SEMICONDUCTOR
PROGRAMMING NOTE

Pulse Width Modulation TPU Function (PWM)

By Kevin Anderson

1 Functional Overview

This output function generates a pulse-width-modulated waveform in which the period and/or the high time can be changed at any time by the CPU. PWM uses two modes of operation: level and normal. In level mode, a 0% or a 100% duty-cycle waveform can be generated. In normal mode, waveforms with duty-cycles between 0% and 100% can be generated.

In general, any changed period or high time is used in subsequent waveform synthesis, after a low-to-high transition. An immediate update is possible in either mode. After an immediate update, the new period and/or high time is reflected in the output waveform during the immediate host-service state, instead of waiting for a subsequent low-to-high transition.

2 Detailed Description

To start a PWM waveform, the CPU configures or updates parameters PWMPER (period desired) and PWMHI (high time desired), then issues an HSR %10 for initialization. After CPU initialization (refer to **Figure 1**), the TPU generates a low-to-high transition and calculates the pulse timing (next fall time, next rise time). The time of the most recent rising edge is moved from parameter PWMRIS to parameter OLDRIS, where it can be read at any time by the CPU. Calculation of the fall time is made by adding OLDRIS to PWMHI. The next rise time is calculated by adding the period desired from PWMPER to the rise time, now in OLDRIS, and then placing the projected new rise time into PWMRIS.
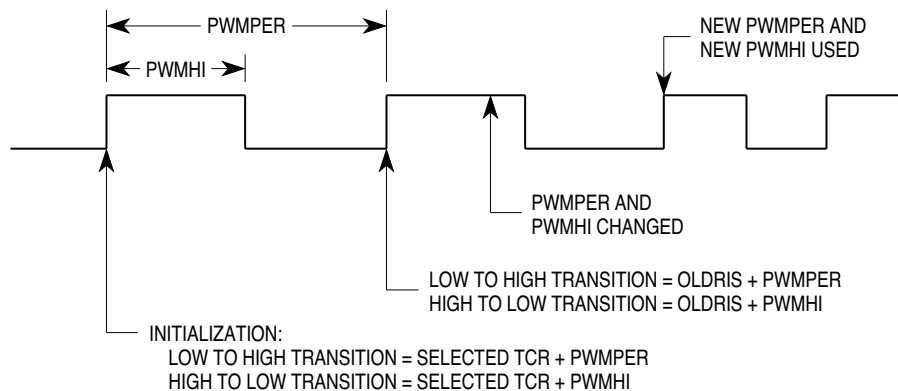

Figure 1 50% Duty Cycle PWM Waveform

In level mode, where the high time in PWMHI is zero (indicating 0% duty cycle) or is equal to or greater than the period (indicating 100% duty cycle), a match without a pin transition is set up for the time (OLDRIS + PWMPER). In normal mode, a match and fall time is set up for the time (OLDRIS + PWMHI), and an interrupt request signal is asserted on each match event if the interrupt enable bit is set. To change the PWM parameters, the CPU coherently writes new 16-bit values to either PWMPER or PWMH. If both PWMPER and PWMH are to be changed, a coherent 32-bit write is required.

MOTOROLA

In both normal and level modes the new parameters are referenced to the next low-to-high transition. An immediate update of either or both parameters may be selected by the CPU by issuing an HSR %01. The immediate result to the waveform depends upon the point at which the immediate update is taken (See **7 Performance and Use of Function**).

A optional CPU interrupt request can be made at the beginning of each pulse in any mode or after an immediate update. This allows the CPU to schedule parameter changes in relationship to a known point in the waveform.

3 Function Code Size

Total TPU function code size determines what combination of functions can fit into a given ROM or emulation memory microcode space. PWM function code size is:

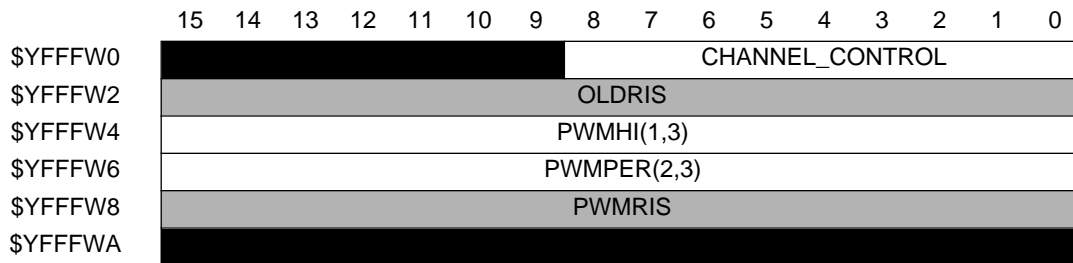32 μ instructions + 8 entries = **40 long words**

4 Function Parameters

This section provides detailed descriptions of function parameters stored in channel parameter RAM. **Figure 2** shows TPU parameter RAM address mapping. **Figure 3** shows the parameter RAM assignment used by the function. In the diagrams, Y = M111, where M is the value of the module mapping bit (MM) in the system integration module configuration register (Y = $7 or $F).

