Freescale Semiconductor
Application Note

AN2612
Rev. 1, 1/2006

PWM Generation Using HCS12 Timer Channels

By Grant M More and Martyn Gallop
MCD Applications, East Kilbride

1 Introduction

Most HCS12 microcontrollers offer pulse width modulation (PWM) generation capability through the use of a dedicated PWM module providing independent PWM output signals (with up to 16-bit resolution). This provision has proved sufficient for most applications. However, where a requirement for more than the available number of dedicated PWM channels exists, or where relatively high frequency and low frequency PWM signals are required simultaneously, the timer module (ECT or TIM) may be used to generate supplementary PWM signals.

This application note demonstrates how to generate PWM signals using the timer module. Ready built example software is provided to demonstrate the principles that are described herein.

Table of Contents

freescale™ semiconductor

2 Principle of PWM Generation Using Timers

The generation of a PWM signal using the dedicated HCS12 PWM module is based on hardware comparisons between register values and free running hardware counters. The timer module offers similar hardware comparison in the form of output compare circuitry. The contents of a register are continually compared to the master free-running timer. When a match occurs, a hardware output event can be configured to take place and an interrupt can then call a service routine. The primary difference between the PWM module and the timer module is that the timer module has only one compare register and can only be configured to trigger a hardware output event on one comparison value at any time. The PWM module has two compare registers and can thus be configured and set to run, toggling the port pin on each of two comparison values, with no core overhead. PWM generation can be achieved with the timer channels by reconfiguring the hardware output event following each successful output compare comparison, effectively replicating the double comparison performed by the PWM module.

Each PWM signal requires a dedicated timer channel with output compare capability. Any channel with this function and a discrete interrupt vector may be used to generate PWM with the method described. Designed as a dedicated module, the PWM module supports a number of specialist PWM features, such as double buffered PWM values, polarity control, emergency shutdown and individual channel prescalers. Some of this functionality may be implemented on the timer with software, however this document focuses on the basic PWM generation mechanism.

3 Configuring the Timer Module

3.1 Timer I/O

The Timer I/O port control registers are located in the Port Integration Module (PIM). Each port can be configured on a pin-by-pin basis and each timer input capture/output compare channel is associated with a single pin. On reset, timer modules are disabled and the appropriate I/O port defaults to a high impedance input.

The initial state of a pin can be defined by configuring the appropriate general purpose I/O pin as an output in the Data Direction Register (DDRT) and writing the Port Data Register (PTx) to the appropriate state. An external pull device is required to control the level during reset.

Setting the Timer Enable (TEN) bit in the Timer System Control Register (TSCR1) enables the timer module. The output compare functionality is disabled in the default module reset state. In this mode, the Data Direction bits (DDRTx) control the I/O state of the pins while the Input Compare logic monitors transitions on the pins. Setting the appropriate bit in the Timer Input Capture/Output Compare Select (TIOS) register enables a timer channel for output compare, as needed for PWM generation.

In output compare mode, the Output Mode (OMn) and Output Level (OLn) bits in the Timer Control Registers (TCTL1/2) simultaneously select the compare event action and enable the connection of the output compare output logic to the relevant pin. If the OMn:OLn control bits for a channel are both zero the DDRTx and PTx bits control the state of the I/O pin. Setting either (or both) of the OMn:OLn bits connects the output compare circuitry to the pin, over-riding the DDRTx and PTx settings. Following a reset, the output state for each output compare circuit is zero.

For PWM generation, the OM bit is set (= 1) so that the output compare output follows the state of the associated OL bit on each compare event. The state of the OL bit is inverted every time the timer channel interrupt is serviced to produce a toggling output.

Clearing or setting the TEN bit disables or enables the timer module respectively, but does not modify the contents of any other timer control registers or the state of the output compare output logic.

3.2 PWM Timing

A number of considerations have to be made when configuring the timer module for PWM. From a high-level point of view, the main considerations are:

- PWM Frequency
- PWM Duty Cycle

In order to generate the required PWM frequency, the bus clock frequency must be known, and the timer prescaler must be set. These values will depend on the range of PWM frequencies that will be generated and the degree of resolution of the PWM signal.

Maximum resolution and PWM frequency are limited by the maximum timer clock frequency. Lower PWM frequencies are limited by the minimum timer clock frequency. This can sometimes result in a trade-off and can be evaluated as shown in Figure 1.

