input pin
  - Synchronized to internal clock T_{OSC}
    - Sample input pin on rising edge of T_{OSC}
    - Input must be high / low for at least $2T_{OSC}$

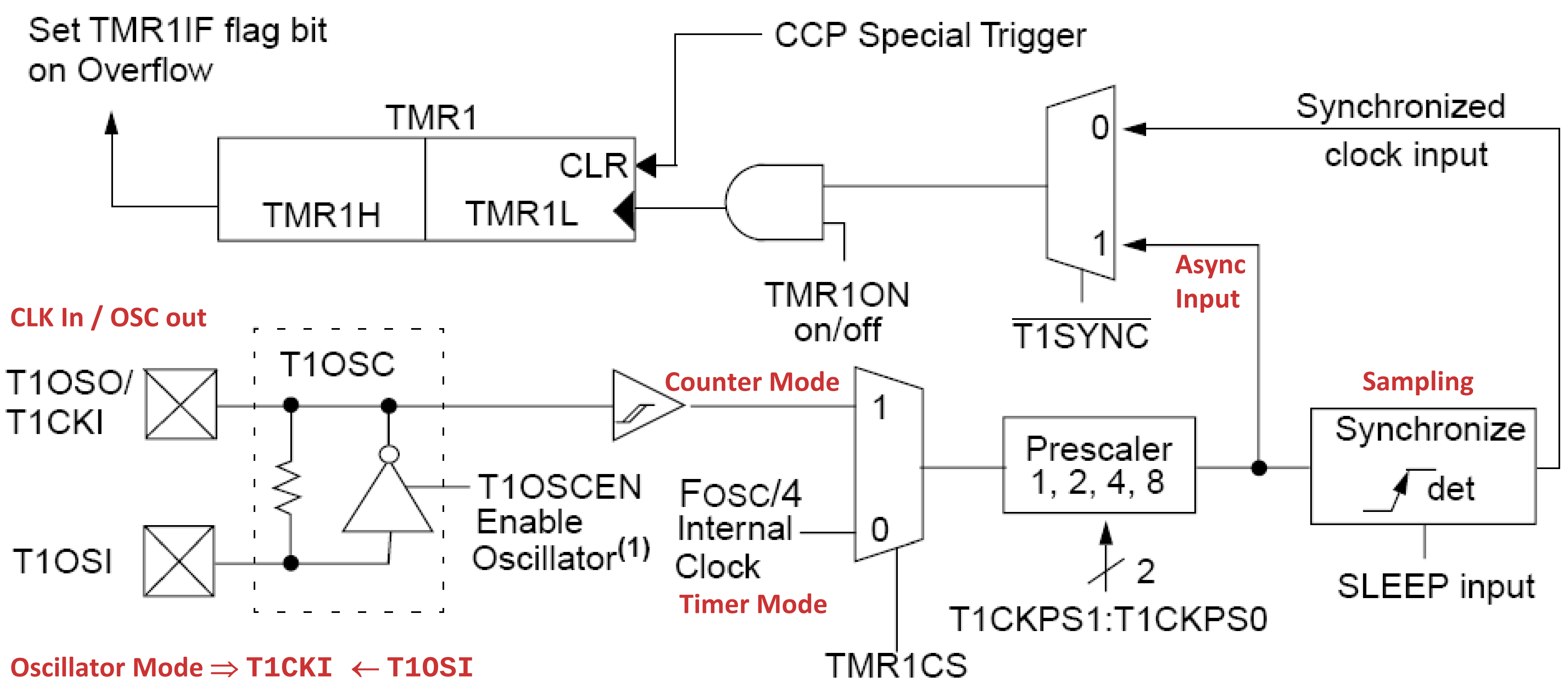


T1CON SFR

▨	▨	T1CKPS1	T1CKPS0	T1OSCEN	<u>T1SYNC</u>	TMR1CS	TMR1ON
7	6	5	4	3	2	1	0

T1CKPS1 **T1CKPS0**	**Timer1 Input Clock Prescale Select**	11 = 1:8 Prescale value 10 = 1:4 Prescale value	01 = 1:2 Prescale value 00 = 1:1 Prescale value
T1OSCEN	**Timer1 Oscillator Enable**	1 = Oscillator mode enabled 0 = Oscillator mode disabled	
<u>T1SYNC</u>	**Timer1 External Clock Input Synchronization Select**	**TMR1CS = 1** (Counter Mode) 1 = Asynchronous Counter Mode 0 = Synchronous Counter Mode **TMR1CS = 0** (Timer Mode) Ignored	
TMR1CS	**Timer1 Clock Source Select**	1 = Counter Mode (count external clock) 0 = Timer Mode (count internal clock F_{OSC} / 4)	
TMR1ON	**Timer1 On**	1 = Enables Timer1 0 = Stops Timer1	

Timer1 Operation



Reading Timer1

```
; All interrupts disabled
    MOVF TMR1H, W   ; W ← high byte
    MOVWF TMPH      ; TMPH ← W
    MOVF TMR1L, W   ; W ← low byte
    MOVWF TMPL      ; TMPL ← W
; TMR1L can roll-over between reads of high and low bytes
    MOVF TMR1H, W   ; W ← high byte again
    SUBWF TMPH, W   ; Verify high byte
    BTFSC STATUS,Z  ; bad read (Z = 0 ⇒ not equal) ⇒ re-do
    GOTO CONTINUE
; New reading ⇒ good value.
    MOVF TMR1H, W   ; W ← high byte
    MOVWF TMPH      ; TMPH ← W
    MOVF TMR1L, W   ; W ← low byte
    MOVWF TMPL      ; TMPL ← W
    ; Re-enable interrupts (if required)
CONTINUE
    ; Continue
```

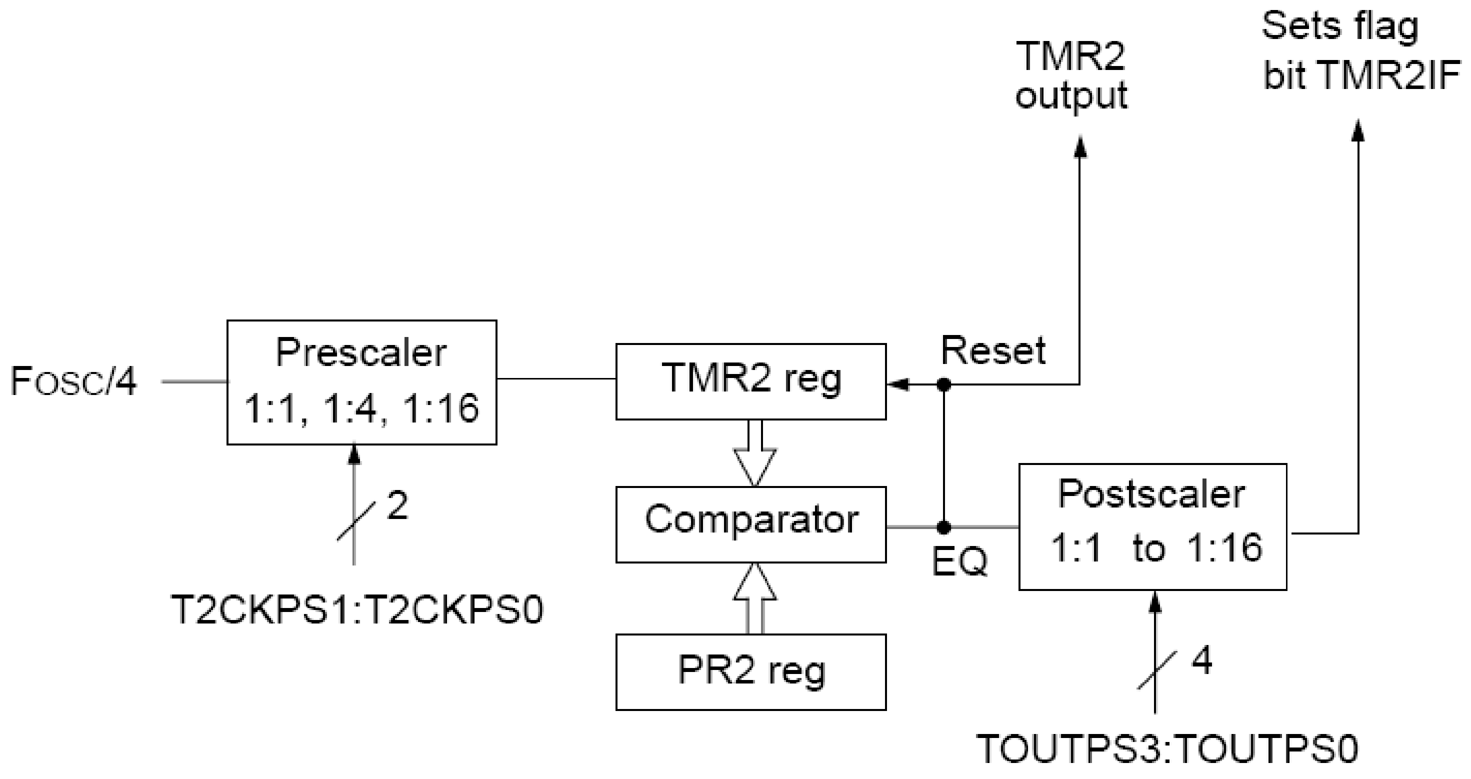
Writing Timer1

```
; All interrupts are disabled
    CLRF TMR1L              ; Clear Low byte
                            ; Prevents rollover to TMR1H
    MOVLW HI_BYTE           ; W ← HI_BYTE
    MOVWF TMR1H, F          ; TMR1H ← W
    MOVLW LO_BYTE           ; W ← LO_BYTE
    MOVWF TMR1L, F          ; TMR1L ← W
; Re-enable interrupts (if required)
CONTINUE
; Continue
```