Channel	Base	Parameter Address							
Number	Address	0	1	2	3	4	5	6	7
0	$YFFF##	00	02	04	06	08	0A	—	—
1	$YFFF##	10	12	14	16	18	1A	—	—
2	$YFFF##	20	22	24	26	28	2A	—	—
3	$YFFF##	30	32	34	36	38	3A	—	—
4	$YFFF##	40	42	44	46	48	4A	—	—
5	$YFFF##	50	52	54	56	58	5A	—	—
6	$YFFF##	60	62	64	66	68	6A	—	—
7	$YFFF##	70	72	74	76	78	7A	—	—
8	$YFFF##	80	82	84	86	88	8A	—	—
9	$YFFF##	90	92	94	96	98	9A	—	—
10	$YFFF##	A0	A2	A4	A6	A8	AA	—	—
11	$YFFF##	B0	B2	B4	B6	B8	BA	—	—
12	$YFFF##	C0	C2	C4	C6	C8	CA	—	—
13	$YFFF##	D0	D2	D4	D6	D8	DA	—	—
14	$YFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	$YFFF##	F0	F2	F4	F6	F8	FA	FC	FE

— = Not Implemented (reads as $00)

Figure 2 TPU Channel Parameter RAM CPU Address Map

Figure 3 shows all of the host interface areas for the PWM function, as well as the parameters, addresses, reference times, and reference sources. This segment lists and defines the parameters for all modes of the PWM time function.

|	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1 0
$YFFFW0	▉▉▉▉▉▉▉	CHANNEL_CONTROL
$YFFFW2	OLDRIS	
$YFFFW4	PWMHI(1,3)	
$YFFFW6	PWMPER(2,3)	
$YFFFW8	PWMRIS	
$YFFFWA	▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉	

Y= Channel number
Parameter Write Access:

(white)	Written by CPU
(light gray)	Written by TPU
(gray)	Written by CPU and TPU
(black)	Unused parameters

Figure 3 Function Parameter RAM Assignment

4.1 CHANNEL_CONTROL

CHANNEL_CONTROL contains the channel latch controls and configures the PSC, PAC, and TBS fields. The PSC field forces the output level of the pin directly without affecting the PAC latches, or forces the output level to the state specified by the PAC latches. The PAC field specifies the pin logic response as either a timer channel input or output. The TBS field configures a channel pin as input or output and configures the time base for output match/input capture events.

15 14 13 12 11 10 9	8 7 6 5	4 3 2	1 0
NOT USED	TBS	PAC	PSC
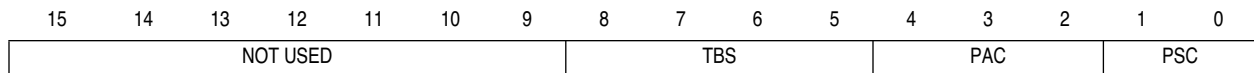
NOTE

This channel must be configured as an output because the PWM function is indeterminate when programmed as an input.

Table 1 CHANNEL_CONTROL Options

TBS	PAC	PSC	Action	
8 7 6 5	**4 3 2**	**1 0**	**Input**	**Output**
		0 0	—	Force Pin as Specified by PAC
		0 1	—	Latches
		1 0	—	Force Pin High
		1 1	—	Force Pin Low
				Do Not Force Any State
	1 x x		Do Not Change PAC	Do Not Change PAC
0 1 x x			—	Output Channel
0 1 0 0			—	Capture TCR1, Compare TCR1
0 1 1 1			—	Capture TCR2, Compare TCR2
1 1 x x			Do Not Change TBS	Do Not Change TBS

The PSC field determines the setting of the pin after initialization. In normal mode, PSC is set to force the pin high. In level mode, where a 0% duty cycle is desired, PSC should be set to force the pin low at initialization. The PAC field specifies the pin logic response as a timer channel output; however, the PWM function does not use the PAC field, but uses direct control by the microcode. CHANNEL_CONTROL must be written by the CPU before initialization.

4.2 OLDRIS

OLDRIS is the time of the previous low-to-high transition. When executing state *Init*, the TPU sets OLD-RIS to the value of either TCR1 or TCR2 as specified in CHANNEL_CONTROL. When PWM is executing in normal mode (PWMPER > PWMHI), the TPU updates OLDRIS at the beginning of each pulse to the time of the last low-to-high transition.

4.3 PWMRIS

PWMRIS is the current calculated rise time calculated at the beginning of the pulse (on the low-to-high transition) by adding OLDRIS to PWMPER. The TPU updates this parameter.

4.4 PWMHI

PWMHI, which is updated by the CPU, is the current pulse high time that may be updated at any time. Estimate for best-case minimum value for PWMHI is greater than 32 system clocks, assuming a single channel operating. When more than one channel is operating, the minimum value for PWMHI depends on TPU configuration (the variables are described in **7 Performance and Use of Function**). The maximum value is $8000. The user should calculate case timing to ensure proper execution of this function.

4.5 PWMPER

PWMPER, which is updated by the CPU, is the current PWM period and is used by the TPU to calculate the next low-to-high transition time. Estimate for best-case minimum value for PWMPER is greater than 32 system clocks, assuming that a single channel is operating. When more than one channel is operating, the minimum value for PWMPER depends on TPU configuration (the variables are described in **7 Performance and Use of Function**). The maximum usable value is that which satisfies the condition: (PWMPER – PWMHI) is less than or equal to $8000. PWMHI and PWMPER must be accessed coherently. The user should calculate the case timing to ensure proper execution of this function. Normal, 100%, and 0% duty cycles are defined as follows.

