Freescale Semiconductor

Order this document
by TPUPN05/D

APPLICATION NOTE

Multichannel PWM TPU Function (MCPWM)

by Jeff Wright

1 Functional Overview

This TPU output function uses externally-gated multiple channels to produce sophisticated pulse-width modulated (PWM) signals. These signals can be used for a variety of applications including motor control. The function allows a user to select edge-aligned or center-aligned timing relationships between multiple PWM wave forms. Center-aligned relationships include dead time and inversion options to support driving H-bridges and inverters. MCPWM can also generate a programmable periodic CPU interrupt request for high time updating.

2 Detailed Description

Standard TPU PWM functions use only one channel to produce a PWM output, but the minimum and maximum pulse widths (other than 0% or 100%) that can be obtained are limited by the latencies of TPU functions running on other channels. To produce a very short pulse, the same channel must be serviced quickly twice in succession, and this may not be possible when there is activity on other channels. Although this restriction does not present a problem in many applications, there are others, including motor control, where high resolution control of the high time is required over the full 0 to 100% duty cycle range. In motor control applications, it is also desirable to have defined timing relationships between multiple PWM signals in order to reduce ripple currents and to prevent shoot-through currents on the phase drivers.

The multichannel PWM function uses two channels that are externally gated (using single EOR gates) together to form a single PWM signal. This allows a full 0% to 100% duty cycle range under a much wider range of TPU operating conditions than other PWM functions.

The MCPWM function can operate in edge-aligned (EA) mode or in center-aligned (CA) mode.

During edge-aligned operation, one TPU channel operates as a master signal and a number (n, in the range 1 to 15) of slave channels are externally gated with the master signal to generate ‘n’ PWM outputs. External EOR gates are used. **Figure 1** is an EA mode schematic.

In EA mode, the PWM outputs have aligned rising edges. This mode is more channel-efficient than CA mode, and is the best choice for applications that require general-purpose high-speed, high-resolution pulse-width modulation. **Figure 2** shows EA mode output waveforms.

