Chapter 9

Timer/Counter Modules

In the PIC18F1220 and PIC18F1320 there is 1×8 bit and 3×16 bit timers, called TMR0, TMR1, TMR2, and TMR3.

We will discuss TMR0, the operation of the other timer/counter modules can be readily understood using the Microchip User Guide.

The Timer0 module has the following features:

- Software selectable as an 8 bit or 16 bit timer/counter.
- Readable and writeable.
- It has a dedicated software programmable prescaler.
- Its clock source can be configured as internal or external.
- It has an interrupt on overflow from 0xFF to 0×00 in 8 bit mode and 0xFFFF to 0×0000 in 16 bit mode.
- There is an edge select for the external clock.

The TMR0 control register is shown in Figure 9.1.

TMR0 can operate as a timer or counter.

The Timer mode is selected by clearing the T0CS bit. When TMR0 is written to it is inhibited for two instruction cycles.

The Counter mode is accessed by setting the T0CS bit. In the counter mode TMR0 will increment on every rising or falling edge of pin RA4/TOCKI selected by T0SE bit.

TMR0 PRESCALER

TMR0 like all timers can run off the internal instruction cycle clock which runs at one-fourth of the clock frequency.

If the clock frequency was 4 MHz then the timing frequency is 1 MHz or a timing period of 1 µs.

If the Prescaler bits T0PS2:T0PS0 are set to 010 (see Figure 9.1) then the timing period is $8 \times 1\,\mu s = 8\,\mu s$.

If we now count 125 of these pulses $125 \times 8\,\mu s$ then we have 1 ms time period.

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	T0PS2	T0PS1	T0PS0
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 an 16-bit timer/counter

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

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 T0PS2:T0PS0: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:258 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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 9.1 TMR0 Register, T0CON.

TIMING AN EVENT

So to time an event in ms, say a positive pulse on RA0

```
1  T0CON=0b11000010;
2  while (PORTAbits.RA0== 1);
3  {
4  TMR0L=131;
5  while (TMR0L!=0);
6  Count=Count++;
7  }
```

- Line1 turns TMR0 on—bit7, sets TMR0 to 8 bit—bit6, internal instruction cycle clock set with bit5, assigns the Prescaler with bit3, selects a prescale value of 8 with T0PS2:0 as 010. See Figure 9.1.
- Line 3–7 executes the while loop every millisecond, while RA0 is high, counting 1 ms periods.

- Line 4 preloads TMR0L with 131 so it overflows in 125 more timing periods, i.e., 1 ms.
- Line 5 waits until TMR0 has overflowed (takes 1 ms) and
- Line 6 increments the Count file, counting milliseconds.

When RA0 goes low the loop finishes and Count contains the number of milliseconds taken.

AN ACCURATE 1 S TIME PERIOD

Unfortunately by juggling the numbers for the internal clock and prescaler we cannot make a time of exactly 1 s. To do this we will need to use an external crystal for accurate timing. The internal oscillator has an accuracy of 1% that is about 15 min every 24 h, not good enough for a clock! The crystal on the other hand has an accuracy of 20 ppm that is about 1.7 s every 24 h.

The microcontroller using the 32.768 kHz watch crystal is shown in Figure 9.2.

The crystal is connected to the OSC1 and OSC2 pins which are also RA7 and RA6, so RA7 and RA6 are no longer available as I/O.

The instruction timing frequency is 32768/4 = 8192 Hz, if we use a prescaler of 256 then this would give a timing frequency of 8192/256 = 32 Hz. So counting 32 timing pulses with TMR0 would give a time of exactly 1 s. Sixteen pulses would be 0.5 s, 160 pulses would be 5 s.

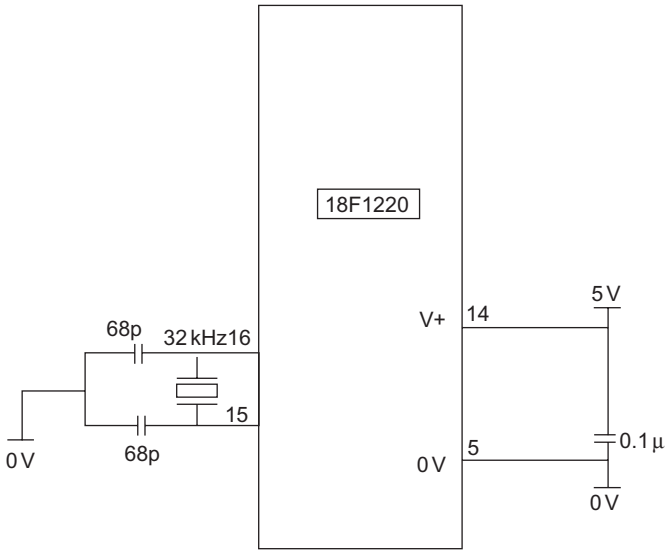

FIGURE 9.2 Connecting the crystal to the 18F1220.

