

# 1 Millisecond Interval Timer Datasheet MSTIMER V 1.2

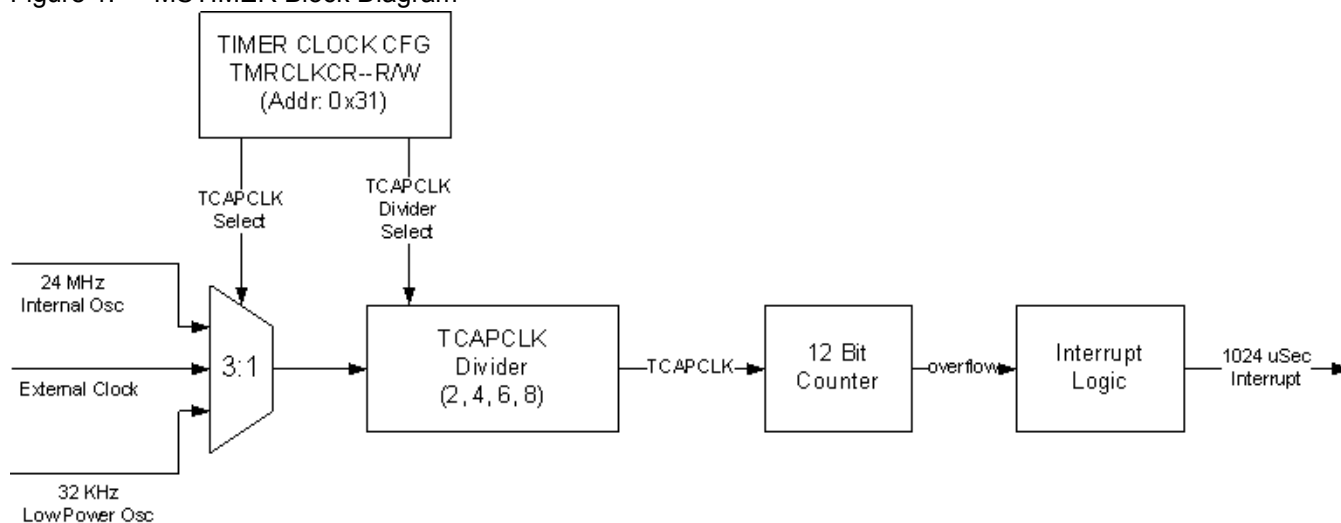
Copyright © 2005-2014 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	API Memory (Bytes)		Pins (per External I/O)
	Flash	RAM	
CY7C639/638/633/602/601xx, CYRF69xx3	19 Bytes	0	None

## Features and Overview

- 12-bit interval timer
- Input clock: enCoRe II TCAPCLK clock
- Provides a 1 Millisecond interrupt with the TCAPCLK Divider = 6 (24 MHz/6 = 4 MHz)
- Source clock rates up to 24 MHz
- Designed for use with the USB User Module for USB Suspend handling

Figure 1. MSTIMER Block Diagram



## Functional Description

The MSTIMER User Module provides convenient access to the "1024 microsecond" feature of the enCoRe II.

## DC and AC Electrical Characteristics

See the device datasheet for your enCoRe II device for the electrical characteristics of the millisecond timer.

## Placement

The MSTIMER User Module occupies the MSTIMER block. Alternate placement is not available.

## Parameters and Resources

The MSTIMER User Module does not have any parameters.

## Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to enable you to deal with the module at a higher level. This section specifies the interface to each function together with the related constants provided by the include files.

The standard method of using this feature is to add custom user code to the interrupt service routine found in the MSTIMERINT.ASM file in the project library directory. In the application code, enable the MSTIMER interrupt by calling the MSTIMER\_EnableInt to begin the periodic calling of the interrupt service routine, and thereby calling the custom user code. When the MSTIMER functionality is no longer needed, call MSTIMER\_DisableInt to disable the interrupt service routine from being called. Enable or disable the MSTIMER interrupt repeatedly as needed by the application.

The API programming routines for MSTIMER follow:

### MSTIMER\_EnableInt

#### Description:

Enables the interrupt mode operation. The MSTIMER\_ISR function in the MSTIMERINT.ASM file is called on 1 ms periods. You must enable global interrupts before the interrupts are actually serviced.

#### C Prototype:

```
void  MSTIMER_EnableInt(void);
```

#### Assembly:

```
lcall  MSTIMER_EnableInt
```

#### Parameters:

None

#### Return Value:

None

#### Side Effects:

This routine modifies the appropriate interrupt enable register in I/O space. You can alter the A and X registers by this function.

## MSTIMER\_DisableInt

**Description:**

Disables the interrupt mode operation.

**C Prototype:**

```
void MSTIMER_DisableInt(void);
```

**Assembly:**

```
lcall MSTIMER_DisableInt
```

**Parameters:**

None

**Return Value:**

None

**Side Effects:**

This routine modifies the appropriate interrupt enable register in IO space. You can alter the A and X registers by this function.

## MSTIMER\_Start

**Description:**

Given for interface consistency only. Does nothing.

**C Prototype:**

```
void MSTIMER_Start(void);
```

**Assembly:**

```
lcall MSTIMER_Start
```

**Parameters:**

None

**Return Value:**

None

**Side Effects:**

You can alter the A and X registers by this function.

## MSTIMER\_Stop

### Description:

Disables the MSTIMER by disabling the interrupt.

### C Prototype:

```
void MSTIMER_Stop(void);
```

### Assembly:

```
lcall MSTIMER_Stop
```

### Parameters:

None

### Return Value:

None

### Side Effects:

You can alter the A and X registers by this function.

## Sample Firmware Source Code

The following code is an example of how to use the MSTIMER User Module. The code sample in assembly language is:

```
include "m8c.inc"      ; part specific constants and macros
include "memory.inc"   ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"  ; PSoC API definitions for all User Modules

export _main

_main:

    M8C_EnableGInt ; Enable Global Interrupts
    lcall MSTIMER_EnableInt; Enable MSTIMER
                                ; MSTIMER interrupts will arise every 1 ms;

.terminate:
    jmp .terminate
```

The same code written in C is:

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

void main(void)
{
    M8C_EnableGInt ; // Enable Global Interrupts
    MSTIMER_EnableInt(); // Enable MSTIMER
                                // MSTIMER interrupts will arise every 1 ms;
}
```

## Version History

Version	Originator	Description
1.2	DHA	Added Version History.
1.2.b	DHA	Added Sample Firmware Source Code section to this user module datasheet.

**Note** PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2005-2014 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.