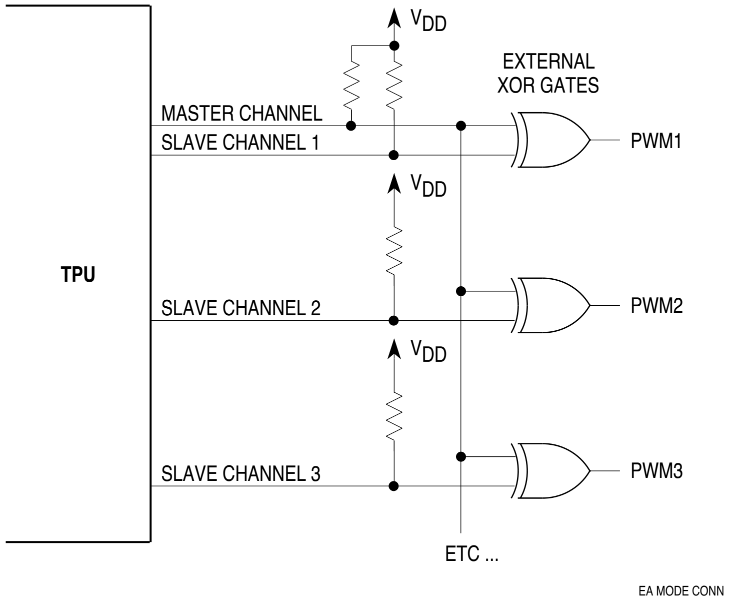


Figure 1 EA Mode External Gate Schematic

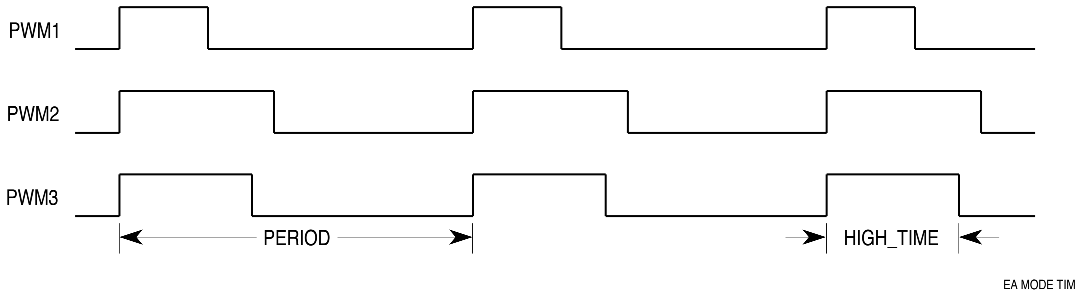


Figure 2 EA Mode Output Waveforms

During center-aligned operation, one TPU channel operates as a master timing channel and a number (n, in the range 2 to 14) slave channels are externally gated in pairs to generate ‘n ≥ 2’ PWM outputs. **Figure 3** is a CA mode schematic.

In CA mode, the PWM outputs have center-aligned high times. To operate in CA mode, the function uses pairs of slave channels that perform slightly different operations. The first channel of the pair is referred to as a Type A slave, and the second channel of the pair is referred to as a Type B slave. In CA mode, the output of the master channel is normally not used. **Figure 4** shows CA mode PWM output waveforms.