0%	$\rightarrow$	PWMHI = 0
100%	$\rightarrow$	PWMPER $\leq$ PWMHI, AND PWMHI $\neq$ 0
Else normal	$\rightarrow$	PWMPER > PWMHI, AND PWMHI $\neq$ 0

5 Host Interface to Function

This section provides information concerning the TPU host interface to the function. **Figure 4** is a TPU address map. Detailed TPU register diagrams follow the figure. In the diagrams, Y = M111, where M is the value of the module mapping bit (MM) in the system integration module configuration register (Y = $7 or $F).

Address	15 8	7 0
$YFFE00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
$YFFE02	TEST CONFIGURATION REGISTER (TCR)	
$YFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
$YFFE06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
$YFFE08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
$YFFE0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
$YFFE0C	CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)	
$YFFE0E	CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)	
$YFFE10	CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)	
$YFFE12	CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)	
$YFFE14	HOST SEQUENCE REGISTER 0 (HSQR0)	
$YFFE16	HOST SEQUENCE REGISTER 1 (HSQR1)	
$YFFE18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
$YFFE1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
$YFFE1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
$YFFE1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
$YFFE20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
$YFFE22	LINK REGISTER (LR)	
$YFFE24	SERVICE GRANT LATCH REGISTER (SGLR)	
$YFFE26	DECODED CHANNEL NUMBER REGISTER (DCNR)	

Figure 4 TPU Address Map

CIER — Channel Interrupt Enable Register   **$YFFE0A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0

CH	Interrupt Enable
0	Channel interrupts disabled
1	Channel interrupts enabled

CFSR[0:3] — Channel Function Select Registers   **$YFFE0C – $YFFE12**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFS (CH 15, 11, 7, 3)				CFS (CH 14, 10, 6, 2)				CFS (CH 13, 9, 5, 1)				CFS (CH 12, 8, 4, 0)			

CFS[4:0] — PWM Function Number (Assigned during microcode assembly)

HSQR[0:1] — Host Sequence Registers   **$YFFE14 – $YFFE16**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15, 7		CH 14, 6		CH 13, 5		CH 12, 4		CH 11, 3		CH 10, 2		CH 9, 1		CH 8, 0	

CH[15:0]	Action Taken
xx	**Not used in this function.**

HSRR[0:1] — Host Service Request Registers   **$YFFE18 – $YFFE1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15, 7		CH 14, 6		CH 13, 5		CH 12, 4		CH 11, 3		CH 10, 2		CH 9, 1		CH 8, 0	

CH[15:0]	Initialization
00	No Host Service (Reset Condition)
01	Immediate Update
10	Initialization
11	Undefined

CPR[1:0] — Channel Priority Registers   **$YFFE1C – $YFFE1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15, 7		CH 14, 6		CH 13, 5		CH 12, 4		CH 11, 3		CH 10, 2		CH 9, 1		CH 8, 0	

CH[15:0]	Channel Priority
00	Disabled
01	Low
10	Middle
11	High

CISR — Channel Interrupt Status Register   **$YFFE20**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0

CH	Interrupt Status
0	Channel interrupt not asserted
1	Channel interrupt asserted

6 Function Configuration

The CPU initializes this time function by the following:

1. Writing CHANNEL_CONTROL, PWMHI, and PWMPER to RAM;
2. Issuing an HSR %10 for initialization; and
3. Enabling channel servicing by assigning a high, middle, or low priority.

The TPU then executes initialization and asserts an interrupt if the interrupt enable bit is set. In the beginning of each period, new pulse parameters are calculated and interrupts are attempted. The CPU should monitor the HSR register (or the channel interrupt) until the TPU clears the service request to 00 before changing any parameters or issuing a new service request to this channel.

In normal mode (PWMPER > PWMHI), the TPU stores the time of the last low-to-high transition in OLD-RIS, which can be read by the CPU. The TPU calculates the pulse timing (next fall time, next rise time) in the beginning of the pulse, after generating a low-to-high transition. An interrupt is then asserted if the interrupt enable bit is set. To change the PWM parameters, the CPU writes PWMPER and/or PWMHI to the parameter RAM. The two parameters must be written as 16-bit values.

Generally, the new parameters are referenced on the next low-to-high transition. For an immediate update of the waveform, the CPU issues HSR %01, which can be issued whenever any previous service request has been serviced, indicated by the HSR bits of the channel at 00. When immediate update is executed, the TPU asserts an interrupt if the interrupt enable bit is set. After issuing either HSR %01 or HSR %10, the CPU should wait for the HSR bits to be cleared by the TPU before changing any parameters or issuing another service request to the channel.

Table 2 Host Service Request Bit Encoding

CH[1:0]	Action Taken
00	No Host Service Request
01	Immediate Update
10	Initialization
11	Undefined

7 Performance and Use of Function

7.1 Performance

Like all TPU functions, PWM function performance in an application is to some extent dependent upon the service time (latency) of other active TPU channels. This is due to the operational nature of the scheduler. When a single PWM channel is in use and no other TPU channels are active, the minimum time between any two pulse edges is greater than 32 CPU clocks. When more TPU channels are active, performance decreases. However, worst-case latency in any TPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the TPU, use the guidelines given in the TPU reference manual and the information in the PWM state timing table below.

Table 3 Pulse Width Modulation Function — State Timing

State Number & Name	Max. CPU Clock Cycles	RAM Accesses by TPU
S1 *Init*	32	4
S2 *Normal_L_H*	24	4
S3 *Normal_H_L*	2	1
S4 *Normal_0*	24	4
S5 *Immed_H*	28	3
S6 *Immed_L*	28	3

7.2 Changing Duty Cycle

The CPU can change the duty cycle at any time once the TPU has completed the initialization state (indicated by HSR %00 or a CPU interrupt request). Changes are made by writing a new high time value to PWMHI in the channel's parameter RAM.

