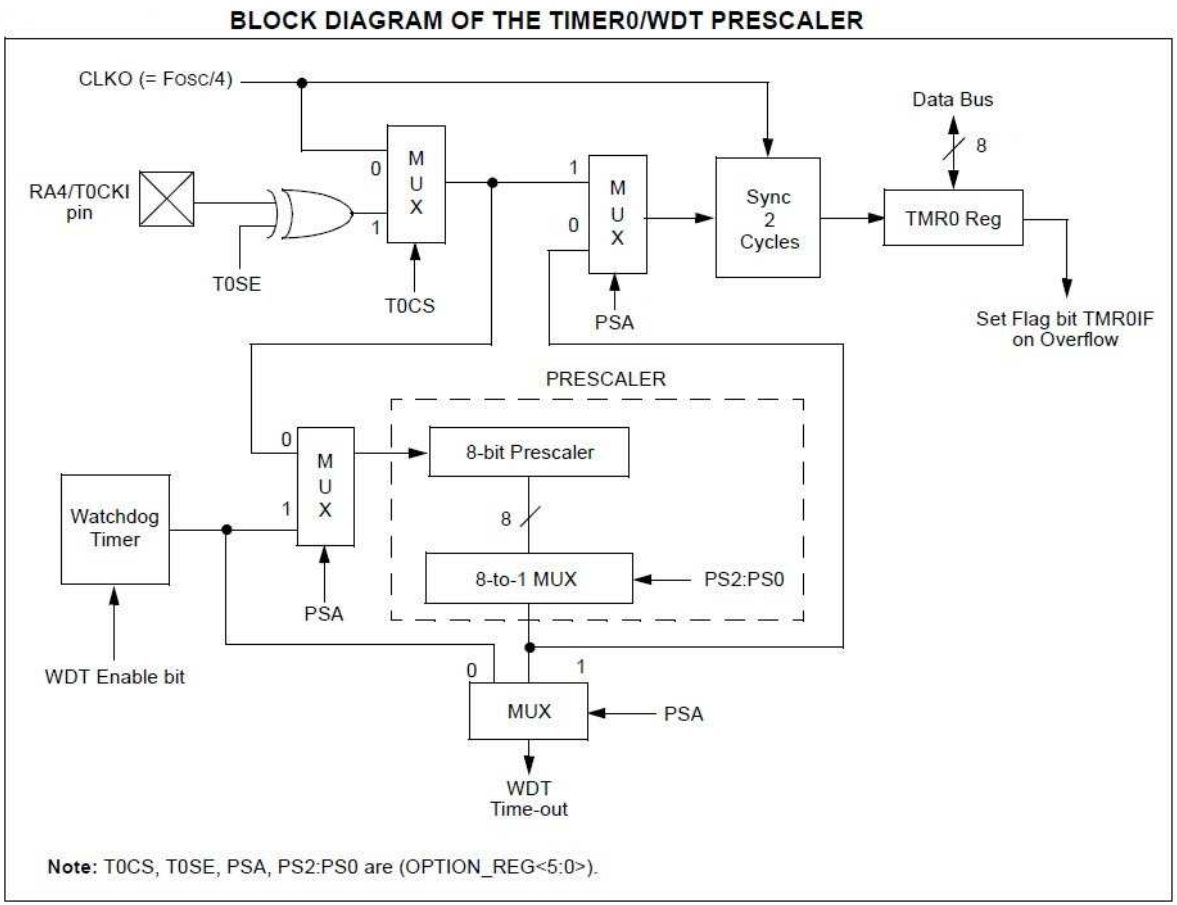
El TIMER 0


BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER
CLKO (= Fosc/4)
RA4/T0CKI pin
T0SE
T0CS
PSA
Sync 2 Cycles
Data Bus
8
TMR0 Reg
Set Flag bit TMR0IF on Overflow
PRESCALER
8-bit Prescaler
8
8-to-1 MUX
PS2:PS0
Watchdog Timer
WDT Enable bit
MUX
WDT Time-out
Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

Características

- Se trata de un registro de 8 bits. (SFR: 01h, 101h)
- Puede trabajar como contador o temporizador.
- Se puede leer o escribir en él.
- Dispone de un preescaler de 8 bit programable por software.
- Puede trabajar con señal de reloj interna o externa.
- Generación de interrupción por desbordamiento (FF->00)
- Flanco programable (ascendente/descendente) para la señal de reloj externa.

TIMER 0: Modo Temporizador (I)

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

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

- El modo temporizador se selecciona poniendo a **cero** el bit **T0CS** (OPTION_REG<5>).
- En modo temporizador, el Timer0 incrementa su valor con cada ciclo de instrucción (sin preescaler).
- Si se escribe en el registro TMR0, éste deja de incrementarse durante 2 ciclos de instrucción.

TIMER 0: Modo Temporizador (II)

Para configurar el TIMER0 en modo temporizador se utiliza la función:

```
setup_timer_0(RTCC_INTERNAL | RTCC_DIV_N );
```

RTCC_INTERNAL, indica el modo temporizador y RTCC_DIV_N configura el preescaler en función de N.

Donde N puede tomar uno de los siguientes valores: 1,2,4,8,16,32,64,128,256.

