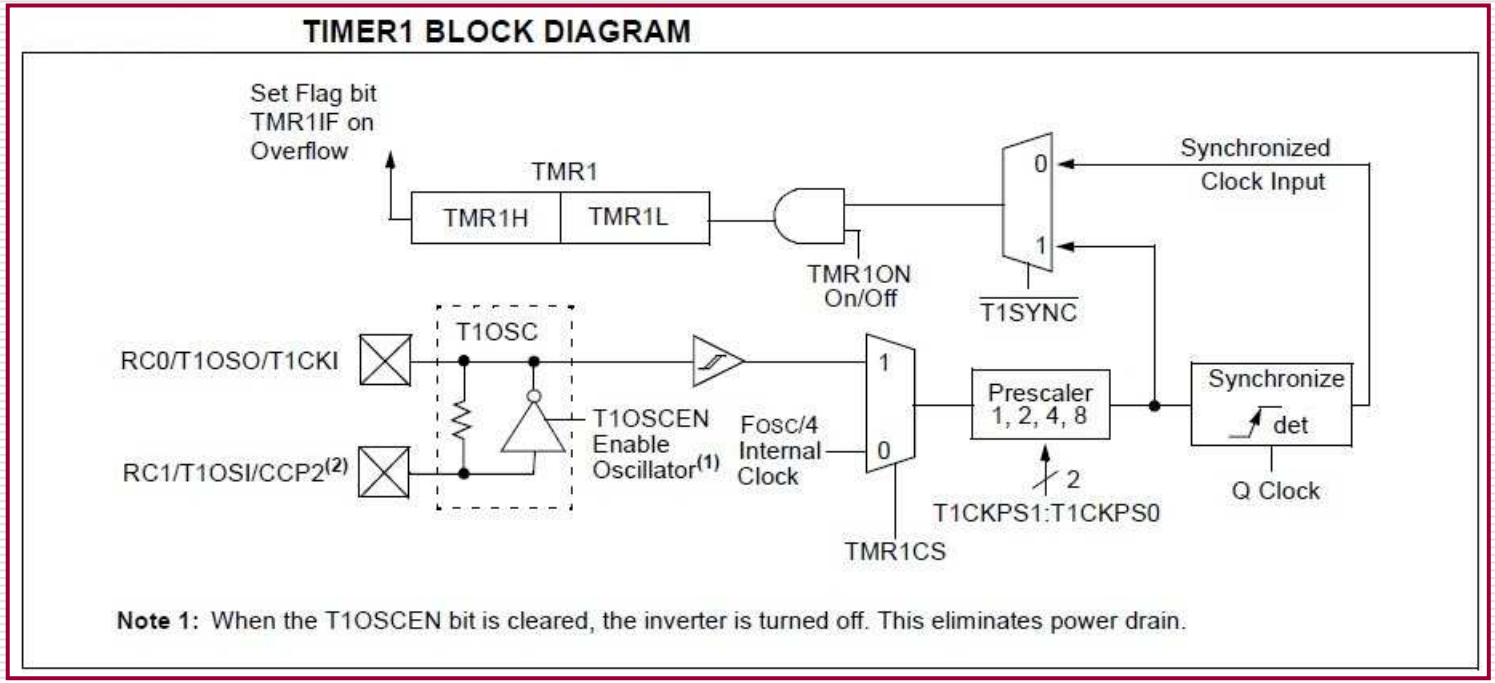
El TIMER 1


TIMER1 BLOCK DIAGRAM
Set Flag bit
TMR1IF on
Overflow
TMR1
TMR1H
TMR1L
TMR1ON
On/Off
0
1
Synchronized
Clock Input
T1SYNC
T1OSC
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2(2)
T1OSCEN
Enable
Oscillator(1)
Fosc/4
Internal
Clock
1
0
Prescaler
1, 2, 4, 8
2
T1CKPS1:T1CKPS0
Synchronize
det
Q Clock
TMR1CS
Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