The minimum duty cycle (and the maximum non-100% duty cycle) is dependent on the number of active TPU channels and the maximum channel latency as discussed above. A 0% duty cycle is generated by setting PWMHI = 0. A 100% duty cycle is scheduled by setting PWMPER less than or equal to PWMHI when PWMHI is not equal to zero.

Duty cycle changes take effect at the completion of the current period unless an immediate update (HSR %01) is also requested. Immediate updates may be requested for any duty cycle including 0% and 100%. A new PWMHI value with an immediate update HSR causes the TPU to change the currently scheduled high-to-low time. This can cause the undesired side effect of an improper duty cycle for one period. **Figure 5** is an example of such a case.

In **Figure 5** the newly requested duty cycle is shorter than the current one. When the immediate HSR is serviced, the TPU schedules a new high-to-low transition time. However, this new value is less than the current TCR value, so that the TPU greater-than-or-equal-to comparator fires, generating an immediate high-to-low transition. At the end of the period, the new high time is again used to calculate the next falling edge, with reference to the latest rise time — from this point, the duty cycle is correct.

PWMHI CHANGED WITH
IMMEDIATE UPDATE

TPU PWM C UP W/ TIM
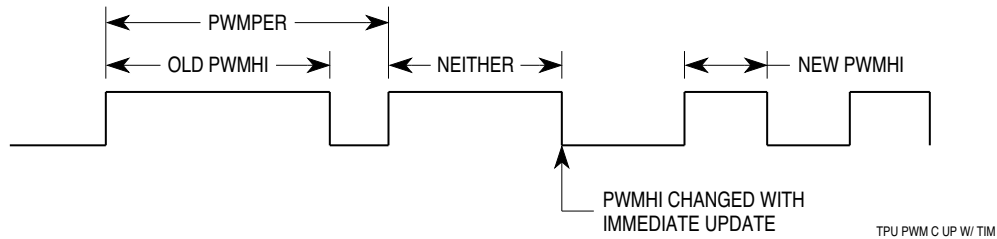
Figure 5 Immediate Duty Cycle Update With A One Period Anomaly

If the update in the above example happens at a point in the period that is before the newly specified fall time, then the new duty cycle occurs as planned in the current cycle, with no intermediate glitches. This is shown in **Figure 6**. If the update occurs at a time after the falling edge of the current pulse the new duty cycle takes effect in the next period. This is shown in **Figure 7**.

PWMHI CHANGED WITH
IMMEDIATE UPDATE

TPU PWM C UP W/O TIM
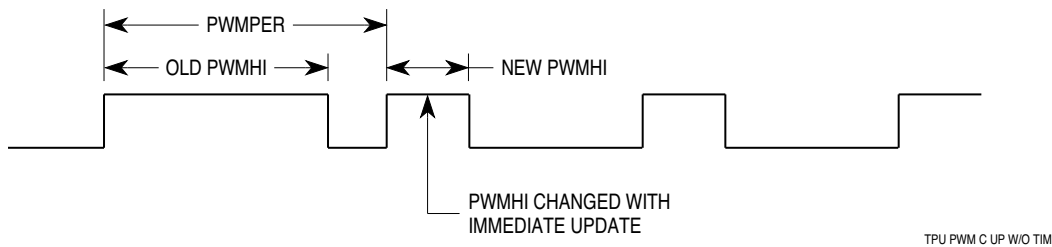
Figure 6 Immediate Duty Cycle Update Without Anomaly

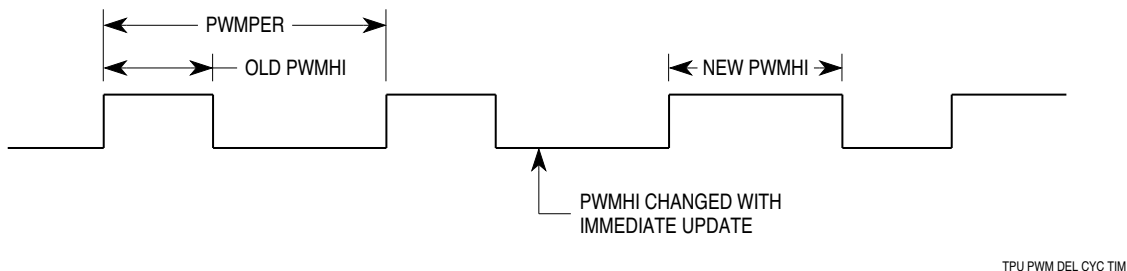PWMHI CHANGED WITH
IMMEDIATE UPDATE

TPU PWM DEL CYC TIM

Figure 7 Immediate Duty Cycle Update Delayed One Period

Many applications are intolerant of the duty cycle glitch described above. For that reason the normal update mode is preferred for most applications.

7.3 Changing Period

Once the TPU has completed the initialization state the CPU may at any time specify a new period by writing to the PWMPER parameter. Unless the CPU also generates an immediate update service request the new period takes effect at the beginning of the next period, as shown in **Figure 1**. That is, a new rise time is calculated at the next low-to-high transition. Thus, the current period is allowed to complete before the new one begins.

If an immediate update is requested in conjunction with a new period, the TPU immediately calculates a new rise time that is applied during the current period. If the new period is longer than the old period the new period takes effect immediately. If the new period is shorter than the old, the current period may actually be shorter than the old period but longer than the new period. An example of this is shown in **Figure 8**.