Timer2

Readable / writable 8-bit timer

- Prescaler
- Period register **PR2**
  - Readable / writable
  - **`TMR2 = PR2`** $\Rightarrow$ reset (**`TMR2`** $\leftarrow$ **`0`**)
- Postscaler



T2CON SFR

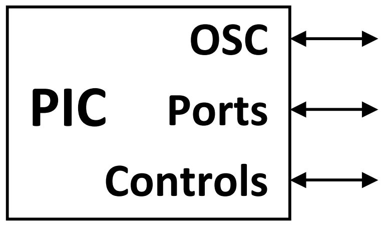
	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
7	6	5	4	3	2	1	0

TOUTPS3:0	**Timer2 Output Postscale Select**	0000 = 1:1 Postscale 0001 = 1:2 Postscale 0010 = 1:3 Postscale 0011 = 1:4 Postscale : 1111 = 1:16 Postscale
TMR2ON	**Timer2 On**	1 = Timer2 is on 0 = Timer2 is off
T2CKPS1:0	**Timer2 Clock Prescale Select**	00 = Prescaler is 1 01 = Prescaler is 4 1x = Prescaler is 16

Ports

I / O pins

Electrical connections to external circuits



Configurable in SFR ADCON1 as

Digital I/O

Analog inputs to A/D converter

Mid-Range PIC configurations

Minimal — Port A (6 pins) + Port B (8 pins)

Maximal — Port A (6 pins), Port B (8 pins), ... , Port G (8 pins)

Special Function Registers

Data

```
PORTA , ... , PORTG
PORTi<x> = data bit on pin x of port i
```

Direction

```
TRISA, ... , TRISG
TRISi<x> = 1 ⇒ pin x of port i is Input
TRISi<x> = 0 ⇒ pin x of port i is Output
```

Port Access

Output

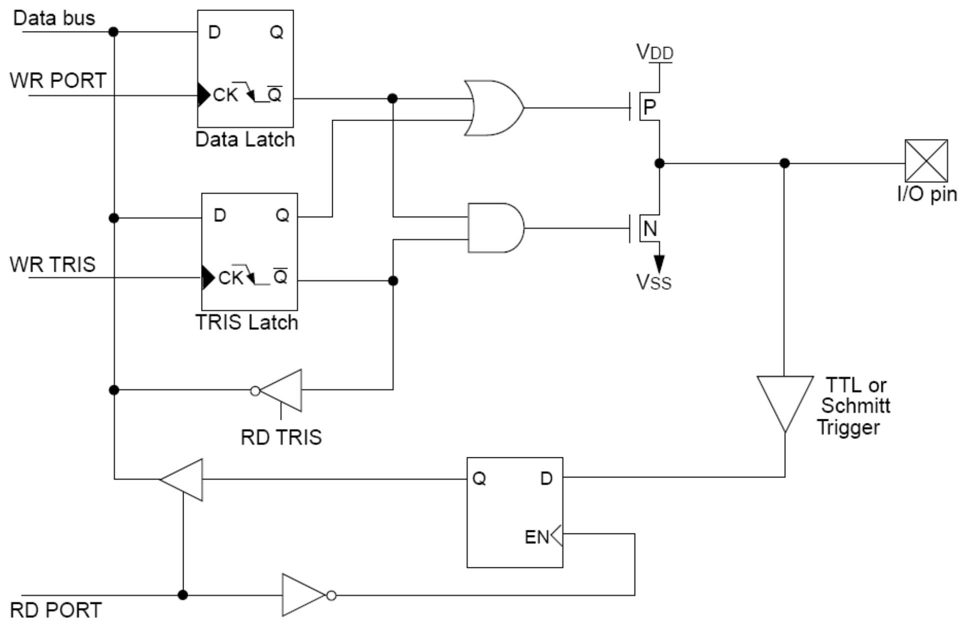
Set **`TRISi<x> = 0`**

Write data bit to **`PORTi<x>`**

Input

Set **`TRISi<x> = 1`**

Read data bit from **`PORTi<x>`**



Order of operations

Read

Reads levels on physical I/O pins (not data register file)

Write

Implemented as read → modify

Causes update of all input data registers from physical I/O pins

Program must read all required inputs before any write

PORTA

5 general purpose I/O pins

- **RA5** and **RA3:RA0**
- Standard electrical behavior
  - TTL input levels and CMOS output drivers

Special input

- **RA4**
- Schmitt trigger input
  - Threshold decision converts input to **`binary (RA4 > threshold)`**
- Open drain output
  - Permits specialize electrical functions on output
  - Wired-OR, analog weighting, ...

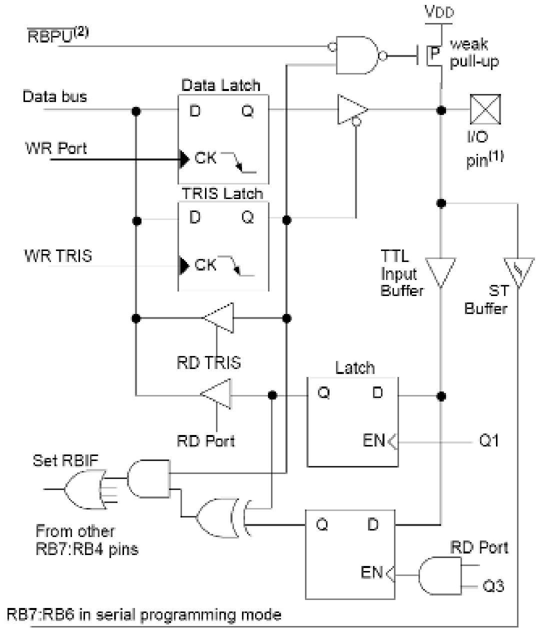
Initializing PORTA

```
CLRF STATUS        ; Bank0
CLRF PORTA         ; Initialize PORTA
BSF STATUS, RP0    ; Select Bank1
MOVLW 0xCF         ; Initialize data directions
                   ; CFh = 11001111
                   ;     = x x Out Out In In In In
MOVWF TRISA        ; PORTA<3:0> = inputs
                   ; PORTA<5:4> = outputs
                   ; TRISA<7:6> always read 0
```

PORTB

8 general purpose I/O pins

- **`RB7:RB0`**
- Standard electrical behavior
  - TTL input levels and CMOS output drivers



Interrupt on change

- Input pins **`RB7:RB4`**
- Inputs compared with previous read of PORTB
- **`OR(compare bits) = 1`** → RB Port Change Interrupt
- Can wake device from SLEEP
  - Example — wake-up on key press
- Clear interrupt
  - Read or write **`PORTB`**
  - Clear flag bit **`RBIF`**