Bus clock frequency = 16 MHz
Required PWM frequency = 100 Hz

Maximum possible timer clock frequency = 100 Hz * 2^{16} = 6.55 MHz

Available timer prescaler values = {1, 2, 4, 8, 16, 32, 64, 128}

Target timer prescaler value = Bus clock frequency / Timer clock frequency
= 16 MHz / 6.55 MHz
= 2.4427

Figure 1. Identification of Optimum Timer Prescaler Value

It is now possible to select a divider value from the list of possible values above. Select the value that is greater than but closest to the target timer prescaler value. In this case, the prescaler would be set at 4. This is the optimal setting for this PWM requirement. It is possible that the timer prescaler may be configured for another system timer requirement; in any case the prescaler cannot be less than the value in Figure 1. Other system timing parameters may need to be adjusted to accommodate this constraint.

In order to configure the timer module for duty cycle, the period of the PWM signal in relation to the number of timer ticks during each PWM cycle must first be calculated. Once this value is calculated, the output compare register values (the timer values for generating the on (mark) and off (space) pulses) can be calculated as shown in Figure 2.

```
PWM Period (timer ticks)       = (Bus clock frequency / Timer prescaler) / PWM Frequency
PWM Mark time (timer ticks)  = (PWM Period / 100) * Duty cycle (as an integer percentage)
PWM Space time (timer ticks)= PWM Period - PWM Mark time
```

Figure 2. Calculation of Mark and Space Times

4 PWM Signal Generation

Once the timer channel is configured, the PWM signal can be generated using the timer channel interrupt. This should be configured to call an interrupt service routine (ISR) to load the timer compare register with the appropriate compare value (mark or space). This is achieved by identifying whether the last action was a negative or a positive edge transition, switching the transition status and loading the compare register with the next appropriate value.

References to the master timer count register can be avoided by simply adding consecutive mark and space values to the timer compare register on successive ISR function calls as shown in Figure 3. Timer roll-over is seamless when using unsigned integer addition. Using the previous compare value as a reference for generating the next compare value allows precise output timing even though the ISR latency may vary.

```
PWMTimerChannelInteruptServiceRoutine
{
   if (Timer.tctl2.bit.ol0 == 1)   /* if channel is set to generate rising edge */
   {
      Timer.tc[0].word += Mark[0];/* set up timer compare for mark time */
                                  /* relative to the last transition */
      Timer.tctl2.bit.ol0 = 0;    /* set pin action to falling edge */
   }
   else
   {
      Timer.tc[0].word += Space[0];/* set up timer compare for space time */
                                  /* relative to the last transition */
      Timer.tctl2.bit.ol0 = 1;    /* set pin action to rising edge */
   }
}
```

Figure 3. Recommended ISR Structure

5 Starting and Stopping PWM Output

Starting the PWM is a task that requires careful consideration. In order to start the PWM generation using the interrupt, it is necessary to configure the first compare event manually. It is necessary to configure a forced compare by setting a compare to switch the output pin to the first transition state. After the initial compare event, interrupts will handle the PWM generation. The HCS12 does not support hardware forced compare, but a forced compare can be configured by setting a normal compare a few cycles ahead of the current free-running timer value. The cycles are necessary to compensate for internal latency within the MCU. The number of cycles will vary depending on the core and module clocks.

When stopping the PWM generation, it is important to consider runt pulses (pulses with width shorter than the prescribed mark or space ratio as appropriate). To avoid these pulses, disable the PWM generation by setting the appropriate local interrupt mask within the associated interrupt service routine. The appropriate state of the pin at stop time can be set by disabling the interrupt in either part of the ISR; either the rising or falling edge portion.

6 Limitations

The MCU must be able to service the timer PWM interrupts in time for the next edge to be configured. This sets an upper limit on the frequency/resolution of PWM signal that can be reproduced, and will vary from system to system depending on MCU application. It is the responsibility of the system designer to satisfy him or herself that the core can update the compare registers in sufficient time using the ISR under maximum core loading conditions.

A secondary limitation is that the use of the timer module incurs a greater degree of core overhead as the timer module has to be serviced at every edge transition (interrupt). This does not happen with the PWM module as the toggling mechanism is independent of the core. The described method of PWM generation is likely to be more suited to slower rate PWM requirements due to the overhead generated by having to service an interrupt for each edge of each PWM signal.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2612
Rev. 1, 1/2006

freescale™
semiconductor