TECNOLÓGICO DE MONTERREY

Subject 7
THE BASIC USE OF TIMERS

Clock

- The CPU of the microcontroller is a sequential digital system and requires a time base to transition between states (fetch and execute)
- The time base is provided by a clock
- The PIC18 has a very complete module and allows the use of several clock sources

Clock Module

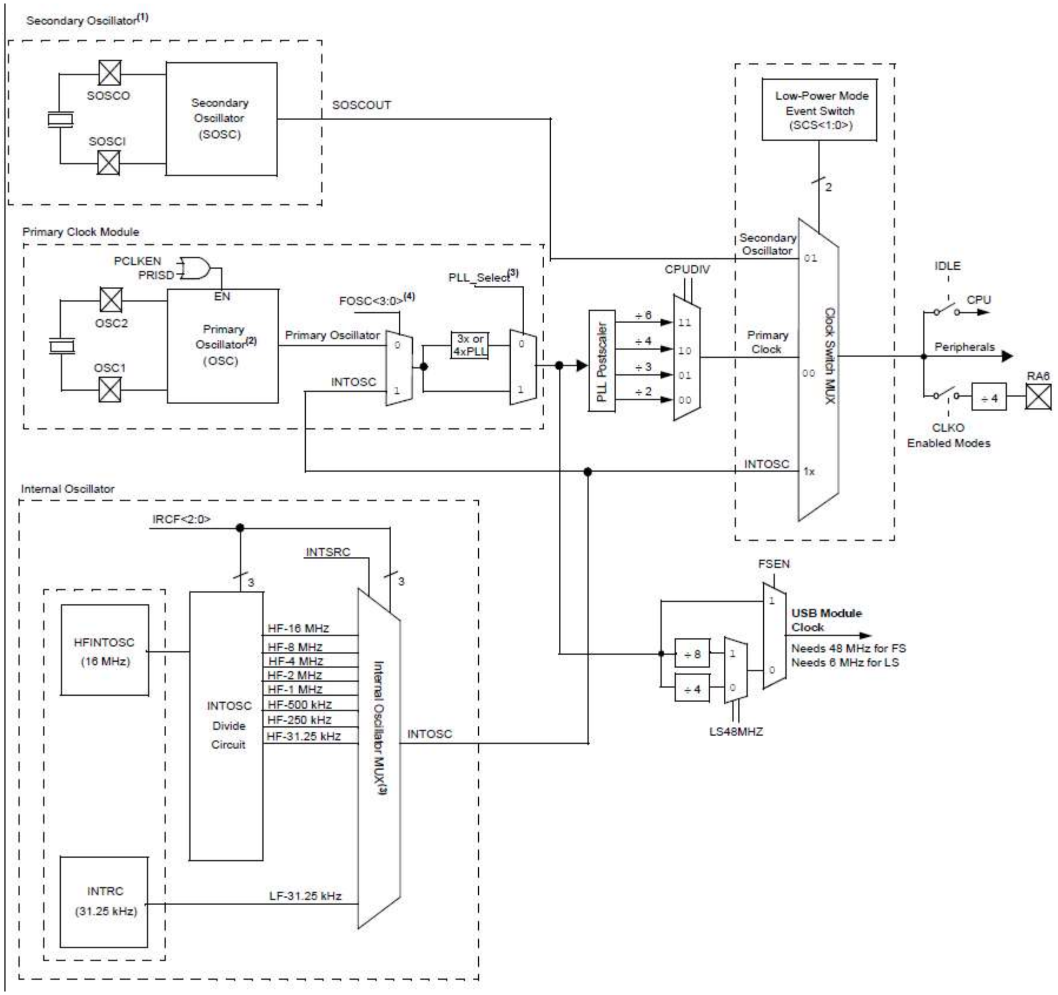
- The clock module allows:
  - Selection in run-time via firmware of multiple clock sources
  - Frequency multiplication using a PLL
  - Oscillator failure detection and recovery (FSCM)
  - Oscillator startup module (OST) that mantains the microcontroller on halt until the oscillator is stable

Oscillators

- The alternative for the oscillators are:

1.	RC	External Resistor/Capacitor
2.	LP	Low-Power Crystal
3.	XT	Crystal/Resonator
4.	INTOSC	Internal Oscillator
5.	HS	High-Speed Crystal/Resonator
6.	EC	External Clock

Oscillators



Flexible Oscillator Structure:

- 3x and 4xPLL Clock Multipliers
- Two External Clock modes, Up to 48 MHz (12 MIPS)
- Internal 31 kHz Oscillator
- Internal Oscillator, 31 kHz to 16 MHz
  - Factory calibrated to ± 1%
  - Self-tune to ± 0.20% max. from USB or secondary oscillator
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Primary Oscillator

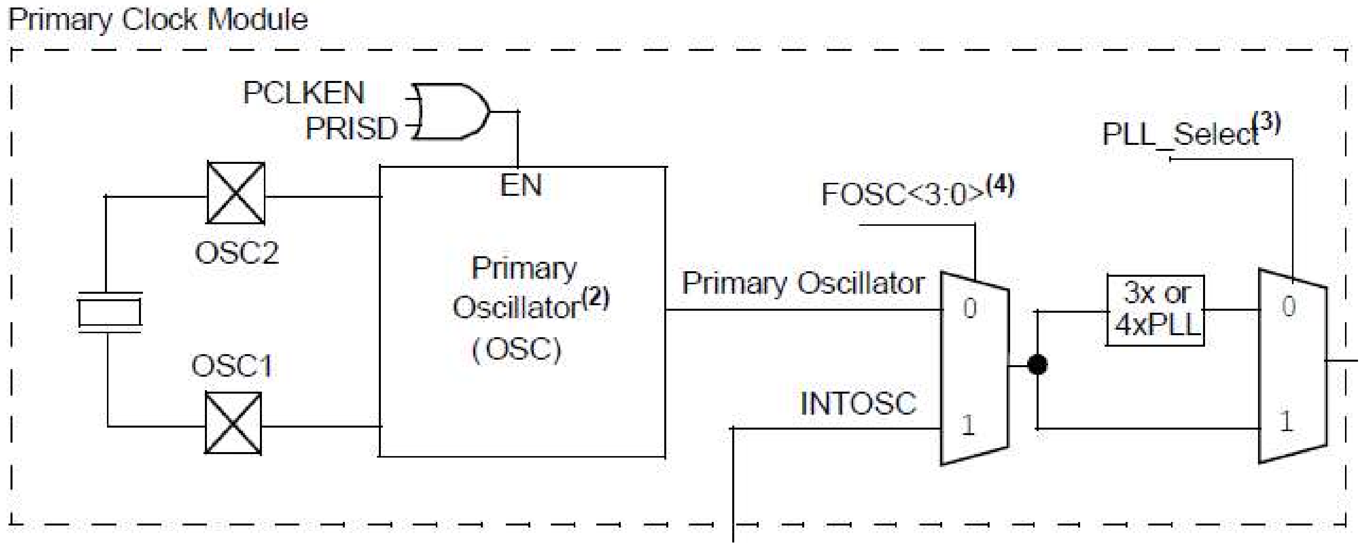
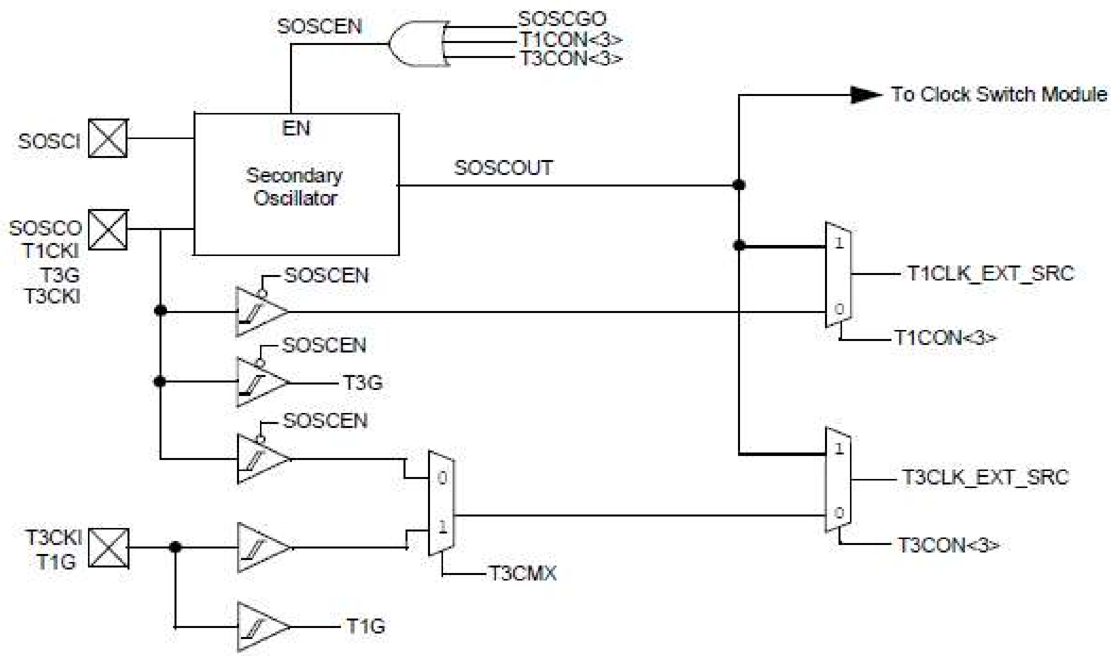
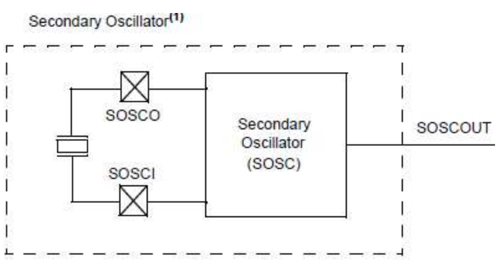