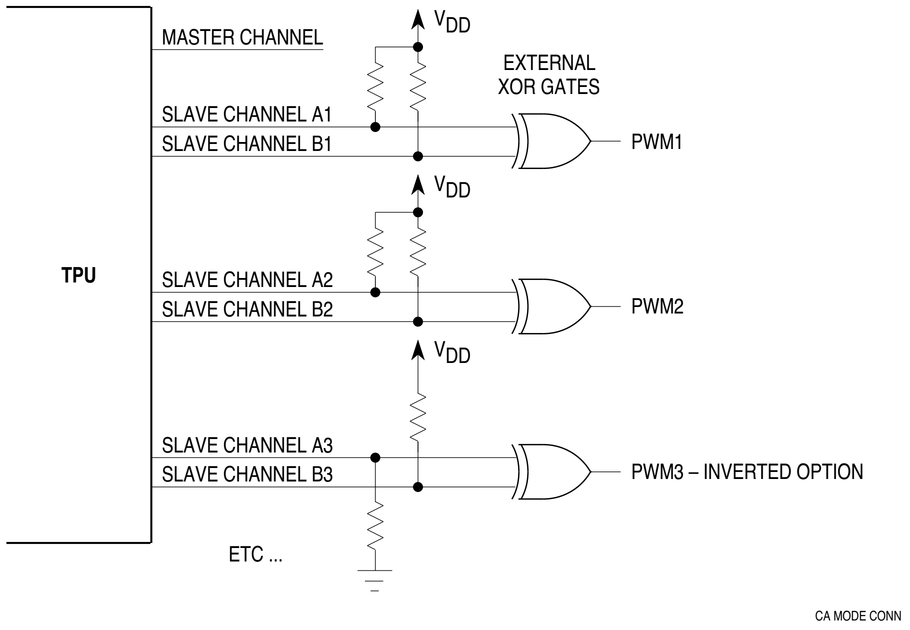


Figure 3 CA Mode External Gate Schematic

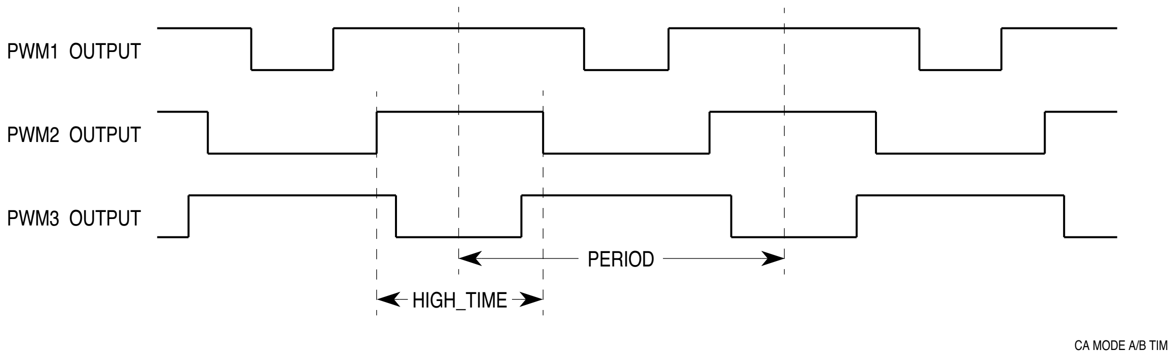


Figure 4 CA Mode Type A and Type B Output Waveforms

CA mode includes optional dead time and inversion functions. Dead time effectively reduces PWM duty cycle slightly — if two CA mode PWM channels, one with a specified dead time and one with no specified dead time, reference the same high time, the channel with dead time specified has a slightly shorter high time and thus does not change state at the same time as the other channel. The user can specify a dead time accurate to one timer clock. The user can choose to invert one or more of the PWM outputs during initialization. These features allow MCPWM to easily drive inverter-type applications. **Figure 5** shows CA mode inversion and dead-time output waveforms.

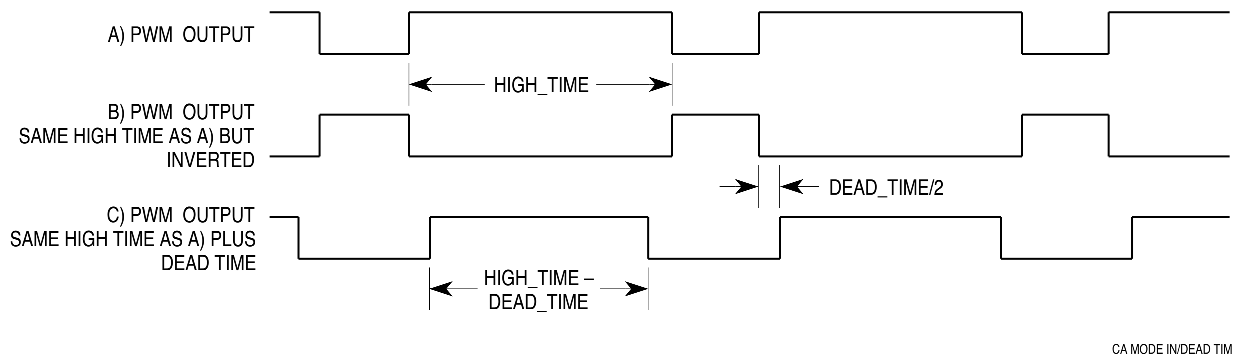

A) PWM OUTPUT
HIGH_TIME
B) PWM OUTPUT SAME HIGH TIME AS A) BUT INVERTED
DEAD_TIME/2
C) PWM OUTPUT SAME HIGH TIME AS A) PLUS DEAD TIME
HIGH_TIME – DEAD_TIME
CA MODE IN/DEAD TIM

Figure 5 CA Mode Inverted and Dead-Time Waveforms

In both operating modes, the master MCPWM channel can be programmed to generate an interrupt service request to the host CPU when a specified number of PWM periods have elapsed. The number of PWM periods is specified by a user programmable 8-bit number. In many applications, this feature can be used to make the CPU update PWM duty cycles at a predetermined rate. The programmable interrupt feature is particularly useful for sine-wave modulated PWM production. The programmable interrupt feature can also be used as a simple periodic interrupt timer for the CPU — in this case, only one channel, programmed as a master, is needed.

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. MCPWM function code size is:

30 μ instructions + 8 entries = **38 long words**

4 Function Parameters

This section provides detailed descriptions of multichannel PWM function parameters stored in channel parameter RAM. **Figure 6** shows TPU parameter RAM address mapping. **Figure 7** shows the parameter RAM assignment used by the function for the various modes of operation. 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 6 TPU Channel Parameter RAM CPU Address Map

Master Channel Parameter Assignment — All Modes

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$YFFFW0	PERIOD															
$YFFFW2	▒ IRQ_RATE								▒ PERIOD_COUNT							
$YFFFW4	LAST_RISE_TIME															
$YFFFW6	LAST_FALL_TIME															
$YFFFW8	RISE_TIME_PTR															
$YFFFWA	FALL_TIME_PTR															
$YFFFWC	■															
$YFFFWE	■															

