M16C/62
Using the M16C/62 Timer in Pulse Period/Width Measurement Mode

1.0 Abstract
Measuring the frequency (1/period) or the pulse width of an input signal is useful in applications such as tachometers, DC motor control, power usage calculations, and so on. The following article describes how to use timer B to measure the period and pulse width of an input waveform, referred to as “Pulse Period/Pulse Width Measurement Mode”.

2.0 Introduction
The M16C/62 is a 16-bit MCU, based on the M16C CPU core, with an impressive list of features including 10-bit A/D, D/A, UARTS, timers, DMA, etc., and up to 256k bytes of user flash. The MCU has 6 B timers and all 6 timers can operate in Pulse Period/Pulse Width Measurement Mode.

Timer B has the following additional modes of operation:
• Timer Mode
• Event Counter Mode

Figure 1 illustrates the operation of timer B. The remainder of this article focuses on setting up timer B0 to measure pulse width, and timer B1 to measure pulse period.

Figure 1 Block Diagram of Timer B
3.0 Pulse Period Measurement

In Period Measurement Mode (e.g. falling edge to falling edge), after the ‘start count flag’ is set, the counter counts up using the selected clock source and every time a falling edge is detected on the TBiIN pin, the value in the counter is transferred to the reload register, the counter is reset to zero, and then continues counting. At the same time, the timer interrupt request bit is set and an interrupt is generated if the timer interrupt priority level is set above the current CPU priority level (if the I flag in the CPU flag register is cleared, the interrupt will not be serviced until the flag is set). If the timer’s counter overflows within a period, it will also generate the interrupt and the MR3 bit in the TBiMR is set to distinguish between the interrupt causes. Note that the measurement is free running and the reload register contains the most recent measurement. The user has the option of polling the TBi register or reading it in an interrupt service routine. Also note that the value of the counter immediately after the ‘start count flag’ is set is indeterminate and an overflow could occur before the first falling edge. Figure 2 illustrates this.

3.1 Pulse Width Measurement

Pulse Width Measurement Mode operates in much the same way except the count register is transferred to the reload register for every edge detected on the TBiIN pin, and the counter resets and resumes counting, as shown in Figure 3. Again, note that the value of the counter immediately after the ‘start count flag’ is set is indeterminate and an overflow could occur before the first falling edge. This measurement is also free running but now the user must determine by software whether the measurement is for the high or low width.

![Figure 2 Operation Timing When Measuring a Pulse Period](image-url)
3.2 Pulse Width Measurement
Pulse Width Measurement Mode operates in much the same way except the count register is transferred to the reload register for every edge detected on the TBiIN pin, and the counter resets and resumes counting, as shown in 0. Again, note that the value of the counter immediately after the ‘start count flag’ is set is indeterminate and an overflow could occur before the first falling edge. This measurement is also free running but now the user must determine by software whether the measurement is for the high or low width.

Figure 3 Operation Timing When Measuring a Pulse Width

4.0 Configuring Pulse Period/Pulse Width Measurement Mode
To configure a timer for Pulse Period/Pulse Width Measurement Mode:

1. Load the timer mode register, TBiMR.
   - Select Measurement Mode: bits TMOD0 = 0, TMOD1 = 1.
   - Set the MR0 and MR1 bits for period or width measurement.
   - Clear the MR2 bit for period or width measurement.
   - MR3 is the timer Bi overflow flag (can be cleared but not set).
   - Select the clock source (f1, f/8, f/32, or fc/32): bits TCK0, TCK1 register.
2. Set the timer ‘interrupt priority level’, TBiIC, to at least 1 if required.
3. Enable interrupts (CPU I flag set).
4. Set the ‘start count flag’ bit, TBiS, in the ‘count start flag’ register, TABSR or TBSR.
It is not necessary to perform these steps in the order listed, but the mode register should be loaded before the 'start count' flag is set. Also, the priority level should not be modified when there is a possibility of an interrupt occurring.

The required registers are shown in Figure 4 through Figure 8.