The program using this would be:

```
1.  //Delay1S.c DW Smith 1-2-12
2.  #include <p18f1220.h>
3.  #pragma config WDT=OFF, OSC=LP, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  void one_sec_delay (void)
6.  {
7.  TMR0L=0;
8.  while (TMR0L<32);
9.  }
10. void main (void)
11. {
12. //SET UP
13.       ADCON1=0×7F;          //all IO are digital or 0b01111111 in binary
14.       TRISA=0b11111111;     //sets PORTA as all inputs
15.       PORTA=0b00000000;     //turns off PORTA outputs, not required, no outputs
16.       TRISB=0b00000000;     //sets PORTB as all outputs
17.       PORTB=0b00000000;     //turns off PORTB outputs, good start position
18.       T0CON=0b11000111;     //sets TMR0 on, 8its, internal clock, prescaler assigned,
                                //prescaler=256
19. one_sec_delay ();
20. while(1);
21. }
```

- Line 3 has been altered to show the new oscillator configuration, OSC=LP, which is the code for the low power 32 kHz crystal.
- Line 18 configures T0CON register.
- Lines 5–9 are the one_sec_delay routine.
- Line 7 sets TMR0L to 0.
- Line 8 waits until TMR0=32.
- Line 19 calls the one_sec_delay
- Line 20 stops the program waiting while "1" which is always true.

AN ACCURATE 1 MIN DELAY

Once again we use the 32.768 kHz crystal with the circuit of Figure 9.2.

Using the prescaler set to 256 gives 32 pulses/s as in the 1 s timer.

For a 1 min delay we need 32 × 60 pulses = 1920 pulses. 1920 pulses cannot be counted in one 8 bit timer so we need to use TMR0 as a 16 bit counter using TMR0H and TMR0L.

1920d is 0 ×0780 , so TMR0H counts to 0 × 07 and TMR0L counts to 0 × 80.

The program to introduce the 1 min delay, **Delay1min.c** is shown below.

```
1.  //Delay1min.c DW Smith 1-2-12
2.  #include <p18f1220.h>
3.  #pragma config WDT=OFF, OSC=LP, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  int TMR0x;
6.  void one_min_delay (void)
```

```
7.    {
8.          TMR0H=0;
9.          TMR0L=0;
10.         while (TMR0H<0×7)          TMR0x=TMR0L;
11.         while (TMR0L<0×80);
12.   }
13.   void main (void)
14.   {
15.         //SET UP
16.         ADCON1=0×7F;               //all IO are digital or 0b01111111 in binary
17.         TRISA=0b11111111;          //sets PORTA as all inputs
18.         PORTA=0b00000000;          //turns off PORTA outputs, not required, no outputs
19.         TRISB=0b00000000;          //sets PORTB as all outputs
20.         PORTB=0b00000000;          //turns off PORTB outputs, good start position
21.         T0CON=0b10000111;          /*sets TMR0 on, 16bits, internal clock, prescaler
                                       assigned, prescaler= 256 */
22.         one_min_delay ();
23.         while(1);
24.   }
```

Lines 6–12 contain the 1 min delay routine.

Lines 8 and 9 set the timer registers TMR0H and TMR0L to zero. NB. TMR0H is buffered and is only written to when TMR0L is written to. So write the data to TMR0H before writing to TMR0L.Writing to TMR0L then clocks both writes.

Line 10 waits until TMR0H has reached 0×07. NB TMR0H is buffered and is only read when TMR0L is read, by TMR0x = TMR0L.

Line 11 waits until TMR0L has reached 0×80.

Line 21 sets T0CON.

24 H CLOCK

Now that we have an accurate 1 min delay we can write the code for a 24 h clock.

We will need to display the time on an alpha numeric display so we will require each of the hours and minutes as single numbers, say:

HourTen HourUnit: MinuteTen MinuteUnit.

The clock works as follows:

- After each 1 min delay, MinuteUnit will be incremented.
- If MinuteUnit = 10, then make MinuteUnit = 0 and add 1 to MinuteTen.
- If MinuteTen = 6, then make MinuteTen = 0 and add 1 to HourUnit.
- If HourUnit = 10 then make HourUnit = 0 and add 1 to HourTen.
- If HourTen = 2 and HourUnit = 4 then make HourTen = 0, HourUnit = 0.