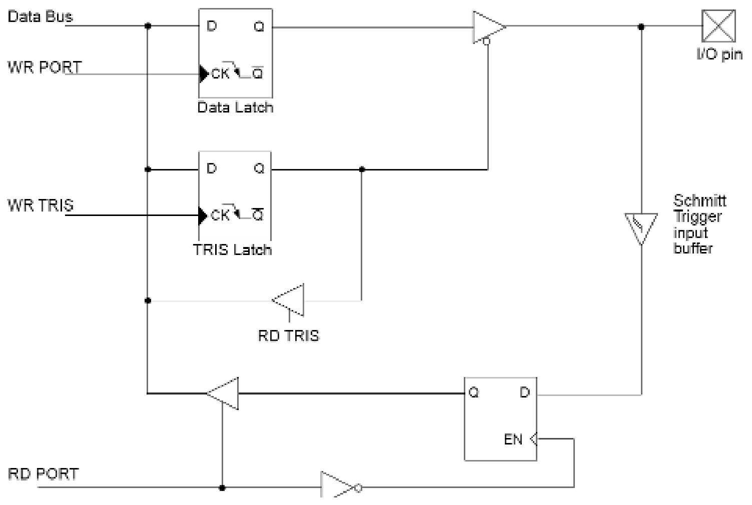
Ports C to G

Ports C to E

8 binary I/O pins

`Ri7:Ri0, i = C, D, E`

Schmitt trigger on each input pin



Ports F and G

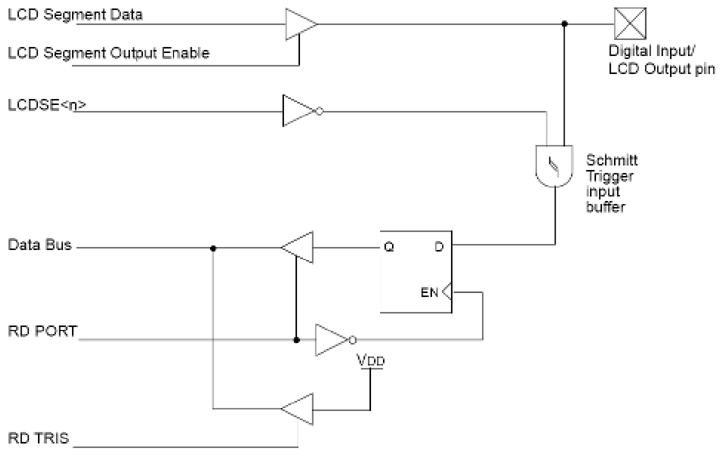
`Ri7:Ri0, i = F, G`

8 binary inputs

Schmitt trigger on each input

8 LCD driver outputs

Direct connection to 7-segment display



LCD Segment Data
LCD Segment Output Enable
Digital Input/ LCD Output pin
LCDSE<n>
Schmitt Trigger input buffer
Data Bus
Q
D
EN
RD PORT
VDD
RD TRIS

Analog-to-Digital (A/D) Converter Module

Converts analog input signals

- Sample and hold
- One of 8 analog inputs (channels)
- Conversion to 8-bit binary number

Analog reference voltage

- Software selectable
  - Device supply voltage
  - Voltage level on V_{REF} pin
- Can operate in sleep mode

Three registers

- A/D Result Register (**`ADRES`**)
- A/D Control Register0 (**`ADCON0`**)
- A/D Control Register1 (**`ADCON1`**)

ADCON0 SFR

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	Resv	ADON
7	6	5	4	3	2	1	0

ADCS1:ADCS0	**A/D Conversion Clock Select** 00 = $f_{OSC}/2$ 01 = $f_{OSC}/8$	10 = $f_{OSC}/32$ 11 = f_{RC} (internal A/D RC osc)
CHS2:CHS0	**Analog Channel Select** 000 = channel 0 (AN0) 001 = channel 1 (AN1) 010 = channel 2 (AN2)	011 = channel 3 (AN3) : 111 = channel 7 (AN7)
GO/$\overline{\text{DONE}}$	**A/D Conversion Status** 1 = in progress	0 = not in progress
Reserved	0	
ADON	**A/D On** 1 = activated	0 = deactivated

ADCON1 SFR

▨	▨	▨	▨	▨	PCFG2	PCFG1	PCFG0
7	6	5	4	3	2	1	0

PCFG2:PCFG0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
000	A	A	A	A	A	A	A	A
001	A	A	A	A	V_{REF}	A	A	A
010	D	D	D	A	A	A	A	A
011	D	D	A	A	V_{REF}	A	A	A
100	D	D	D	D	A	D	A	A
101	D	D	D	D	V_{REF}	D	A	A
11x	D	D	D	D	D	D	D	D

A	Port pin configured for analog input
D	Port pin configured for digital I/O
AN3 = V_{REF}	Conversion compares to reference voltage V_{REF} = voltage on AN3
AN3 = D	Conversion compares to reference voltage V_{REF} = device supply voltage

Operation of A/D Converter

Configure A/D module

- Analog pins + voltage reference + digital I/O in **ADCON1**
- Select A/D input channel (**ADCON0**)
- Select A/D conversion clock (**ADCON0**)
- Activate A/D module (**ADCON0**)

Configure A/D interrupt (optional)

- Clear **ADIF**
- Set **ADIE** + **GIE**

Start conversion

- Set **GO/DONE** bit (**ADCON0**)

Wait for A/D conversion to complete

- Poll **GO/DONE** until cleared or wait for A/D interrupt

Read result

- A/D Result register (**ADRES**)

Repeat

A/D Conversion

```
BSF STATUS, RP0         ; Bank1
CLRF ADCON1             ; Configure inputs as analog
BSF PIE1, ADIE          ; Enable A/D interrupts
BCF STATUS, RP0         ; Bank0
MOVLW 0xC1              ; C1h = 11000001
MOVWF ADCON0            ; Internal RC, A/D active, Channel 0
BCF PIR1, ADIF          ; Clear A/D interrupt flag
BSF INTCON, PEIE        ; Enable peripheral interrupts
BSF INTCON, GIE         ; Enable all interrupts
      ;
      ; Wait required sampling time for selected input
      ;
BSF ADCON0, GO          ; ADCON0<2> ← 1 ⇒ Start conversion
      ; On completion ADIF bit ← 1 and GO/DONE ← 0
```

Comparator

Two analog comparators

Inputs shared with I/O pins

Access via **CMCON** SRF (device-dependent file address)

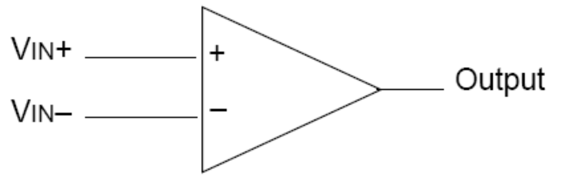
Operation

Analog input at $\mathbf{V_{IN}+ < V_{IN}-} \Rightarrow$ **output = binary 0**

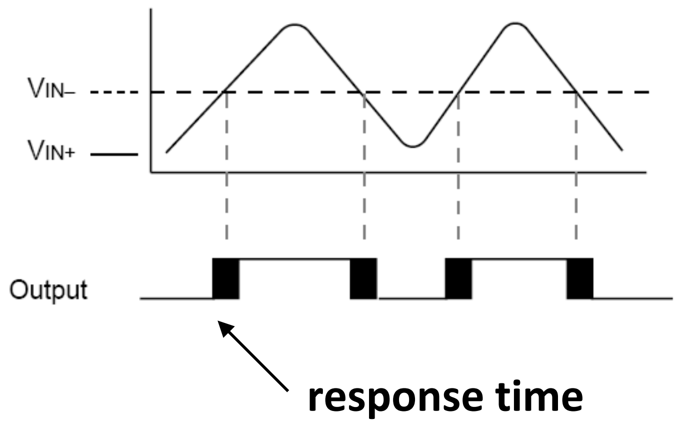
Analog input at $\mathbf{V_{IN}+ > V_{IN}-} \Rightarrow$ **output = binary 1**

CMCON SFR

C2OUT	C1OUT	▨	▨	CIS	CM2	CM1	CM0
7	6	5	4	3	2	1	0

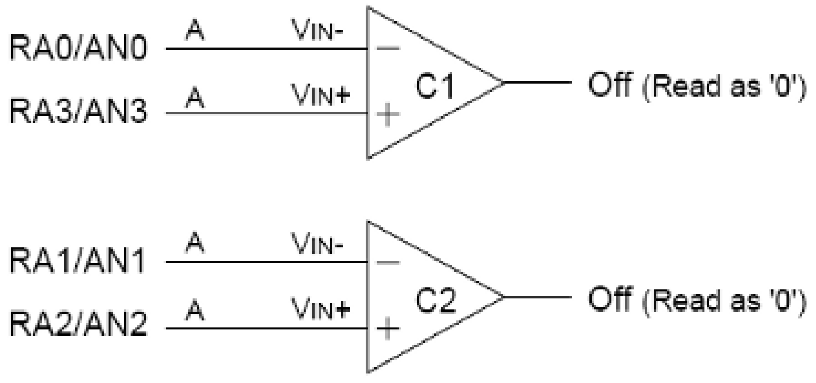


C2OUT **C1OUT**	**Comparator Outputs**	$\mathbf{1 = V_{IN}+ > V_{IN}-}$ $\mathbf{0 = V_{IN}+ > V_{IN}-}$
CIS **CM2:CM0**	**Comparator Input Switch**	**See table on following slides**

