Modifier Version Date Changes

Ross Kristof v0.0 3/15/11 Created

Ross Kristof v0.1 3/21/11 Updated

Ross Kristof v0.2 4/14/11 Updated

Ross Kristof v0.3 5/5/11 Updated

Timer Library Reference Manual

**Introduction:**

This document covers the use and configuration of the timer functionality of the Cerebot 32MX4 and Cerebot 32MX7 microcontrollers.

**Timer Overview:**

The timers included with the Cerebot 32MX4 and Cerebot 32MX7 are used to synchronize events in your program. They support both interrupt driven events as well as direct access.

**Timer Availability & Proper Usage:**

The Cerebot 32MX4 has 5 16-bit timers available for usage. Timers 2 and 3 and timers 4 and 5 can be combined to form 2 32-bit timers (Timer23 and Timer45). It is important to note that, when using a 32-bit timer, the two 16-bit timers will be unavailable.

**Timer Library Functions:**

**Description**

The standard usage functions for Timers such as initializing and reading/writing from a timer are contained here. The timer library is divided into two feature sets:

Basic:

The basic timer functionality can be set by doing the following:

1. Right click on your project file (in the TimerExample.mcw window in the example)
2. Select **Build Options…**
3. Click on the **MPLAB PIC32 C Compiler** tab
4. Under **Preprocessor Macros** click **Add…**
5. Enter **TIMER\_n** where **n** is the timer number you want

**unsigned char fnTimer1Setup(unsigned int ulMS)**

parameters:

ulMS - Duration of the timer in milliseconds

returns:

0 on failure

1 on success

Initializes Timer1 and enables Timer1’s interrupt.

**void fnTimer1IntHandler()**

parameters:

none

This is the interrupt handler function for Timer1. The user must define this function.

**unsigned int fnReadTimer1()**

parameters:

none

returns:

value of the timer, in ms.

Read the current time value of the timer.

**unsigned char fnSetTimer1(unsigned int ulMS)**

parameters:

ulMS - Duration of the timer in milliseconds

returns:

0 for failure

1 for success

This function will sets the time value of the timer.

**void fnCloseTimer1()**

parameters:

none

Closes the timer

Advanced:

The advanced timer functionality can be set by doing the following:

1. Right click on your project file (in the TimerExample.mcw window in the example)
2. Select **Build Options…**
3. Click on the **MPLAB PIC32 C Compiler** tab
4. Under **Preprocessor Macros** click **Add…**
5. Enter **ADVTIMER\_n** where n is the timer number you want

**IMPORTANT!!!**

If you are using both the ADVTIMER\_**n** and TIMER\_**n** together, the fnAdvTimer**n**IntHandler must be used instead of fnTimer**n**IntHandler.

**unsigned char fnAdvTimer1Setup(unsigned int ulConfig, unsigned int ulPeriod)**

parameters:

ulConfig - bitwise or of configuration constants. See Microchip timer library for more information

ulPeriod - period of the timer

returns:

0 on failure

1 on success

Initializes Timer1 and enables Timer1’s interrupt

**void fnAdvTimer1IntHandler()**

parameters:

none

The interrupt handler function for Timer1. The user must define this function.

**unsigned int fnAdvReadTimer1()**

parameters:

none

returns:

value of the timer

gets the current value of the timer.

**void fnAdvSetTimer1(unsigned int ulValue)**

parameters:

ulValue - the new value for the timer

Sets the value of timer1

**unsigned int fnAdvReadTimer1Period()**

parameters:

none

returns:

period of the timer

gets the period of the timer.

**unsigned char fnAdvSetTimer1Period(unsigned int ulPeriod)**

parameters:

ulPeriod - period of the timer

returns:

0 on failure

1 on success

Sets the Timer1 period value

**void fnAdvConfigTimer1 (unsigned int ulConfig)**

parameters:

ulConfig - bitwise or of configuration constants. See Microchip timer library for more information

configures the timer to the passed config

**void fnAdvEnableTimer1()**

parameters:

none

Enables the timer

**void fnAdvDisableTimer1()**

parameters:

none

Disables the timer

**void fnAdvTimer1IntPriority(unsigned int ulPriority)**

parameters:

ulPriority - a value of T1\_INT\_PRIOR\_XXX

Sets the interrupt priority of the timer

**void fnAdvCloseTimer1()**

parameters:

none

Closes the timer