DOCUMENTATION for TIMER module

Ed Carryer
Rev : January 10, 2001 to remove reference to limited function version for
cockroaches
December 29, 2001 to remove example that used SES.

PURPOSE OF MODULE

This module provides for 8 independent timer channels. These timers can be started, stopped and tested for completion. All timers share a common time base, and that timebase is selectable from the available RTI rates defined below.

INTERFACE

Hardware/Output Specifications

This is a software only module, with no hardware interactions.

Defined Constants

These constants are used with TMR_Init() to set the timebase for all of the timers.

TMR_RATE_4MS	4.1mS
TMR_RATE_8MS	8.19 mS
TMR_RATE_16MS	16.38mS
TMR_RATE_32MS	32.77mS

MODULE FUNCTIONS

TMR_Init

Prototype: void TMR_Init(unsigned char Rate)
Parameters: unsigned char Rate
set to one of the TMR_RATE_XX defines to set the RTI rate
Returns: None.

Description
Initializes the timer module by attaching the RTI interrupt to the response routine. Must be called before using any of the other TMR routines.

TMR_InitTimer

Prototype: signed char TMR_InitTimer(unsigned char Num, unsigned int NewTime)
Parameters: unsigned char Num, the number of the timer to start
unsigned int NewTime, the number of tick to be counted
Returns: -1 if the requested timer does not exist, 0 otherwise.

Description
Sets the NewTime into the chosen timer and clears any previous event flag and sets the timer active to begin counting.

TMR_SetTimer

Prototype: signed char TMR_SetTimer(unsigned char Num, unsigned int NewTime)
Parameters: unsigned char Num, the number of the timer to set.
Returns: -1 if requested timer does not exist, 0 otherwise

Description
Sets the time for a timer, but does not make it active.

TMR_StartTimer

Prototype: signed char TMR_StartTimer(unsigned char Num)
Parameters: unsigned char Num the number of the timer to start
Returns: signed char -1 for error 0 for success

Description
Sets the active flag in TMR_ActiveFlags to start a timer that was set or to resart a stopped timer.

TMR_StopTimer

Prototype: signed char TMR_StopTimer(unsigned char Num)
Parameters: unsigned char Num the number of the timer to stop.
Returns -1 for error (timer doesn't exist) 0 for success.

Description
Clears the bit in TMR_ActiveFlags associated with this timer. This will cause it to stop counting.

TMR_IsTimerActive

Prototype: signed char TMR_IsTimerActive(unsigned char Num)
Parameters: unsigned char Num the number of the timer to check
Returns: -1 if requested timer is not valid
0 if timer is not active
1 if it is active

Description
This functions is used to determine if a timer is currently counting.

TMR_IsTimerExpired

Prototype: signed char TMR_IsTimerExpired(unsigned char Num)
Parameters: unsigned char Num, the number of the timer to test.
Returns -1 if reqiested timer does not exist
0 if not expired
1 if expired

Description
This function tests the flags to determine if the requested timer has expired.

TMR_ClearTimerExpired

Prototype: signed char TMR_ClearTimerExpired(unsigned char Num)
Parameters unsigned char Num,
the timer whose event flag should be cleared.
Returns -1 if requested timer does not exist
0 otherwise

Description
Clears the appropriate bit in TMR_EventFlags to show that the event has been serviced.

CONSTRAINTS/NOTES

1. TMR_Init must be called before the module becomes active.

2. The time-base of all timers is the same .

THEORY OF OPERATION

This module operates using RTI interrupt as the time-base. At each occurrence of the interrupt, the 'time remaining' count for each of the active timers is decremented. If the count for a timer goes to 0, the Expired flag for that timer is set, and the timer is made inactive.

Usage Example

```
#include <timer.h>

/* TIME_OUT_DELAY  = 10 S w/ 8mS interval */
#define TIME_OUT_DELAY 1220

void main(void)
{
   unsigned int i;

   puts("Starting\n");
   TMR_Init(TMR_RATE_8MS);
   TMR_InitTimer(0, TIME_OUT_DELAY);
   TMR_InitTimer(1, TIME_OUT_DELAY);
   TMR_InitTimer(2, TIME_OUT_DELAY);
   TMR_InitTimer(3, TIME_OUT_DELAY);
   TMR_InitTimer(4, TIME_OUT_DELAY);
   TMR_InitTimer(5, TIME_OUT_DELAY);
   TMR_InitTimer(6, TIME_OUT_DELAY);
   TMR_InitTimer(7, TIME_OUT_DELAY);
   while(TMR_IsTimerExpired(0) != 1)
      ;
   puts("Timed Out\a\n");
   TMR_InitTimer(7, TIME_OUT_DELAY);
   for (i=0;i<10,000 ;i++ )
   {/* kill some time */
   }
   TMR_StopTimer(7);
   if (TMR_IsTimerActive(7) != 0)
      puts("Timer Stop Failed\n");
   else
      puts("Timer Stop Succeded\n");
   TMR_StartTimer(7);
   while(TMR_IsTimerExpired(7) != 1)
      ;

   puts("Timed Out Again\a\n");

}
```