---

**Figure 4 Timer Bi Mode Register in Pulse Period/Pulse Width Measurement Mode**

**Figure 5 Timer Bi Register**
Using the M16C/62 Timer in Pulse Period/Width Measurement Mode

### Count Start Flag

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA0S</td>
<td>038016</td>
<td>Timer A0 count start flag</td>
</tr>
<tr>
<td>TA1S</td>
<td></td>
<td>Timer A1 count start flag</td>
</tr>
<tr>
<td>TA2S</td>
<td></td>
<td>Timer A2 count start flag</td>
</tr>
<tr>
<td>TA3S</td>
<td></td>
<td>Timer A3 count start flag</td>
</tr>
<tr>
<td>TA4S</td>
<td></td>
<td>Timer A4 count start flag</td>
</tr>
<tr>
<td>TB0S</td>
<td></td>
<td>Timer B0 count start flag</td>
</tr>
<tr>
<td>TB1S</td>
<td></td>
<td>Timer B1 count start flag</td>
</tr>
<tr>
<td>TB2S</td>
<td></td>
<td>Timer B2 count start flag</td>
</tr>
</tbody>
</table>

**Figure 6 Count Start Flag Register**

### Timer B3, 4, 5 count start flag

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB3S</td>
<td>034016</td>
<td>Timer B3 count start flag</td>
</tr>
<tr>
<td>TB4S</td>
<td></td>
<td>Timer B4 count start flag</td>
</tr>
<tr>
<td>TB5S</td>
<td></td>
<td>Timer B5 count start flag</td>
</tr>
</tbody>
</table>

**Figure 7 Timer B3, 4, 5 Count Start Flag Register**

### Interrupt Control Register (Note 2)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILVL0</td>
<td></td>
<td>Interrupt priority level select bit</td>
</tr>
</tbody>
</table>

**Figure 8 Interrupt Control Register**

Note 1: Rewrite the interrupt control register only if it will not generate an interrupt request for that register. See precautions listed at end of the Interrupts chapter.

Note 2: This bit can only be accessed for reset (=0) Set (=1) cannot be accessed.
Using the M16C/62 Timer in Pulse Period/Width Measurement Mode

5.0 Reference

Renesas Technology Corporation Semiconductor Home Page
http://www.renesas.com

E-mail Support
support_apl@renesas.com

Data Sheet
• M16C/62 datasheets, 62aeds.pdf

User’s Manual
• NC30 Ver. 4.0 User’s Manual, NC30UE.pdf
• M16C/60 and M16C/20 C Language Programming Manual, 6020EC.pdf
• Application Note: Writing Interrupt Handlers in C for the M16C

6.0 Software Code

Below is a program written for Renesas’ NC30 compiler to illustrate how to configure Pulse
Period/Pulse Width Measurement Mode. The program can measure up to about a 16msec period and runs on
the MSV1632 Starter Kit Board. Using the KD30 debugger, the program can be 'stopped' and the global
variables ‘widthlow’, ‘width_hi’, and ‘period’ viewed from the global watch window.

To become familiar with this mode, try changing the clock source or even switch to a different timer (e.g. TB2,
TB3, etc.).

/*****************************************************************************
* File Name: period_width.c
* Content: Example program using Timer B in 'pulse width / period measurement
* mode'. This program is written for the "Pulse Period/Width
* Measurement" application note. Timer B0 is configured to measure
* pulse width(TB0in pin), and timer B1, measures the period(TB1in pin).
* Tested using a 250Hz square wave with a 0.5msec pulse width(high).
* This program works with the MSV1632 starter kit board.
Using the M16C/62 Timer in Pulse Period/Width Measurement Mode

Compiled with NC30 ver. 3.20.00.

All timing based on 16 Mhz Xtal

Copyright, 2003 Renesas Technology Corporation, Inc.

#include "sfr62.h"

#define B1TIME_CONFIG 0x42  /* 01000010 value to load into timer B1 mode register
  |||||||||_ TMOD0,TMOD1: PULSE MEASUREMENT MODE
  |||||||____ MR0,MR1:     PULSE PERIOD MODE
  ||||||_______MR2:         = 0 FOR PULSE MEASUREMENT
  ||||_______ MR3:         OVERFLOW FLAG
  ||________ TCK0,TCK1:   F DIVIDED BY 8 SELECTED */