TABLE 3-1: PLL_SELECT TRUTH TABLE

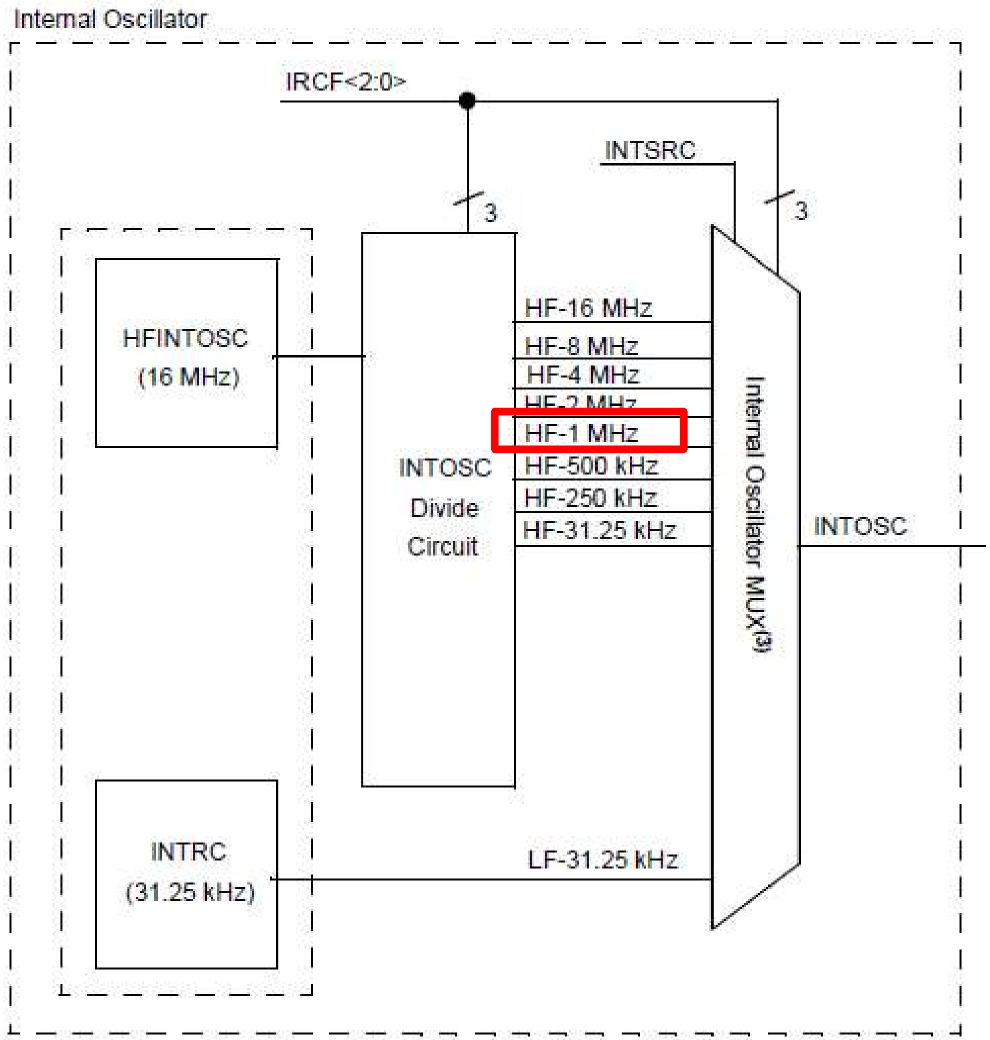
Primary Clock MUX Source	FOSC<3:0>	CFGPLLEN	PLLSEL	PLLEN	SPLLMULT	PLL_Select
External Clock (ECHIO/ECHCLKO)	010x	1	1	x	x	3xPLL[1]
			0	x	x	4xPLL[2]
HS Crystal (HSH)	0010	0	x	1	1	3xPLL[1]
					0	4xPLL[2]
INTOSC (INTOSCIO, INTOSCCLKO)	100x			0	x	OFF
Fosc (all other modes)	xxxx	x	x	x	x	OFF

Secondary Oscillator





The internal oscillator



External sources

FIGURE 3-4: EXTERNAL CLOCK (EC) MODE OPERATION

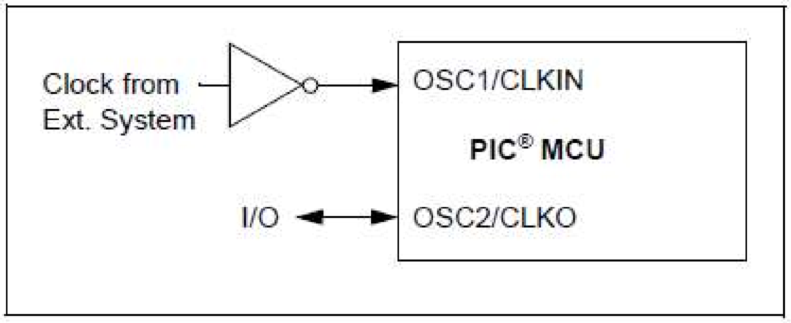
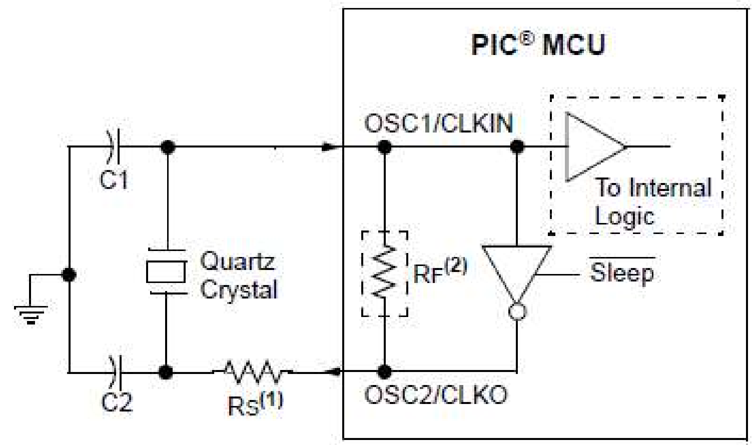


FIGURE 3-5: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)

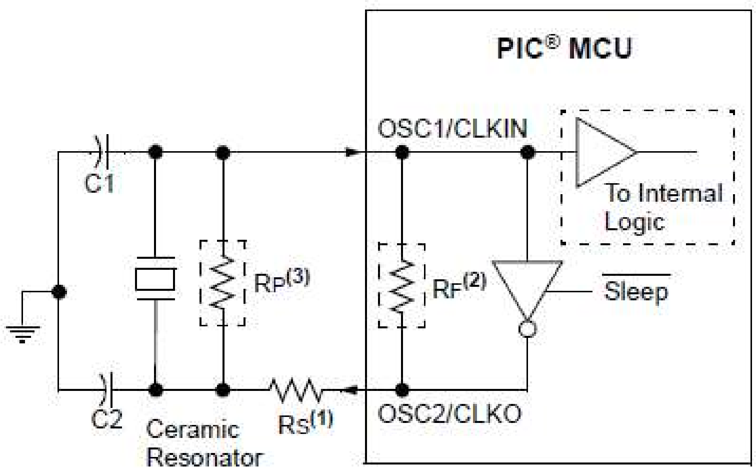


Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

External Sources

FIGURE 3-6: CERAMIC RESONATOR OPERATION (XT OR HS MODE)

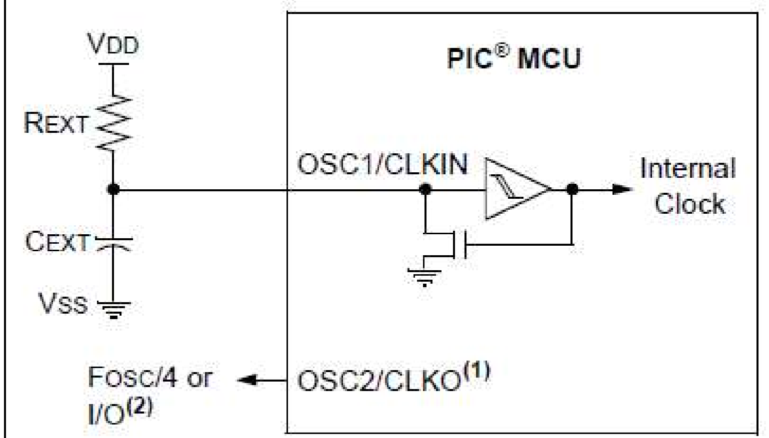


Note 1: A series resistor (RS) may be required for ceramic resonators with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

3: An additional parallel feedback resistor (RP) may be required for proper ceramic resonator operation.

FIGURE 3-7: EXTERNAL RC MODES



Recommended values: $10\ k\Omega \le R_{EXT} \le 100\ k\Omega$
$C_{EXT} > 20\ pF$

Note 1: Alternate pin functions are listed in **Section 1.0 "Device Overview"**.

2: Output depends upon RC or RCIO clock mode.

How to set the internal clock

- The default oscillator is the internal at 1Mhz.
- Tipically, internal oscillators in low en microcontrollers exibits much greater error in the 1 to 10% compared with crystals that have a ppm rating
- In the PIC18 the error can go as high as +/-5%
- The error can be reduced using with tuning using the Active Clock Tuning but you still need to use the secondary external crystal at 32.768Khz

The PIC18 internal osc error

HF-INTOSC Accuracy[1]					
	-2	±1	+2	%	+0°C to +70°C
	-3	—	+2	%	+70°C to +85°C
	-5	—	+5	%	-40°C to 85°C

The PIC18 ACT

HF-INTOSC Accuracy with Active Clock Tuning (ACT)					
	-0.20	±0.05	+0.20	%	-40°C to +85°C[2], Active Clock Tune is enabled and locked.

How to change the internal clock frequency

3.3 Register Definitions: Oscillator Control

REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-0	R/W-1	R/W-1	R-q	R-0	R/W-0	R/W-0
IDLEN	IRCF<2:0>			OSTS(1)	HFIOFS	SCS<1:0>	
bit 7							bit 0

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>:** Internal RC Oscillator Frequency Select bits
111 = HFINTOSC – (16 MHz)
110 = HFINTOSC/2 – (8 MHz)
101 = HFINTOSC/4 – (4 MHz)
100 = HFINTOSC/8 – (2 MHz)
011 = HFINTOSC/16 – (1 MHz)(2)
010 = HFINTOSC/32 – (500 kHz)
001 = HFINTOSC/64 – (250 kHz)

If INTSRC = 1:
000 = HFINTOSC/512 – (31.25 kHz)

If INTSRC = 0:
000 = INTRC – (31.25 kHz)

bit 3 **OSTS:** Oscillator Start-up Time-out Status bit
1 = Device is running from the clock defined by FOSC<3:0> of the CONFIG1H register
0 = Device is running from the internal oscillator (HFINTOSC or INTRC)

bit 2 **HFIOFS:** HFINTOSC Frequency Stable bit
1 = HFINTOSC frequency is stable
0 = HFINTOSC frequency is not stable

bit 1-0 **SCS<1:0>:** System Clock Select bit
1x = Internal oscillator block
01 = Secondary (SOSC) oscillator
00 = Primary clock (determined by FOSC<3:0> in CONFIG1H).

Note 1: Reset state depends on state of the IESO Configuration bit.
2: Default output frequency of HFINTOSC on Reset.

How to set the internal clock

For example, in the following code I am changing the value of the Fosc to 16 Mhz:

```
//FUNCTION TO INIT THE OSCILLATOR AT 16 MHZ
void init_osc(void){

      OSCCON = 0b01110010;


}
```

IRCF<2:0>: Internal RC Oscillator Frequency Select bits

111 = HFINTOSC – (16 MHz)
110 = HFINTOSC/2 – (8 MHz)
101 = HFINTOSC/4 – (4 MHz)
100 = HFINTOSC/8 – (2 MHz)
011 = HFINTOSC/16 – (1 MHz)(2)
010 = HFINTOSC/32 – (500 kHz)
001 = HFINTOSC/64 – (250 kHz)

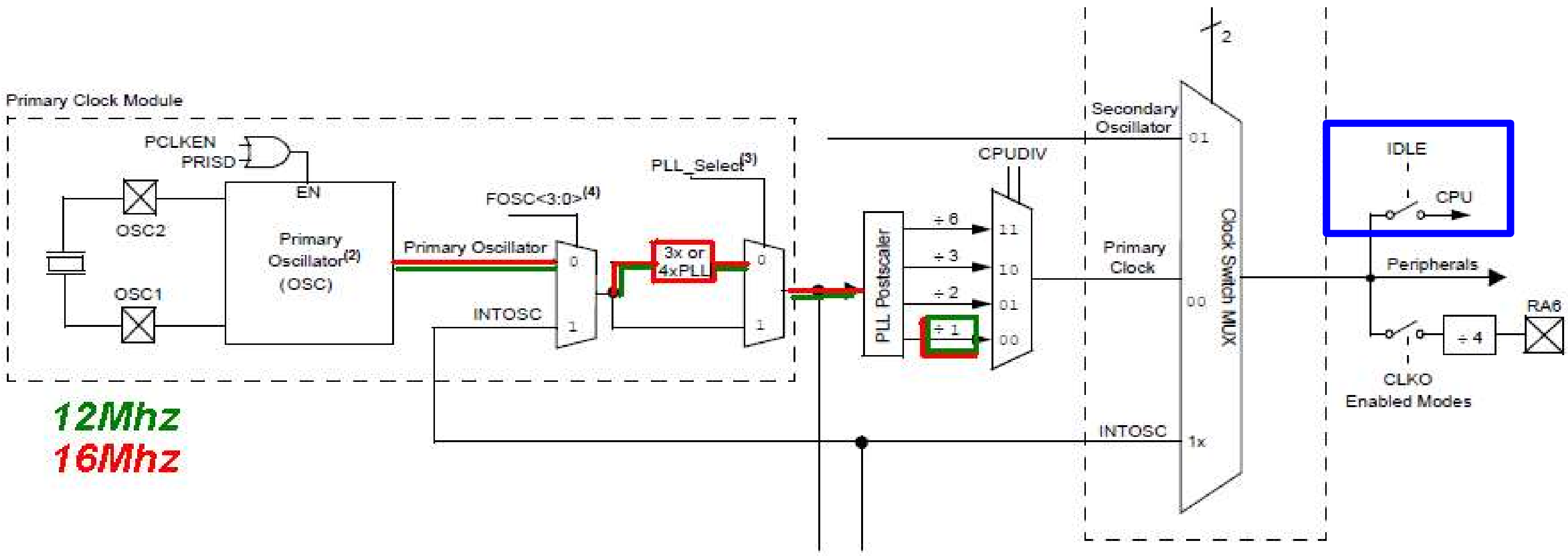
The MSB is don care for our application
The Green bits define the frequency
The Blue bits are read only
The Red bit tell the microcontroller to use the internal oscillator, so even if the configuration bits (See Subject 13) set in the PIC flash say to use the crystal, the microcontroller will witch to the internal when the function is executed. So if you want to use an external crystal then the Red bits must be equal to 00.

