

# Timing Performance of TPU I/O Hardware

by Richard Soja and Sharon Darley

## INTRODUCTION

This application note describes the timing relationships between Time Processor Unit (TPU) I/O pins and the system clock used to drive the TPU module. These relationships, rather than the software latencies associated with event scheduling, define actual hardware performance. A working example is provided in the text. The example shows how an output pulse from one TPU pin can be triggered by an input edge applied to another TPU pin, with no software overhead. The delay between the trigger edge and output pulse is strictly defined by constant hardware delays associated with the TPU I/O pins.

## TPU INPUT AND OUTPUT

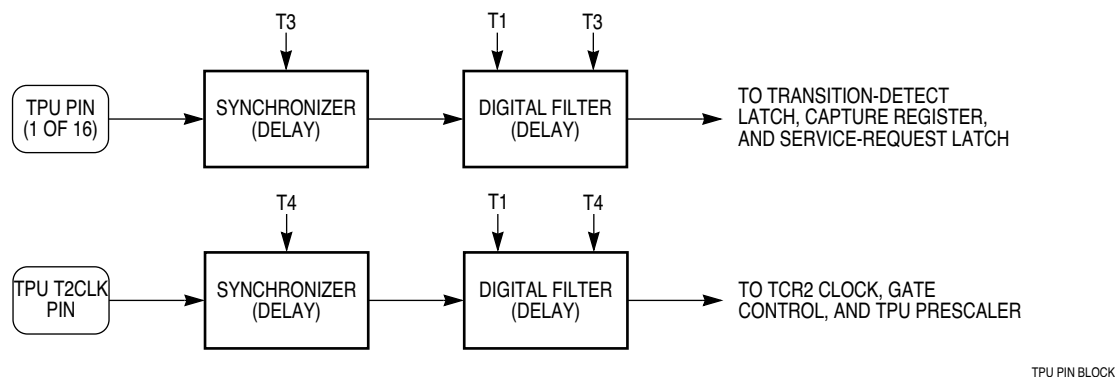
The TPU has a total of 17 pins. Sixteen of the pins, designated TPUCH0 to TPUCH15, may be assigned to timing functions. The 17th pin, T2CLK, can be used either to connect an external clock source to increment the internal TCR2 counter, or to gate the internal clock to TCR2.

In most applications, the function to which a pin is assigned determines whether the pin is an input or an output. However, it is possible to execute a function without making use of an assigned pin. For example, when the Read TCR1 and TCR2 mode of the Output Compare (OC) function is used, the CPU can issue a service request to the OC function without using pin hardware. While a pin would normally remain assigned to input or output during application operating time, specialized functions could reassign a pin during execution. TPU hardware supports this capability, but availability depends on the function.

## DIGITAL FILTER AND SYNCHRONIZER

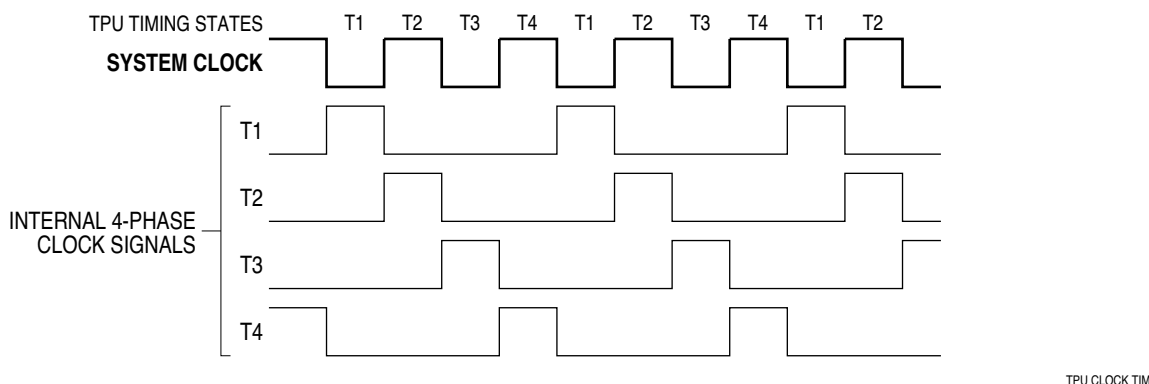
To give the TPU a certain amount of noise immunity, each input pin, including the T2CLK pin, has a digital filter that rejects short-duration pulses. Preceding each filter is a synchronizer, which re-times the incoming signal to the internal system clock. Figure 1 is a block diagram of input pin configuration. Both the digital filter and the synchronizer delay incoming signals. The amount of delay is quantifiable, and depends on the phase relationship between the input edge and the appropriate timing state of the TPU system clock.





**Figure 1 TPU I/O Pin Block Diagram**

The TPU system clock is directly related to the internal system clock used by all other on-chip modules. This internal clock signal is made available to users on the CLKOUT pin. The TPU system clock can be decomposed into four timing states; T1, T2, T3 and T4. Each timing state is one-half of a system clock cycle in duration. Clock timing relationships are shown in Figure 2.