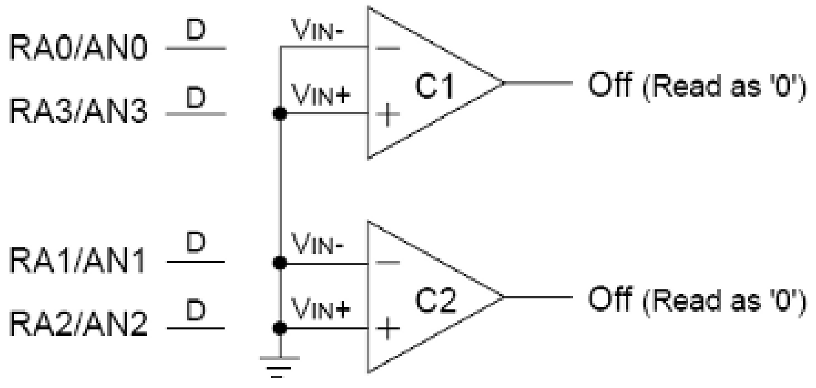


Comparator Modes — 1

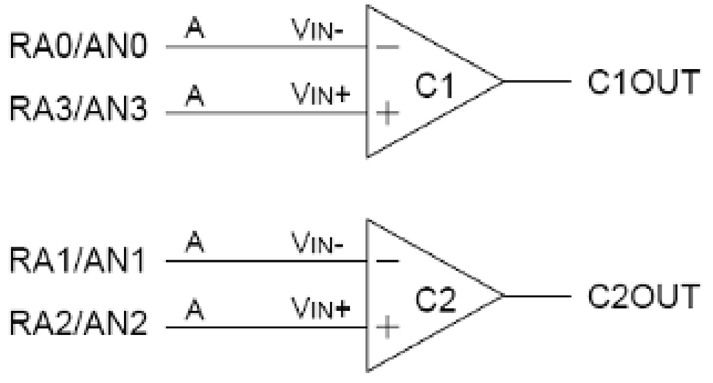
CM2:CM0 = 000
Comparators Reset (POR Default Value)



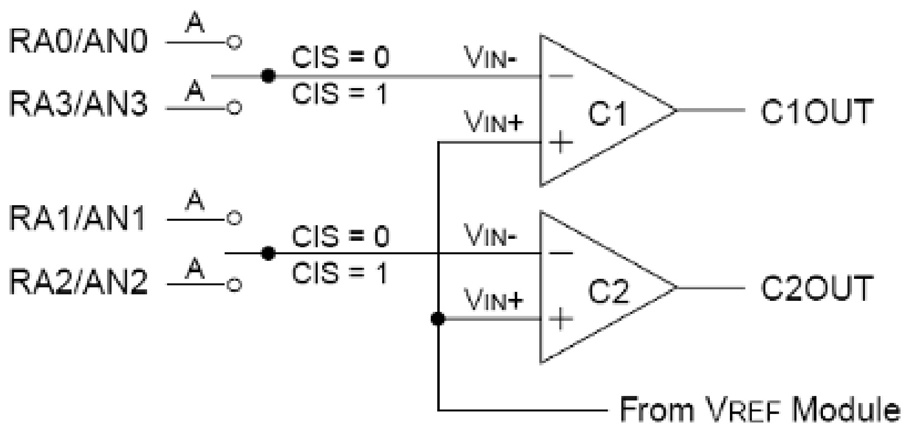
CM2:CM0 = 111
Comparators Off



CM2:CM0 = 100
Two Independent Comparators

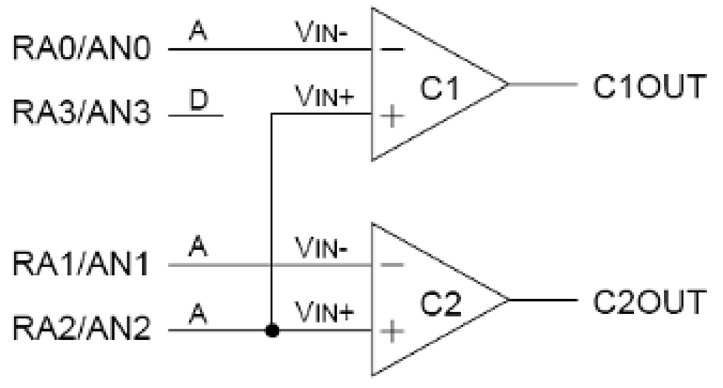


CM2:CM0 = 010
Four Inputs Multiplexed to Two Comparators

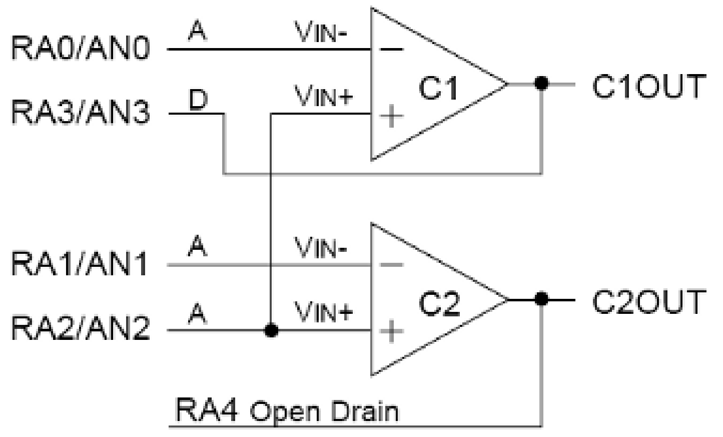


Comparator Modes — 2

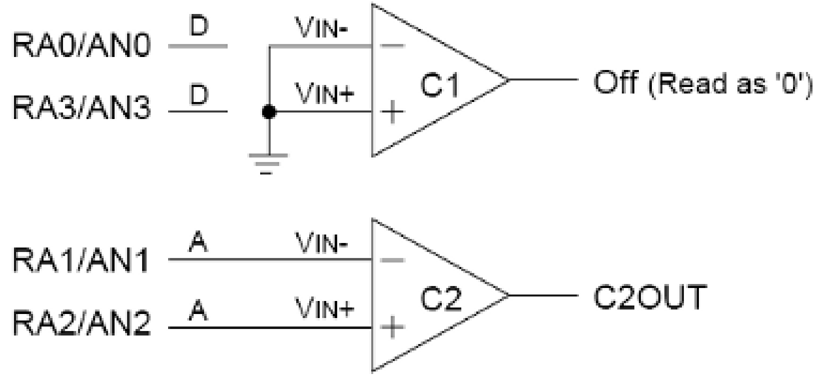
CM2:CM0 = 011
Two Common Reference Comparators



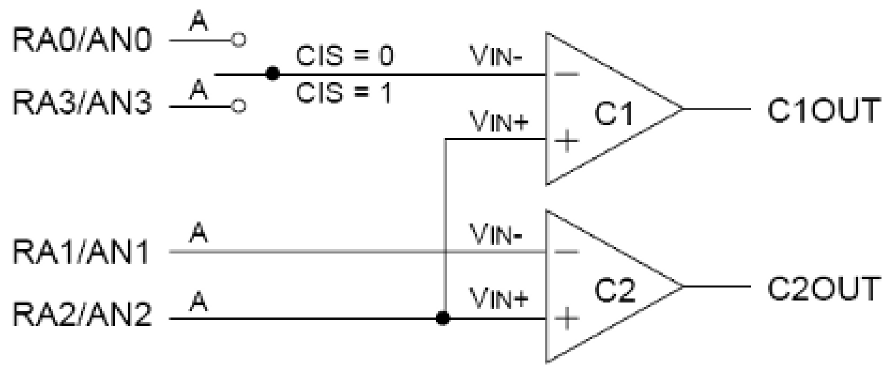
CM2:CM0 = 110
Two Common Reference Comparators with Outputs



CM2:CM0 = 101
One Independent Comparator



CM2:CM0 = 001
Three Inputs Multiplexed to Two Comparators



Initialize Comparator

```
FLAG_REG EQU 0x20                ; FLAG_REG points to address 20h
     CLRF FLAG_REG               ; flag register ← 0
     CLRF PORTA                  ; PORTA ← 0
     ANDLW 0xC0                  ; Mask comparator bits W<5:0> ← 0
     IORWF FLAG_REG,F            ; FLAG_REG ← FLAG_REG OR W
     MOVLW 0x03                  ; Init comparator mode
     MOVWF CMCON                 ; CM<2:0> = 011 (2 common reference)
     BSF STATUS,RP0              ; Bank1
     MOVLW 0x07                  ; Initialize data direction
     MOVWF TRISA                 ; Set RA<2:0> as inputs
                                 ; RA<4:3> as outputs
                                 ; TRISA<7:5> read 0
     BCF STATUS,RP0              ; Bank0
     CALL DELAY 10               ; 10ms delay
     MOVF CMCON,F                ; Read CMCON (enter read mode)
     BCF PIR1,CMIF               ; Clear pending interrupts
     BSF STATUS,RP0              ; Bank1
     BSF PIE1,CMIE               ; Enable comparator interrupts
     BCF STATUS,RP0              ; Bank0
     BSF INTCON,PEIE             ; Enable peripheral interrupts
     BSF INTCON,GIE              ; Global interrupt enable
```