TIMER 1: Características

- Se trata de un contador/temporizador de 16 bits.
- Puede trabajar como contador o temporizador.
- Está implementado en dos registos TMR1H y TMR1L que son de lectura/escritura.
- Los registros se incrementan desde 0000h hasta FFFFh y cuando se desbordan se activa la bandera **TMR1IF**. (PIR1<0>).
- Generación de interrupción por desbordamiento (FFFF->0000) que puede ser habilitada/deshabilitada con el bit **TMR1IE** del registro (PIE1<0>)

TIMER 1: Modo Temporizador (I)

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

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

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

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

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

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

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

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

- El modo temporizador se selecciona poniendo a **cero** el bit **TMR1CS** (T1CON<1>).
- En modo temporizador, el Timer0 incrementa su valor con cada ciclo de instrucción (sin preescaler).
- Estado habilitado/deshabilitado a través del bit **TMR1ON** (T1CON<0>).
- Posibilidad de reinicio por el módulo de CCP.

TIMER1: Modo Temporizador (II)

Para configurar el TIMER1 en modo temporizador en CCS se utiliza la función:

```
setup_timer_1(T1_INTERNAL | T1_DIV_BY_N );
```

T1_INTERNAL, indica el modo temporizador y T1_DIV_BY_N configura el preescaler en función de N.

Donde N puede tomar uno de los siguientes valores: 1,2,4,8.

TIMER 1: Modo Contador (I)

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit 7							bit 0

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

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

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

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

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

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

- El modo contador se selecciona poniendo a **uno** el bit **TMR1CS** (T1CON<1>).
- En modo contador, el Timer0 incrementa su valor con cada flanco de subida que se produce en RC0/T1OSO/T1CK.
- Puede operar de forma síncrona o asíncrona en función del bit **T1SYNC** (T1CON<2>).

TIMER 1: Modo Contador (II)

Para configurar el TIMER1 en modo contador en CCS se utiliza la función:

```
setup_timer_1( modo | T1_DIV_BY_N );
```

- modo puede tomar los valores:

T1_DISABLED, (Timer 1 desabilitado)
T1_EXTERNAL, (Timer 1 como contador asíncrono en RC0)
T1_EXTERNAL_SYNC (Timer 1 como contador síncrono en RC0/RC1)

- El valor de N puede ser uno de los siguientes: 1,2,4,8.

TIMER 1: Lectura y escritura

Para leer el contenido del TIMER1 se utiliza la función:
get_timer1();

Ejemplo:
```
int16 valor;                 // Declarar una variable de 16 bits
valor=get_timer1();     // Asignamos el valor del timer a la variable
```

Para escribir un valor en el registro TIMER0 se utiliza la función:
set_timer1(valor);

Ejemplo:
```
int valor=1500;          // Declarar una variable de 8 bits
set_timer1(valor);      // Asignamos el valor 1500 al Timer 1
```

o simplemente:
set_timer1(1500);

TIMER1: Generador de interrupción

PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

TMR1IE: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

TMR1IF: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

- Para habilitar la interrupción del TIMER1 hay que poner a **uno** el bit **TMR1IE** del registro PIE1.
- La interrupción se produce cuando el Timer pasa de 0xFFFF a 0x0000. En ese momento el bit **TMR1IF** se pone a uno PIR1<0>.
- El bit **TMR1IF** debe ponerse a cero por software antes de salir de la rutina de atención a la interrupción.

TIMER1: Generador de interrupción (CCS)

Para habilitar la interrupción del TIMER1 se utilizan las funciones:

```
setup_timer_1(T1_INTERNAL| T1_DIV_BY_N); // N=1,2,4 o 8
enable_interrupts(INT_TIMER1);
enable_interrupts(GLOBAL);
```

Para deshabilitar la interrupción:

```
disable_interrupts(INT_TIMER1);
```

La función de atención a la interrupción es:

```
#int_TIMER1
int TIMER1_isr()
{

}
```

Interrupción del TIMER1 en CCS (I)

Ejemplo:
Usar el Timer1 sin preescaler para realizar un programa que genere por el pin RB7 un tren de pulsos de frecuencia variable en función del valor del pin RA0. Para RA0=0 -> F=100Hz y para RA0=1 -> F= 200Hz. Al mismo tiempo en el LCD se mostrará el texto "F=100Hz en pin RB7" o "F=200Hz en pin RB7" según corresponda.