¿How change the oscillator that will be used after reset?

Using what the Configuration Registers (They are stored in a section of the Flash memory)

Also using SFR registers called OSCCON, OSCCON2, OSCTUNE

Can I increase the internal frequency relative to the source oscillator ?



(12Mhz x 4)/1 = 48Mhz

(16Mhz x 3)/1 = 48Mhz

However, avoid this, since this could affect the communication speed that is used to transfer your code to the microcontroller

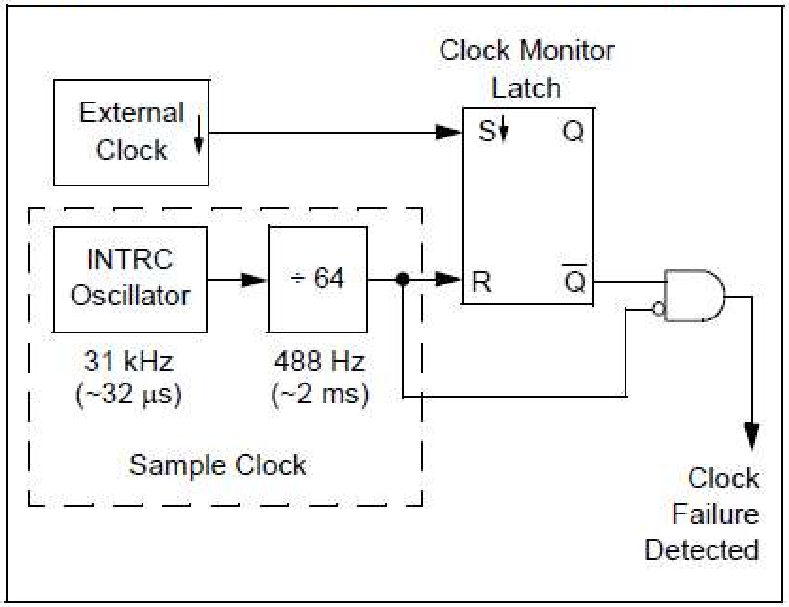
3.7 Register Definitions: Oscillator Tuning

REGISTER 3-3: OSCTUNE: OSCILLATOR TUNING REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPLLMULT	TUN<6:0>						
bit 7							bit 0

Fail safe

FIGURE 3-9: FSCM BLOCK DIAGRAM



3.13 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the CONFIG1H Configuration register. The FSCM is applicable to all external oscillator modes (LP, XT, HS, EC, RC and RCIO).

DS41412E-

Revisar DS30684A-page 47

Timers

- When you need to count or measure time, you can use instruction cycles since you know how much time an instruction requires to execute (4/FOSC)
- This method does not allow the execution of more code since the code will be the only thing it will be executed during this task, also you must use assembly (or a function written on this language) to guarantee more precision and repeatability

Timers

- In a digital system, time is represented by the count of a timer (Timer ticks).
- Timers are useful for :
  - Register the time between events
  - Generate periodic interrupts for precise time base
  - Measurement of pulse widths and periods.
  - Measurement of duty cycle and frequency
  - Waveform generation
  - Time reference
  - Counting events (Counter mode of timers)

Timers

- Timers available on the PIC18F45K50
    - TIMER 0 configurable to 8 or 16 bits
    - TIMER 1 (16 bits)
    - TIMER 2 (8 bits)
    - TIMER 3 (16 bits)

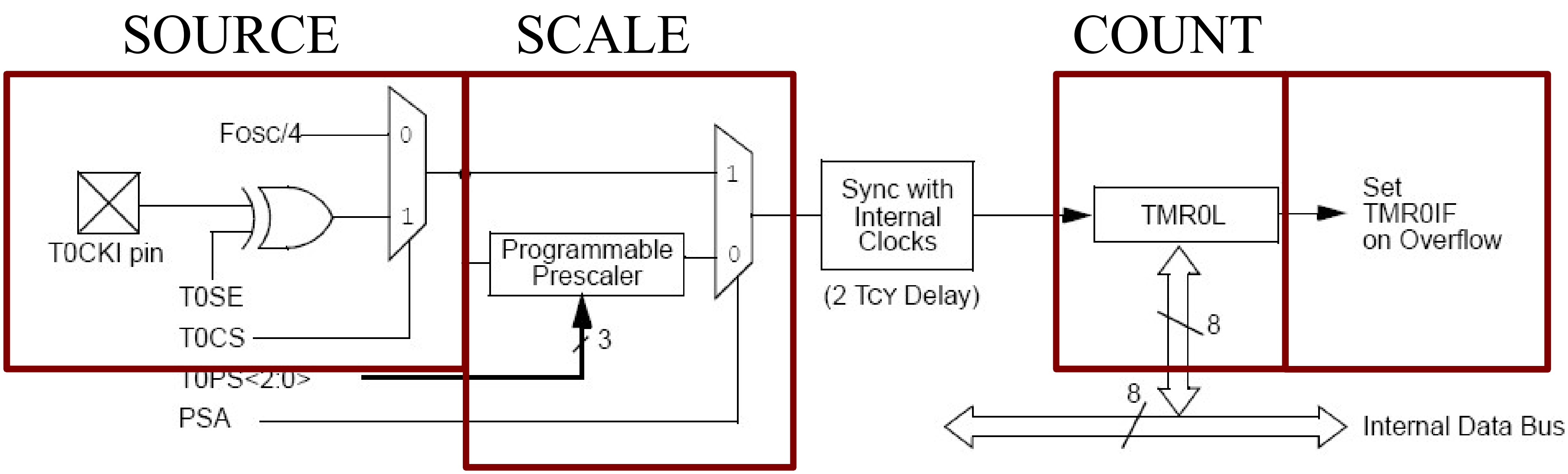
Bits of the timer

- The amount of counts on the count register of the timer provides the resolution or how big is the number to count before rolling over.
- An 8 bit timer will count 0x00 to 0xFF (256 counts considering 0)
- An 16 bits will count from 0x0000 to 0xFFFF (65536 counts including the 0)

Timer 0

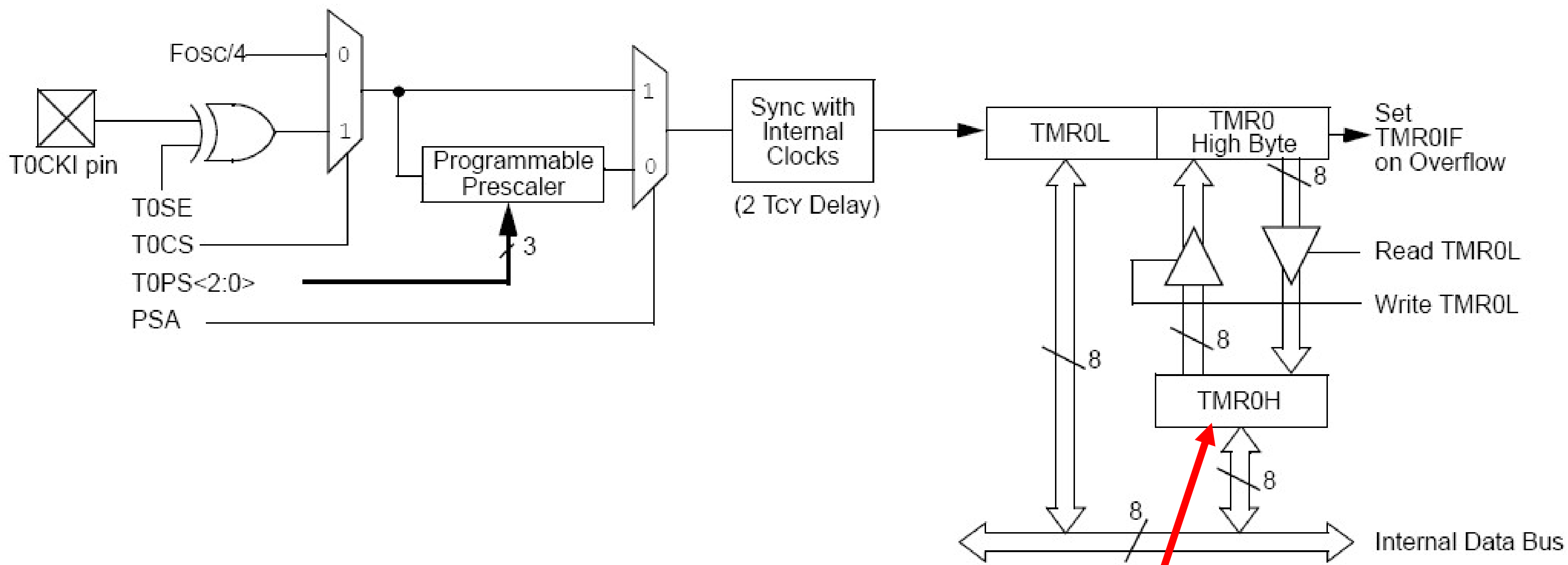
- Configurable to 8 or 16 bits
- Can count using an internal or external clock source
- When using internal source is called TIMER, when external is a COUNTER
- The clock source can be pre-scaled (divided by a factor)

Timer0 (8 bits)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Timer0 (16 bits)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Holds the high part since the data bus is 8 bits not 16 bits

Control register for Timer0

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>		
bit 7							bit 0

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

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

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

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

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

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Other registers related to Timer0

TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER0

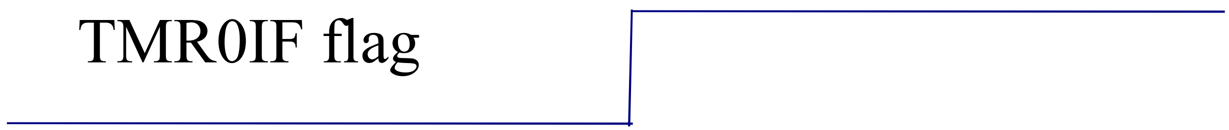
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	IOCIE	TMR0IF	INT0IF	IOCIF	120
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	IOCIP	121
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>			161
TMR0H	Timer0 Register, High Byte								—
TMR0L	Timer0 Register, Low Byte								—
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	156

Operation of Timer 0

- The timer will start counting starting from the value stored in the count register TMR0
- Each cycle of the clock source , the value of TMR0 is incremented by 1 (once pre-scaled if such is the case). Each count is commonly know as a timer-tick
- When the timer rolls from (0xFF ot 0xFFFF) to 0x0 the TMR0IF flag is set to a logic 1.
- The flag can be polled by code or automatically genarate an interrupt if desired.

