M16C/26

Using Timer B 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, etc. The following article describes how use timers 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 Renesas M30262 is a 16-bit MCU based on the M16C/60 series CPU core. The MCU features include up to 64K bytes of Flash ROM, 2K bytes of RAM, and 4K bytes of Virtual EEPROM. The peripheral set includes 10-bit A/D, UARTS, Timers, DMA, and GPIO. The MCU has 8 timers that consists of five Timer A's and three Timer B's. Only the three Timer B's 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 shows a block diagram of timer B. The remainder of this document will focus on setting up timer B0 to measure pulse width and timer B1 to measure pulse period.

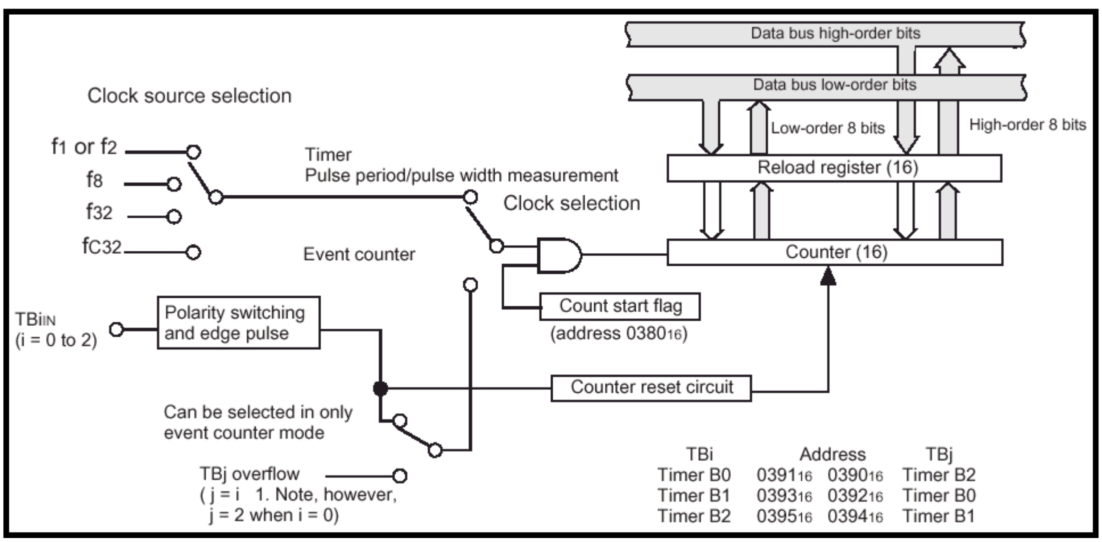

Figure 1 Block Diagram of Timer B

3.0 Pulse Period/ Pulse Width Measurement Mode Description

As can be seen Figure 1, timer TBi register consists of two parts, a counter and a reload register. In Measurement Mode, when an effective edge appears on the TBiIN pin, the count value is transferred to the reload register and the CPU can read this value by performing a read on the TBi register. The measured time is the counter value (TBi) divided by the frequency of the clock source (fi). Two period measurement options are available, that measure from falling edge to falling edge or rising edge to rising edge. For width measurement, the measurement is taken at both edges and software determines if the measured value is for the high width or low.

3.1 Pulse Period Measurement

In Period Measurement Mode (e.g. falling edge to falling edge), after the 'start count flag' is set, the counter starts counting 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 is indeterminate immediately after the 'start count flag' is set and an overflow could occur before the first falling edge. Figure 2 illustrates this operation.