USART

Universal Synchronous / Asynchronous Receiver / Transmitter

- Serial Communications Interface (SCI)
  - PC serial port / modem
- Transmit
  - Parallel $\rightarrow$ serial
  - Data byte as 8 serial bits
- Receive
  - Serial $\rightarrow$ parallel
  - Assemble 8 bits as data byte

Modes

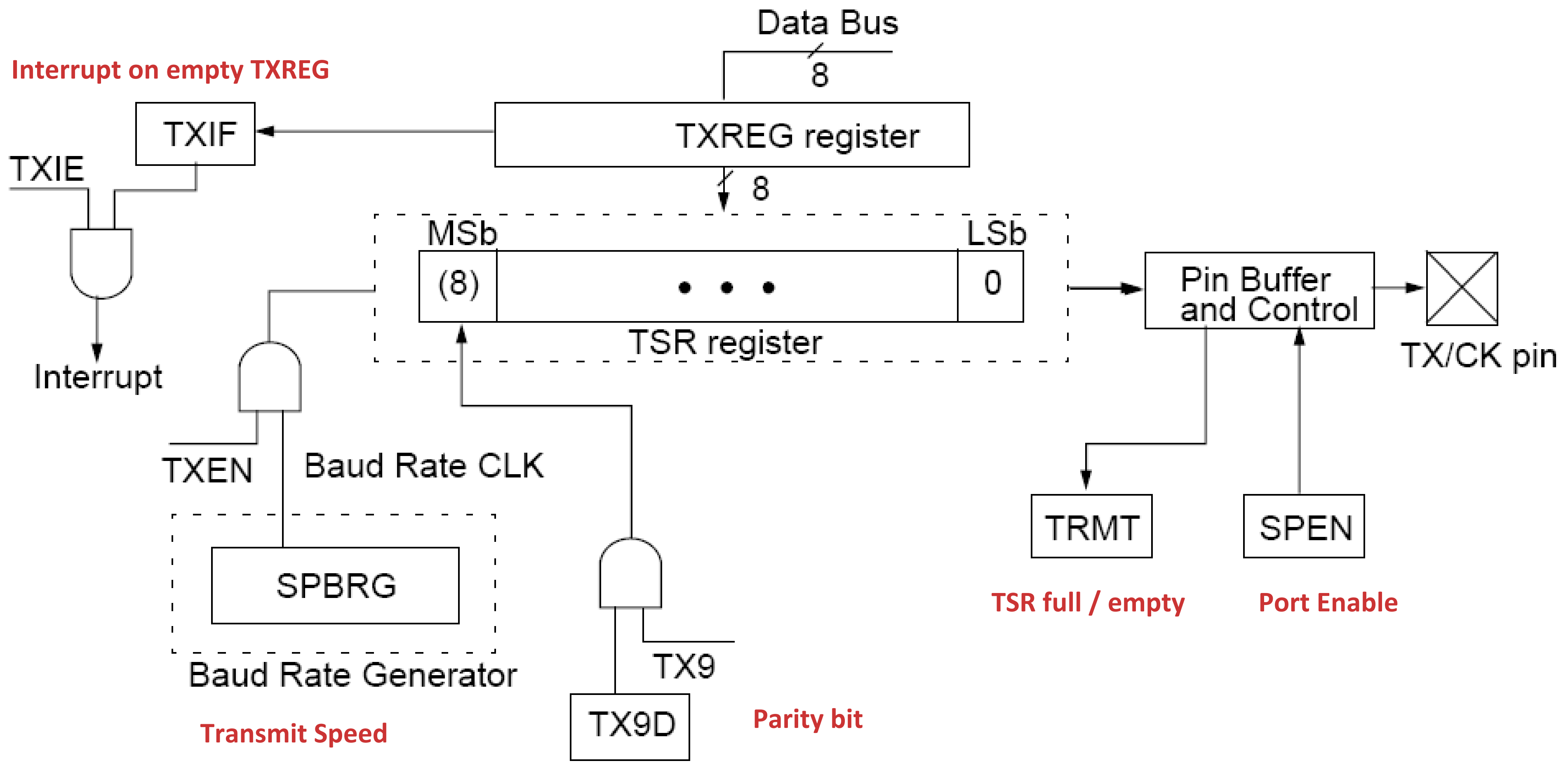
- Asynchronous
  - Full duplex — simultaneous transmit + receive
- Synchronous
  - Half duplex — transmit or receive
  - Master — synchronize data to internal clock
  - Slave — synchronize data to external clock

USART Transmit Operation

Data

Byte → **TXREG** → framing **TSR** → bit **FIFO** → **TX** pin

Framing — add start bit / parity bit



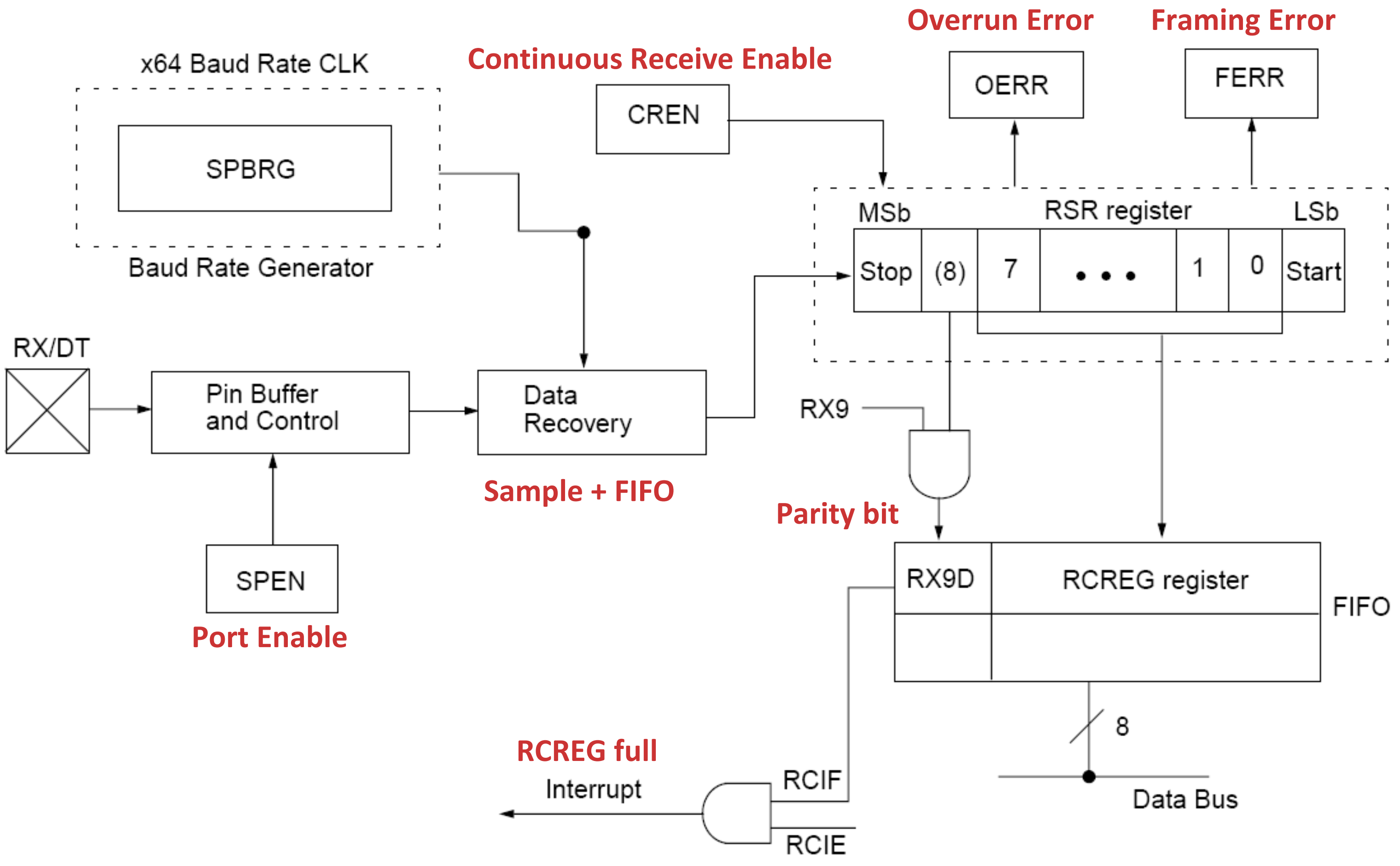
USART Receive Operation

Data

RX pin → bit **`FIFO`** → **RSR** → **RCREG** → byte

Framing

Identify data between stop bits



Capture / Compare / PWM (CCP) Module

SFRs

CCP control register (**CPCON**)