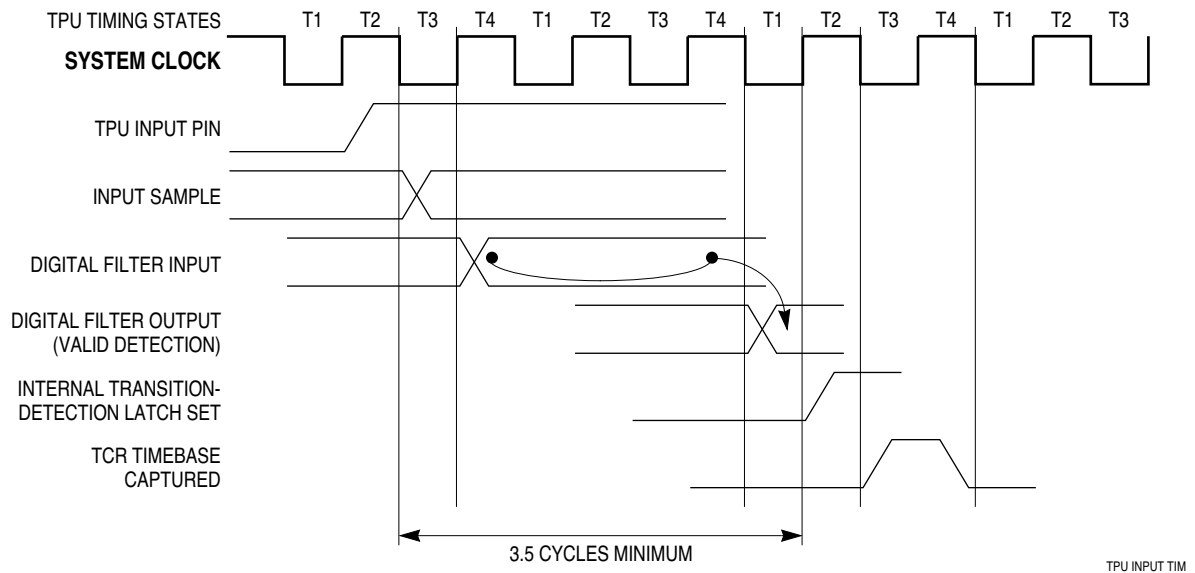


**Figure 2 TPU I/O Pin Clock Timing**

Each timing state performs a specific action within the TPU. Timing states are used by the microengine and execution unit as well as pin hardware. The synchronizers and digital filters are clocked by different combinations of signals derived from T1, T3 and T4. The T2CLK synchronizer uses T4 and the associated digital filter uses T1 and T4. Each of the 16 input pin synchronizers uses T3 and each of the associated digital filters use T1 and T3.

Timing for the input pins (refer to Figure 3) is such that an input is sampled at the start of T3 and is applied to the digital filter at the end of T3. Because the pin is sampled in this way, the synchronizer effectively quantizes an input pulse into integral numbers of two-system-clock-cycle periods (that is, 0, 2, 4, 6, etc., cycles in width). In addition, this sampling method causes a delay between the input pin and the input to the digital filter that ranges from a minimum of 0.5 system clock cycle to a maximum of 2.5 system clock cycles. The exact amount of delay depends on the alignment between the input signal and the TPU T3 timing state. This is discussed in detail later.