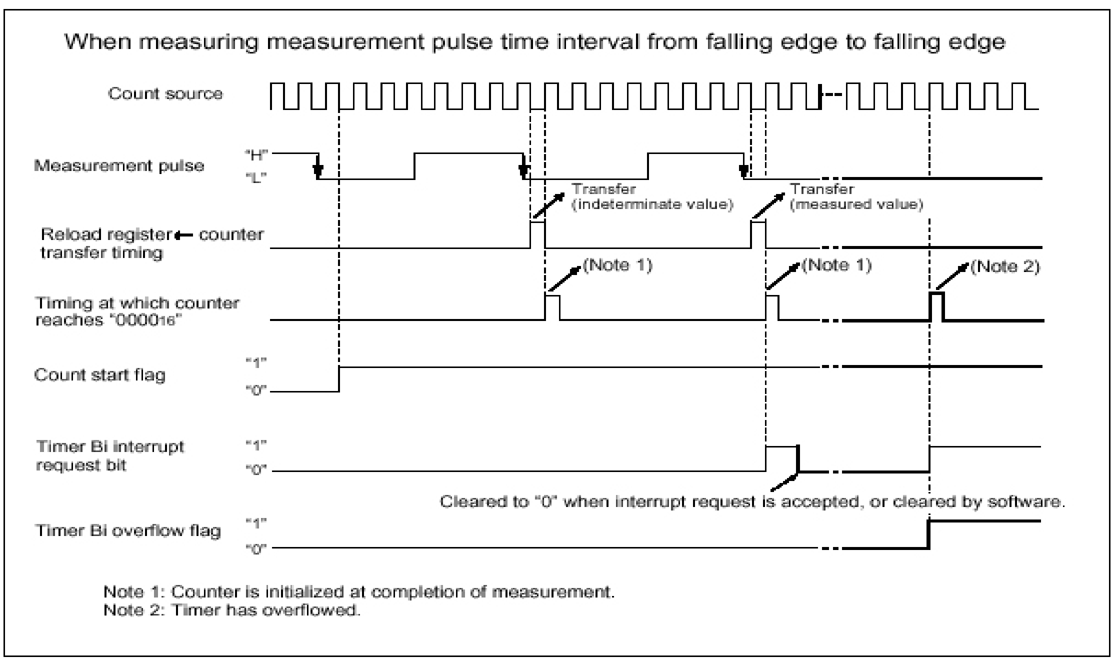

Figure 2 Operation timing when measuring a pulse period

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 Figure 3. Again, note that the value of the counter is indeterminate immediately after the 'start count flag' is set 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.

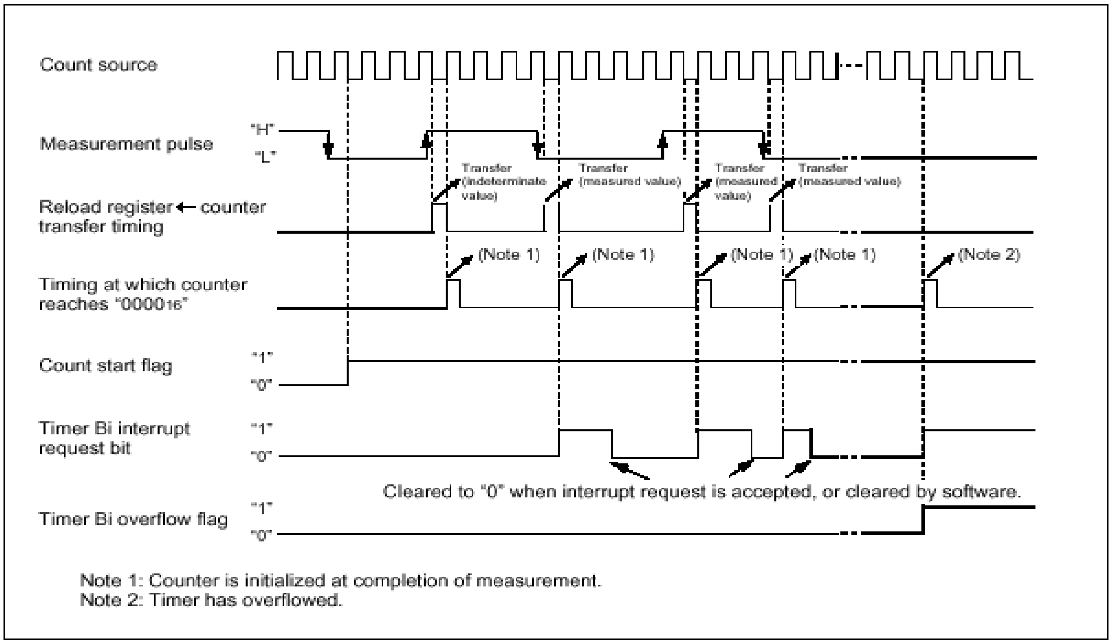

Figure 3 Operation timing when measuring a pulse width

3.3 Configuring Pulse Period/ Pulse Width Measurement Mode

The steps to configure timer B for Pulse Period/ Pulse Width Measurement Mode are shown below.

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.**

For the most part, the order shown above is not important. However, 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 chance of an interrupt occurring.

Figure 4 to Figure 7 show the registers for configuring Timer B for Pulse Period/Width Measurement.

Timer Bi register (Note)

(b15) b7 ... (b8) b0 b7 ... b0

Symbol	Address	When reset
TB0	0391_{16}, 0390_{16}	Indeterminate
TB1	0393_{16}, 0392_{16}	Indeterminate
TB2	0395_{16}, 0394_{16}	Indeterminate

Function	Values that can be set	R	W
ÄTimer mode Counts the timer's period	0000_{16} to $FFFF_{16}$	O	O
ÄEvent counter mode Counts external pulses input or a timer overflow	0000_{16} to $FFFF_{16}$	O	O
ÄPulse period / pulse width measurement mode Measures a pulse period or width	——	O	X

Note: Read and write data in 16-bit units.

Figure 4 Timer B-related registers

Timer Bi mode register

b7 b6 b5 b4 b3 b2 b1 b0: _ _ _ _ _ _ 1 0

Symbol	Address	When reset
TBiMR(i=0 to 2)	$039B_{16}$ to $039D_{16}$	$00XX0000_2$

Bit symbol	Bit name	Function	R	W
TMOD0	Operation mode select bit	b1 b0 1 0 : Pulse period / pulse width measurement mode	O	O
TMOD1			O	O
MR0	Measurement mode select bit	b3 b2 0 0 : Pulse period measurement (Interval between measurement pulse's falling edge to falling edge) 0 1 : Pulse period measurement (Interval between measurement pulse's rising edge to rising edge) 1 0 : Pulse width measurement (Interval between measurement pulse's falling edge to rising edge, and between rising edge to falling edge) 1 1 : Must not be set.	O	O
MR1			O	O
MR2	0 (Must always be 0 in pulse period/pulse width measurement mode; i = 0)		O (Note 2)	O
	Nothing is assigned (i = 1, 2). In an attempt to write to this bit, write 0 . The value, if read, turns out to be indeterminate.		X (Note 3)	X
MR3	Timer Bi overflow flag (Note 1)	0 : Timer did not overflow 1 : Timer has overflowed	O	X
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : f_{C32}	O	O
TCK1			O	O

Note 1: It is indeterminate when reset. Assume that the count start flag condition is 1 and then the timer Bi overflow flag becomes 1 . If the timer Bi mode register has a write access after next count cycle of the timer from the above condition, the timer Bi overflow flag becomes 0 . This flag cannot be set to 1 by software.
Note 2: Timer B0.
Note 3: Timer B1, timer B2.

Figure 5 Timer Bi mode register in pulse period / pulse measurement mode

Count start flag

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	When reset
TABSR	0380_{16}	00_{16}

Bit	Bit symbol	Bit name	Function	R	W
b0	TA0S	Timer A0 count start flag	0 : Stops counting 1 : Starts counting	○	○
b1	TA1S	Timer A1 count start flag		○	○
b2	TA2S	Timer A2 count start flag		○	○
b3	TA3S	Timer A3 count start flag		○	○
b4	TA4S	Timer A4 count start flag		○	○
b5	TB0S	Timer B0 count start flag		○	○
b6	TB1S	Timer B1 count start flag		○	○
b7	TB2S	Timer B2 count start flag		○	○

Figure 6 Count start flag register

Interrupt control register (Note2)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	When reset
TBiIC(i=3 to 5)	0045_{16} to 0047_{16}	$XXXXX000_2$
TAiIC(i=0 to 4)	0055_{16} to 0059_{16}	$XXXXX000_2$
TBiIC(i=0 to 2)	$005A_{16}$ to $005C_{16}$	$XXXXX000_2$

Bit	Bit symbol	Bit name	Function	R	W
b0	ILVL0	Interrupt priority level select bit	b2 b1 b0 0 0 0 : Level 0 (interrupt disabled) 0 0 1 : Level 1 0 1 0 : Level 2 0 1 1 : Level 3 1 0 0 : Level 4 1 0 1 : Level 5 1 1 0 : Level 6 1 1 1 : Level 7	○	○
b1	ILVL1			○	○
b2	ILVL2			○	○
b3	IR	Interrupt request bit	0 : Interrupt not requested 1 : Interrupt requested	○	○ (Note 1)
b7–b4	Nothing is assigned. In an attempt to write to these bits, write “0”. The value, if read, turns out to be indeterminate.			–	–

Note 1: This bit can only be accessed for reset (= 0), but cannot be accessed for set (= 1).
Note 2: To rewrite the interrupt control register, do so at a point that dose not generate the interrupt request for that register. For details, see the precautions for interrupts.

Figure 7 Interrupt control register

4.0 Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

support_apl@renesas.com

Data Sheets

- M16C/26 datasheet, M30262eds.pdf

User's Manual

- KNC30 Users Manual, KNC30UE.PDF
- M16C/60 and M16C/20 C Language Programming Manual, 6020EC.PDF
- Writing interrupt handlers in C for the M16C Application Note
- MSV30262-SKP or MSV-Mini26-SKP Quick start guide
- MSV30262-SKP or MSV-Mini26-SKP Users Manual
- MDECE30262 or MSV-Mini26-SKP Schematic

5.0 Software Code

A sample program written in C and compiled using the KNC30 compiler to illustrate how to configuring Pulse Period/ Pulse Width Measurement Mode is shown below. The program can measure up to about a 16ms period and runs on the MSV30262 Starter Kit Board. Using KD30, the values of the global variables 'widthlow', 'width_hi', and 'period' can be viewed from the global watch window.

To get 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 MSV30262 starter kit board.
*
*        Compiled with KNC30.
*
*        All timing based on 20 Mhz Xtal
*
```

```
*==============================================================================
*      $Log:$
*============================================================================*/
#include "sfr26.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 apropriate variable.
****************************************************************************** */

void TimerB0Int(void)
{
  if (mr3_tb0mr ==1){               // check for timer overflow
    tb0mr = B0TIME_CONFIG;          // if so clear flag and
    return;                         // data invaild, 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 illistrate
               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:  init()
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;
   tb1s = 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/26
;
;       Copyright 2003 Renesas Technology America, Inc.
;        All Rights Reserved.
;
;       Written by T.Aoyama
;       Modified  for use on MSV30262 Starter Kit.
;       sect30.inc     : section definition
;       This program is applicable when using KD30 and the ROM Monitor.
;
;        $Id:
;
;*****************************************************************************
                                          :
                                          :

       .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)
                                          :
                                          :
```