OLD PWMPER   NEITHER   NEW PWMPER

OLD PWMHI   NEW PWMHI

PWMPER AND PWMHI CHANGED
WITH IMMEDIATE UPDATE

TPU PWM P UP W/ TIM
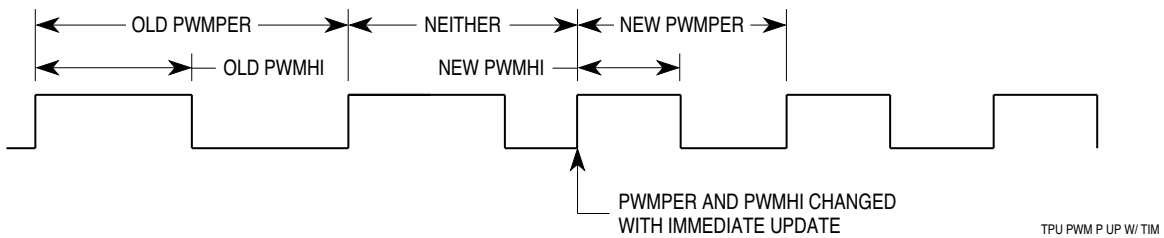
Figure 8 Immediate Period Update With Single Period Anomaly

In this example a new PWMPER and PWMHI time have been requested at the time indicated. Since the current pulse high time has expired, a shorter high time has no impact on this cycle. However, the newly calculated low-to-high time (OLDRIS + PWMPER) is now less than the current time and the TPU greater-than-or-equal-to comparator immediately generates the rising edge. At this point new high-to-low and low-to-high times are calculated and the new correct period and high time are in effect.

If the update had occurred earlier in the period the result would have been different. This is illustrated in **Figure 9**. Here, the update occurs such that both the newly scheduled rising and falling edges are in the future. Thus both will occur in the proper places and the one cycle anomaly is eliminated.
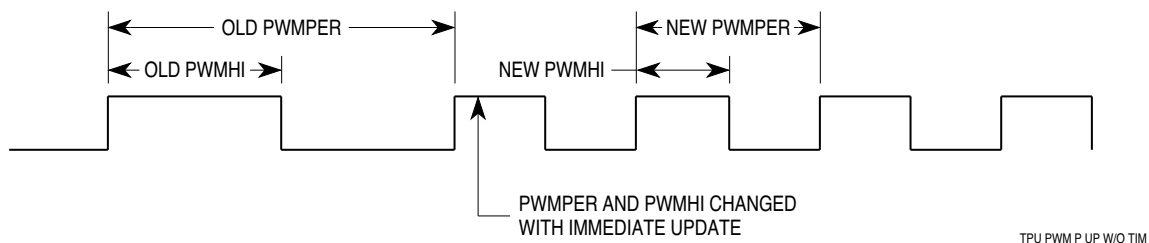

OLD PWMPER   NEW PWMPER

OLD PWMHI   NEW PWMHI

PWMPER AND PWMHI CHANGED
WITH IMMEDIATE UPDATE

TPU PWM P UP W/O TIM

Figure 9 Immediate Period Update With No Anomaly

Remember that updates made without using the immediate update feature always take effect on the next rising edge and no anomalous behavior occurs. If an application requires both the period and high time to be updated coherently, it is best to enable the CPU interrupt before the update. During the interrupt service the new period and high time can be updated. The interrupt occurs at the beginning of the period, so as long as the interrupt service finishes before the end of the period, the new period and high time take effect during the same cycle. Without this time reference, the parameter updates could straddle the end of a period and one would take effect a cycle ahead of the other. This could cause the single cycle anomalies discussed earlier.

7.4 Counting Periods

The TPU generates a CPU interrupt service request during the channel service at the beginning of each period. The CPU can respond to these requests to keep track of how many periods have elapsed. In this way, new pulse widths can be scheduled at a known position in time.

7.5 Stopping the Function

Once PWM operation is initialized on a channel, it runs without CPU intervention until a reset occurs. If it is necessary to turn off a PWM channel, the CPU can write zeros to the channel function select bits in registers CFSR[0:3]. This disables the function on the channel. Another way to disable output is to select 0% or 100% duty cycle in the channel parameter RAM. In this case the PWM continues to run and receive channel service but no transitions are seen on the pin.

8 PWM Examples

The following examples give an indication of the capabilities of the PWM function. Each example includes a description of the example, a diagram of the initial parameter RAM content, initial control bit settings, and a diagram of the output waveform.

8.1 Example A

8.1.1 Description

Generate a 50% duty cycle waveform with a period equal to $800 TCR1 clocks on channel 1.

8.1.2 Initialization

Disable channel 1 by clearing the priority bits (CPR1[3:2]). Select PWM function by programming the function select register for channel 1 (CFSR3[7:4]). Configure the parameter RAM for channel 1 as shown below. Write HSRR1[3:2] = %10 to initialize the channel on the first channel service. Write the priority bits (CPR1[3:2] to high, medium, or low priority to begin channel service.