Operation

TMR0IF flag



00h,01h,02h,……..FDh,FEH,FFH,00h,01h,02h,……..FDh,FEH,FFH,

Initial value of TMR0 = 0x00

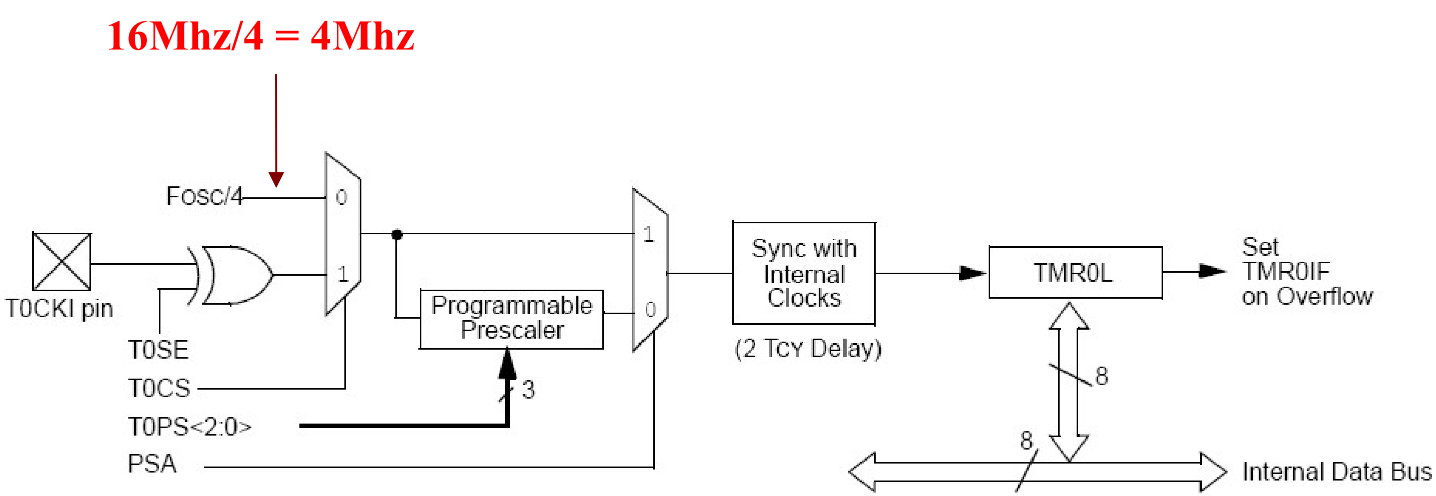
TMR0IF flag

25h,26h,27h,.FDh,FEH,FFH,00h,01h,02h,---.FDh,FEH,FFH,

Initial value of TMR0 = 0x25

Using Timer0

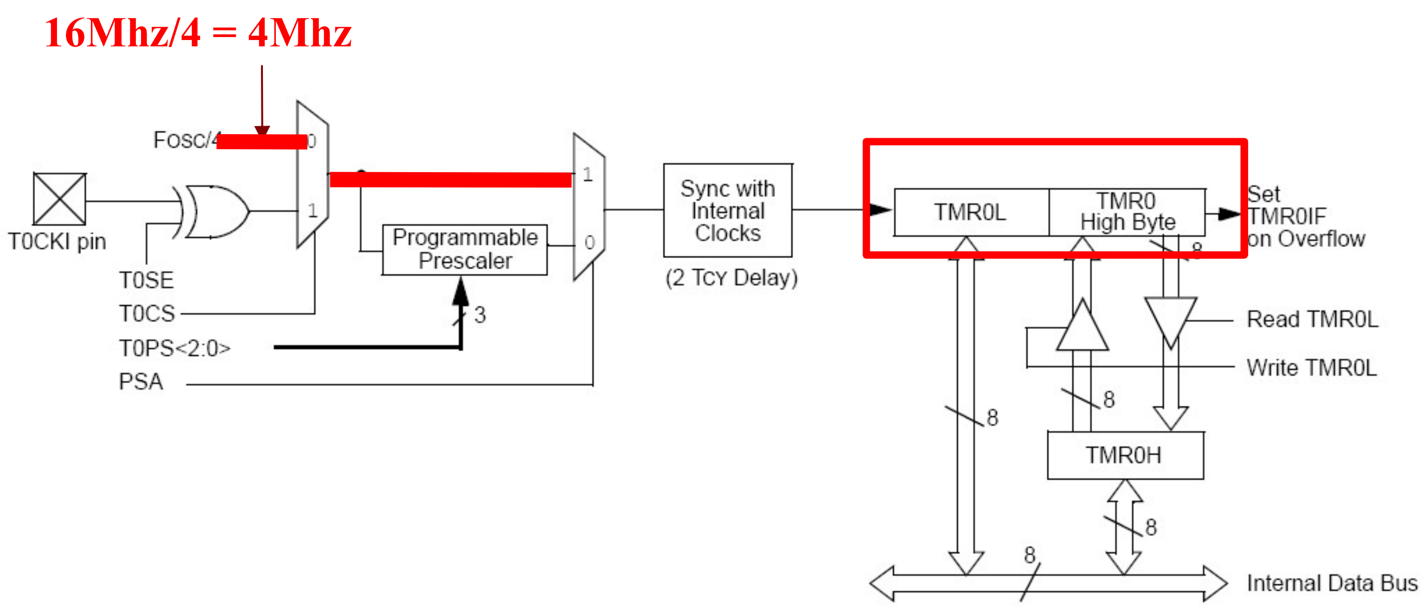
- Make a delay subroutine that used the Timer 0. The delay must be 100msec and the oscilador is 16Mhz. Use the subroutine to generate a 5Hz wave on port D.



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Using Timer0

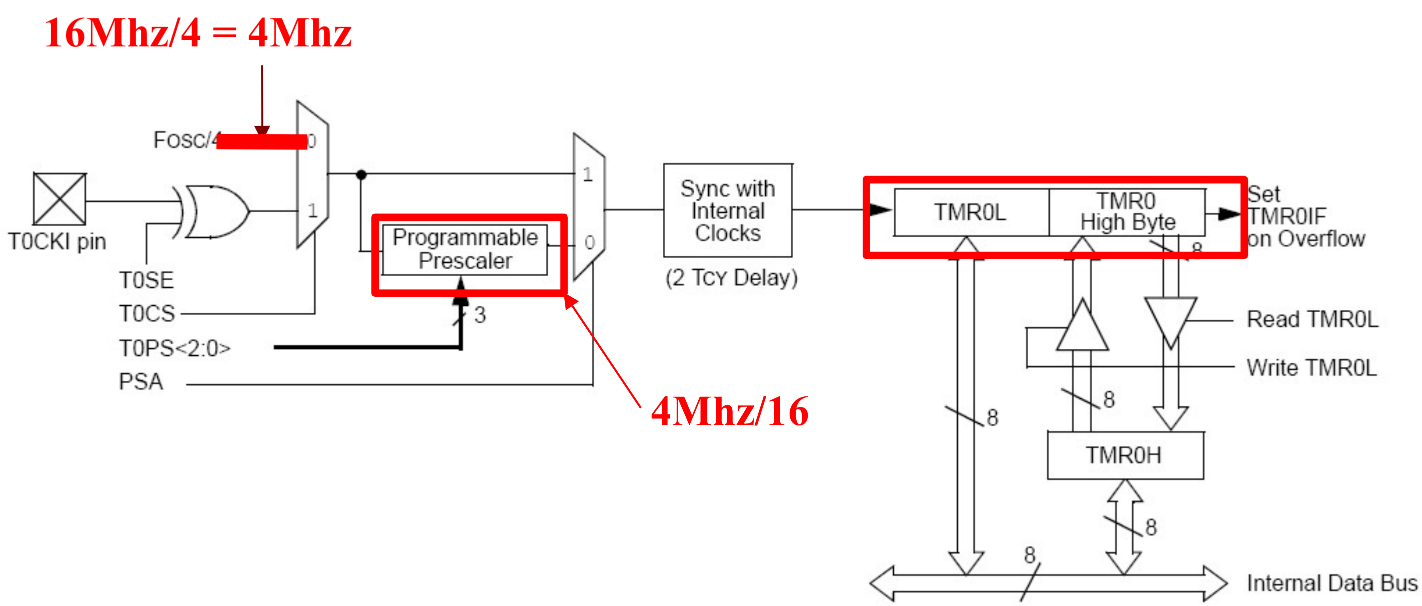
- Lets check if we can implement without pre-scaler
- Each count (timer tick) will be 1/4Mhz = 0.25usec
- We need to count 100ms, then 0.1/0.25usec = 400,000 ticks (counts). But the maximum value is 65535



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Using Timer0

- The pre-scaler divides the frequency in 2,4,8,16,32,64,128,256
- Lets try a value of 16, each count (timer tick) will be 16/4Mhz = 4usec
- If we want to count100ms, then 0.1/4usec = 25,000 ticks. This can be stored in the16 bit register. So is a Win !!



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Control registerT0CON

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>		
bit 7 1	0	0	0	0	0	1	1 bit 0

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

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

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

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

Starts the count when set to 1

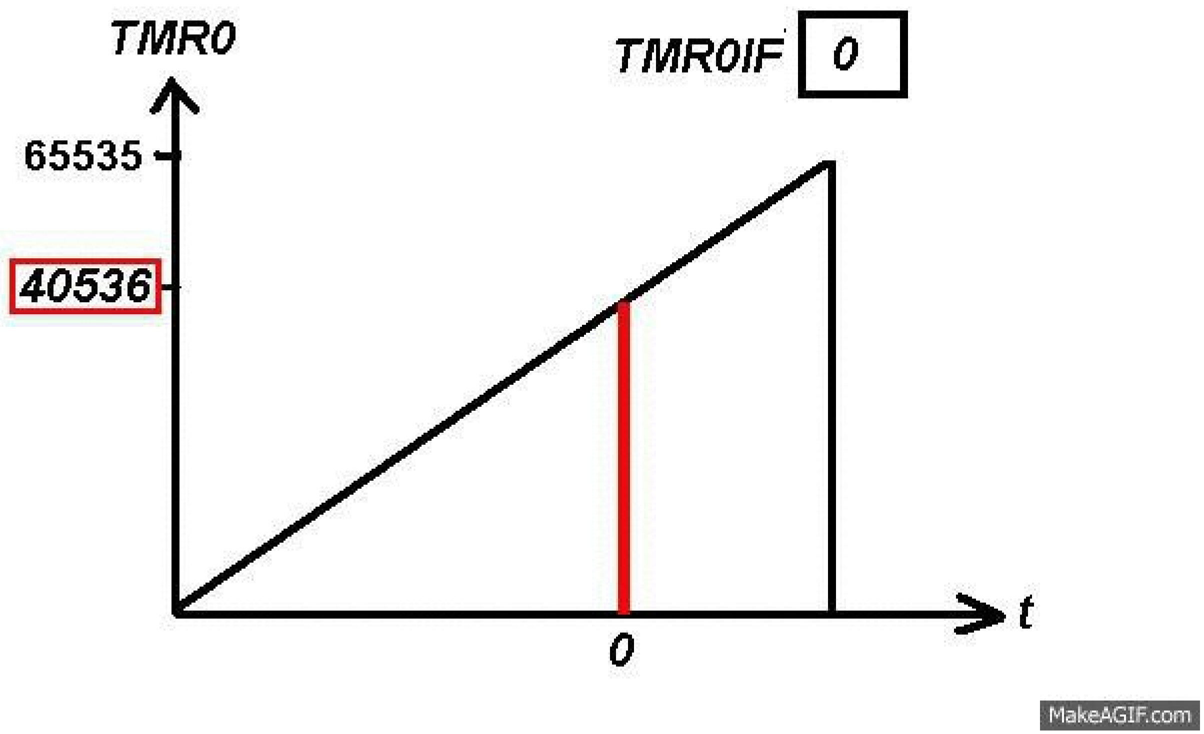
bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Configuring TMR0 count register