Interrupción del TIMER1 en CCS (II)

Paso 1: Configuración del TIMER1 como temporizador sin preescaler y habilitar la interrupción (dentro de la función main()):

```
setup_timer_1(T1_INTERNAL| T1_DIV_BY_1);
enable_interrupts(INT_TIMER1);
enable_interrupts(GLOBAL);
```

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

```
int F=0;                                  // Partimos con la frecuencia baja
#int_TIMER1
int TIMER1_isr()
{
output_toggle(PIN_B7);
if(F==0) set_timer1(65535-5000);  // Recargamos para 5ms ->100Hz
else
set_timer1(65535-2500);              // Recargamos para 2,5ms ->200Hz
return 0;
}
```

Interrupción del TIMER1 en CCS (III)

Paso 3: Completamos la función main() con los mensajes del LCD y la captura del estado del pin RA0:

```
while(true)
{
if(input(PIN_A0))  {                    // PIN_A0=1 -> F=200Hz,T=5ms,T/2=2,5ms
  if(primera_vez==1 || F==0  ) {   // Para evitar el parpadeo del LCD, solo se
    LCD_Borrar();                       // escribe cuando se produce el cambio.
    printf(LCD_Putc,"F=200 Hz");
    F=1;
    primera_vez=0;
   }
  }
 else {                                 // PIN_A0=0 -> F=100Hz,T=10ms,T/2=5ms
   if(primera_vez==1 || F==1)      // Para evitar el parpadeo del LCD, solo se
   {                                    // escribe cuando se produce el cambio.
    LCD_Borrar();
    printf(LCD_Putc,"F=100 Hz");
    F=0;
    primera_vez=0;
   }
  }}
```