Table 4 PWM Channel Parameter RAM

	15								8							0	
$YFFF10	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	CH_CNTL
$YFFF12	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	PWMHI
$YFFF16	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	PWMPER
$YFFF18	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF1A	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

8.1.3 Output Waveforms
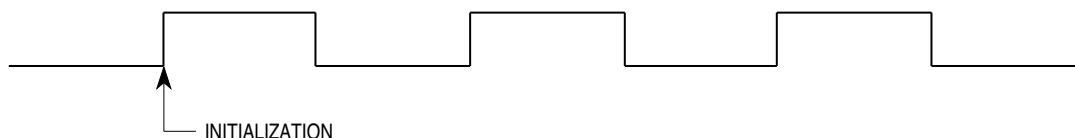
INITIALIZATION

TPU PWM EXA TIM

8.2 Example B

8.2.1 Description

The waveform in Example A is running on Channel 1. Change the duty cycle to 25% using normal update mode.
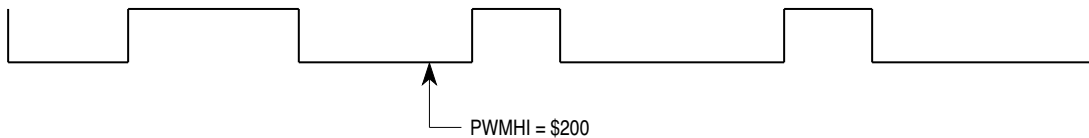
8.2.2 Initialization

Change the value in PWMHI as shown in below. The new duty cycle becomes effective in the period following the update.

Table 5 PWM Channel Parameter RAM

	15						8								0	
$YFFF10	UNCHANGED															CH_CNTL
$YFFF12	UNCHANGED															OLDRIS
$YFFF14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	PWMHI
$YFFF16	UNCHANGED															PWMPER
$YFFF18	UNCHANGED															PWMRIS
$YFFF1A	UNCHANGED															

8.2.3 Output Waveforms

PWMHI = $200

TPU PWM EXB TIM

8.3 Example C

8.3.1 Description

Change the waveform in Example B to 100% duty cycle. Use the immediate update mode and a CPU interrupt so that the update takes effect on the cycle that generates the interrupt (assumes that the interrupt latency and service time is less than the current duty cycle).

8.3.2 Initialization

Enable the channel to generate a CPU interrupt (CIER[1] = 1). During interrupt service set PWMHI = PWMPER as shown below and signal an immediate update (HSRR1[3:2] = 01).

Table 6 PWM Channel Parameter RAM

	15						8								0	
$YFFF10	UNCHANGED															
$YFFF12	UNCHANGED															
$YFFF14	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	PWMHI
$YFFF16	UNCHANGED															
$YFFF18	UNCHANGED															
$YFFF1A	UNCHANGED															

8.3.3 Output Waveforms

CIER[1] = 1   PWMHI = PWMPER

IRQ

TPU PWM EXC TIM
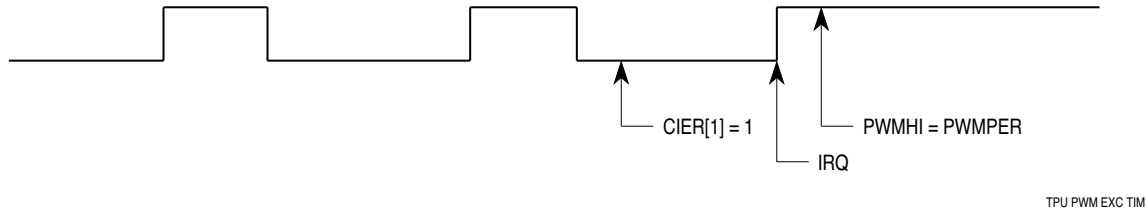
9 Function Algorithm

The PWM time function consists of the following six states.

9.1 STATE 1 — *Init*

This state is entered as a result of HSR %10, which initializes the pulse parameters and channel latches and generates an interrupt when *Init* is completed. Start time of the pulse is set to the current TCR time. The 100% or 0% duty-cycle pulse relationships are checked and processed; flag0 is set to indicate this condition. The PSC field determines the setting of the pin after initialization. In normal mode, PSC is set to force the pin high; in level mode where a 0% duty cycle is desired, PSC should be set to force the pin low at initialization.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 10xxxx
Match Enable: Disable

        Configure channel latches via CHANNEL_CONTROL
        Store current TCR (indicated in CHANNEL_CONTROL) in OLDRIS
        Clear flag0
        Set PAC to high to low
        Calculate and store next rise time
            PWMRIS = OLDRIS + PWMPER
        If PWMHI = 0 (0% duty cycle) then {
            Assert flag0   /* level mode */
            Set pin low
            Set PAC to don't change on match
            Generate a match on PWMRIS
        }
        If PWMHI $\geq$ PWMPER (100% duty cycle) then {
            Assert flag0
            Set pin high
            Set PAC to don't change on match
            Generate a match on PWMRIS
        If (PWMHI > 0) and (PWMHI < PWMPER) then {   /* normal mode */
            Generate a match (fall time) = OLDRIS + PWMHI
        }
        Assert interrupt request

9.2 STATE 2 — *Normal_L_H*

In this state the TPU sets the fall time of the pulse and calculates the new rise time. This state is entered as a result of one of the following events:

1. When in normal mode (0% < duty cycle < 100%, indicated by flag0 equals zero) after a match occurs and a low-to-high transition results;
2. When in level mode (100% duty cycle, indicated by flag0 equals one) after a match occurs and the pin is high. The 0% or 100% duty-cycle condition is checked and processed. The parameters are rechecked at the rate of PWMPER to determine if they have been updated from the case of 100% duty cycle.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001010
         HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001011
Match Enable: Don't Care

      Store transition time into OLDRIS
      Calculate and store next rise time
          PWMRIS = OLDRIS + PWMPER
      Assert flag0
      Set PAC to high to low
      If PWMHI = 0 (0% duty cycle) then {
          Assert flag0   /* level mode */
          Set pin low
          Set PAC to don't change on match
          Generate a match on PWMRIS
      }
      If PWMHI ≥ PWMPER (100% duty cycle) then {
          Assert flag0   /* level mode */
          Set pin high
          Set PAC to don't change on match
          Generate a match on PWMRIS
      }
      If (PWMHI > 0) and (PWMHI < PWMPER) then {
          Generate a match (fall time) on OLDRIS + PWMHI
      }
      Assert interrupt request

9.3 STATE 3 — *Normal_H_L*

This state is entered after a match occurs and a high-to-low transition results. In this state, the TPU sets the rise time of the pulse.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001000
Match Enable: Don't Care

      Set PAC to low to high
      Generate a match on PWMRIS

9.4 STATE 4 — *Normal_0*

This state is entered in level mode (indicated by flag0 equals one) when the pin is low, after a match event occurs. A match on next period time is set up. The 0% or 100% duty cycle condition is checked and processed and an interrupt is generated. The parameters are rechecked at the rate of PWMPER to determine if they have been updated from the case of 0% duty cycle. If normal mode is to be resumed, a low-to-high transition is projected for the current match time plus PWMPER.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001001
Match Enable: Don't Care

      Calculate and store next rise time
          OLDRIS = ERT
          PWMRIS = OLDRIS + PWMPER
      Set PAC to low to high
      Clear flag0
      If PWMHI = 0 (0% duty cycle) then {
          Assert flag0   /* level mode */
          Set pin low
          Set PAC to don't change on match
          Generate a match on PWMRIS

```
}
If PWMHI ≥ PWMPER (100% duty cycle) then {
    Assert flag0
    Set pin high
    Set PAC to don't change on match
    Generate a match on PWMRIS
}
If (PWMHI > 0) and (PWMHI < PWMPER) then {
    Generate a match on next rise time = OLDRIS + PWMPER
}
Assert interrupt request (period time)
```

9.5 STATE 5 — *Immed_H*

This state is entered as a result of HSR%01 when the pin is asserted. This state causes an immediate update of the high time of the pulse starting from OLDRIS. The case of 0% or 100% duty cycle pulse is checked and processed, and an interrupt is generated.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 01xx1x
Match Enable: Disable

```
Calculate and store next rise time
    PWMRIS = OLDRIS + PWMPER
Clear flag0
Set PAC to high to low
If PWMHI = 0 (0% duty cycle) then {
    Assert flag0 (level mode)
    Set pin low
    Set PAC to don't change on match
    Generate a match on PWMRIS
}
If PWMHI ≥ PWMPER (100% duty cycle) then {
    Assert flag0 (level mode)
    Set pin high
    Set PAC to don't change on match
    Generate a match on PWMRIS
}
If (PWMHI > 0) and (PWMHI > PWMPER) then {
    Generate a match on next rise time = OLDRIS + PWMHI
}
Assert interrupt request (period time)
```

9.6 STATE 6 — *Immed_L*

This state is entered as a result of HSR %01 when the pin is low. This state causes an immediate update of the low time of the pulse starting from OLDRIS. The case of 0% or 100% duty cycle is checked and processed and an interrupt is generated.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 01xx0x
Match Enable: Disable

```
Calculate and store next rise time
    PWMRIS = OLDRIS + PWMPER
Set PAC to low to high
Clear flag0
If PWMHI = 0 (0% duty cycle) then {
    Assert flag0
    Set pin low
```

Set PAC to don't change on match
        Generate a match on PWMRIS
    }
    If PWMHI $\geq$ PWMPER (100% duty cycle) then {
        Assert flag0
        Set pin high
        Set PAC to don't change on match
        Generate a match on PWMRIS
    }
    If (PWMHI > 0) and (PWMHI < PWMPER) then {
        Generate a match on next rise time = OLDRIS + PWMPER
    }
    Assert interrupt request (period time)

The table below shows the PWM transitions listing the service request sources and channel conditions from current state to next state. **Figure 10** illustrates the flow of PWM states, including the initialization and immediate update states.

Table 7 PWM State Transition Table

Current State	HSR	M/TSR	LSR	Pin	Flag0	Next State
All States	10	—	—	—	—	S1 *Init*
	01	—	—	1	—	*S5 Immed_H*
	01	—	—	0	—	S6 *Immed_L*
S1 *Init*	00	1	—	0	0	S3 *Normal_H_L*
	00	1	—	1	1	*S2 Normal_L_H*
	00	1	—	0	1	S4 *Normal_0*
S2 *Normal_L_H*	00	1	—	0	0	S3 *Normal_H_L*
	00	1	—	1	1	*S2 Normal_L_H*
	00	1	—	0	1	S4 *Normal_0*
S3 *Normal_H_L*	00	1	—	1	0	S2 *Normal_L_H*
S4 *Normal_0*	00	1	—	1	—	S2 *Normal_L_H*
	00	1	—	0	1	*S4 Normal_0*
S5 *Immed_H*	00	1	—	0	0	S3 *Normal_H_L*
	00	1	—	1	1	*S2 Normal_L_H*
	00	1	—	0	1	S4 *Normal_0*
S6 *Immed_L*	00	1	—	1	—	S2 *Normal_L_H*
	00	1	—	0	1	*S4 Normal_0*
Unimplemented	11	—	—	—	—	—
Conditions	00	0	1	—	—	—

NOTES:

1. Conditions not specified are "don't care."

2. HSR = Host service request
   LSR = Link service request
   M/TSR = Either a match or transition (input capture) service request occurred (m/tsr = 1) or neither occurred (m/tsr = 0).