- Since we need to count 25,000 ticks and our count registrar can hold up to 65536
- We need to count starting from 65536-25000= 40536
- So the initial value of count registerTMR0 must be 40536 or 0x9E58

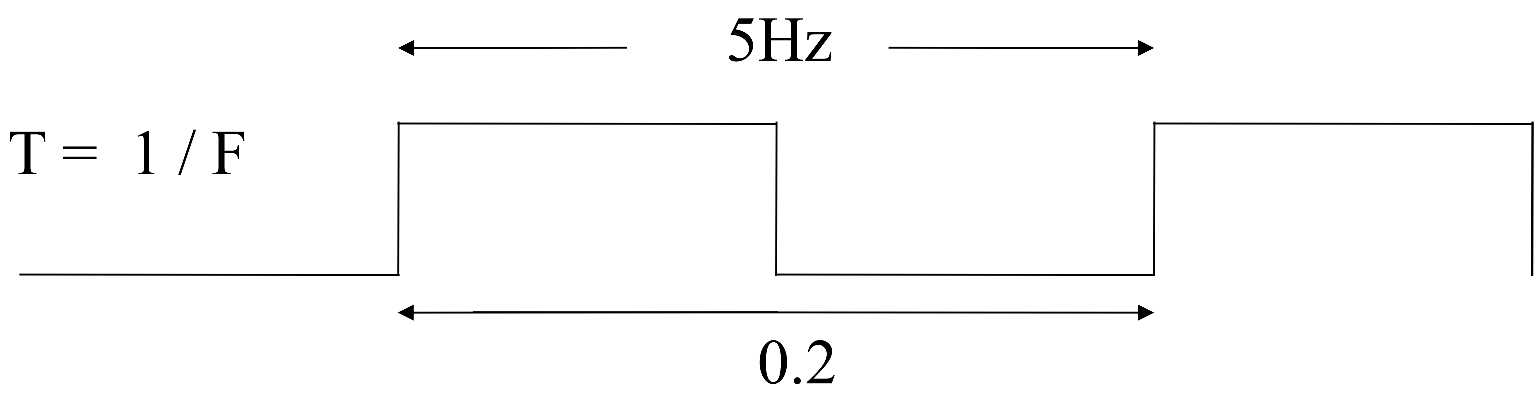
Configuring TMR0 count register



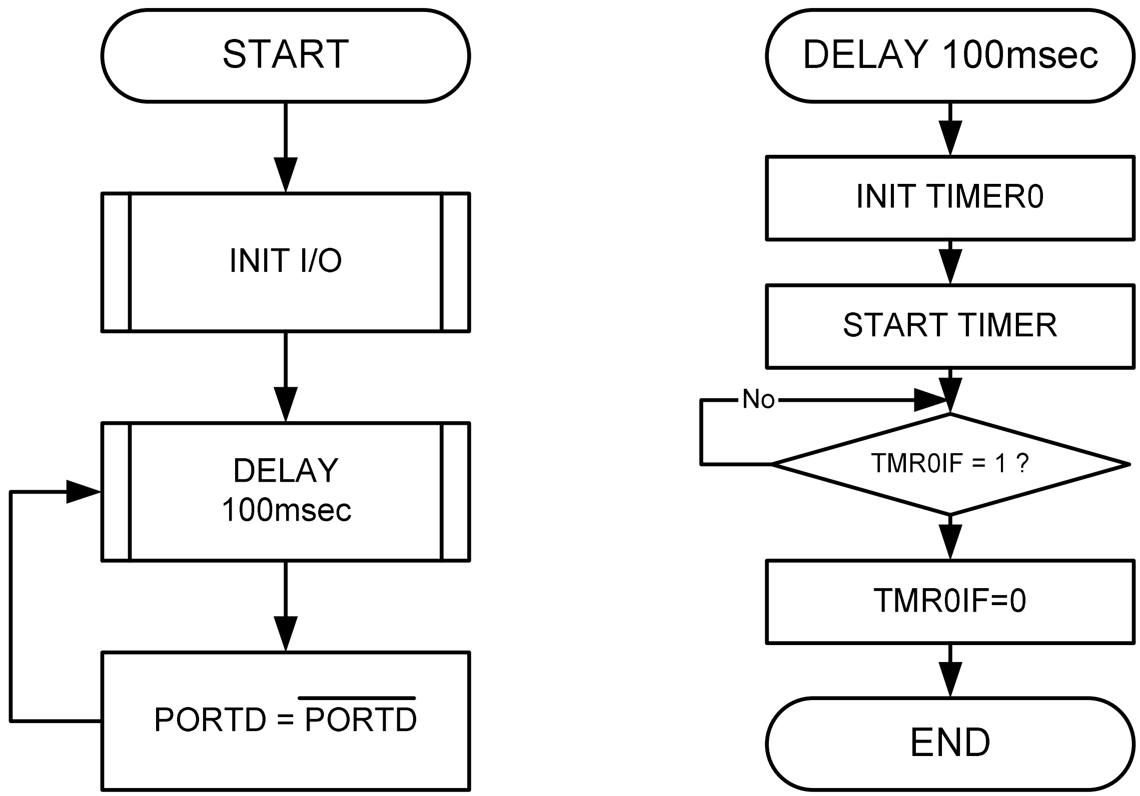
Code

- Init the count register TMR0 to define the initial value of the count
- Init the control register T0CON to define the timer behavior
- Wait for the flag TMR0IF that indicates the overflow (roll over)
- Use our “delay” function to generate 5Hz square wave

Code



Code


START
INIT I/O
DELAY 100msec
PORTD = PORTD
DELAY 100msec
INIT TIMER0
START TIMER
No
TMR0IF = 1 ?
TMR0IF=0
END

Code

```
#include<xc.h>
void delay_100ms(void);           //This function with generate 100ms delay
void init_ports(void);            //Funtion inits port and interal oscillator freq
#define LED_D1 PORTAbits.RA5      //Define a name to the port (optional)
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Main program
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void main(void){

 init_ports();                   //Init the oscilltor and ports

while(1){
    delay_100ms();
    LED_D1 = LED_D1 ^ 0x01;
} //from while(1)
}
```