The code is illustrated below in **24HourClock.C**

```
1     // 24HourClock.C by DW Smith, 20 February 2012
2     //E on B7, RS on B6, D4-7 on A0-3. A7 and A6 being used by crystal.
```

```
#include <p18f1220.h>
#pragma config WDT=OFF, OSC=LP, PWRT=ON, LVP=OFF, MCLRE=OFF
#include <delays.h>
#include <dwsLCD2.h>
int TMR0x;
int HourTen=0, HourUnit=0, MinuteTen=0, MinuteUnit=0;
void one_min_delay (void)
{
TMR0H =0;          //when writing load TMR0H first, written on TMR0 write.
TMR0L=0;
while (TMR0H<0×7) TMR0x=TMR0L;
while (TMR0L<0×80);
}
void main (void)
{
//SET UP
ADCON1=0×7F;                    //all IO are digital or 0b01111111 in binary
TRISA=0b00100000;               //sets PORTA
PORTA=0b00000000;               //turns off PORTA outputs, not required, no outputs
TRISB=0b00000000;               //sets PORTB as all outputs
PORTB=0b00000000;               //turns off PORTB outputs, good start position
T0CON=0b10000111; //sets TMR0 on, 16bits, internal clock, prescaler assigned, prescaler=256
// this code configures the display
WriteCmd (0×02);                                  // sets 4bit operation
WriteCmd (FOUR_BIT & LINES_5×7);                  // sets 5×7 and multiline operation.
WriteCmd (CURSOR_BLINK);                          // blinks cursor
WriteCmd (CURSOR_RIGHT);                          // moves cursor right
WriteCmd (CLEAR_SCREEN);
   while(1)
{
one_min_delay ();
MinuteUnit=MinuteUnit +1;
if (MinuteUnit == 10)
   {
MinuteUnit=0;
MinuteTen=MinuteTen+1;
if (MinuteTen == 6)
      {
MinuteTen=0;
HourUnit=HourUnit +1;
if (HourUnit == 10)
         {
HourUnit=0;
HourTen=HourTen+1;
         }
      }
   }
if (HourTen == 2 && HourUnit == 4)
   {
HourUnit=0;
HourTen=0;
   }
WriteChar(0×30 + HourTen);
```

```
56    WriteChar(0×30 + HourUnit);
57    WriteChar(0×3A);              //colon
58    WriteChar(0×30 + MinuteTen);
59    WriteChar(0×30 + MinuteUnit);
60       }
61    }
```

Explanation of the Code

- Line 2 comments that the crystal occupies RA6 and RA7. Note new position of E and RS. LCD.h changed to LCD2.h with new positions for E and RS.
- Line 4 now shows the oscillator is the low power 32.768 kHz crystal, OSC = LP.
- Line 6 has included dwsLCD2.h
- Line 7 defines TMR0x where TMR0L is going to be read into. Enabling the TMR0H buffer to be read.
- Line 8 declares HourTen, HourUnit, MinuteTen, and MinuteUnit as integers and sets them to 0. They could have been set to any 24 h clock value or set with a keypad.
- Lines 9–15 is the 1 min delay code.
- The Main code is between lines 17 and 61.
- Line 24 configures the Timer0 Control Register.
- Lines 26–30 configures the Alpha Numeric Display.
- Line 33–53 updates the time every minute.
- Lines 55–59 writes the time to the display.

The 16 bit TMR0 can time 0xFFFF = 65,535 pulses.

With a 32.768 kHz oscillator that is a maximum time of 65,535 × 1/32 s = 2048S = 34 min.

Longer times can of course be computed by counting these 34 min times.

30 MIN DELAYS AND LONGER

As we have seen using a 32.768 kHz crystal and setting the prescaler to/256 gives timing pulses of 32/s. In 30 min (1800 s) we would have 32 × 1800 = 57,600 pulses.

57600 in hex is 0xE100, so when TMR0H reaches 0xE1 and TMR0L reaches 0 we will have had 30 min.

The code for this is:

```
Void Delay30 min(void)
{
TMR0H=0;
TMR0L=0;
while(TMR0H<0xE1) TMR0temp=TMR0L;          //to read TMR0H we
must read TMR0L
}
```

Now that we have a 30 min delay a **1 h delay** would be:

```
for(int i =0; i<2; i++)          // loops until i=2.
{
Delay30 min(void)
}
```

A **24 h delay** would be:

```
for(int i =0; i<48; i++)          // loops until i=48
{
Delay30 min(void)
}
```

A **1 week delay** would be:

```
for(int i =0; i<336; i++)          // loops until i=336
{
Delay30 min(void)
}
```

How long do you want? 1 year, 365 days=365×24=8760 hours= 17520×30 min

A **1 year delay** would be:

```
for(int i =0; i<17520; i++)          // loops until i=17520
{
Delay30 min(void)
}
```

Try doing a **10 year delay**, if it doesn't work, let me know!! I hope I'm still around.