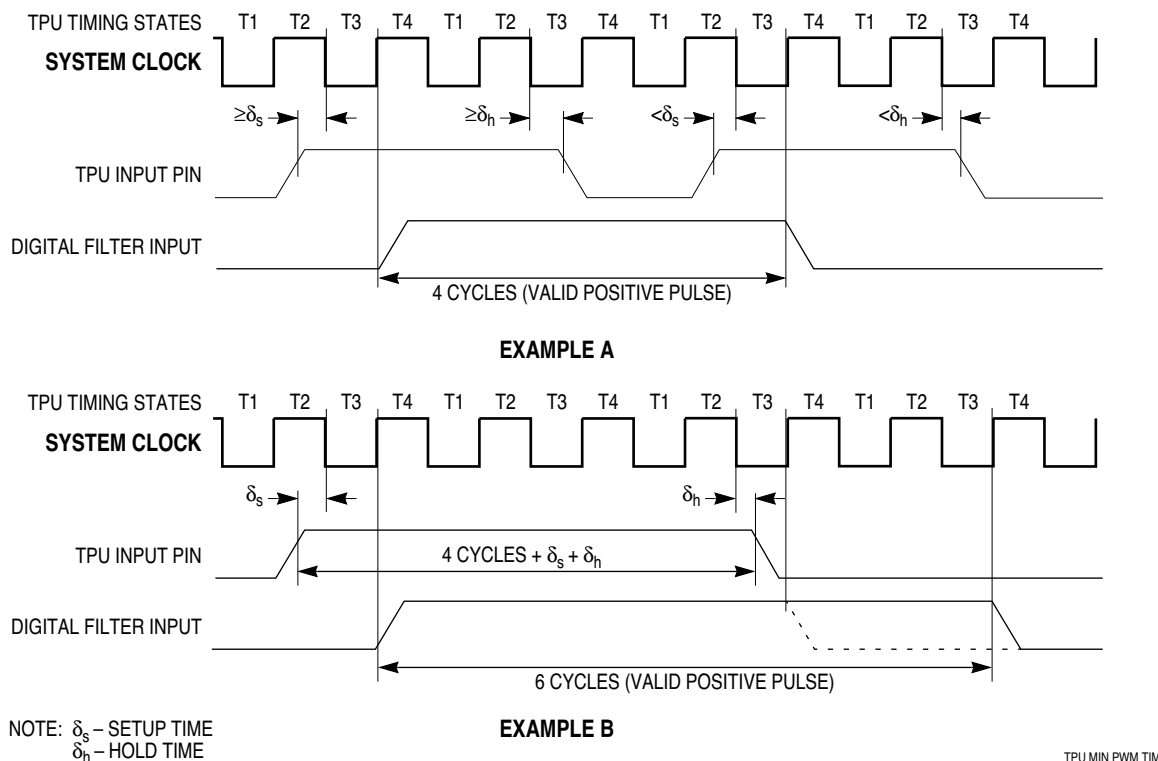
The digital filter is basically a two-stage latch followed by some voting logic with a latched output. For an input to be validated by the digital filter, it must be stable for more than two system clock cycles. An input to the digital filter of exactly two cycles duration is rejected. Since synchronizer output occurs in increments of two cycles, the minimum pulse accepted by the digital filter is four cycles wide. The total delay between an input transition at the pin, and the setting of the internal transition detection latch (TDL) is a minimum of 3.5 cycles and a maximum of 5.5 cycles. The timebase capture occurs 0.5 clock cycles later.



**Figure 3 TPU Input Timing**

### MINIMUM PULSE WIDTHS

Figure 4 shows possible combinations of input timing, based on various pulse widths and timing relationships between a TPU input pin and the T3 sampling point. The following description also applies to T2CLK input, except that the T2CLK pin is sampled in the T1 state.



**Figure 4 Minimum Pulse-Width Timing Scenarios**

One input pin timing scenario can “mislead” the TPU and cause it to operate incorrectly. Example A (Figure 4) shows the theoretical case of an input pulse rate which could be recognized as a valid signal at half the frequency. This is akin to the aliasing effects seen in signal processing applications. The first positive pulse is sampled on two consecutive T3 states because it meets the setup timing for the first T3 state and the hold timing for the second T3 state. The pulse is therefore recognized as a valid positive pulse, as it quantizes internally to four system clock cycles. The second positive pulse is not detected because it does not meet the setup and hold requirements for the next two T3 states. If the input signal were to repeat in this manner, and maintain phase synchronism with the system clock, the TPU would detect a valid signal at half the input rate. Even when the input frequency of these pulses is well within the limits defined by TPU latency, if the input positive pulse widths are as shown in Example A, the input hardware will detect an alias frequency. The same situation applies if the input waveform consists of short negative pulses in place of short positive pulses.

As the first pulse in Example A shows, pulses that are just greater than two system clock cycles in width can be validly detected, provided they meet the minimum setup and hold timings for consecutive sampling points. In this particular case, the input effectively must be synchronized to the system clock. In most practical applications, however, the input pulse would be completely asynchronous to the system clock.

To ensure that at least two consecutive samples of the same pin level are always made under worst case conditions, the input pulse width must be at least four system clock cycles plus the sum of the minimum setup and hold times, as shown in Example B in Figure 4. Here, the input pulse rises just before the first T3 state, and falls just after the third T3 state. This means that three samples are made. However, any incremental shift in phase between the pulse and the system clock will violate either the minimum setup or hold requirements, depending on whether the phase shift is positive or negative. This will result in the minimum of two samples being taken, which will still be detected as valid.