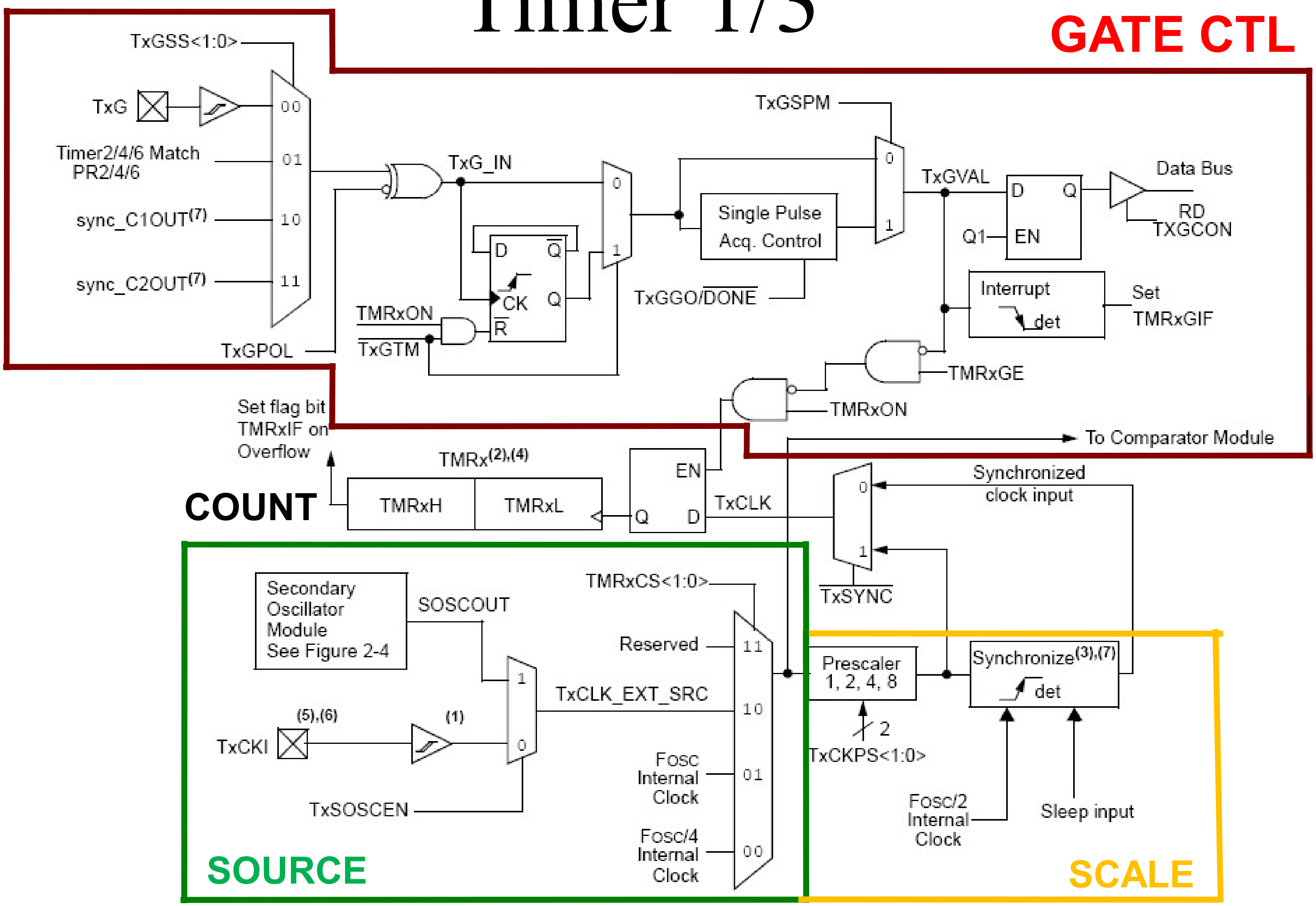
Code

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Delay function
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void delay_100ms(void){
        //Count 40536 or 0x9E58 ticks
    TMR0H = 0x9E;            //High byte 0x9E58
    TMR0L = 0x58;            //Low bte de 0x9E58
    INTCONbits.TMR0IF = 0;   //Clear the timer overflow flag
    //Configure the timer
    //16 bits
    //Set a 16 pre-scaler
    T0CON = 0b10000011;
    while(INTCONbits.TMR0IF == 0); //Wait for overflow
    T0CON = 0X00;                  //Stop the timer
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Function that inits the ports used and oscilator
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void init_ports(void){
    OSCCON = 0b01111110;     //Set the board oscillator to 16Mhz
    //Output RA5 --> LED
    TRISAbits.TRISA5 = 0;   //Output
    ANSELAbits.ANSA5 = 0;   //Digital
```

Use of interrupts

- We can implement the later code using interrupts.
- In the ISR we could have the timer initialization and the change of state of the port.

Timer 1/3


GATE CTL
TxGSS<1:0>
TxG
00
01
10
11
Timer2/4/6 Match PR2/4/6
sync_C1OUT(7)
sync_C2OUT(7)
TxGPOL
TxG_IN
TMRxON
TxGTM
D
Q
CK
R
TxGSPM
Single Pulse Acq. Control
TxGGO/DONE
TxGVAL
EN
Q1
Data Bus
RD TXGCON
Interrupt det
Set TMRxGIF
TMRxGE
TMRxON
To Comparator Module
Set flag bit TMRxIF on Overflow
COUNT
TMRx(2),(4)
TMRxH
TMRxL
TxCLK
Synchronized clock input
TxSYNC
SOURCE
Secondary Oscillator Module See Figure 2-4
SOSCOUT
TMRxCS<1:0>
Reserved
TxCLK_EXT_SRC
TxCKI
(5),(6)
(1)
TxSOSCEN
Fosc Internal Clock
Fosc/4 Internal Clock
Prescaler 1, 2, 4, 8
TxCKPS<1:0>
2
Synchronize(3),(7) det
Fosc/2 Internal Clock
Sleep input
SCALE

Timer 1/3

13.0 TIMER1/3 MODULE WITH GATE CONTROL

The Timer1/3 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-Bit Read/Write Operation
- Time base for the Capture/Compare function
- Special Event Trigger (with CCP/ECCP)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-pulse mode
- Gate Value Status
- Gate Event Interrupt

Control register for Timer 1/3

13.13 Register Definitions: Timer1/3 Control

REGISTER 13-1: TxCON: TIMER1/3 CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/0	R/W-0/u
TMRxCS<1:0>		TxCKPS<1:0>		SOSCEN	$\overline{\text{TxSYNC}}$	RD16	TMRxON
bit 7							bit 0

bit 7-6 **TMRxCS<1:0>:** Timer1/3 Clock Source Select bits
11 = Reserved. Do not use.
10 = Timer1/3 clock source is pin or oscillator:
If SOSCEN = 0:
External clock from TxCKI pin (on the rising edge)
If SOSCEN = 1:
Crystal oscillator on SOSCI/SOSCO pins
01 = Timer1/3 clock source is system clock (FOSC)
00 = Timer1/3 clock source is instruction clock (FOSC/4)

bit 5-4 **TxCKPS<1:0>:** Timer1/3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **SOSCEN:** Secondary Oscillator Enable Control bit
1 = Dedicated secondary oscillator circuit enabled
0 = Dedicated secondary oscillator circuit disabled

bit 2 **$\overline{\text{TxSYNC}}$:** Timer1/3 External Clock Input Synchronization Control bit
TMRxCS<1:0> = 1X
1 = Do not synchronize external clock input
0 = Synchronize external clock input with system clock (FOSC)

TMRxCS<1:0> = 0X
This bit is ignored. Timer1/3 uses the internal clock when TMRxCS<1:0> = 0X

bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer1/3 in one 16-bit operation
0 = Enables register read/write of Timer1/3 in two 8-bit operation

bit 0 **TMRxON:** Timer1/3 On bit
1 = Enables Timer1/3
0 = Stops Timer1/3
Clears Timer1/3 Gate flip-flop

Control regiser for Timer 1/3

REGISTER 13-2: TxGCON: TIMER1/3 GATE CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W/HC-0/u	R-x/x	R/W-0/u	R/W-0/u
TMRxGE	TxGPOL	TxGTM	TxGSPM	TxGGO/$\overline{\text{DONE}}$	TxGVAL	TxGSS<1:0>	
bit 7							bit 0

bit 7 **TMRxGE:** Timer1/3/5 Gate Enable bit
If TMRxON = 0:
This bit is ignored
If TMRxON = 1:
1 = Timer1/3/5 counting is controlled by the Timer1/3/5 gate function
0 = Timer1/3/5 counts regardless of Timer1/3/5 gate function

bit 6 **TxGPOL:** Timer1/3/5 Gate Polarity bit
1 = Timer1/3/5 gate is active-high (Timer1/3/5 counts when gate is high)
0 = Timer1/3/5 gate is active-low (Timer1/3/5 counts when gate is low)

bit 5 **TxGTM:** Timer1/3/5 Gate Toggle Mode bit
1 = Timer1/3/5 Gate Toggle mode is enabled
0 = Timer1/3/5 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1/3/5 gate flip-flop toggles on every rising edge.

Control register for Timer 1/3

REGISTER 13-2: TxGCON: TIMER1/3 GATE CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W/HC-0/u	R-x/x	R/W-0/u	R/W-0/u
TMRxGE	TxGPOL	TxGTM	TxGSPM	$TxGGO/\overline{DONE}$	TxGVAL	TxGSS<1:0>	
bit 7							bit 0

bit 4 **TxGSPM:** Timer1/3/5 Gate Single-Pulse Mode bit
1 = Timer1/3/5 gate Single-Pulse mode is enabled and is controlling Timer1/3/5 gate
0 = Timer1/3/5 gate Single-Pulse mode is disabled

bit 3 **$TxGGO/\overline{DONE}$:** Timer1/3/5 Gate Single-Pulse Acquisition Status bit
1 = Timer1/3/5 gate single-pulse acquisition is ready, waiting for an edge
0 = Timer1/3/5 gate single-pulse acquisition has completed or has not been started
This bit is automatically cleared when TxGSPM is cleared.

bit 2 **TxGVAL:** Timer1/3/5 Gate Current State bit
Indicates the current state of the Timer1/3/5 gate that could be provided to TMRxH:TMRxL.
Unaffected by Timer1/3/5 Gate Enable (TMRxGE).

bit 1-0 **TxGSS<1:0>:** Timer1/3/5 Gate Source Select bits
00 = Timer1/3/5 Gate pin
01 = Timer2/4/6 Match PR2/4/6 output (See Table 12-6 for proper timer match selection)
10 = Comparator 1 optionally synchronized output (sync_C1OUT)
11 = Comparator 2 optionally synchronized output (sync_C2OUT)

Associated registers for Timer 1/3

TABLE 13-5: REGISTERS ASSOCIATED WITH TIMER1/3 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	155
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	IOCIE	TMR0IF	INT0IF	IOCIF	120
IPR1	ACTIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	129
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	130
IPR3	—	—	—	—	CTMUIP	USBIP	TMR3GIP	TMR1GIP	131
PIE1	ACTIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	126
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	127
PIE3	—	—	—	—	CTMUIE	USBIE	TMR3GIE	TMR1GIE	128
PIR1	ACTIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	123
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	124
PIR3	—	—	—	—	CTMUIF	USBIF	TMR3GIF	TMR1GIF	125
PMD0	—	UARTMD	USBMD	ACTMD	—	TMR3MD	TMR2MD	TMR1MD	64
T1CON	TMR1CS<1:0>		T1CKPS<1:0>		SOSCEN	$\overline{\text{T1SYNC}}$	RD16	TMR1ON	174
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/$\overline{\text{DONE}}$	T1GVAL	T1GSS<1:0>		175
T3CON	TMR3CS<1:0>		T3CKPS<1:0>		SOSCEN	$\overline{\text{T3SYNC}}$	RD16	TMR3ON	174
T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/$\overline{\text{DONE}}$	T3GVAL	T3GSS<1:0>		175
TMRxH	Timer1/3 Register, High Byte								—
TMRxL	Timer1/3 Register, Low Byte								—
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	156
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0	156

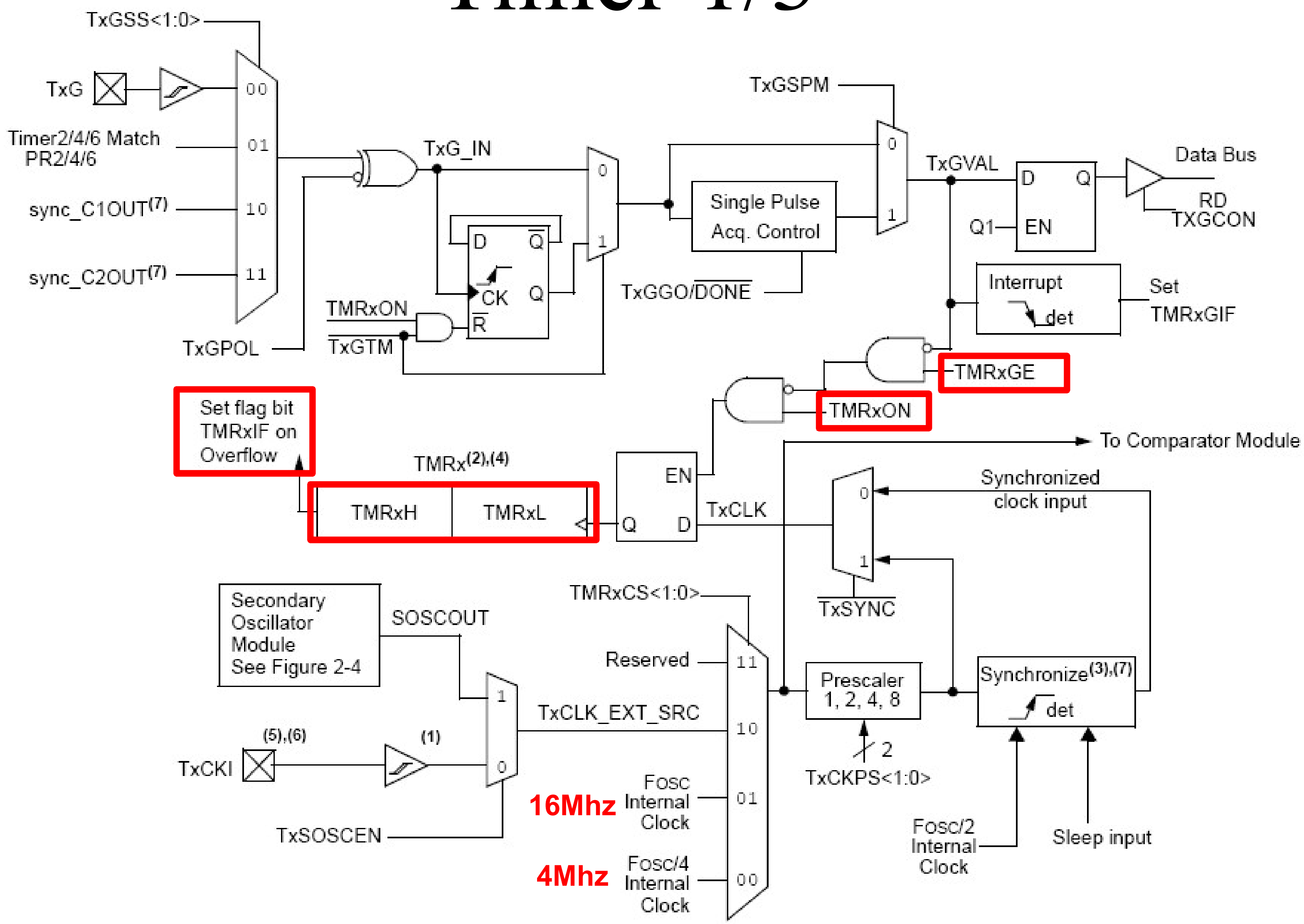
TABLE 13-6: CONFIGURATION REGISTERS ASSOCIATED WITH TIMER1/3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
CONFIG3H	MCLRE	SDOMX	—	T3CMX	—	—	PBADEN	CCP2MX	391

Using Timer1/3

- Make a delay function using Timer 3. The delay must be 10msec and the oscillator is 16Mhz. Use the funtion to generate a 50Hz signal on pin RD0
- The timer will be started by firmware (gate control disabled)

Timer 1/3


TxGSS<1:0>
TxG
Timer2/4/6 Match PR2/4/6
sync_C1OUT(7)
sync_C2OUT(7)
00
01
10
11
TxGPOL
TxG_IN
TMRxON
TxGTM
D
Q
CK
R
0
1
TxGSPM
Single Pulse Acq. Control
TxGGO/DONE
TxGVAL
Data Bus
RD TXGCON
Q1
EN
Interrupt det
Set TMRxGIF
TMRxGE
TMRxON
Set flag bit TMRxIF on Overflow
TMRx(2),(4)
TMRxH
TMRxL
To Comparator Module
Synchronized clock input
TxCLK
TxSYNC
Secondary Oscillator Module See Figure 2-4
SOSCOUT
TMRxCS<1:0>
Reserved
TxCLK_EXT_SRC
Prescaler 1, 2, 4, 8
Synchronize(3),(7) det
TxCKPS<1:0>
TxCKI
(5),(6)
(1)
TxSOSCEN
16Mhz
Fosc Internal Clock
4Mhz
Fosc/4 Internal Clock
Fosc/2 Internal Clock
Sleep input

Timer 3

- As in Timer 0, Timer 3/1 will count starting from an initial value on the count register TMR3 and can hold up to 0xFFFF ticks
- When the roll from 0xFFFF to 0x0000 the flag TMR3IF will be set to 1
- If we need to count certain amount of ticks until the flag is set, the count register TMR3 will be:
  – TMR3 = 65536 - COUNTS

Timer 3

- As in Timer 0, Timer 3/1 will count starting from an initial value on the count register TMR3 and can hold up to 0xFFFF ticks
- When the roll from 0xFFFF to 0x0000 the flag TMR3IF will be set to 1
- If we need to count certain amount of ticks until the flag is set, the count register TMR3 will be:
  - TMR3 = 65536 - COUNTS

Timer 3

- The time it will take since the timer is started until the TMR3IF flag is set is give by:

T_{TM3IF} = Tosc * Prescale * COUNTS

$T_{TM3IF} = \underbrace{Tosc * Prescale}_{Ttick} * [65536 - TMR3]$

- If we need the timer to count using an internal source, Tosc can be 4/Fosc or 1/Fosc

Timer 3

- To configure the timer we can play with Tclock, Prescale and count register TMR3 to reach the desired value

T_{TM3IF} = Tclock * Prescale* [65536 – TMR3]

- Supposing we use Tclock = 4/Fosc = 0.25usec
- If we use a Prescale value of 1:

$10x10^{-3} = 0.25x10^{-6} *1 * [65536 – TMR3]$

- The value of the count register TMR3:

$TMR3 = [65536 - (10x10^{-3})/(0.25x10^{-6} *1)]$

TMR3 = [65536-40000] = 25536 = 0x63C0

Gate control (timer start)

REGISTER 13-2: TxGCON: TIMER1/3 GATE CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W/HC-0/u	R-x/x	R/W-0/u	R/W-0/u
TMRxGE	TxGPOL	TxGTM	TxGSPM	TxGGO/$\overline{\text{DONE}}$	TxGVAL	TxGSS<1:0>	
bit 7 0							bit 0

bit 7 **TMRxGE:** Timer1/3/5 Gate Enable bit
If TMRxON = 0:
This bit is ignored
If TMRxON = 1:
1 = Timer1/3/5 counting is controlled by the Timer1/3/5 gate function
0 = Timer1/3/5 counts regardless of Timer1/3/5 gate function

bit 6 **TxGPOL:** Timer1/3/5 Gate Polarity bit
1 = Timer1/3/5 gate is active-high (Timer1/3/5 counts when gate is high)
0 = Timer1/3/5 gate is active-low (Timer1/3/5 counts when gate is low)

bit 5 **TxGTM:** Timer1/3/5 Gate Toggle Mode bit
1 = Timer1/3/5 Gate Toggle mode is enabled
0 = Timer1/3/5 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1/3/5 gate flip-flop toggles on every rising edge.

Timer 1/3 Control

13.13 Register Definitions: Timer1/3 Control

REGISTER 13-1: TxCON: TIMER1/3 CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/0	R/W-0/u
TMRxCS<1:0>		TxCKPS<1:0>		SOSCEN	$\overline{\text{TxSYNC}}$	RD16	TMRxON
bit 7 0	0	0	0	0	X	X	1 bit 0

bit 7-6 **TMRxCS<1:0>:** Timer1/3 Clock Source Select bits
11 = Reserved. Do not use.
10 = Timer1/3 clock source is pin or oscillator:
If SOSCEN = 0:
External clock from TxCKI pin (on the rising edge)
If SOSCEN = 1:
Crystal oscillator on SOSCI/SOSCO pins
01 = Timer1/3 clock source is system clock (FOSC)
00 = Timer1/3 clock source is instruction clock (FOSC/4)

bit 5-4 **TxCKPS<1:0>:** Timer1/3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **SOSCEN:** Secondary Oscillator Enable Control bit
1 = Dedicated secondary oscillator circuit enabled
0 = Dedicated secondary oscillator circuit disabled

bit 2 **$\overline{\text{TxSYNC}}$:** Timer1/3 External Clock Input Synchronization Control bit
TMRxCS<1:0> = 1X
1 = Do not synchronize external clock input
0 = Synchronize external clock input with system clock (FOSC)

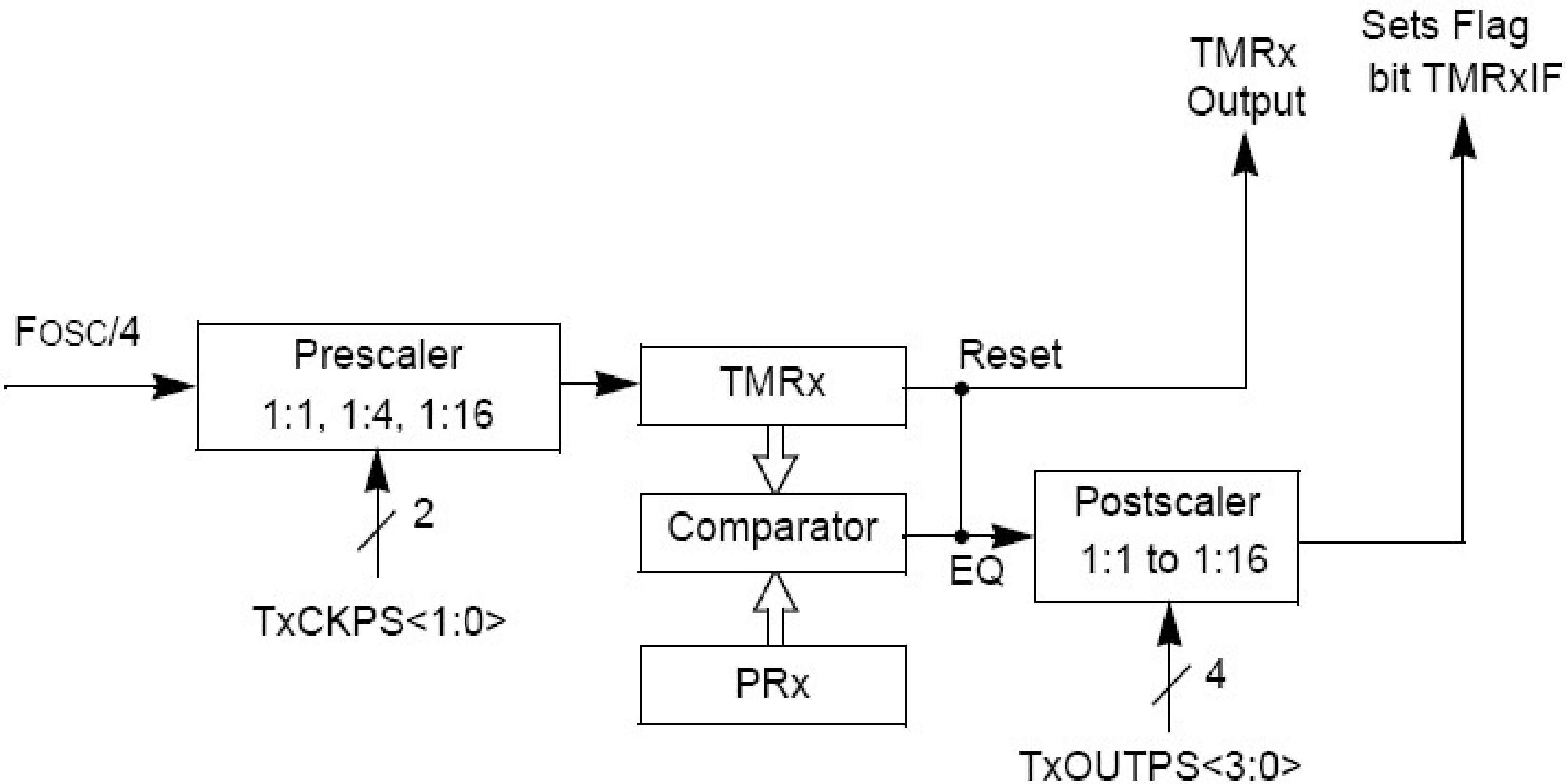
TMRxCS<1:0> = 0X
This bit is ignored. Timer1/3 uses the internal clock when TMRxCS<1:0> = 0X

bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer1/3 in one 16-bit operation
0 = Enables register read/write of Timer1/3 in two 8-bit operation

bit 0 **TMRxON:** Timer1/3 On bit
1 = Enables Timer1/3
0 = Stops Timer1/3
Clears Timer1/3 Gate flip-flop

