Keywords: MAXQ, Timer, Timers, Timer 2, Counter, PWM,

APPLICATION NOTE 3205

Using Timers in the MAXQ Family of Microcontrollers

Aug 18, 2004

Abstract: This application note describes how to set up and use the Type 2 Timers in the MAXQ™ Family of Microcontrollers for different applications. It includes source code for reference.

Introduction

The MAXQ family of microcontrollers has three types of timers: Timer 0, Timer 1, and Timer 2.

The MAXQ Timer 0 type is modeled after the Timer 0 type common on many 8051 microcontollers. The MAXQ Timer 1 type is modeled after the 8051 Timer 2 type. Most MAXQ products have a new timer called Timer 2 that is unique to the MAXQ family. This application note details how to set up and use this new Timer 2 for different purposes, and includes some source code for reference.

Overview

Functionality
The three main uses of the timer are to generate output waveforms, to count transitions of an input signal (including counting system clock transitions thus functioning as a timer), and to time an input signal. This section covers methods to use the configuration modes of Timer 2 to perform these timer functions.

Compare
In this mode, the counter is sourced internally by either the system clock or an alternate clock (typically the 32,768 Hz RTC clock), either of which may be optionally prescaled by 1, 2, 4, 8, 16, 32, 64, or 128. The counter is then used to control the output of the primary and/or secondary timer pins to generate various waveforms.

By changing the values in the reload (T2R) and compare (T2C) registers, the frequency and duty cycle of the output waveforms can be modified. In this way, the MAXQ processor can generate a pulse-width modulated (PWM) waveform. The outputs can be selectively enabled and the starting polarity can be inverted. The limits on frequency and duty cycle are determined by the frequency of the clock selected as source (whether the system or alternate clock) and the clock divisor selected. The minimum pulse width is one clock cycle (selected by setting the compare and reload values to the same value or by setting the compare value to FFFEh) of the prescaled source clock. The maximum pulse width is 65,536 cycles of the prescaled source clock. With a prescale factor of 128, the maximum pulse width is 8,388,608 system clock periods - over eight seconds for a 1 MHz system clock. Using the alternate clock with a lower frequency can increase this even further.

Compare mode also provides the ability to generate single pulses by use of the single shot (SS2) capability. Single shot allows the firmware to set up the length of the pulse before it is triggered, precluding the requirement for firmware to time the pulse and determine when it should end. Gating allows the primary pin to trigger a single pulse on the secondary output or to turn the counter on and off, allowing a PWM output on the secondary pin to be modified by an incoming signal. The compare mode can also be used to generate recurring interrupts on a specific schedule.

Capture

In this mode, the counter is sourced internally but is used to count or time the duration of an input signal on the primary timer pin. The counter can be gated and triggered on either or both edges of the input signal, allowing flexibility in timing pulses, single events, or reccurring waveforms. The value in the T2C register can be used to calculate the period of the measured event.

Counter

In this mode, the primary timer pin sources the clock for the counter. In this mode, the counter counts the transitions on the primary pin, either the rising or falling edges, or both. The secondary pin can be used to output a waveform, which toggles when the count overflows and when the counter matches a specific value. Interrupts can also be generated in these two cases.

Dual Eight-Bit Modes

Timer 2 can be operated in sixteen-bit mode (in which case there is just one counter available) or in eight-bit mode, which treats the counter as two eight-bit counters, which can be used independently. This allows for three additional functions for the timer circuitry. When in eight-bit mode, the primary counter (the high counter) can be in used for Compare, Capture, or Counter functions while the secondary counter (the low counter) can be used for compare or PWM output.

Dual Compare

In this mode, each eight-bit counter counts individually and can be used to output a waveform, but each counter is sourced from the same internal clock source. Different frequencies and duty cycles of waveforms can be output on the two timer pins by using different reload and compare values for each eight-bit counter.

Capture/PWM

In this mode, one eight-bit counter (the high counter) works in capture mode to time the duration of an input signal. The other eight-bit counter (the low counter) operates in compare mode and can be used to output a waveform.

Counter/PWM

In this mode, one eight-bit counter (the high counter) works in counter mode to count transitions of an input signal. The other eight-bit counter (the low counter) operates in compare mode and can be used to output a waveform.