W = Channel number

Slave Channel Parameter Assignment — Edge-Aligned Mode

$YFFFW0	PERIOD
$YFFFW2	HIGH_TIME
$YFFFW4	■
$YFFFW6	HIGH_TIME_PTR
$YFFFW8	RISE_TIME_PTR
$YFFFWA	FALL_TIME_PTR
$YFFFWC	■
$YFFFWE	■

W = Channel number

Parameter Write Access

□	Written by CPU
░	Written by TPU
▒	Written by CPU and TPU
■	Unused parameters

Figure 7 MCPWM Function Parameter RAM Assignment (1 of 3)

Slave Channel A Parameter Assignment — Non-Inverted Center-Aligned Mode

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$YFFFW0	PERIOD															
$YFFFW2	▨ NXT_B_RISE_TIME															
$YFFFW4	▨ NXT_B_FALL_TIME															
$YFFFW6	DEAD_TIME								HIGH_TIME_PTR							
$YFFFW8	RISE_TIME_PTR															
$YFFFWA	FALL_TIME_PTR															
$YFFFWC	⬛															
$YFFFWE	⬛															

W = Channel number

Slave Channel B Parameter Assignment — Non-Inverted Center-Aligned Mode

$YFFFW0	HIGH_TIME															
$YFFFW2	▨ CURRENT_HIGH_TIME															
$YFFFW4	▨ TEMP_STORAGE															
$YFFFW6	⬛															
$YFFFW8	B_FALL_TIME_PTR															
$YFFFWA	B_RISE_TIME_PTR															
$YFFFWC	⬛															
$YFFFWE	⬛															

W = Channel number

Parameter Write Access

⬜	Written by CPU
░	Written by TPU
▨	Written by CPU and TPU
⬛	Unused parameters

Figure 7 MCPWM Function Parameter RAM Assignment (2 of 3)

Slave Channel A Parameter Assignment — Inverted Center-Aligned Mode

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$YFFFW0	PERIOD															
$YFFFW2	NXT_B_FALL_TIME															
$YFFFW4	NXT_B_RISE_TIME															
$YFFFW6	DEAD_TIME								HIGH_TIME_PTR							
$YFFFW8	FALL_TIME_PTR															
$YFFFWA	RISE_TIME_PTR															
$YFFFWC	■															
$YFFFWE	■															

W = Channel number

Slave Channel B Parameter Assignment — Inverted Center-Aligned Mode

$YFFFW0	HIGH_TIME								
$YFFFW2	CURRENT_HIGH_TIME								
$YFFFW4	TEMP_STORAGE								
$YFFFW6	■								
$YFFFW8	B_FALL_TIME_PTR								
$YFFFWA	B_RISE_TIME_PTR								
$YFFFWC	■								
$YFFFWE	■								

W = Channel number

Parameter Write Access

□	Written by CPU
░	Written by TPU
▒	Written by CPU and TPU
■	Unused parameters

Figure 7 MCPWM Function Parameter RAM Assignment (3 of 3)

5 Master Channel Parameters —All Modes

5.1 PERIOD

This parameter is written by the CPU before initialization. It defines the period of the PWM output in TCR1 clocks. PERIOD has a maximum permissible value of $4000 and a minimum value that is dependent on TPU latency. See **11 Performance and Use of MCPWM Function** for more information. If PERIOD is changed while the function is running, there may be a period of indeterminate output from the EOR gates. This is due to the delay between the master and slave channels changing period. The channel actually produces a square wave output with a period of two times the value in PERIOD, but the PWM resulting from the combination of the master with a slave channel has a period equal to PERIOD.

5.2 IRQ_RATE

IRQ_RATE determines the frequency of periodic interrupt requests to the host CPU. IRQ_RATE is specified in the number of PWM periods required between each interrupt; any 8-bit value is valid. This parameter is written by the CPU prior to initialization. If no periodic interrupt is required then the interrupt enable bit for the master channel should be negated. If IRQ_RATE = $00 then an interrupt request is generated every 256 PWM periods.