```c
#include<xc.h>
void delay_10ms(void);              //Function that generates 10mse delay
void init_ports(void);              //Init the oscillator and the ports
#define LED_D1 PORTAbits.RA5     //Define a name to the port (optional)

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Main program
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
main(){
init_ports();             //Init ports and osc
while(1){
    delay_10ms();
    LED_D1 = LED_D1 ^ 0x01;
} //from while(1)
} //from main() TEMA_07_PIC_6.C

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Delay function
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

void delay_10ms(void){
        //You must count 25536 O 0x63C0 timer ticks
        TMR3H = 0x63;               //High part 0x9E58
        TMR3L = 0xC0;               //Low part  0x9E58
        T3GCONbits.TMR3GE = 0;      //Timer starts via firmware
        PIR2bits.TMR3IF = 0;        //Clear the TMR3 overflow flag
        T3CON = 0b00000001;         //Configure ans start the timer
        while( PIR2bits.TMR3IF == 0); //Wait for overflow
        T3CON = 0X00;               //Stop the timer
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Function that inits the ports used and oscilator
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void init_ports(void){

    OSCCON = 0b01111110;     //Set the board oscillator to 16Mhz

    //Output RA4 --> LED
    TRISAbits.TRISA5 = 0;   //Output
    ANSELAbits.ANSA5 = 0;   //Digital