Options

Single Shot (SS2)

This option allows the counter to run as if the run bit (T2R) had been turned on, but only until the next overflow of the counter. The counter then reverts to using the run bit to determine if it will continue counting. When in eight-bit mode, this bit applies ONLY to the primary eight-bit counter (high counter).

Gating Enable (G2EN)

This option allows the counter to be turned off (that is, gated) for a period of time without the firmware needing to manually toggle the run bit (T2R). In compare and counter mode, gating applies to the source clock. In capture mode, gating applies to the reload event.

The gating is controlled by the value on the primary pin. This requires the primary pin to be an input (T2OE [0]=0), so the output waveform (if needed) must be output on the secondary (or B) pin. Gating control is only available on the primary input.

Polarity Select
The polarity bits (T2POL [0] and T2POL [1]) can be used to invert the output waveforms. When used for this function, they must be set before the respective output enable bits (T2OE [0] and T2OE [1]) are set. Setting the polarity bits after the enable bits are set has no effect. The T2POL [0] bit can also be used to invert the gating condition on the primary pin when it is not used as an output. The T2POL [0] bit also an additional meaning in capture mode when both edges are defined for capture. In this case it defines which edge starts a single shot cycle, and inhibits reloading on one edge if gating is enabled.

Output Enable
The output enable bits (T2OE [0] and T2OE [1]) determine if the primary and secondary pins are actively driven by the timer circuitry. When the output enable bits are turned off, these pins can be read as inputs using the respective Port Input register. When the primary pin is not enabled for output, it can be used to gate (turn off) the counter.

Clock selection
The T2CI bit (alternate clock select) and T2DIV [2:0] bits (clock divisor) are used to set the clock source and prescale factor used by the counter. T2CI is used to select between the system clock and the alternate clock. Either of these can then be prescaled by a factor of 2^n (n is the value from 0 to 7 that is stored in T2DIV).

Using the Timer

The Counter Registers
The counter has three registers associated with it in sixteen-bit mode and three additional registers when in eight-bit mode. These registers can be read and written in either mode (eight- or sixteen-bit), but the low registers behave differently when in eight-bit mode. When operating in sixteen-bit mode, T2V, T2R and T2C serve as the sixteen-bit counter, reload and compare registers, respectively. When operating in eight-bit mode, these three registers serve as the low-order eight-bit registers. The high-order registers are represented by the T2H, T2RH and T2CH for the counter, reload and compare registers, respectively. Additionally, these registers take on the role of the primary counter, with the T2V, T2R and T2C registers assuming a secondary role.

This means that any code that uses the timer registers will operate significantly differently when the timer is in eight-bit mode versus sixteen-bit mode. Therefore it is recommended that you use either eight-bit mode or sixteen-bit mode exclusively for a particular timer. If you need to use it in both modes at different times, using separate functions for each mode will cause less confusion for the programmer.

In sixteen-bit mode, the T2V register contains the current count. This register is incremented with the selected clock edge(s) when either the run bit (TR2) or the single shot bit (SS2) are turned on and gating is not active (G2EN=0). If gating is active (G2EN=1) then in addition to either TR2 or SS2 being on, the input on the primary pin must be of opposite polarity to the primary polarity bit (T2POL [0]) for the counter to increment.

The T2R register holds the reload value for the counter. This value is automatically inserted into the counter (T2V) whenever it overflows (has reached FFFFh and is due to increment again).

Control and Configuration Registers

There are three control and configuration registers: T2CFG, T2CNA, and T2CNB. T2CFG contains general

configuration information.

T2CI	T2DIV[2]	T2DIV[1]	2DIV[0]	T2MD	CCF[1]	CCF[0]	C/T2
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

The C/T2 bit (counter/timer select) selects whether the timer will function in counter mode or timer mode (capture, compare, and capture with compare output are sub-modes of the timer mode). In timer mode, the CCF [1:0] bits (capture/compare select) determine if the timer is in compare mode (CCF [1:0] = 00) or capture mode (CCF [1:0] = 01, 10, or 11). In counter mode, the CCF bits determine which edges - falling, rising, or both - will be counted. In counter mode, a value of 00 in the CCF bits is not used since the counter would have nothing to count. The mode select bit (T2MD) determines if the timer will operate as one sixteen-bit timer or two separate eight-bit timers. When set, two eight-bit timers are selected. The secondary timer is always a compare/PWM timer.

The system clock or the alternate clock (32 kHz RTC clock in some MAXQ implementations) can be selected as the source clock, and each of these can be prescaled as necessary. The alternate clock select bit (T2CI) defaults to 0, which selects the system clock. Setting this bit selects the alternate clock.

The prescaler bits (T2DIV [2:0]) select the clock divisor, which ranges from 1 to 128. The formula for the prescaler is 2_n, where n is the value in T2DIV [2:0].

T2CNA contains gating enable, single shot, reload enable, run enable, low run enable, primary output polarity, primary output enable, and interrupt enable bits.

ET2	T2OE0	T2POL0	TR2L	TR2	CPRL2	SS2	G2EN
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

The gating enable bit (G2EN) allows the counter to be selectively disabled. The single shot bit (SS2) allows the timer to run until the next overflow condition, at which point the timer halts.

The capture and reload bit (CPRL2) instructs the timer to capture its value into its capture register and reload the value from the reload register on an external edge. CPRL2 is not used in the compare and counter modes.

The run enable bit (TR2) allows the primary counter to run and the low run enable bit (TR2L) allows the secondary counter to run when in eight-bit mode.

The primary polarity select bit (T2POL0) selects the initial polarity of the primary output. Changing this bit after the output has been enabled via T2OE0 has no effect. Setting the primary output enable bit (T2OE0) turns on the output for the primary pin and sets its value equal to the value in the polarity bit (T2POL0).

Setting the primary interrupt enable (ET2) allows interrupts to be generated provided that they are enabled for the timer's module (set the appropriate bit in the IMR register) and global interrupts have been enabled (IC register bit 0 set to 1). Interrupts are generated when the primary counter overflows (reaches FFFFh) or matches the compare register. In these cases the appropriate bit (TF2 for overflow or TCC2 for a compare) will be set and should be reset by firmware in the interrupt handler. Failing to reset these bits will cause repeated interrupts until they are reset or the interrupt has been disabled.

T2CNB contains compare and overflow flags, the secondary interrupt enable, and the secondary output polarity and enable bits.

ET2L	T2OE1	T2POL1	Reserved	TF2	TF2L	TCC2	TC2L
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

The capture/compare flag (TCC2) is set when the primary counter value matches the compare value.

The low capture/compare flag (TC2L) is similar to TCC2, but is set only when in eight-bit mode and the low or secondary counter matches the low compare value.

The overflow flag (TF2) is set when the primary counter overflows. The low overflow flag (TF2L) is similar to TF2, but is set only when in eight-bit mode and the low or secondary counter overflows. The secondary polarity select bit (T2POL1) selects the initial polarity of the secondary or B output pin. Changing this bit after the output has been enabled via T2OE1 has no effect. Setting the secondary output enable bit (T2OE1) turns on the output for the secondary pin and sets its value equal to the value in the polarity bit (T2POL1). The secondary output is not directly linked to the secondary counter, since in sixteen-bit mode the primary counter sources it but in eight-bit mode the low counter sources it.

Setting the secondary interrupt enable (ET2L) allows interrupts to be generated when the TF2L or TC2L bits are set by an overflow (TF2L) or compare (TC2L) of the secondary or low eight-bit counter when in eight-bit mode. The ET2L bit is not used in sixteen-bit mode.

Examples

Compare Example 1 - Output a waveform with gating
The following code will output a signal with a frequency of 100 Hz and a duty cycle of 1/3. The code was written for a clock speed of 4.9152 MHz. The reload value of 4000h (16384 decimal) provides C000h (49152 decimal) clock cycles between the reload and the overflow and subsequent reload for a period of 10 ms (100 Hz). The compare value of C000 gives us 32768 clock cycles (C000h - 4000h) or 6.7 ms after the reload that the compare (and consequently, the pulse edge) will occur.

The T2POL1 bit sets the initial value since we are using the secondary output, the primary being used for gating. The T2POL0 bit selects the gating level in this case. With T2POL1 set to 0, the initial value of the output is 0, so the output is low for 6.7 milliseconds followed by high for 3.3 milliseconds. The 1/3 duty cycle waveform will be output as long as the primary pin is held high. When the primary pin is pulled low the counter will stop counting and will hold the secondary pin at the current level. The waveform can be inverted (making a 2/3 duty cycle pulse) by setting T2POL1 to 1. The gating level can be changed to active high by setting T2POL0 to 1.

```
move    T2V0, #04000h ; set to reload value to keep first pulse
  ; from being extra long
        move    T2R0, #04000h ; reload value
        move    T2C0, #0C000h ; compare value

        move    T2CFG0, #000h ;
; 0000,0000 - use system clock (0), divide by 1 (000),
;       16 bit mode (0), compare mode (00), c/t2=timer (0)

move    T2CNB0, #040h
; 0100,0000 - ET2L off - low interrupts not available in 16-bit mode (0),
;       secondary OE is on (1), POL1 = low starting value (0), reserved (0)
;       TF2 is not used (0), TF2L is not used (0), TCC2 is not used (0),
;       TC2L is not used (0)

move    T2CNA0, #009h
; 0000,1001 - ET2 off - interrupts not needed (0),
;       primary OE off as primary pin is used for gating (0),
;       POL0 low, gated when primary pin is low (0), TR2L is not needed (0)
```

```
;        TR2 on, run enabled (1), CPRL2 is not needed (0),
;        SS2 is not needed (0), gating enabled (1)
```

Compare Example 2 - Single shot pulse
The following code was written for a MAXQ2000 part, which has 3 separate Timer 2s. It uses the third timer, which is in module 4. It is also written for a system clock frequency of 4.9152 MHz and will output a low pulse with a width of two milliseconds when triggered, with the output normally being high. At the end of the pulse an interrupt will be generated to indicate that the pulse has finished. This code uses the timer in eight-bit mode and also demonstrates use of the prescaler.

To obtain the desired two millisecond period, a prescale value of 64 and a timer period of 154 prescaled clocks (4915200 / 64 * 0.002 = 153.6) are used. Selecting a compare value of 66h (100h - 9Ah) gives us 9Ah (154 decimal) ticks before the counter overflows from FFh to 00. Setting the reload value to 65h causes the pulse to start 1 tick after we set the SS2 (single shot) bit.

```
SetupPulse:
        ; This code sets up the timer and should be run once
        ; set up Int handler
        move    IV, #IntHandler     ; Set interrupt vector.
        move    IC.0, #1            ; Enable global interrupts.
        move    IMR.4, #1           ; Enable interrupts for module 3.
        ; timer 0 is in module 3, timer 1 & 2 are in module 4

        move    T2CFG2, #068h ;
; 0110,1000 -- use system clock (0), divide by 64 (110),
;        8 bit mode (1), compare mode (00), c/t2=timer (0)

        move    T2CNB2, #000h
; 0000,0000 -- ET2L off, low interrupts not needed (0), secondary OE off (0),
;        T2POL1 = not used (0), reserved(0)
;        TF2 is generated by the timer (0), TF2L is not used (0), TCC2 is not
used (0),
;        TC2L is not used (0)

        move    T2CNA2, #0E0h
; 1110,0000 -- ET2 on, interrupt will be generated (1), primary OE on (1),
;        T2POL0 is high (1), TR2L is not needed (0)
;        TR2 off, run not enabled (0), CPRL2 is not needed (0), SS2 is set
later (0),
;        gating disabled (0)

        move    T2H2, #065h ; set to reload value
        move    T2RH2, #065h ;
        move    T2CH2, #066h ; 0x100 - 0x66(compare value) = 0x9A = 154 ticks
```

The following code should be called whenever the pulse is to be triggered.

```
TriggerPulse:
        move T2CNA2.1, #1 ; set the single shot bit to start the timer
ret
The following is a piece of the interrupt code.
IntHandler:
move c, T2CNB2.3
jump nc, NonTimerInt
move T2CNB2.3, #0 ; turn off overflow bit so interrupt is serviced
; code for end of pulse here...

NonTimerInt:
        ; other interrupt code here.

reti
```

Compare Example 3 - Timed interrupts
The following code will generate two interrupts, one every 125 microseconds and a second one every

millisecond. It uses one timer split into two eight-bit timers. Since both eight-bit timers will be running from the same input clock, we want to pick a clock divisor that will allow both timers to have a count of less than 256 (the maximum for a eight-bit counter). Using the alternate clock as source provides us with a way to get both timings without needing to divide down the system clock, at the cost of some accuracy: the actual timing is closer to 122 microseconds for the high counter and 0.98 milliseconds for the low counter since the alternate clock runs at 32768 Hz and no even divisor for the required periods is available. We will get 1024 interrupts per second from the low counter and 8192 interrupts per second from the high counter. Both high and low interrupts are enabled and the type of interrupt is identified by the TF2 and TF2L bits which must be cleared to service the interrupt.

```
; This code sets up the timer and should be run once
        ; set up Int handler
        move    IV, #IntHandler     ; Set interrupt vector.
        move    IC.0, #1            ; Enable global interrupts.
        move    IMR.3, #1           ; Enable interrupts for module 3.
        ; timer 0 is in module 3

        move    T2CFG0, #088h ;
; 1000,1000 -- use 32 kHz clock (1), divide by 1 (000),
;       8 bit mode (1), compare mode (00), c/t2=timer (0)

        move    T2CNB0, #080h
; 1000,0000 -- ET2L on, low interrupt will be generated (1), secondary OE off
(0),
;       T2POL1 = not used (0), reserved(0)
;       TF2 is generated by the timer (0), TF2L is generated by timer (0),
;       TCC2 is not used (0), TC2L is not used (0)

        move    T2CNA0, #080h
; 1000,0000 -- ET2 on, interrupt will be generated (1), primary OE off (0),
;       T2POL0 is not needed (0), TR2L off, will be set later (0)
;       TR2 off, will be set later (0), CPRL2 is not needed (0),
;       SS2 is not needed (0), gating disabled (0)

        move    T2H0, #0FCh ; set to reload value to keep first pulse from
being extra long
        move    T2RH0, #0FCh ; reload value (0x100 - 0xFC = 4 ticks)
        move    T2CH0, #000h ;

        move    T2V0, #0E0h ; set to reload value to keep first pulse from
being extra long
        move    T2R0, #0E0h ; reload value (0x100 - 0xE0 = 32 ticks)
        move    T2C0, #000h ;

        move    ACC, T2CNA0 ; turn on high run and low run (TR2, TR2L)
        or      #018h ;
        move    T2CNA0, ACC
```

The following is part of the interrupt handler code:

```
IntHandler:
        move c, T2CNB0.3
        jump nc, No8KHz
        ; code for 8KHz interrupt here
        move T2CNB0.3, #0


No8KHz:
        move c, T2CNB0.2
        jump nc, No1KHz
        ; code for 1KHz interrupt here
        move T2CNB0.2, #0


No1KHz:
        ; other interrupt code here
```

```
reti
```

Capture Example - Time an incoming waveform

This example times an incoming signal. This code uses the second timer (they are numbered 0,1, and 2) and is set up to time the duration of a high pulse. It does not start timing until a rising edge is seen. The CPRL2 bit enables a reload upon capture so that subsequent pulses can also be timed. The T2POL [0] and SS2 bits have slightly different meanings in this mode. The SS2 bit (single shot) is used to inhibit counting until the beginning edge is detected. This enables the timer to be set up at any time, even when the input is currently high since the timer will not start until the next rising edge. The T2POL [0] bit selects the gating level instead of the output polarity as in the previous examples. A gating level of 0 prevents the counter from running while the input is low. This example is geared for longer pulses and divides the system clock by 128. With a system clock frequency of 4.9152 MHz, the timer has a resolution of approximately 26 microseconds, and can count to 1.7 seconds before overflowing.

The following code sets up the capture.

```
; set up Int handler
        move    IV, #IntHandler        ; Set interrupt vector.
        move    IC.0, #1               ; Enable global interrupts.
        move    IMR.4, #1              ; Enable interrupts for module 3.
            ; timer 0 is in module 3, timer 1 & 2 are in module 4

        ; repeated capture of high pulse
        move T2CFG1, #074h ;
; 0111,0100 -- use system clock (0), divide by 128 (111),
;       16 bit mode (0), capture on falling edge (10), c/t2=timer (0)

        move T2CNA1, #08Fh ;
; 1000,1111 -- ET2 on, interrupt will be generated (1), primary OE off (0),
;       T2POL0 (gating) at low (0), TR2L off, will be set later (0)
;       TR2 on (1), CPRL2 (reload on capture) is on (1),
;       SS2 on (1), gating enabled (1)

        move    T2CNB1, #000h
; 0000,0000 -- ET2L off, low interrupt not used (0), secondary OE off (0),
;       T2POL1 = not used (0), reserved(0)
;       TF2 is generated by the timer (0), TF2L is not used (0),
;       TCC2 is not used (0), TC2L is not used (0)
```

The following is part of the interrupt handler code:

```
IntHandler:
        ; looking for Timer2 interrupt...

move c, T2CNB1.1 ; capture/reload flag
jump nc, NoCapture
move T2CNB1.1, #0
; put code for capture event here
move ACC, T2C1 ; capture value now in ACC

NoCapture:
move c, T2CNB1.3 ; overflow flag
jump nc, NoOverflow
        move T2CNB1.3, #0 ;
        ; put code to deal with overflow here
        ; pulse was too long to measure with current clock speed and divisor
        ; can add 65536 to a 32-bit value to keep counting

NoOverflow:
        ; put other interrupt code here

reti
```

Counter Example - Count incoming transitions with interrupt and output waveform
The following example counts incoming pulses on the primary pin and generates an interrupt after every
eight pulses. It also controls an output waveform on the secondary pin, which is toggled once every eight
input pulses.

```
        ; This code sets up the timer and should be run once
        ; set up Int handler
        move    IV, #IntHandler    ; Set interrupt vector.
    move    IC.0, #1               ; Enable global interrupts.
    move    IMR.3, #1              ; Enable interrupts for module 3.
        ; timer 0 is in module 3

        move    T2V0, #0FFF8h ; set to reload value
        move    T2R0, #0FFF8h ; reload value 0x10000 - 0x0ffff = 8 ticks
        move    T2C0, #00000h ;

        move    T2CFG0, #003h ;
; 0000,0011 -- use system clock (0), divide by 1 (000),
;       16-bit mode (0), rising edge (01), c/t2=counter (1)

        move    T2CNB0, #060h
; 0110,0000 -- ET2L not used (0), secondary OE on (1),
;       T2POL1 start at high (1), reserved(0)
;       TF2 is generated by the timer (0), TF2L is not used (0),
;       TCC2 is generated by the timer (0), TC2L is not used (0)

        move    T2CNA0, #088h
; 1000,1000 -- ET2 on, interrupt will be generated (1), primary OE not used
(0),
;       T2POL0 is not used (0), TR2L is not needed(0)
;       TR2 on (1), CPRL2 is not needed (0),
;       SS2 is not needed (0), gating disabled (0)
```

The following is part of the interrupt handler

```
IntHandler:

        move c, T2CNB0.3
        jump nc, NoTimer
        move T2CNB0.3, #0 ; service interrupt
        ; put code for every 8 pulses here

NoTimer:
        ; other interrupt code here

reti
```

Some Common Pitfalls to Avoid

When in compare mode, if the compare and reload values are equal a second transition on the output
occurs one clock cycle after a reload occurs. While this is a valid option when you want a pulse width of
just one clock cycle, it's easy to let this happen without intending it since they both default to the same
value of 0. If the compare value is not to be used it should be set outside of the range used by the timer.
Normally setting the compare value to something less than the reload value will do this. If this is not
possible (due to using a reload value of 0) then the compare value should be set to FFFFh in sixteen-bit
mode or FFh in eight-bit mode. This causes the compare and overflow events to happen on the same
timer clock cycle preventing the second transition of the output.

Output enables should be turned on before or at the same time as the run enable. Setting the run enable
before the output enable can cause the output to be inverted since it is possible for interrupts to cause the
code to be suspended (while the interrupt is serviced) in between the time the timer is set to run and when
the output enable is asserted. This can cause a compare or overflow event to happen before the output is

enabled, which causes the output to be of the opposite polarity. This usually happens with the B or secondary output since the output enable (T2OE1) is in the T2CNB register while the run bit (TR2) and low run bit (TR2L) are in the T2CNA register. In this case the T2CNB register should be set up first or the T2CNA register should be set up with the run bits set to zero and then set the run bits after all registers have been configured.

