TIMER FUNCTIONS

Original by F.Zia, rewritten for the PIC18F4550 by M.F.v.Lieshout, March 2006

Information sources:

The information in this document is obtained from the following Microchip manuals:

- PIC18F4550 Datasheet
- PICmicro® 18C MCU Family Reference Manual
- MPLAB C18 C Compiler Libraries

Function Prototypes:

For a detailed description of these functions, please see:
Section 2.9 Timer Functions, in MPLAB C18 C Compiler Libraries manual.

```
#include <timers.h>
void OpenTimer0( unsigned char config0 ); // Configure and enable timer x.
void OpenTimer1( unsigned char config1 );
void OpenTimer2( unsigned char config2 );
void OpenTimer3( unsigned char config3 );
void WriteTimer0( unsigned int value16 ); // Write a value into timer x.
void WriteTimer1( unsigned int value16 );
void WriteTimer2( unsigned char value_8 );
void WriteTimer3( unsigned int value16 );
unsigned int ReadTimer0( void ); // Read the value of timer x.
unsigned int ReadTimer1( void );
unsigned char ReadTimer2( void );
unsigned int ReadTimer3( void );
void CloseTimer0( void ); // Disable timer x.
void CloseTimer1( void );
void CloseTimer2( void );
void CloseTimer3( void );
```

Notes:

1. Timers are the most versatile internal peripheral devices in a PIC18F4550. They can be used to time internal or external events through polling or by generating an interrupt. Some timers can be configured as counters for counting internal or external events such as switch closure or a pulse train. In addition, some timers are used as part of other internal peripherals to control their operation as outlined below.

2. Timer register(s) for `ReadTimerx` and `WriteTimerx` functions:

- Timer0 (16 bit): TMR0L,TMR0H
- Timer1 (16 bit): TMR1L,TMR1H
- Timer2 (8 bit): TMR2
- Timer3 (16 bit): TMR3L,TMR3H

3. When using a timer in 8-bit mode that may be configured in 16-bit mode (e.g., TIMER0), the upper byte is not guaranteed to be zero. The user may wish to cast the result to a char for correct results. For example:

```
// Example of reading a 16-bit result from a
// 16-bit timer operating in 8-bit mode:
unsigned int result;
result = (unsigned char) ReadTimer0();
```

4. TIMER0 features:

a. Software selectable as an 8-bit or 16-bit timer/counter
b. Readable and writable
c. Dedicated 8-bit software programmable prescaler
d. Clock source selectable to be external or internal
e. Interrupt on overflow from FFh to 00h (FFFFh to 0000h in 16-bit mode)

f. Edge select for external clock
5. TIMER1 features:
a. 16-bit timer/counter (two 8-bit registers; TMR1H and TMR1L)
b. Readable and writable (both registers)
c. Internal or external (e.g. additional 32 kHz crystal) clock select. In real-time applications, SLEEP does not disable the external TIMER1 oscillator
d. Interrupt-on-overflow from FFFFh to 0000h
e. RESET from CCP module special event trigger
6. TIMER2 Features:
a. 8-bit timer (TMR2 register)
b. 8-bit period register (PR2)
c. Readable and writable (both registers)
d. Software programmable prescaler (1:1, 1:4, 1:16)
e. Software programmable postscaler (1:1 to 1:16)
f. Interrupt on TMR2 match of PR2
g. SSP module optional use of TMR2 output to generate clock shift
7. TIMER2 Notes:
a. Using the prescaler and postscaler at their maximum settings, the overflow time is the same as a 16-bit timer.
b. If the PR2 register = 00h, the TMR2 register will not increment (Timer2 cleared).
c. When T2CON is written, TMR2 does not clear.
d. During sleep, TMR2 will not increment. The prescaler will retain the last prescale count, ready for operation to resume after the device wakes from sleep.
8. TIMER3 features:
a. 16-bit timer/counter (two 8-bit registers; TMR3H and TMR3L)
b. Readable and writable (both registers)
c. Internal or external clock select
d. Interrupt-on-overflow from FFFFh to 0000h
e. RESET from CCP module trigger
9. TIMER3 Notes:
a. You should never write to TIMER3 in situations where that could cause the loss of time. In most cases, that means you should not write to the **TMR3L** register, but if the conditions are ok, you may write to the **TMR3H** register, e.g. if you want the Timer3 overflow interrupt to be sooner than the full 16-bit time-out.