TIMER 0: Modo Contador (I)

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

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

- El modo contador se selecciona poniendo a **uno** el bit **T0CS** (OPTION_REG<5>).
- En modo contador, el Timer0 incrementa su valor con cada flanco de (subida/bajada) que se produce en RA4/TOCK1.
- Para **TOSE=1** flanco descendente y para **TOSE=0** flanco descendente.

TIMER 0: Modo Contador (II)

Para configurar el TIMER0 en modo contador se utiliza una de las funciones siguientes:

```
setup_timer_0(RTCC_EXT_L_TO_H | RTCC_DIV_N );
```

O bien:

```
setup_timer_0(RTCC_EXT_H_TO_L | RTCC_DIV_N );
```

RTCC_EXT_L_TO_H, configura el modo contador y hace que el registro TIMER0 se incremente con cada flanco ascendente en RA4.

RTCC_EXT_L_TO_H, configura el modo contador y hace que el registro TIMER0 se incremente con cada flanco descendente en RA4.

Lectura y escritura del TIMER0

Para leer el contenido del TIMER0 se utiliza la función:

```
get_timer0();
```

Ejemplo:

```
int valor;                  // Declarar una variable de 8 bits
valor=get_timer0();         // Asignamos el valor del timer a la variable
```

Para escribir un valor en el registro TIMER0 se utiliza la función:

```
set_timer0(valor);
```

Ejemplo:

```
int valor=156;              // Declarar una variable de 8 bits
set_timer0(valor);          // Asignamos el valor 156 al timer
```

o simplemente:

```
set_timer0(156);
```

Interrupción del TIMER0

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

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

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

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

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

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

- Para habilitar la interrupción del TIMER0 hay que poner a uno el bit **TMR0IE** del registro INTCON.
- La interrupción se produce cuando el registro TMR0 pasa de 0xFF a 0x00. En ese momento el bit **TMR0IF** se pone a uno.
- El bit **TMR0IF** debe ponerse a cero por software antes de salir de la rutina de atención a la interrupción.
- La interrupción del Timer0 no saca al pic del modo sleep ya que en este modo el Timer0 permanece desactivado.

Interrupción del TIMER0 en CCS (I)

Para habilitar la interrupción del TIMER0 se utilizan las funciones:

```
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_N);    // N=1,2,4,…..
enable_interrupts(INT_TIMER0);
enable_interrupts(GLOBAL);
```

Para deshabilitar la interrupción:

```
disable_interrupts(INT_TIMER0);
```

La función de atención a la interrupción es:

```
#int_TIMER0
int TIMER0_isr()
{

}
```

Interrupción del TIMER0 en CCS (II)

Ejemplo:

Utilizar el Timer 0 como temporizador con un pre-escaler de 256 y un retardo base de 50ms para realizar un programa que muestre un contador binario en el PORTB a intervalos de 250ms.

Interrupción del TIMER0 en CCS (III)

Paso 1: Configuración del TIMER0 como temporizador, programar el preescaler y habilitar la interrupción (dentro de la función main()):

```
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
enable_interrupts(INT_TIMER0);
enable_interrupts(GLOBAL);
```

Paso 2: Programar la función de interrupción, antes de la función main():

```
int8 conta=0,valor=0;
#int_TIMER0
TIMER0_isr()
{
conta++;
if(conta==5)            // Cuando interrumpe 5 veces han pasado ±250ms
   {                    // 5x50ms=250ms
    valor++;
    conta=0;
   }
set_timer0(256-195);  // Recarga para interrumpir cada 195x256us=49920us
}
```

Interrupción del TIMER0 en CCS (IV)

Paso 3: Completamos la función main() con los mensajes del LCD y la configuración de los puertos.

```
////////////  MENSAJES DE TEXTO ///////////////
char msg1[]="Int. de 250ms";
char msg2[]="con TIMER-0";
void main()
{
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
enable_interrupts(INT_TIMER0);
enable_interrupts(GLOBAL);
set_tris_b(0x00);
output_b(0x00);
set_timer0(256-195);
LCD_Init();
LCD_Escribecadena(msg1);
LCD_Cambiolinea();
LCD_Escribecadena(msg2);
while(true)                    // Bucle infinito
 {
  output_b(valor);
 }
}
```

```
// Interrupción del TIMER0 //
#include <16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#include <lcd_monibot.c>

////////////  FUNCIONES DE INTERRUPCIÓN //////////
int8 conta=0,valor=0;
#int_TIMER0
TIMER0_isr()
{
conta++;
if(conta==5)                // Cuando interrumpe 5 veces han pasado ±250ms
    {                       // 5x50ms=250ms
     valor++;
     conta=0;
    }
set_timer0(256-195);  // Recarga para interrumpir cada 195x256us=49920us
return 0;
}

////////////  MENSAJES DE TEXTO //////////////
char msg1[]="Int. de 250ms";
char msg2[]="con TIMER-0";

void main()
{
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
enable_interrupts(INT_TIMER0);
enable_interrupts(GLOBAL);
set_tris_b(0x00);
output_b(0x00);
set_timer0(256-195);
LCD_Init();
LCD_Escribecadena(msg1);
LCD_Cambiolinea();
LCD_Escribecadena(msg2);

while(true)                     // Bucle infinito
 {
  output_b(valor);
 }
}
```

Programa completo realizado sin utilizar el asistente

Captura de la simulación: