M16C/26

Using the Watchdog Timer

1.0 Abstract

The following article introduces and shows an example of how to set up and use the watchdog timer on the M16C/26 microcontroller (MCU).

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 M16C/26 MCU has a built-in watchdog timer, which can be used for a variety of applications. For most applications, it is used to recover MCU processing from a program that is out of control. In some cases, it can be used to preserve processor or firmware status after an application runs out of control.

In this example application, we show you how to set up the watchdog timer, the watchdog interrupt vector, and how the application uses the watchdog timer. This example was written for the MSV30262-Board with an oscillator frequency Xin = 20 MHz.

3.0 Watchdog Timer Demo

This section discusses the watchdog timer demo setup and how it works. The key components of the program are discussed in the next section. A program listing appears later in the article.

3.1 M16C/26 Watchdog Timer

The M16C/26 watchdog timer is a 15-bit counter using BCLK as the clock source. BCLK and the watchdog prescaler control the length of time before the timer expires. This BCLK-prescaler combination can be used for a wide range of watchdog timing requirements. The block diagram of the watchdog timer is shown in Figure 1.

A hardware watchdog interrupt is generated after the timer expires and the program executes the watchdog interrupt routine. To prevent the watchdog timer from expiring, the Watchdog Timer Start Register (WDTS) must be written before the timer underflows. For example, if the watchdog timer is set up for 1.6s, the WDTS register must be written to within 1.6s so that the timer does not expire.

For this demo, the timer was set up for 1.678s.

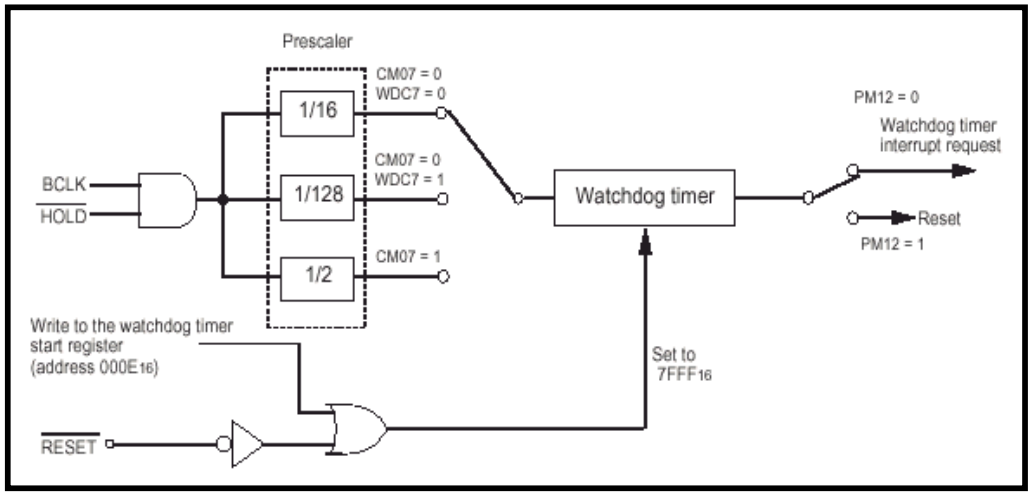

Figure 1 Watchdog Timer Block Diagram

3.2 Watchdog Underflow Effects

After the watchdog timer expires or underflows, an interrupt or reset is generated depending on the value of the PM12 bit of the Processor Mode Register 1. If PM12 is set (PM12 = 1), a reset is generated. If PM 12 is cleared (PM12 = 0), an interrupt is generated. For this demo, PM12 is cleared so an interrupt is generated.

An interrupt service routine must be in place for the program to execute when a watchdog interrupt occurs. This interrupt routine can be used to store program parameters or register status in RAM. As an added fail-safe feature, the M16C/26 MCU chip is automatically reset if there is a second successive underflow of the watchdog timer. Furthermore, The bit-5 (WDC5) of the watchdog timer control (WDC) register may be used to distinguish between a cold start from a warm start.

3.3 The Demo Application

This application note concentrates on the generation as well as prevention of interrupts from watchdog timer. The demo uses two timers (Timer A0, A1), the AD converter using AN1, and I/O ports. Timer A0's output is used as the clock source of Timer A1. Timer A1 is preloaded with the ADC value of the R46 potentiometer and is then used to set up how fast the LED's LED3-5 blink and the WDTS register is written to. The I/O ports are used to turn on or off the LED's LED3-5 and to read the status of the switches SW2–SW4.

By adjusting R46 from full clockwise (CW) position to full counterclockwise position (CCW), the period the WDTS is written to varies also. The LEDs will be blinking fast at full CW and very slow (about 5s interval) at full CCW. At full CCW, the time period of when WDTS is written is greater than 2.097s, which will then trigger a watchdog interrupt. However, still at full CCW, if any of the SW2-SW4 switches is pressed within 2s, the watchdog timer is restarted and thus, a watchdog timer interrupt is not generated.

4.0 Watchdog Timer Setup

A watchdog timer interrupts after a certain time has expired. As mentioned earlier, the M16C/26 watchdog timer can be set up for various time periods by configuring the BCLK and watchdog prescaler. The equations to calculate the period based on the BCLK source are shown in Figure 2. These parameters are discussed in the following subsections. For more detailed information, see the M16C/26 datasheet.

With XIN chosen for BCLK

$$\text{Watchdog timer period} = \frac{\text{prescaler dividing ratio (16 or 128) X watchdog timer count (32768)}}{\text{BCLK}}$$

With XCIN chosen for BCLK

$$\text{Watchdog timer period} = \frac{\text{prescaler dividing ratio (2) X watchdog timer count (32768)}}{\text{BCLK}}$$

Figure 2 Calculating the Watchdog Timer Period

4.1 BCLK

The clock source of the timer is BCLK, which is the CPU clock for the M16C/26. The value of BCLK can be modified by changing the oscillator circuits of the device or by changing setting in the clock control registers (see "Clock Control" from the datasheet). BCLK can use Xin (f1), XCin (fc), or clock divider output (f2, f4, f8, f16, f32). Modifying the BLCK will then modify the frequency the timer counts down and processor operating speed.

For this demo, the clock divider output f8 was used as the BLCK. With an Xin frequency of 20 MHz, BCLK frequency is 2.5 MHz.

4.2 Prescaler

Besides BLCK, the other parameter that can adjust the timer is the watchdog prescaler. The prescaler further divides BLCK for larger time periods. The prescaler that can be used differs depending on whether Xin or XCin is used as the BCLK source. If Xin is used, the prescaler can be either a div 16 (divided by 16) or div 128 (divided by 128). If XCin is used, the prescaler is fixed to div 2 (divided by 2).

For this demo, since Xin is used for the BCLK, the prescaler used is div 128.

4.3 Timer Count

Besides BLCK and the prescaler, the other parameter is the timer count. This parameter, however, cannot be modified. Regardless of what value is written to the WDTS register, the default value of 07FFFh (32768) is loaded into the timer.

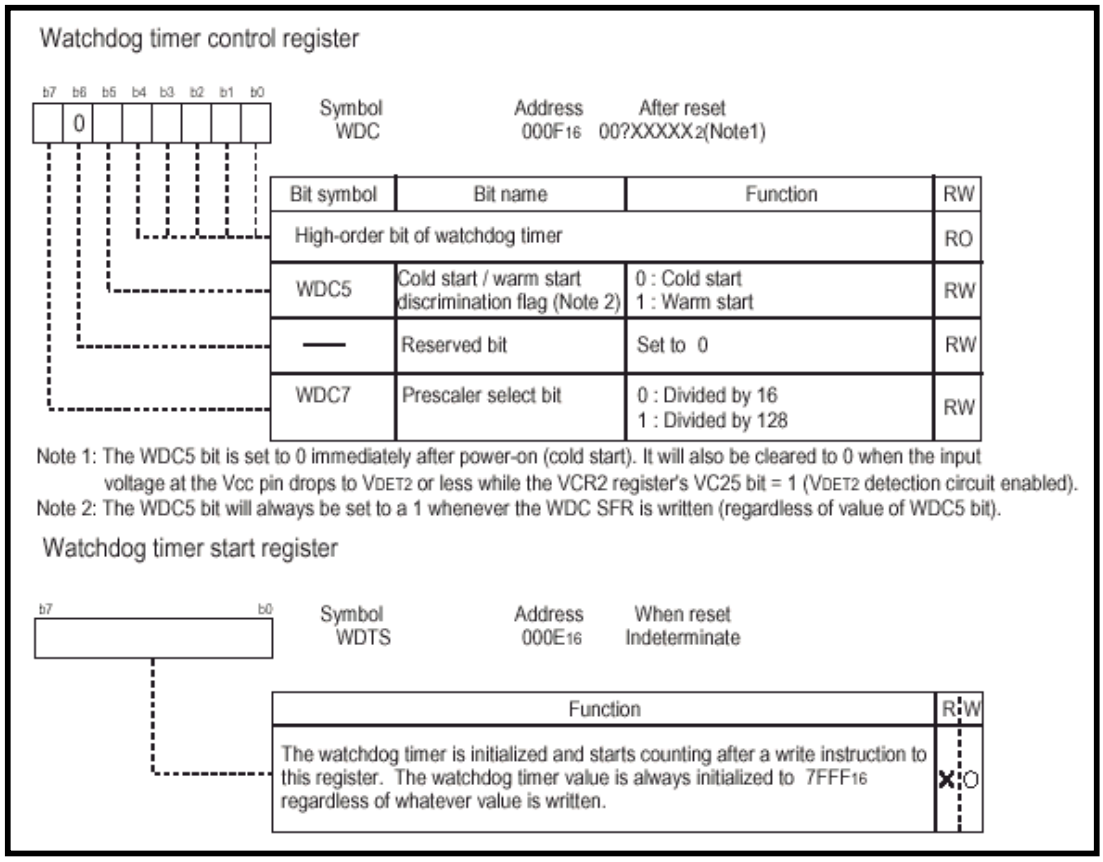

Figure 3 Watchdog Timer control and start registers

5.0 Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

support_apl@renesas.com

Data Sheets

- M16C/26 datasheets, M30262eds.pdf

User's Manual

- M16C/20/60 C Language Programming Manual, 6020c.pdf
- M16C/20/60 Software Manual, 6020software.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

6.0 Software Code

The example program was written to run on the MSV30262 Board but could be modified to implement in a user application. The program is written in C and compiled using the KNC30 Compiler.

```
/***********************************************************************
*
*       File Name: main.c
*
*       Content:   This program blinks the three LEDs (D3, D4, & D5) sequentially.
*       The blink rate is controlled by the R46 (Analog Adjust) potentiometer
*       connected to AN1 of the M16C/26 ADC. Turn the R46 clockwise or
*       counter-clockwise to change the speed of LED switching. Extreme CCW
*       position of the potentiometer generates interrupt from WD Timer
*       and turns ON all LEDs simultaneously. However, rotating the potentiometer
*       CCW while pressing any of switches -2, -3 or -4 prevents the WD Timer
*       from generating its interrupt and the 3 LEDs continue to blink even
*       at the extreme CCW position of the potentiometer
*
*       Date:  10-31-2002
*       This program was written to run on the MDECE30262 Board for MSV30262-SKP.
*
*       Copyright 2003 Renesas Technology America, Inc.
*       All rights reserved
*
*=====================================================================
*       $Log:$
*=====================================================================*/

#include "sfr262.h"     // M16C/26 special function register definitions
#pragma INTERRUPT       TimerA1_ISR

/* LEDs */
#define red_led         p7_0
#define yellow_led      p7_1
#define green_led       p7_2
/* SWITCHES */
#define sw2                     p10_5
#define sw3                     p10_6
#define sw4                     p10_7
```

```
void TimerA1_ISR(void);         //Interrupt Service Routine for Timer A1
void mcu_init(void);    // routine that initializes MCU

void WD_Init();         //routine that initializes watchdog operation
void WD_Loop_ISR(void);//routine when a watchdog interrupt is generated

/***********************************************************************
Name:   main
Parameters:
Returns:
Description:  main program loop and initialization
***********************************************************************/

main() {
        WD_Init();          /* intialize Watchdog timer */
        mcu_init();             /* initialize MCU */

/*************** PROGRAM LOOP ************************************/
        while(1){
                int value;
                adst=1; /* Start A2D conversion */
                while( adst==1); /* Wait for A/D start bit to return to 0 */
                value=ad1; /* Read value from A/D register and pre-load Timer1 */
                ta1=value; /* This value is used to vary the blink rate */


                if(sw2==0 || sw3==0 || sw4==0) { //check if any switch is pressed
                        wdts = 0; //restart Watchdog Timer to continue blinking of LEDs
                }
        }
}

/***********************************************************************
Name:   TimerA1_ISR
Parameters:
Returns:
Description: This Timer A1 interrupt routine writes to WD Timer and prevents it from interrupting.
It also varies the sequential blinking rate of LED's
D3, D4, & D5.
***********************************************************************/
void TimerA1_ISR(void){
        static unsigned int out1=0;

        wdts = 0;       // re-start watchdog timer

        ++out1;
        if( out1 > 3 )
                out1=0;
```

```
        //blink a LED
        switch (out1){

                case 1: /* green on */
                                red_led = 1;
                                yellow_led = 1;
                                green_led = 0;
                                break;

                case 2: /* yellow on */
                                red_led = 1;
                                yellow_led = 0;
                                green_led = 1;
                                break;

                case 3: /* red on */
                                red_led = 0;
                                yellow_led = 1;
                                green_led = 1;
                                break;

                default:        /* all LED's off */
                                red_led = 1;
                                yellow_led = 1;
                                green_led = 1;
        }
}

/***********************************************************************
Name:           mcu_init
Parameters:     None
Returns:        None
Description: Initialization routine for the different MCU peripherals. See
             settings for details.
***********************************************************************/
void mcu_init(void) {

   /* LED initialization */
   pd7_0 = 1;           // Change LED ports to outputs (connected to LEDs)
   pd7_1 = 1;
   pd7_2 = 1;

   red_led = 1; // turn off LEDs
   green_led = 1;
   yellow_led = 1;

   /* Configure Timer A0 - 5ms (millisecond) counter */
   ta0mr = 0x80;        // Timer mode, f32, no pulse output
   ta0ud =0;
   ta0 = 2499;  // 4ms time period for Timer A0
                        // 4 ms x 20MHz/32 = (2499+1)
```

```
   /* Configure Timer A1 - Timer A0 used as clock */
   ta1mr = 0x01;        // Event Counter mode, no pulse output
   ta1 = 0;
   trgsr = 0x02;        // Timer A0 as event trigger
                        // Max interrupt interval of TA1 at Max ADC value of 0x3FF
                        // = 1024x0.004 = 4.096s > 1.678s timeout of WD Timer

   ta0s = 1;            // Start timer A0
   ta1s = 1;            // Start Timer A1

   /* Configure ADC - AN1 (R46 Analog Adjust Pot) */
   adcon0 = 0x01;       // AN1, one-shot mode, software trigger
   adcon1 = 0x28;       // 10-bit mode, Vref connected.
   adcon2 = 0x01;       // Sample and hold enabled

   asm("FCLR I");       // disable irqs before setting irq registers

   ta1ic = 3;           // Set the timer A1's interrupt priority to level 3

   asm("FSET I");       // enable interrupts

   return;
}

void WD_Init(){ //Initialize Watchdog Timer

        cm06 = 1;       //BCLK = (20/8)MHz = 2.5 MHz (Xin div by 8, default)

        wdc7 = 1;       //prescaler is div by 128
                        //Watchdog Timer period = (32,768 x 128) /(2.5 MHz) = 1.678s

        wdts = 0;       //start Watchdog Timer by writing any value to
                        //wdts register (value always resets to 0x7fff = 32,768 when
                        //written to)
}

void WD_Loop_ISR(void){ //turn ON all LEDs
        while(1){
                red_led = 0;
                yellow_led = 0;
                green_led = 0;

                wdts=0;    //writing in WD Timer prevents it from interrupting again
                             //the second interrupt from the WD Timer would have
     //reset the MCU
        }
}
```

In order for this program to run properly, the Watchdog Timer and TimerA1 interrupt vector needs to point to the service routines for these interrupts. The interrupt vector table information is included in the startup file "sect30.inc". Insert the function label "TimerA1_ISR" and the function label " WD_Loop_ISR" into the interrupt vector table locations as shown below.

```
;***********************************************************************
;
;  sect30.inc :    Customized section and macro definitions for the M30262
;                  (M16C/26) microcontroller using the NC30 compiler.
;
;  Description :  This file is specific to the M30262 microcontroller and adapted
;                 for use with the MSV30262 Starter Kit. UART1 interrupt
;                 vectors are used for the Starter Kit debugger.
;
;***********************************************************************

;-----------------------------------------------------------------
; variable vector section
;-----------------------------------------------------------------
           :
           :
.lword  dummy_int               ; TIMER A0 (for user)
.glb    _TimerA1_ISR
.lword  _TimerA1_ISR            ; TIMER A1 (for user)
.lword  dummy_int               ; TIMER A2 (for user)
.lword  dummy_int               ; TIMER A3 (for user)
           :
           :
;=================================================================
; fixed vector section
;-----------------------------------------------------------------
.org    0fffdch

.glb    _WD_Loop_ISR

UDI:
        .lword  dummy_int
OVER_FLOW:
        .lword  dummy_int
BRKI:
        .lword  dummy_int
ADDRESS_MATCH:
        .lword  dummy_int
SINGLE_STEP:
        .lword  dummy_int
WDT:
        .lword  _WD_Loop_ISR
DBC:
        .lword  dummy_int
NMI:
        .lword  dummy_int
RESET:
        .lword  start
;***********************************************************************
```

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.