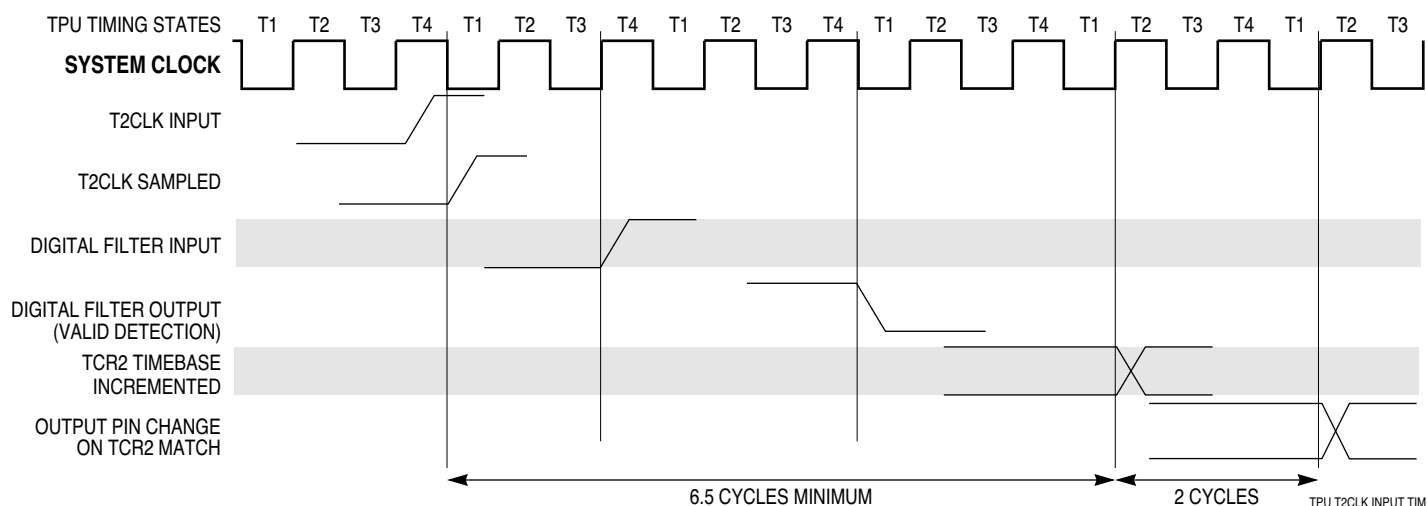
If the asynchronous input pulse width is just less than four system clock cycles plus the setup and hold times, the pulse will at times be aligned such that neither the setup nor hold requirements are met. Under these circumstances, only one T3 sample is made, so that the pulse is undetected. The minimum setup and hold times are specified as 15 ns each. This means that the minimum input pulse width guaranteed to be

detected under all conditions is 30 ns plus four system clock cycles. Since this applies to both low and high pulses, a square wave applied to the input pin at the minimum pulse width must have a perfect 50% duty cycle. If it does not, the frequency must be reduced to a value which assures the minimum pulse width. Because all TPU functions are synthesized in microcode, the minimum period/pulse width which actually can be measured is very dependent upon function latency.

The previous discussion is most relevant when considering the maximum rate at which TCR2 can be clocked by the T2CLK pin. T2CLK input timing is shown in Figure 5.

T2CLK synchronizer and digital filter hardware are identical to the other pins, except that T1 and T4 are used for re-timing and sampling of the input signal applied to the pin. In addition to the delays through the synchronizer and digital filter, the T2CLK input acquires delays due to the prescaler. The minimum delay between a rising edge on the input pin, and the incrementing of TCR2 is 6.5 system clock cycles. This occurs when T2CLK input rises just before the falling edge of T4, i.e., when the minimum set up time requirements have been met. This is the situation shown in Figure 5. The maximum delay is 8.5 cycles, which occurs when the T2CLK input rises just after the falling edge of T4.

The TCR2 timebase bus changes in timing state T2 (as does the TCR1 timebase). If a match results from this new timebase value, a pin configured for output will change level in the next T2 timing state. This is also shown in Figure 5. The maximum delay between T2 rising edge and output pin valid is 23 ns.