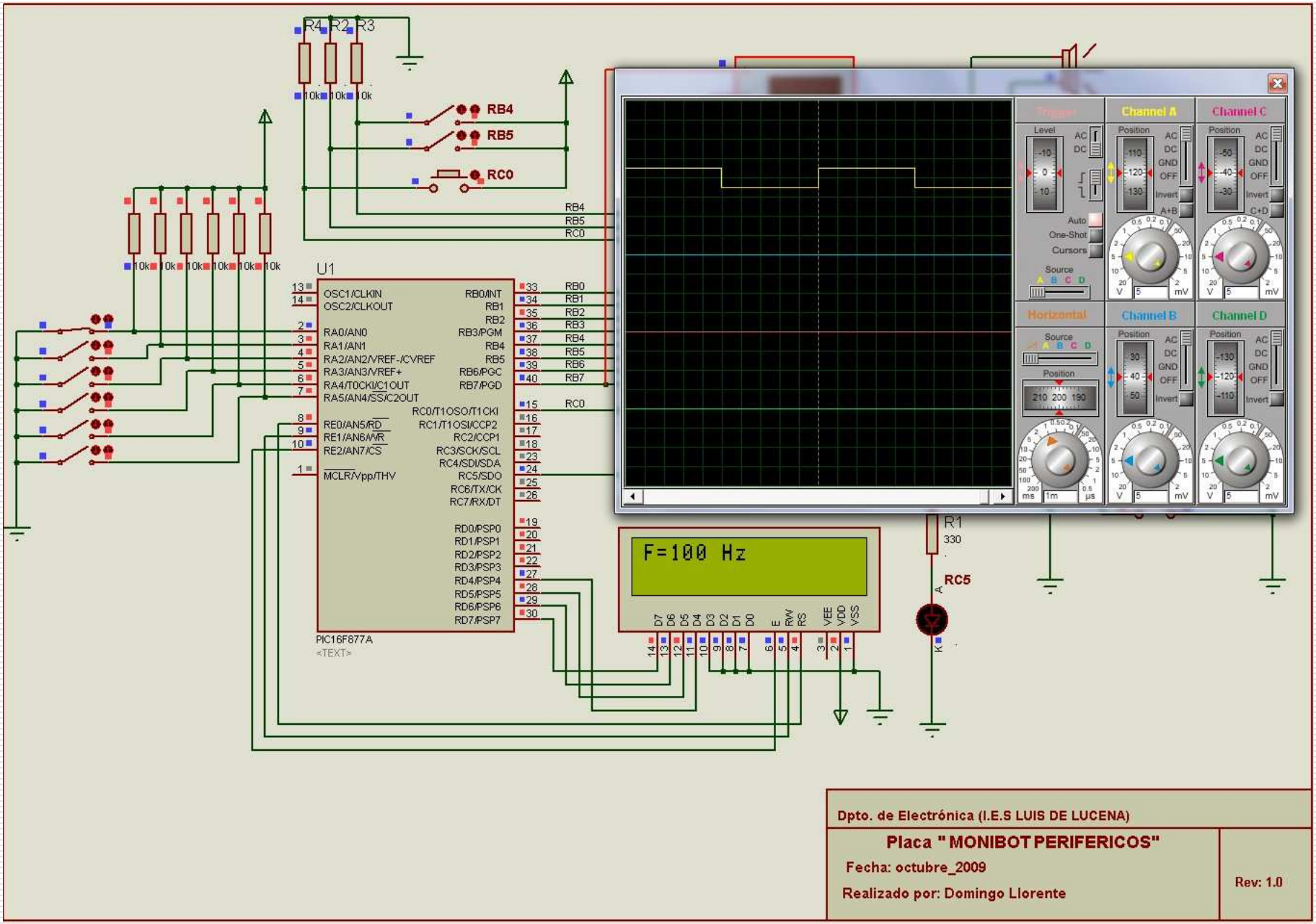
Programa completo (parte 1/2)

```
////////////////////////////////////////////////////////////////////////////////
// Programa......: inte_timer1.c                                              //
// Plataforma hw.: Placa monibot 16F877A                                      //
// Fecha.........: noviembre-2010                                             //
// Programado por: Domingo Llorente                                           //
// Descripción...: Programa que genera un tren de pulsos de freq. variable en //
// el pin RB7. Para RA0=0 =>100Hz, para RA0=1 => 200Hz                        //
////////////////////////////////////////////////////////////////////////////////
#include <16F877A.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP
#use delay(clock=4000000)
#include <lcd_monibot.c>
#use fast_io(b)              // Opcional usar fast o standar io
#use fast_io(a)

////////////  FUNCIóN DE INTERRUPCIÓN //////////
int F=0;   // Partimos con la frecuencia baja
#int_TIMER1
int TIMER1_isr()
{
output_toggle(PIN_B7);
if(F==0) set_timer1(65535-5000);      // Recargamos para 5ms ->100Hz
else
set_timer1(65535-2500);               // Recargamos para 2,5ms ->200Hz
}

int primera_vez=1;

```

Programa completo (parte 2/2)

```
28
29  void main()
30  {
31  setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);    // Timer1,interno,preescaler=1
32  enable_interrupts(INT_TIMER1);              // Habilitamos int. del timer1
33  enable_interrupts(GLOBAL);                  // Habilitar interrupciones
34  set_timer1(65535-5000);                     // Recargamos para 5ms ->100Hz
35  setup_adc_ports(NO_ANALOGS);                // Deshabilitamos las ent. analogicas
36  set_tris_b(0x00);                           // RB0..RB7 como salidas
37  set_tris_a(0xFF);                           // RA0..RA5 como entradas
38  LCD_Init();
39
40  while(true)
41  {
42  if(input(PIN_A0))                 // PIN_A0=1 -> F=200Hz,T=5ms,T/2=2,5ms
43    {
44    if(primera_vez==1 || F==0  )    // Para evitar el parpadeo del LCD, solo se
45     {                              // escribe cuando se produce el cambio.
46      LCD_Borrar();
47      printf(LCD_Putc,"F=200 Hz");
48      F=1;
49      primera_vez=0;
50     }
51    }
52   else                             // PIN_A0=0 -> F=100Hz,T=10ms,T/2=5ms
53    {
54    if(primera_vez==1 || F==1)      // Para evitar el parpadeo del LCD, solo se
55     {                              // escribe cuando se produce el cambio.
56      LCD_Borrar();
57      printf(LCD_Putc,"F=100 Hz");
58      F=0;
59      primera_vez=0;
60     }
61    }
62  }
63  }
```

Simulación (100Hz)



Simulación (200Hz)