KEY:

HSR	M/TSR	LSR	PIN	FLAG0	FLAG1
XX	X	X	X	X	X

HSR = 10

S1
INIT
01XXXX

M/T = 1
PIN = 0
FLAG0 = 0

M/T = 1
PIN = 0
FLAG0 = 1

M/T = 1
PIN = 1
FLAG0 = 1

S3
NORMAL_H_L
001000

M/T = 1
PIN = 1
FLAG0 = 0

M/T = 1
PIN = 0
FLAG0 = 0

S2
NORMAL_L_H
00101X

M/T = 1
PIN = 1
FLAG0 = 1

M/T = 1
PIN = 1

M/T = 1
PIN = 0
FLAG0 = 1

M/T = 1
PIN = 0
FLAG0 = 1

S4
NORMAL_0
001001

M/T = 1
PIN = 0
FLAG0 = 0

M/T = 1
PIN = 0
FLAG0 = 1

M/T = 1
PIN = 1
FLAG0 = 1

M/T = 1
PIN = 0
FLAG0 = 1

M/T = 1
PIN = 1

S5
IMMED_H
01XX1X

S6
IMMED_L
01XX0X

HSR = 01
PIN = 1

HSR = 01
PIN = 0
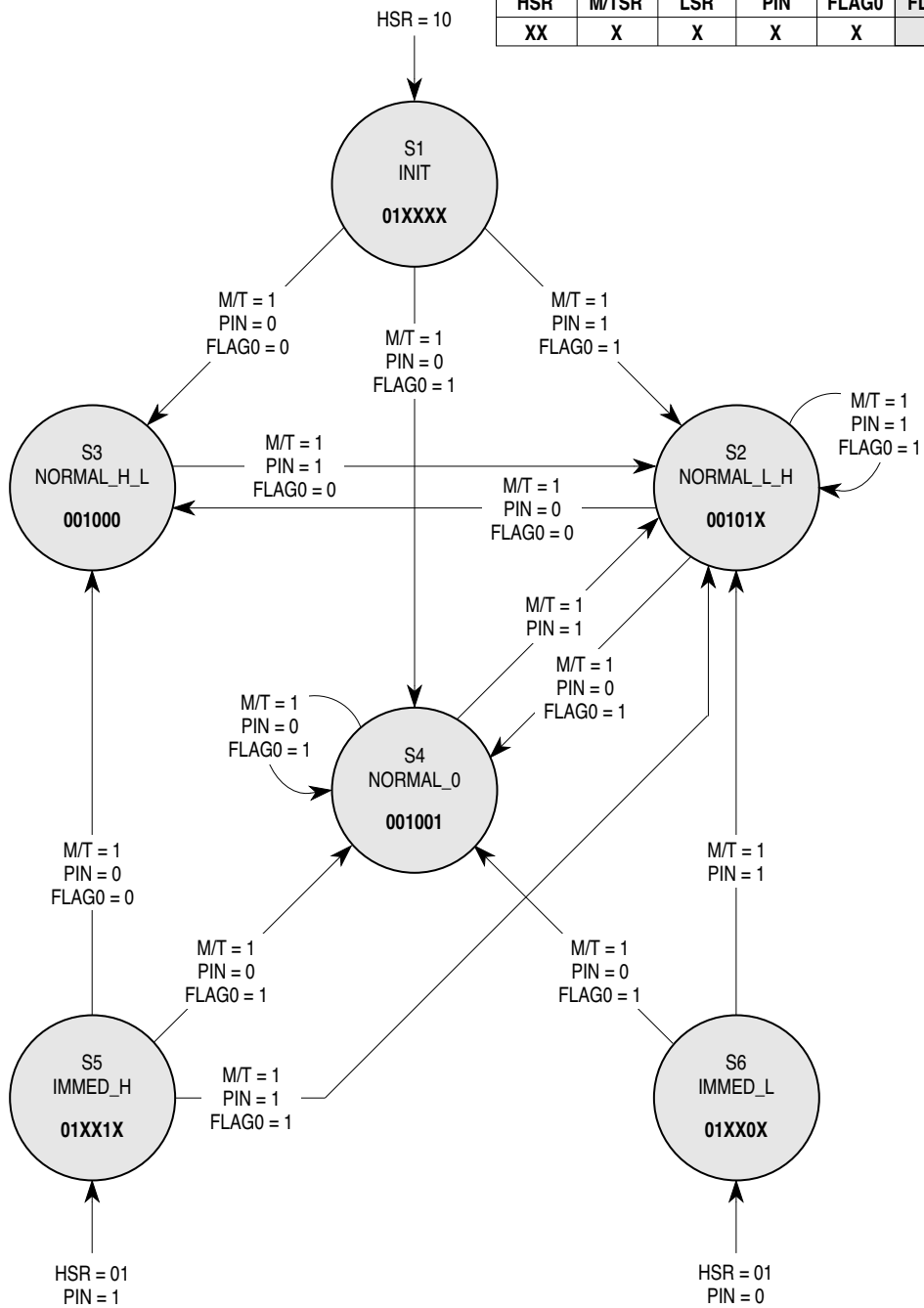
1028A

Figure 10 PWM State Diagram

NOTES

NOTES

MOTOROLA