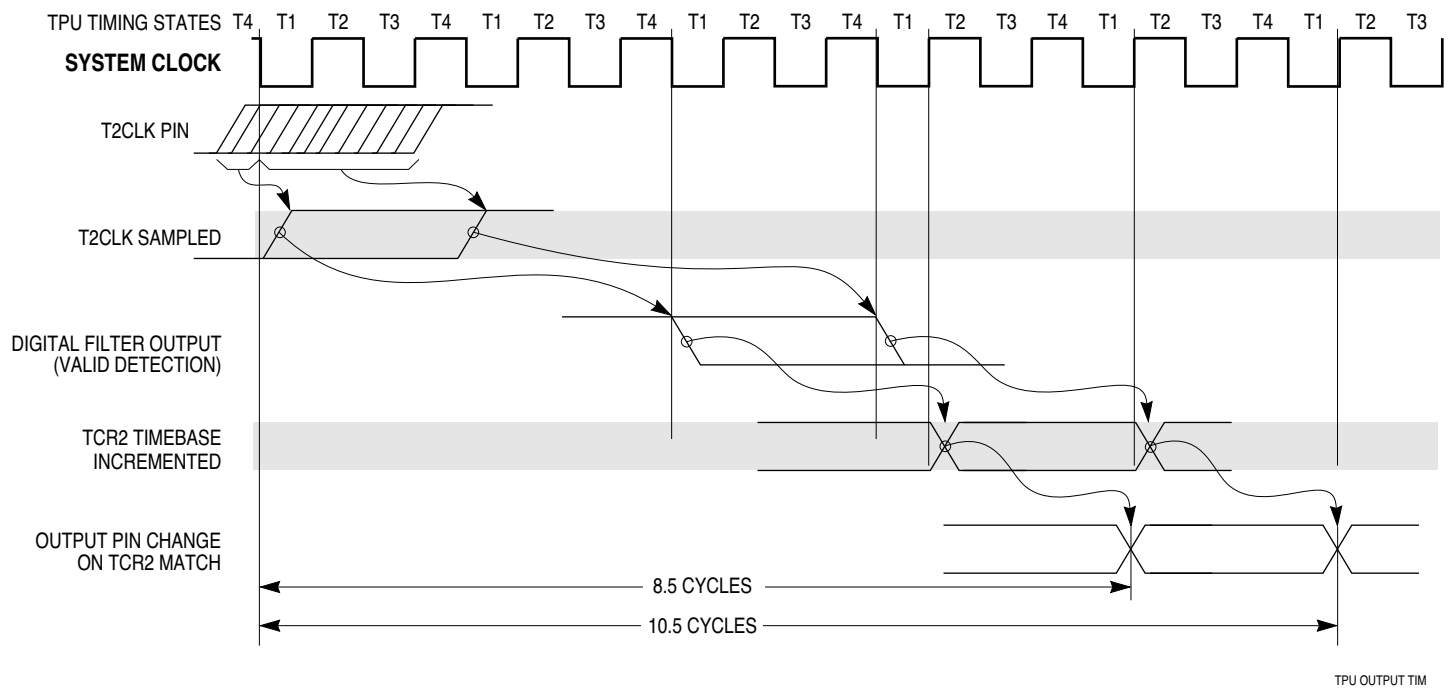
**Figure 5 T2CLK Input Timing**

## INCREMENTING T2CLK

### Asynchronously

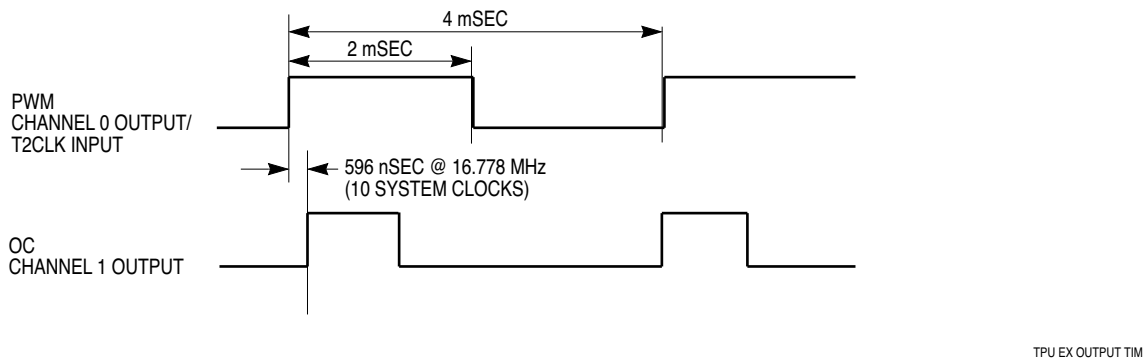
Consider an application where an OC function is configured to match on TCR2, and TCR2 is incremented by an external clock which is asynchronous to the TPU system clock. If the new value of the TCR2 timebase results in a match event, the OC pin should change state in the next T2 time state. However, due to the asynchronism of T2CLK input, the output pin transition will occur a minimum of 8.5 and maximum of 10.5 system clock cycles after the rising edge of T2CLK input. Thus, if T2CLK is derived from an external source which is asynchronous to the TPU system clock, the OC signal will jitter over a two-cycle range with respect to the T2CLK input, as shown in Figure 6. Exact timing depends on whether the input edge occurs before or after the referenced sample point at the end of T4.

**Figure 6 T2CLK Incremented Asynchronously**



### Synchronously

If the T2CLK input is derived from another TPU (output) pin, there is no jitter on the OC transition. The timing relationship between the T2CLK input and OC match is now fixed, since the signal applied to the T2CLK input is timed to the T2 timing state. In this case, the delay between T2CLK rising edge input and the OC transition is exactly ten cycles, as shown in Figure 7.



**Figure 7 T2CLK Incremented Synchronously**

## PRACTICAL EXAMPLE

The following example illustrates two aspects of TPU hardware capabilities. One is the ability to trigger output edges with no microcode execution, and the other is the ability to cause a match on one of two timebases while capturing the other.

The example generates single shot output pulses triggered by each rising transition of a pulse train. Figure 7 shows the timing relationships between output and trigger waveforms. The delay between the trigger edge and the first output edge of the pulse is purely a function of delays caused by the TPU hardware synchronizer and digital filter, as explained in previous sections of this application note. There is no software latency caused by microcode execution. This is achieved by using the OC function in conjunction with the T2CLK input pin. The OC function is first initialized to generate a rising edge on a match with the current value of TCR2 plus one. The trigger signal is applied to T2CLK input, and when a rising edge occurs, TCR2 is incremented, a match occurs on the OC function, and the associated pin goes high. All of this takes place without microcode execution. However, the resulting match event will eventually cause the appropriate state in the OC function to be executed.