CCPR High byte / Low byte (**CCPRH / CCPRL**)

I/O pin

CPP pin (**CPPx**) — device-dependent pin configured in **TRIS** SFR

Capture Mode

Captures 16-bit value of register **TMR1** on **CCPx** event

Every falling edge / rising edge / 4th rising edge / 16th rising edge

Triggers interrupt

Compare mode

Compare 16-bit **(CCPR == TMR1)**

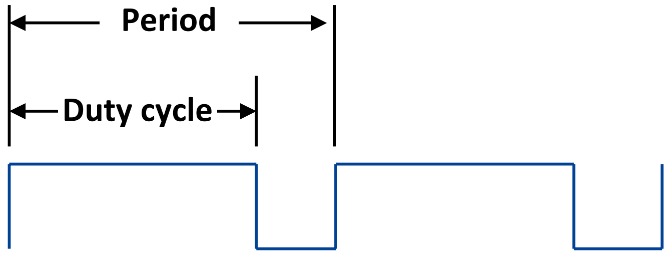
Configurable response on match

CCPx $\leftarrow$ **0 / 1**

Interrupt with no change on **CPPx**

Pulse Width Modulation (PWM) mode

Generates duty cycle waveform on **CCPx**



CCPxCON SFR

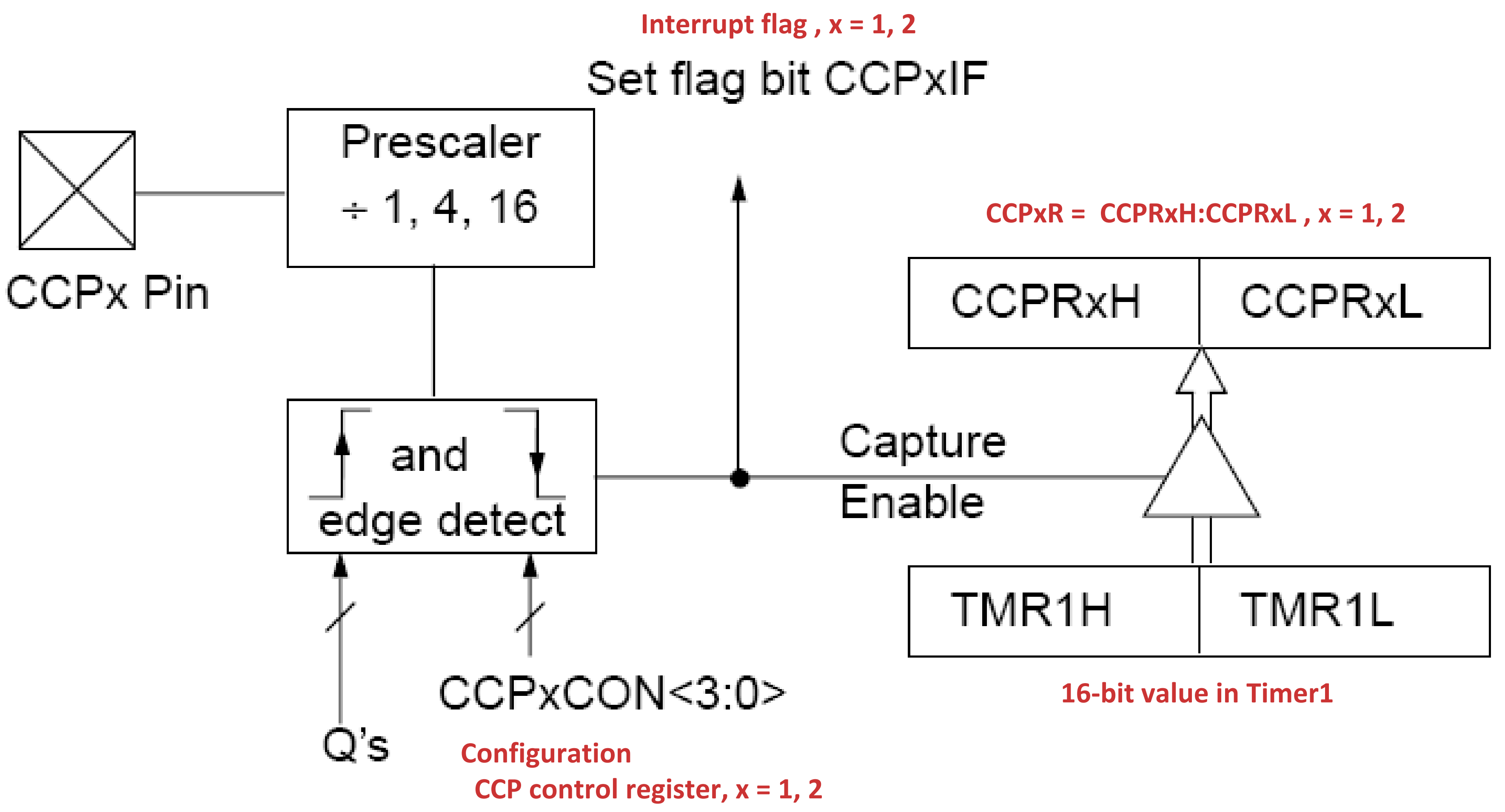
7	6	5	4	3	2	1	0
▨	▨	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0

DCxB1:DCxB0	**PWM Duty Cycle bit1 and bit0**	DCx1:DCx0 of 10-bit PWM duty cycle DCx9:DCx2 in CCPRxL
CCPxM3:CCPxM0	**CCPx Mode Select bits**	0000 = Capture/Compare/PWM off (resets CCPx module) 0100 = Capture mode, every falling edge 0101 = Capture mode, every rising edge 0110 = Capture mode, every 4th rising edge 0111 = Capture mode, every 16th rising edge 1000 = Compare mode, CCP low to high 1001 = Compare mode, CCP high to low 1010 = Compare mode, software interrupt on match 1011 = Compare mode, Trigger special event 11xx = PWM mode

Capture Mode

Event

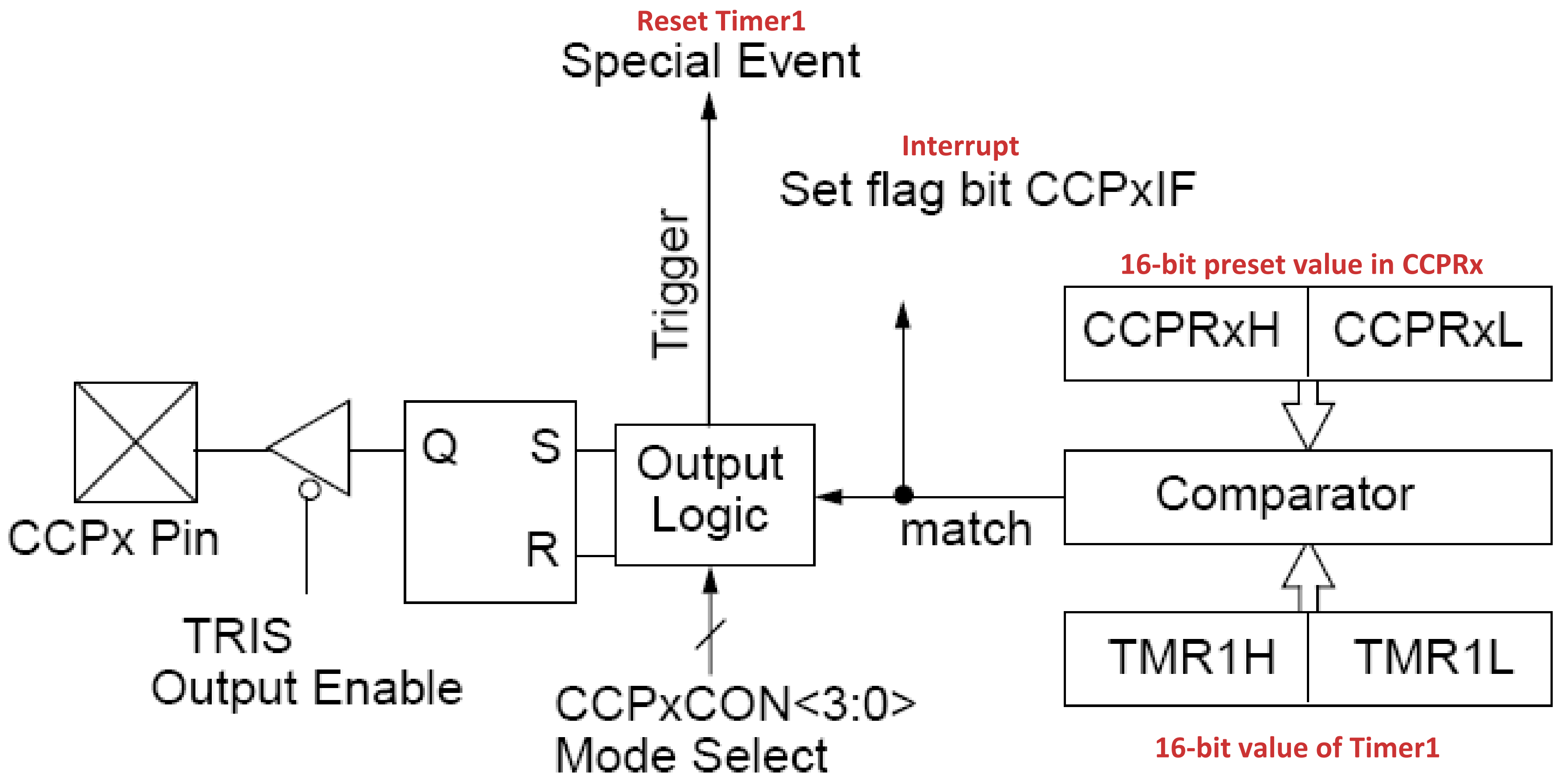
Input pin CCPx divided by prescaler sampled on rising / falling edge