#define B0TIME_CONFIG 0x4a  /* 01001010 value to load into timer B0 mode register
  |||||||||_ TMOD0,TMOD1: PULSE MEASUREMENT MODE
  |||||||____ MR0,MR1:     PULSE WIDTH MODE
  ||||||_______MR2:         = 0 FOR PULSE MEASUREMENT
  ||||_______ MR3:         OVERFLOW FLAG
  ||________ TCK0,TCK1:   F DIVIDED BY 8 SELECTED */

#define CNTR_IPL 0x03   // TB0 priority interrupt level

int period,widthlow,width_hi;

//prototypes
void init(void);

#pragma INTERRUPT /B TimerB0Int
void TimerB0Int(void);
/****************************************************************************
Name: TimerB0Int()
Parameters: none
Returns: nothing
Description: Timer B0 Interrupt Service Routine. The overflow flag is check
to determine if the TB0 register contains valid data. If so, the
input is tested to determine if the value in the TB0 register
is the high pulse width or low width and stored in the appropriate
variable.
**************************************************************************** */

void TimerB0Int(void)
{
  if (mr3_tb0mr ==1)  // check for timer overflow
  { tb0mr = B0TIME_CONFIG;    // if so clear flag and
      return;    // data invalid, so leave
  }
  if (p9_0== 1)
      widthlow = tb0 ;            // if input now hi, just measured a low width
  else
      width_hi = tb0;
}

****************************************************************************
Name: main()
Parameters: none
Returns: nothing
Description: initializes variables. Then the variable 'period' is constantly
updated with the period count in timer TB1. This is to illustrate
that the period measurement is free running. Note that the first
few times TB1 is read, the data may not be valid.
****************************************************************************
void main (void)
{

    init();
    while (1)
    {
        period = tb1; // period measured in polled mode
    }
}

/******************************************************************************
* Name:  initial()
* Parameters: none
* Returns:  nothing
* Description: Timer TB0 setup for pulse width interrupts and TB1 configured for
  * pulse period measurement (no interrupts).
******************************************************************************/
void init()
{

    /* the following procedure for writing an Interrupt Priority Level follows that as
described in the M16C
data sheets under 'Interrupts' */

    _asm (" fclr i"); // turn off interrupts before modifying IPL
    tb0ic |= CNTR_IPL; // use read-modify-write instruction to write IPL
    tb0mr = B0TIME_CONFIG;
    _asm (" fset i");

    tb0s = 1; //start counting

    tb1mr = B1TIME_CONFIG;
    tbls = 1; //start counting
}
In order for this program to run properly, timer B0’s interrupt vector needs to point to the function. The interrupt vector table is near the end of the startup file “sect30.inc”. Insert the function label “TimerB0Int” into the interrupt vector table at vector 26 as shown below.

```
;*******************************************************************************
;                  C Compiler for M16C/62
;                  Copyright,2003 Renesas Technology Corporation, Inc.
;                  All Rights Reserved.
;
; Modified for use on MSV1632 Starter Kit.
; sect30.inc       : section definition
; This program is applicable when using KD30 and the ROM Monitor.
;
;lword  dummy_int  ; timer A0(for user)(vector 21)
;lword  dummy_int  ; timer A1(for user)(vector 22)
;lword  dummy_int  ; timer A2(for user)(vector 23)
;lword  dummy_int  ; timer A3(for user)(vector 24)
;lword  dummy_int  ; timer A4(for user)(vector 25)
.glb  _TimerB0Int
.lword  _TimerB0Int  ; timer B0(for user)(vector 26)
.lword  dummy_int  ; timer B1(for user)(vector 27)
.lword  dummy_int  ; timer B2(for user)(vector 28)
.lword  dummy_int  ; int0 (for user)(vector 29)
.lword  dummy_int  ; int1 (for user)(vector 30)
.lword  dummy_int  ; int2 (for user)(vector 31)
.lword  dummy_int  ; vector 32 (for user or MR30)
.lword  dummy_int  ; vector 33 (for user or MR30)
.lword  dummy_int  ; vector 34 (for user or MR30)
.lword  dummy_int  ; vector 35 (for user or MR30)
.lword  dummy_int  ; vector 36 (for user or MR30)
.lword  dummy_int  ; vector 37 (for user or MR30)
;*******************************************************************************
```
Keep safety first in your circuit designs!

- Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
- The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
- Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination. Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.