In this example, the OC function is set up to generate an interrupt to the CPU. The interrupt occurs only when the appropriate OC state is executed by microcode, and not at the time the match event occurred. This means there is a delay caused by software before the CPU can respond to the pin transition. When the interrupt occurs, the CPU interrupt handler redefines the OC parameter RAM so that the OC pin will go low after a programmed duration. The duration is defined in terms of the free running counter, TCR1. This means that:

- a) The timebase reference for the OC match hardware must be changed from TCR2 to TCR1.
- b) The point in time that the OC pin should go low must now be referenced from the 'real time' when the match on TCR2 occurred.

Fortunately, TPU pin hardware operates in a way which makes this simple to deal with, without the need to use other pins. The hardware on any one pin is able to match on one of the two timebases and at the same time capture the other. So, in this example, the OC function is initialized to match on TCR2 and capture TCR1. When the OC interrupt occurs as a result of the trigger edge, the CPU interrupt handler switches the OC match timebase from TCR2 to TCR1. In addition, pin operation is also redefined to cause a falling transition after a programmable delay referenced from the TCR1 value captured when the match occurred. When the delay expires, the pin goes low and the subsequent interrupt causes the interrupt handler to set up the original initial conditions, which include matching on TCR2 and capturing TCR1.

Both TPU software overhead and CPU software overhead are associated with scheduling the falling edge—software latencies incurred by the appropriate OC state execution and CPU interrupt handler response time define minimum pulse width. In contrast, maximum delay between the trigger point and the rising edge is 10.5 system clock cycles. With a 16.778 MHz system clock, this is equivalent to 626 ns. The maximum trigger rate is a function of software overhead, in response to both rising and falling edges of the OC pin.

To ease the evaluation of the timing features provided by the TPU in this example, the trigger signal for the one-shot pulse is obtained from TPU Channel 0. Channel 0 is configured as a pulse width modulated (PWM) output, and the output is connected directly to T2CLK.

The one-shot pulse output is obtained from Channel 1. Channel 1 is configured for OC function, in host-initiated pulse mode, and thus generates a single transition at a specified delay from a reference time.

The CPU must make a host service request to 'arm' the function. This results in the TPU scheduling a match to occur on a particular timebase, and perform a CPU-specified action on the pin. Initially, and after every OC falling edge, the CPU sets up the OC to match on the current TCR2 timebase value plus one. When the next rising edge occurs on T2CLK input, the TCR2 value is incremented, a match occurs and the pin goes high. The subsequent interrupt causes the CPU to reconfigure the OC function to generate a falling transition at a CPU-defined offset from the time that the pin went high, thus defining the high time for the pulse. The actual time the OC pin goes high is captured by the pin hardware as the value of TCR1 at the time of the last match on TCR2.

Because the trigger is derived from the TPU PWM function, the delay between this signal and the rising edge of the OC pulse is a constant ten system clock cycles, as shown in Figure 7. If an external asynchronous trigger source were applied to T2CLK input, the delay between its rising edge and that of the OC would jitter between 8.5 and 10.5 system clocks.

## **DETAILED DESCRIPTION OF TPU SETUP**

### **CHANNEL 0**

Channel 0 generates the trigger signals which are input to the TCR2 counter. It must be physically connected to the external T2CLK pin. Channel 0 is configured as a PWM function with 50% duty cycle PWM. Its timebase is TCR1. Parameter RAM for Channel 0 is initialized as follows:

#### **CHANNEL\_CONTROL = \$92**

This field determines whether a pin is an input or an output. When a pin is configured as an output, as in this example, CHANNEL\_CONTROL also determines the initial pin state when the host service request is accepted, the new pin state when a match occurs, the timebase used for the match, and the timebase used for the capture. The example value causes Channel 0 to capture and match on TCR1, forces the TPUCH0 pin low upon initialization, and sets it up to go high when the next match occurs.

#### **PWMHI = \$400**

This parameter determines the high time, or pulse width, of the PWM signal. The hexadecimal number stored in this parameter can be converted to a decimal value and then multiplied by the TCR1 period to get the actual period in seconds. In this example, the default state of the prescaler and PSCK fields in the TPUMCR register are used, giving a pulse high time of approximately 2 ms with a 16.778 MHz system clock.

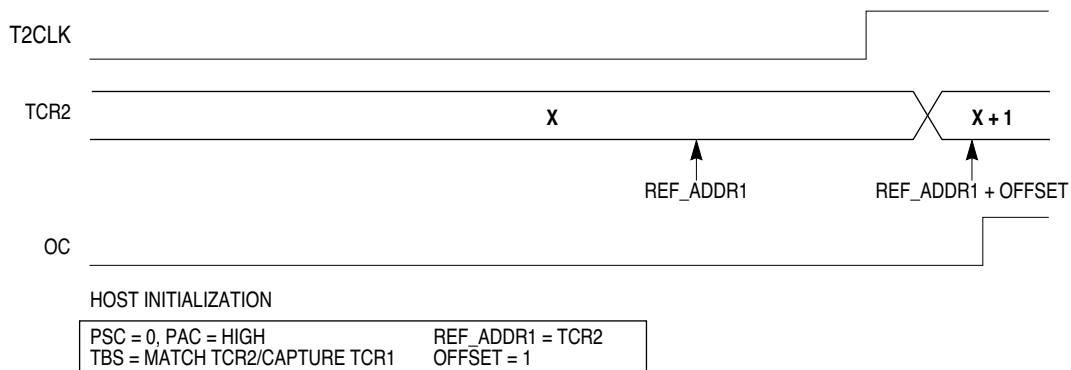
#### **PWMPER = \$800**

PWMPER determines the period of the PWM signal. In this example, using the default state of the prescaler and PSCK fields in the TPUMCR register gives a period of approximately 4 ms with a 16.778 MHz system clock.