Compare Mode

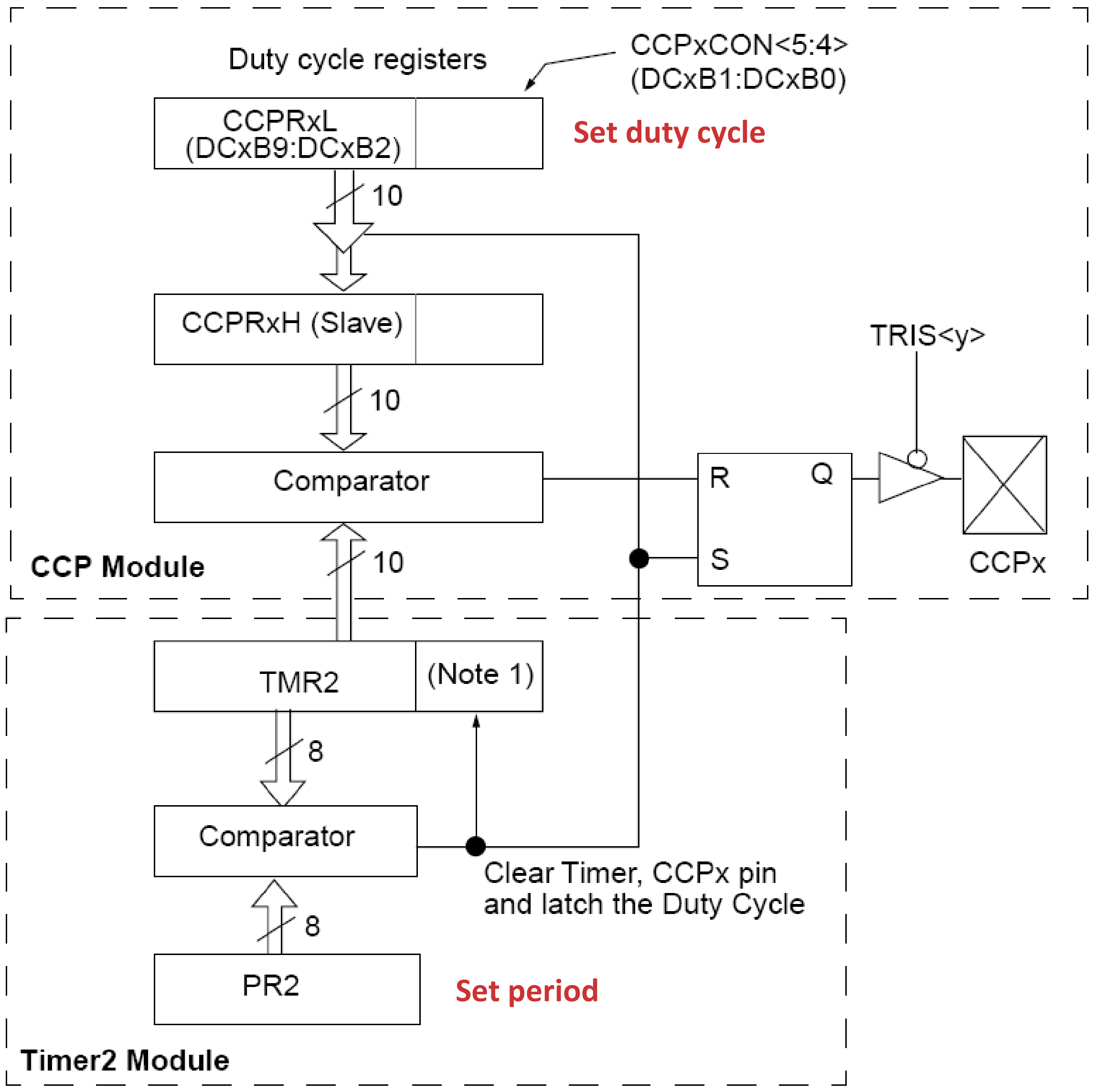
16-bit compare

```
(CCPRx == TMR1), x = 1, 2
```



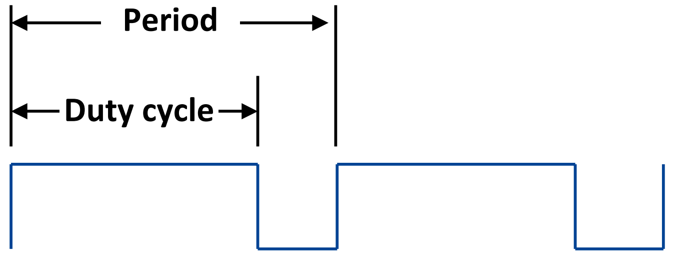
Pulse Width Modulation (PWM) mode

Generates duty cycle waveform



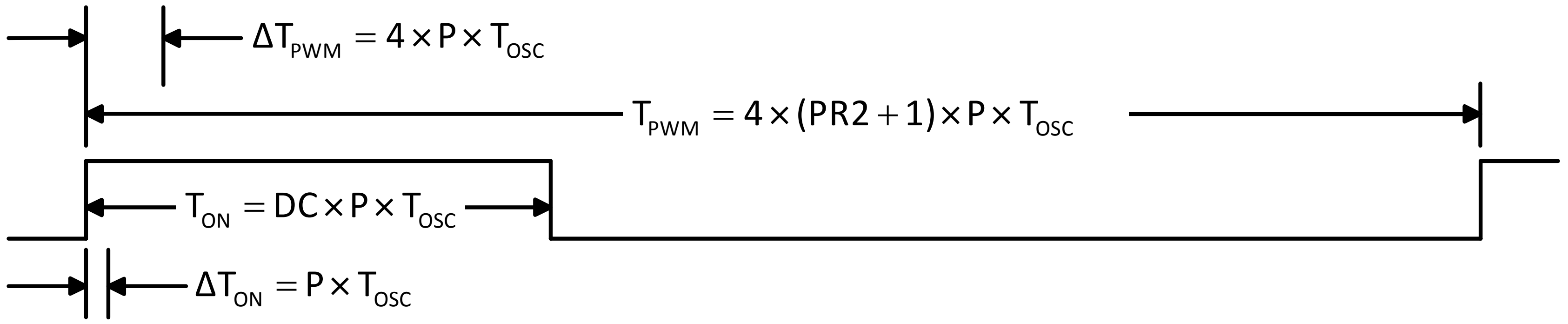
$$\text{Period} = (\text{PR2} + 1) \times 4 \times \text{prescale} \times T_{\text{OSC}}$$

$$\text{Duty cycle} = \text{DC} \times \text{prescale} \times T_{\text{OSC}}$$



$$0 \le \text{PR2} \le 255$$
$$0 \le \text{DC} \le 1023$$

Duty Cycle


$\Delta T_{PWM} = 4 \times P \times T_{OSC}$
$T_{PWM} = 4 \times (PR2 + 1) \times P \times T_{OSC}$
$T_{ON} = DC \times P \times T_{OSC}$
$\Delta T_{ON} = P \times T_{OSC}$

Duty Cycle

$$\frac{T_{ON}}{T_{PWM}} = \frac{DC \times P \times T_{OSC}}{4 \times \left(PR2 + 1\right) \times P \times T_{OSC}} = \frac{DC}{4 \times \left(PR2 + 1\right)}$$

$$\frac{T_{ON}}{T_{PWM}} \leq 1 \Rightarrow DC \leq 4 \times \left(PR2 + 1\right)$$