}
```

Timer 2


TMRx
Output
Sets Flag
bit TMRxIF
FOSC/4
Prescaler
1:1, 1:4, 1:16
TMRx
Reset
2
Comparator
EQ
Postscaler
1:1 to 1:16
TxCKPS<1:0>
PRx
4
TxOUTPS<3:0>

Timer 2

14.0 TIMER2 MODULE

The Timer2 module incorporates the following features:

- 8-bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match with PR2, respectively
- Optional use as the shift clock for the MSSP module

Control register for Timer 2

REGISTER 14-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>	
bit 7							bit 0

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

bit 6-3 **TxOUTPS<3:0>:** TimerX Output Postscaler Select bits

0000 = 1:1 Postscaler
0001 = 1:2 Postscaler
0010 = 1:3 Postscaler
0011 = 1:4 Postscaler
0100 = 1:5 Postscaler
0101 = 1:6 Postscaler
0110 = 1:7 Postscaler
0111 = 1:8 Postscaler
1000 = 1:9 Postscaler
1001 = 1:10 Postscaler
1010 = 1:11 Postscaler
1011 = 1:12 Postscaler
1100 = 1:13 Postscaler
1101 = 1:14 Postscaler
1110 = 1:15 Postscaler
1111 = 1:16 Postscaler

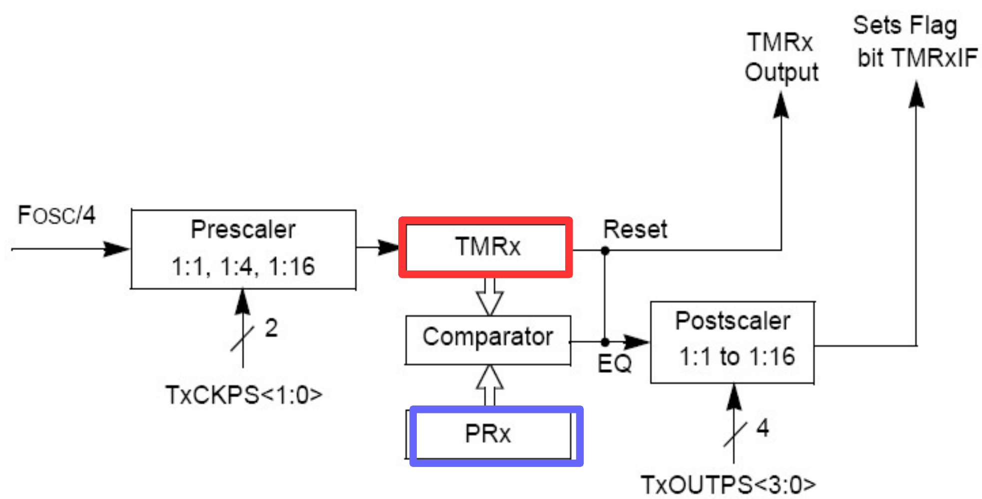
bit 2 **TMRxON:** TimerX On bit

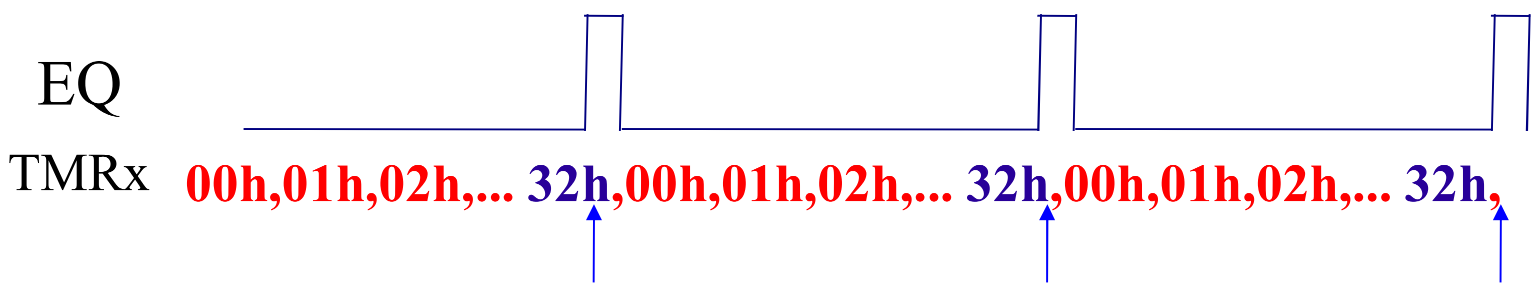
1 = TimerX is on
0 = TimerX is off

bit 1-0 **TxCKPS<1:0>:** Timer2-type Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Operation





Value in Prx = 0x32

Associated registers for Timer 2

TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

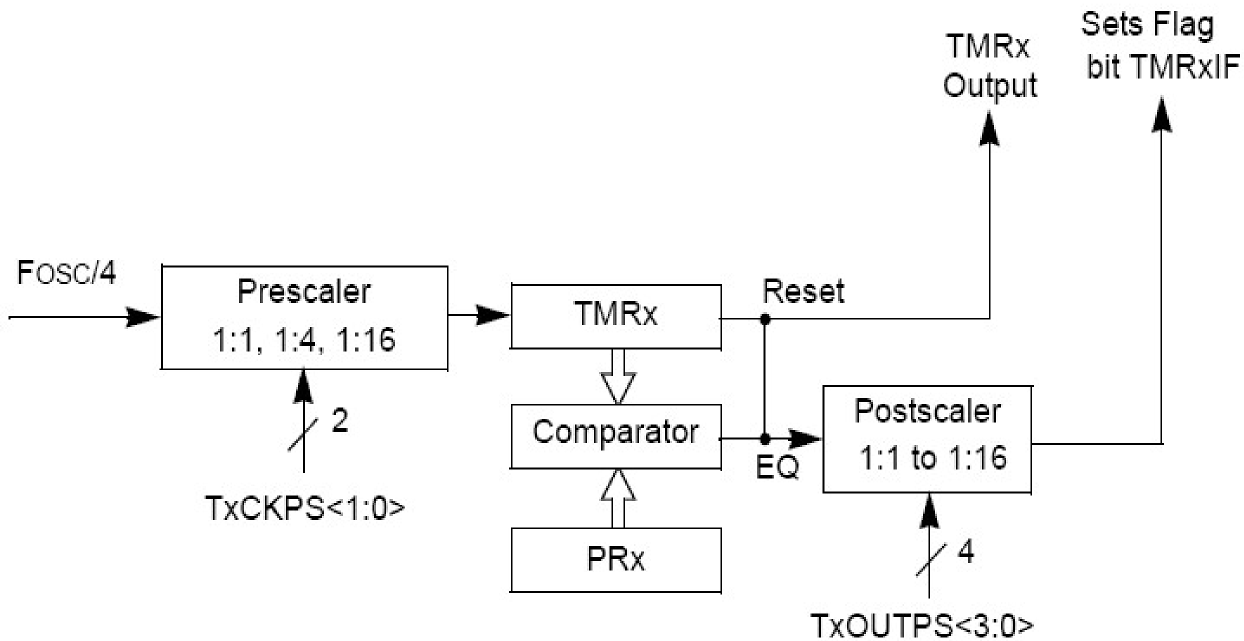
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	IOCIE	TMR0IF	INT0IF	IOCIF	120
IPR1	ACTIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	129
PIE1	ACTIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	126
PIR1	ACTIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	123
PMD0	—	UARTMD	USBMD	ACTMD	—	TMR3MD	TMR2MD	TMR1MD	64
PR2	Timer2 Period Register								—
T2CON	—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>		174
TMR2	Timer2 Register								—

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by Timer2.

Using Timer 2

- From previous example , add the required code to generate a 500Hhz square wave on RD1 using Timer 2 and interrupts

Using Timer2



Using Timer 2

- Timer 2 has a resolution of 8 bits.
- The count register is compared with the “period” register PR2
- Then the TMR2 count register is equal to PR2 register an internal signal reset signal that sets TMR2 to 0 and continues counting
- The internal signal reset signal occurrences can be also post-divided
- The frequency division is the generated by the combination of TMR2 and PR2 + 1;

Using Timer 2

- The time it will take to set the TMR2IF flag from the point it was started will be given by:

$$T_{TM2IF} = Tosc * 4 * Prescale * (PR2+1) * Postscale$$

- We can play with Prescale, PR2 and Postscale

Using Timer 2

- For our problem we need to generate a 500Hz. Since each interrupt we will change the state of the port, we need an interrupt frequency of 1000Hz. We will fist try to configure with a pre-scale in the middle, in this case 4
- We need $T_{TM2IF} = 1/1000 = 1x10^{-3}$
- Since $Fosc = 16x10^6$, $Tosc = 1/Fosc = 6.25x10^{-8}$
- Given the later:

T_{TM2IF} = Tosc *4* Prescale *(PR2+1)*Postscale

$1x10^{-3}$ = $6.25x10^{-8}$*4* [4] *(PR2+1)*Postscale

0000 = 1:1 Postscaler
0001 = 1:2 Postscaler
0010 = 1:3 Postscaler
0011 = 1:4 Postscaler
0100 = 1:5 Postscaler
0101 = 1:6 Postscaler
0110 = 1:7 Postscaler
0111 = 1:8 Postscaler
1000 = 1:9 Postscaler
1001 = 1:10 Postscaler
1010 = 1:11 Postscaler
1011 = 1:12 Postscaler
1100 = 1:13 Postscaler
1101 = 1:14 Postscaler
1110 = 1:15 Postscaler
1111 = 1:16 Postscaler

Using Timer 2

T_{TM2IF} = Tosc *4* Prescale*(PR2+1)*Postscale

$1x10^{-3}$ = $6.25x10^{-8}$*4* [4] *(PR2+1)*Postscale

(PR2+1)*Postscale = 1000

- If we make (PR2+1) = 125 and Postscale= 8
- 125* 8 = 1000
- PR2 will have a value of 124 decimal

0000 = 1:1 Postscaler
0001 = 1:2 Postscaler
0010 = 1:3 Postscaler
0011 = 1:4 Postscaler
0100 = 1:5 Postscaler
0101 = 1:6 Postscaler
0110 = 1:7 Postscaler
0111 = 1:8 Postscaler
1000 = 1:9 Postscaler
1001 = 1:10 Postscaler
1010 = 1:11 Postscaler
1011 = 1:12 Postscaler
1100 = 1:13 Postscaler
1101 = 1:14 Postscaler
1110 = 1:15 Postscaler
1111 = 1:16 Postscaler

Control register Timer 2

REGISTER 14-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>	
bit 7 X	0	1	1	1	1	0	1 bit 0

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

bit 6-3 **TxOUTPS<3:0>:** TimerX Output Postscaler Select bits
0000 = 1:1 Postscaler
0001 = 1:2 Postscaler
0010 = 1:3 Postscaler
0011 = 1:4 Postscaler
0100 = 1:5 Postscaler
0101 = 1:6 Postscaler
0110 = 1:7 Postscaler
0111 = 1:8 Postscaler
1000 = 1:9 Postscaler
1001 = 1:10 Postscaler
1010 = 1:11 Postscaler
1011 = 1:12 Postscaler
1100 = 1:13 Postscaler
1101 = 1:14 Postscaler
1110 = 1:15 Postscaler
1111 = 1:16 Postscaler

bit 2 **TMRxON:** TimerX On bit
1 = TimerX is on
0 = TimerX is off

bit 1-0 **TxCKPS<1:0>:** Timer2-type Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

```
#include<xc.h>

void delay_100ms(void);          //This function with generate 10ms delay
void init_ports(void);           //Funtion inits port and interal oscillator freq
void init_timer2(void);          //Funtion inits timer 2
void high_priority_ISR(void);    //High priority interrup function

#define LED_D1 PORTAbits.RA5     //Define a name to the port (optional)
#define LED_D2 PORTAbits.RA4     //Define a name to the port (optional)

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Main program
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
main(){
init_ports();                   //Init port and other sutuff
init_timer2();                  //Init timer 2
while(1){
    delay_100ms();
    LED_D1 = LED_D1 ^ 0x01;
} //from while(1)
} //from main
```

```c
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Delay function
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void delay_100ms(void){
        //You must count 25536 O 0x63C0 timer ticks
        TMR3H = 0x63;                   //High part 0x9E58
        TMR3L = 0xC0;                   //Low part  0x9E58
        T3GCONbits.TMR3GE = 0;          //Timer starts via firmware
        PIR2bits.TMR3IF = 0;            //Clear the TMR3 overflow flag
        T3CON = 0b00000001;             //Configure ans start the timer
        while( PIR2bits.TMR3IF == 0); //Wait for overflow
        T3CON = 0X00;                   //Stop the timer
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Function that inits the ports used and oscilator
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void init_ports(void){
    OSCCON = 0b01111110;     //Set the board oscillator to 16Mhz
    //Output RA5 --> LED
    TRISAbits.TRISA5 = 0;   //Output
    ANSELAbits.ANSA5 = 0;   //Digital
    //Output RA4 --> LED
    TRISAbits.TRISA4 = 0;   //Output
    //Port 4 is always digital, so no ANSELAbits.ANES6 config requried
}
```

```c
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ Function that inits TIMER2
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void init_timer2(void){
    RCONbits.IPEN = 1;      //Enables priority in interruptions
    PIE1bits.TMR2IE = 1;    //Enable interrupt on match for timer2
    IPR1bits.TMR2IP = 1;    //Interrup is high priority
    PIR1bits.TMR2IF = 0;    //Clear the interrupt flag
    INTCONbits.GIEH = 1;    //Habilitacion global de las de prioridad alta
    INTCONbits.GIEL = 0;    //No hay ninguna asignada por el momento
    PR2 = 124;              // 125 * 8 * 4 * (4 * 6.25 e -8) = 1e-3
    T2CON = 0b00111101;     //Post scale 8, Timer On y Prescale = 4
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//+ ISR for TIMER2
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 void __interrupt (high_priority)  high_priority_ISR(void){
       LED_D2 = LED_D2 ^ 0x01;
      PIR1bits.TMR2IF = 0;      //Apagar bandera
 }
```