### **CHANNEL 1**

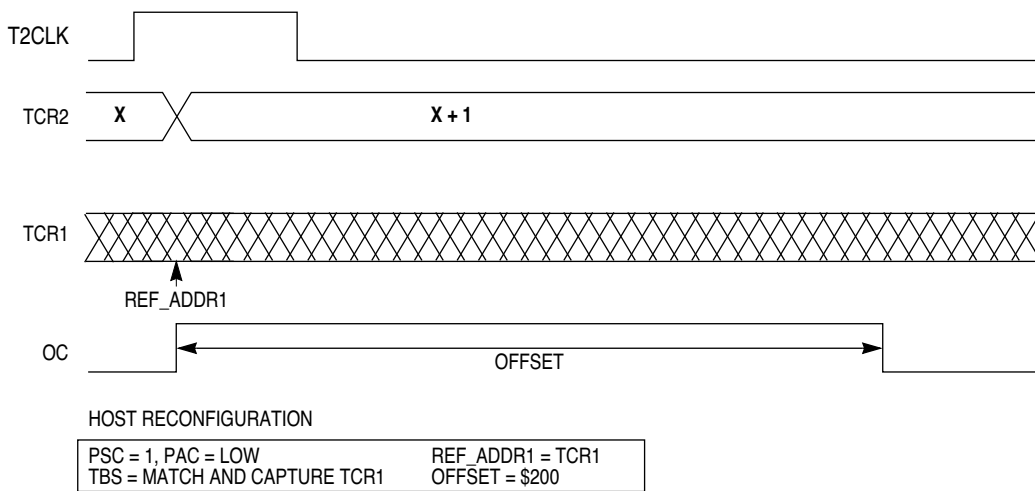
This channel is configured as an OC function using the host-initiated pulse mode. To synthesize the one-shot pulse triggered by an input on T2CLK, the channel is reconfigured on every edge it generates, using a CPU interrupt handler. Each rising edge is synchronized to TCR2, and each falling edge is synchronized to TCR1 referenced from its own most recent rising edge. Figure 8 and Figure 9 show how the interrupt handler synthesizes the one-shot pulse by changing the OC parameters.





TPU OC HOST R INIT TIM

**Figure 8 OC Host-Initiated Pulse Mode to Schedule Rising Edge**



TPU OC HOST F INIT TIM

**Figure 9 OC Host-Initiated Pulse Mode to Schedule Falling Edge**

### CHANNEL\_CONTROL = \$A6

The example value sets the output pin to low, selects TCR2 as the match timebase, and sets up the output pin to go high when a match occurs. The capture timebase is TCR1. Since the pin is an output, the only source of captures is when the match event occurs.

### OFFSET = 1

This value defines the number of increments of the value pointed to by REF\_ADDR1 which must elapse before a match occurs. In this example, only one increment of TCR2 is required. The value can be any integer in the range 0 to \$8000.

## REF\_ADDR1 = \$EE

This parameter is a pointer to an address in parameter RAM. The value in REF\_ADDR1 is the least significant byte of that address. The example value is the address where the TCR2 counter value is stored when each host service request is executed. Thus, the TCR2 value at the most recent host service request is added to OFFSET to form the next match value.

## SEQUENCE OF EVENTS

The first host service request initializes Channel 0 to generate a continuous PWM waveform. The CPU also sets up the OC function on Channel 1 to make a low-to-high transition after one TCR2 count (one PWM rising edge) by initializing the CHANNEL\_CONTROL parameter and setting REF\_ADDR1 to point to the stored value of TCR2. This value is updated by the TPU after the CPU issues a host service request on Channel 1. Since this channel is configured to match TCR2 and capture TCR1, it will capture the associated pin's rising edge time in TCR1 counts, and store it in the parameter RAM defined as ACTUAL\_MAT\_TIME.

After the host service request is made, interrupts are enabled on Channel 1, which causes the TPU to generate an interrupt request on each OC transition. The same CPU interrupt handler processes both low-to-high and high-to-low transitions. The handler checks which transition occurred by reading the PAC and PSC bit fields in Channel 1 CHANNEL\_CONTROL parameter.