Controllability

$$\frac{\Delta T_{ON}}{T_{PWM}} = \frac{P \times T_{OSC}}{4 \times \left(PR2 + 1\right) \times P \times T_{OSC}} = \frac{1}{4 \times \left(PR2 + 1\right)}$$

Resolution

$$DC \leq 4 \times \left(PR2 + 1\right) \Rightarrow 2^{r} = \frac{T_{ON}}{T_{OSC}} \leq 4 \times \left(PR2 + 1\right) \times P \Rightarrow r = \frac{\log\left(4 \times \left(PR2 + 1\right) \times P\right)}{\log\left(2\right)}$$

Duty Cycle Example

Frequency and duty cycle from given parameters

$$f_{OSC} = 20\text{ MHz} \Rightarrow T_{OSC} = \left(20\text{ MHz}\right)^{-1} = 50\text{ ns}$$

$$P = 1$$

$$PR2 = 63 \Rightarrow T_{PWM} = 4 \times 64 \times 50\text{ ns} = 12.8\ \mu\text{s}$$

$$f_{PWM} = \left(12.8\ \mu\text{s}\right)^{-1} = 78.125\text{ kHz}$$

$$DC = 32 \Rightarrow T_{ON} = 32 \times 50\text{ ns} = 1.6\text{ ms}$$

$$\frac{T_{ON}}{T_{PWM}} = \frac{32}{4 \times 64} = 0.125 = 12.5\%$$

$$\frac{\Delta T_{ON}}{T_{PWM}} = \frac{1}{4 \times 64} = \frac{1}{256} = 0.00390625$$

$$2^r = \frac{T_{ON}}{T_{OSC}} \le 4 \times 64 \Rightarrow r = \frac{\log\left(256\right)}{\log\left(2\right)} = 8$$

PWM Example

Choosing parameters

Oscillator frequency = 4 MHz
Produce output with
1 kHz frequency (T_{PWM} = 1 ms)
10% duty cycle

Internal oscillator

$f_{OSC} = 4\ \text{MHz} \Rightarrow T_{OSC} = 0.25\ \mu\text{s}$

PWM frequency

$T_{PWM} = 1\ \text{ms} = 4 \times (PR2 + 1) \times P \times 0.25\ \mu\text{s} = (PR2 + 1) \times P \times 1\ \mu\text{s}$

Require $(PR2 + 1) \times P = 1000$

Preset and PR2

$P = 4 \Rightarrow PR2 + 1 = 250 \Rightarrow PR2 = 249 = \text{0xF9}$

$\Delta T_{ON} = P \times T_{OSC} = 1\ \mu\text{s}$

Duty cycle = 10 %

$T_{ON} = 0.10 \times 1\ \text{ms} = 100\ \mu\text{s}$

$DC = 0.10 \times 4 \times (PR2 + 1) = 100 = \text{0x064} \Rightarrow DCH = \text{0x19} = 25 \quad DCL = 0$

PWM Example

Code

```
            List p = 16F873
            include "P16F873.INC"
Init_pwm:   movlw 0x01             ; Stop Timer2
            movwf T2CON            ; Prescaler ← 4
            clrf CCP1CON           ; Reset module CCP1
            clrf TMR2              ; Timer2 ← 0
            movlw .25              ; 10% duty cycle
            movwf CCPR1L           ; DC1B9:DC1B2
            bsf STATUS, RP0        ; Bank 1
            movlw .249             ; Timer2
            movwf PR2
            bcf PIE1, TMR2IE       ; Disable Timer2 interrupt
            bcf PIE1, CCP1IE       ; Disable CCP1 interrupt
            bcf TRISC, 2           ; Pin CCP1 = output
            bcf STATUS, RP0        ; Bank 0
            clrf PIR1              ; Clear interrupt flags
            movlw 0x0C             ; CCP1 in PWM mode
            movwf CCP1CON          ; DC1B1:DC1B0 ← 0
            bsf T2CON, TMR2ON      ; Start Timer2
            return
TON_pwm:    movwf CCPR1L           ; Call to change
            return                 ; Duty cycle
            end
```

Data EEPROM

Additional long term data memory

- Internal EEPROM

Indirect addressing

- Not directly mapped in register file space
- Access through SFRs
  - **EECON1**
    - Control bits
  - **EECON2**
    - Initiates read / write operation
    - Virtual register — not physically implemented
  - **EEDATA**
    - 8-bit data for read / write
  - **EEADR**
    - Access address in EEPROM
    - 8-bit address $\Rightarrow$ 256 EEPROM locations

EECON1 SFR

▨	▨	▨	EEIF	WRERR	WREN	WR	RD
7	6	5	4	3	2	1	0

EEIF	**Write Operation Interrupt Flag**	1 = Write operation completed 0 = Write operation not complete / not started
WRERR	**Error Flag**	1 = Write operation prematurely terminated 0 = Write operation completed
WREN	**Write Enable**	1 = Allows write cycles 0 = Inhibits write to data EEPROM
WR	**Write Control**	1 = Initiates write cycle 0 = Write cycle to data EEPROM is complete (cleared by hardware)
RD	**Read Control**	1 = Initiates read 0 = Does not initiate an EEPROM read (cleared by hardware)

EECON2 SFR

Not physical register

Read **`EECON2`** $\rightarrow$ **`0`**

SFR access to EEPROM write hardware

Data EEPROM write sequence

```
EEDATA ← data
EEADR ← address_for_write
W ← 55h
EECON2 ← W
W ← AAh
EECON2 ← W                ; EECON2 ← 55AAh
EECON1<WR> ← 1            ; initiate (write control bit set)
```

EEPROM Read / Write / Verify — 1

Read

```
BCF STATUS, RP0          ; Bank0
MOVLW CONFIG_ADDR        ; Address in Data EEPROM
MOVWF EEADR              ; Set read address
BSF STATUS, RP0          ; Set Bank1
BSF EECON1, RD           ; Initiate EEPROM Read
BCF STATUS, RP0          ; Set Bank0
MOVF EEDATA, W           ; W ← EEDATA
```

Write

```
BSF STATUS, RP0          ; Bank1
BCF INTCON, GIE          ; Disable INTs
BSF EECON1, WREN         ; Enable write
MOVLW 55h                ; W ← 55h
MOVWF EECON2             ; EECON2 ← W
MOVLW AAh                ; W ← AAh
MOVWF EECON2             ; EECON2 ← W
BSF EECON1,WR            ; Set WR bit (initiates write)
BSF INTCON, GIE          ; Enable INTs
```

EEPROM Read / Write / Verify — 2

Verify

```
    BCF STATUS, RP0          ; Bank0
    MOVF EEDATA, W           ; copy write request data to W
    BSF STATUS, RP0          ; Bank1
  READ
    BSF EECON1, RD           ; Initiate read
    BCF STATUS, RP0          ; Bank0
    SUBWF EEDATA, W          ; W ← write request – read
    BTFSS STATUS, Z          ; Skip next if (Z == 1)
    GOTO WRITE_ERR           ; Handle write error
```

PIC Configuration Bits — 1

Determines certain device modes

- Oscillator mode, WDT reset, copy protection

Sets device state on power-up

- Configured during EEPROM programming
  - Mapped to program memory location 2007h
- Not accessible at run time

CP1:CP0	Code Protection	11 = Code protection off 10 = device dependent 01 = device dependent 00 = memory code protected
DP	Data EEPROM Code Protection	1 = Code protection off 0 = Data EEPROM Memory code protected
BODEN	Brown-out Reset Enable	1 = BOR enabled 0 = BOR disabled
$\underline{\mathrm{PWRTE}}$	Power-up Timer Enable	1 = $\underline{\mathrm{PWRT}}$ disabled 0 = $\underline{\mathrm{PWRT}}$ enabled
$\underline{\mathrm{MCLRE}}$	$\underline{\mathrm{MCLR}}$ (master clear) Pin Function	1 = Pin function = $\underline{\mathrm{MCLR}}$ 0 = Pin function = digital I/O

PIC Configuration Bits — 2

`WDTE`	Watchdog Timer Enable	1 = WDT enabled 0 = WDT disabled
`FOSC1:FOSC0`	Oscillator Selection For devices with no internal RC	11 = RC oscillator 10 = HS oscillator 01 = XT oscillator 00 = LP oscillator
`FOSC2:FOSC0`	Oscillator Selection For devices with internal RC	111 = EXTRC oscillator, with CLKOUT 110 = EXTRC oscillator 101 = INTRC oscillator, with CLKOUT 100 = INTRC oscillator 011 = Reserved 010 = HS oscillator 001 = XT oscillator 000 = LP oscillator