Appendix: Table of bit settings for various modes

MODE	T2MD	C/T2	CCF[1:0]	T2OE[0]	T2OE[1]	T2POL[0]	T2POL[1]	G2EN	SS2	CPRL2
16 bit compare with optional gating	0	0	00	0 = primary output not enabled	0 = secondary output not enabled 1 = secondary output enabled	Defines gating level 0 = low 1 = high	Defines initial polarity of secondary output 0 = low 1 = high	0 = no gating 1 = gating enabled on primary pin	1 = Single shot 0 = use TR2 for run	0 Not used
16 bit compare with primary output enabled (no gating available)	0	0	00	1 = primary output enabled	0 = secondary output not enabled 1 = secondary output enabled	Defines initial polarity of primary output 0 = low 1 = high	Defines initial polarity of secondary output 0 = low 1 = high	0 Not usable with primary output enabled	1 = Single shot 0 = use TR2 for run	0 Not used
16 bit capture Capture on one edge only	0	0	01 = Capture on rising edge 10 = Capture on falling edge	0 Not used	0 Not used	Gating level on primary pin 0 = low 1 = high	0 Not used	1 = Gating enabled 0 = Gating disabled	1 = Single shot 0 = use TR2 for run	Capture and reload 0 = off 1 = on
16 bit capture Capture on both edges, no reload	0	0	11 = Capture on both eges	0 Not used	0 Not used	Gating level on primary pin 0 = low 1 = high	0 Not used	1 = Gating enabled 0 = Gating disabled	1 = Single shot 0 = use TR2 for run	0 = Capture and reload off
16 bit capture Capture on both edges, reload on both	0	0	11 = Capture on both edges	0 Not used	0 Not used	Start/stop edge 0 = start/stop on rising 1 = start/stop on falling	0 Not used	0 = reload on all edges	1 = Single shot T2POL[0] defines start/stop edge 0 = use TR2 for run	1 = Capture and reload on
						Non			1 =	

16 bit capture Capture on both edges, reload on one edge	0	0	11 = Capture on both edges	0 Not used	0 Not used	reload edge and start/stop edge 0 = falling (start/stop on rising) 1 = rising (start/stop on falling)	0 Not used	1 = reload on one edge only T2POL[0] defines capture only edge	Single shot T2POL[0] defines start/stop edge 0 = use TR2 for run	1 = Capture and reload on
16 bit counter	0	1	01 = rising 10 = falling 11 = both 00 = none (Not Used)	0 Not used Primary pin is input only	0 = secondary output not enabled 1 = secondary output enabled	0 Not used Primary pin is input only	Defines initial polarity of secondary output 0 = low 1 = high	0 Not used Do not set to 1	0 Not used Do not set to 1	0 Not used
Dual 8 bit compare with gating	1	0	00	0 = primary output not enabled	Secondary counter output enable 0 = not enabled 1 = output enabled	Defines gating level 0 = low 1 = high	Defines initial polarity of secondary counter outpu 0 = low 1 = high	0 = no gating 1 = gating enabled on primary pin	0 Not used	0 Not used
Dual 8 bit compare with no gating	1	0	00	1 = primary output enabled	Secondary counter output enable 0 = not enabled 1 = output enabled	Defines initial polarity of primary output 0 = low 1 = high	Defines initial polarity of secondary counter output 0 = low 1 = high	0 Not usable with primary output enabled	1 = Single shot 0 = use TR2 Only applies to primary counter	0 Not used
8 bit capture + 8 bit compare	1	0	01, 10, 11 The primary counter acts just as in 16 bit mode (except with only 8 bit resolution)	See 16 bit capture modes	Secondary counter output enable 0 = not enabled 1 = output enabled	See 16 bit capture modes	Defines initial polarity of secondary counter output 0 = low 1 = high	See 16 bit capture modes	See 16 bit capture modes	See 16 bit capture modes
8 bit counter +			01 = rising 10 = falling	0 Not used Primary	Secondary counter output enable	0 Not used	Defines initial polarity of secondary	0 Not used	0 Not used	0 Not

8 bit compare	1	1	11 = both 00 = none (Not Used)	pin is input only	0 = not enabled 1 = output enabled	Primary pin is input only	counter output 0 = low 1 = high	Do not set to 1	Do not set to 1	used

Related Parts

MAXQ2000	Low-Power LCD Microcontroller	Free Samples

More Information

For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

---