If the pin on Channel 1 has made a low-to-high transition, the channel is reconfigured to both match and capture TCR1 and to make the pin go low on the next match. REF\_ADDR1 is changed to point to ACTUAL\_MAT\_TIME, which is the TCR1 time captured at the last match. Also, the parameter OFFSET is changed to \$200. This causes the falling transition to be \$200 TCR1 counts from the rising transition (i.e., the OC pulse width is \$200 TCR1 counts). Lastly, the CPU issues another host service request to start the function. When the pin eventually goes low, the TPU will generate another interrupt to the CPU.

When the Channel 1 pin goes low, parameter RAM is reinitialized to the values set up on the first occasion: match TCR2 and capture TCR1, pin transition from low to high upon a match, OFFSET is reset to one TCR2 count, and REF\_ADDR1 is changed back to point to the TCR2.

The cycle then repeats: the next rising edge on T2CLK input triggers the OC output pin transition, causing the CPU interrupt handler to again reconfigure the OC parameter RAM.

## EXAMPLE CODE LISTING

The following example code is written for CPU32-based microcontrollers with the standard TPU functions (those on the MC68332 or MC68332A, **not** the MC68332G). It must be modified to use with CPU16-based microcontrollers. The function numbers may vary with different mask sets.

```

TPUMCR      equ      $fffe00
TICR        equ      $fffe08
CIER        equ      $fffe0a
CISR        equ      $fffe20
CFSR3       equ      $fffe12
HSQR1       equ      $fffe16
CPR1        equ      $fffe1e
HSRR1       equ      $fffe1a
PRAM0_0     equ      $ffff00
PRAM0_1     equ      $ffff02
PRAM0_2     equ      $ffff04
PRAM0_3     equ      $ffff06
PRAM1_0     equ      $ffff10
PRAM1_1     equ      $ffff12
PRAM1_2     equ      $ffff14


                org      $4000
                move.w    #$00e9,(CFSR3).l    ;channels 0 and 1 output compare
                move.w    #$0000,(HSQR1).l    ;matches and pulses scheduled
                move.w    #$000f,(CPR1).l     ;high priority to both channels
                move.w    (CISR).l,d0         ;read and clear interrupt status register
                move.w    #$0000,(CISR).l
                move.l    #INT,($604).l       ;starting address of interrupt routine
                                           ;(assuming VBR = $400)
                ori.w     #$0005,(TPUMCR).l   ;IARB field = 5
                move.w    #$0680,(TICR).l     ;interrupt level 6, vector $80
                andi.w    #$f5ff,SR          ;mask out interrupts at level 5 and below
                                           ;assuming reset state of SR

                move.w    #$0400,(PRAM0_2).l  ;PWMHI = $400 TCR1 counts for ch 0
                move.w    #$0800,(PRAM0_3).l  ;PWMPER = $800 TCR1 counts for ch 0
                move.w    #$00a6,(PRAM1_0).l  ;ch1: psc=0,pac=high, capture
                                           ;TCR1/match TCR2
                move.w    #$0092,(PRAM0_0).l  ;use TCR1 as timebase
                move.w    #$0001,(PRAM1_1).l  ;TCR2 offset from REF_ADDR1 for ch 1
                move.w    #$00ee,(PRAM1_2).l  ;REF_ADDR1 = TCR2 for channel 1
                move.w    #$0006,(HSRR1).l    ;ch 0: init to start PWM, ch 1 will go
high
                move.w    #$0002,(CIER).l     ;enable interrupt for channel 1
mnloop        bra mnloop

INT           move.w    (CISR).l,d6           ;read and clear interrupt
                andi.w    #$fffd,(CISR).l
                move.w    (PRAM1_0).l,d0
                cmpi.b    #$ae,d0            ;see whether ch1 should go low or high
                bne      init_high

init_low      move.w    #$0089,(PRAM1_0).l    ;ch1: capture TCR1/match
                                           ;TCR1, psc=high, pac=low
                move.w    #$200,(PRAM1_1).l   ;TCR1 offset from REF_ADDR1 for ch 1
                move.w    #$001a,(PRAM1_2).l  ;REF_ADDR1 = ACTUAL_MAT_TIME for
                                           ;channel 1
                move.w    #$0004,(HSRR1).l    ;channel 1 will go low
bra EN
init_high     move.w    #$00a6,(PRAM1_0).l    ;ch1: psc=0, pac=high, capture
                                           ;TCR1/match TCR2
                move.w    #$0001,(PRAM1_1).l  ;TCR2 offset from REF_ADDR1 for ch 1
                move.w    #$00ee,(PRAM1_2).l  ;REF_ADDR1 = TCR2 for channel 1
                move.w    #$0004,(HSRR1).l    ;channel 1 will go high
EN            RTE

```

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.  **MOTOROLA** is a registered trademark of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

**TO OBTAIN ADDITIONAL PRODUCT INFORMATION:**

**USA/EUROPE:** Motorola Literature Distribution;  
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447

**JAPAN:** Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki,  
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315

**HONG KONG:** Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,  
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

**MFAX:** RMFAX0@email.sps.mot.com - TOUCHTONE (602) 244-6609

**INTERNET:** <http://www.mot.com>



**MOTOROLA**