5.3 PERIOD_COUNT

This parameter is used by the TPU as a PWM period counter for periodic CPU interrupt requests. PERIOD_COUNT is automatically reset to zero after each request is generated (when PERIOD_COUNT = IRQ_RATE), but should be initialized to a start value (usually $FF) by the host CPU prior to issuing the initialization HSR. Thereafter the parameter must not be written. An initialization value of $FF is used instead of $00 because PERIOD_COUNT is also incremented during the initialization state.

5.4 LAST_RISE_TIME

The LAST_RISE_TIME parameter contains the event time of the latest rising edge on the master MCPWM channel. It is updated by the TPU when the rising edge is serviced. This parameter is referenced by the slaves to maintain the selected timing relationship between channels. It must never be written by the CPU. This parameter is not recommended for interpretation by the user.

5.5 LAST_FALL_TIME

The LAST_FALL_TIME parameter contains the event time of the latest falling edge on the master MCPWM channel. It is updated by the TPU when the falling edge is serviced. This parameter is referenced by the slaves to maintain the selected timing relationship between channels. It must never be written by the CPU. This parameter is not recommended for interpretation by the user.

5.6 RISE_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the master channel LAST_RISE_TIME parameter. For example, if channel 4 is the MCPWM master channel, then the value $44 would be stored in RISE_TIME_PTR.

5.7 FALL_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the master channel LAST_FALL_TIME parameter. For example, if channel 4 is the MCPWM master channel, then the value $46 would be stored in FALL_TIME_PTR.

6 Edge-Aligned Mode Slave Parameters

6.1 PERIOD

This parameter is written by the CPU before initialization. It defines the period of the PWM output in TCR1 clocks. PERIOD has a maximum permissible value of $4000 and a minimum value that is dependent on TPU latency. See**11 Performance and Use of MCPWM Function** for more information. The channel actually produces a square wave output with a period of two times the value in PERIOD, but the PWM resulting from the combination of the master with a slave channel has a period equal to PERIOD. The PERIOD parameters of all slaves must have the same value as that of the master that they are referencing. If PERIOD is changed while the function is running, there may be a period of indeterminate output from the EOR gates, due to the delay between the master and slave channels changing period.

6.2 HIGH_TIME_PTR

This parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the parameter containing the PWM high time. It is used by the TPU as an address pointer to obtain the HIGH_TIME parameter. For example, if the high time is contained in parameter 1 of channel 2 then the value $22 is placed in HIGH_TIME_PTR. Normally the high time parameter resides in slave channel parameter RAM, but it could be anywhere in PRAM — this allows the output of another TPU function to be used as PWM high time.

6.3 HIGH_TIME

As stated above, HIGH_TIME can reside anywhere in PRAM, but would normally be located in parameter 1 of the slave channel. HIGH_TIME is specified in TCR1 clocks over the range 0 to PERIOD, representing 0 to 100% duty cycle. No tests are performed for HIGH_TIME values outside of this valid range — such values result in an indeterminate output from the EOR gates. Valid value testing, if required, is the responsibility of the CPU. HIGH_TIME can be written at any time by the CPU.

6.4 RISE_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the master channel LAST_RISE_TIME parameter. For example, if channel 4 is the MCPWM master channel, then the value $44 would be stored in RISE_TIME_PTR.

6.5 FALL_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the master channel LAST_FALL_TIME parameter. For example, if channel 4 is the MCPWM master channel, then the value $46 would be stored in FALL_TIME_PTR.

7 Center-Aligned Mode Slave Type a Parameters

Some slave A parameters change address when the inversion option is specified. Study parameter diagrams carefully.

7.1 PERIOD

This parameter is written by the CPU before initialization. It defines the period of the PWM output in TCR1 clocks. PERIOD has a maximum permissible value of $4000 and a minimum value that is dependent on TPU latency. See**11 Performance and Use of MCPWM Function** for more information. The PERIOD parameters of all slaves must have the same value as that of the master that they are referencing. If PERIOD is changed while the function is running, there may be a period of indeterminate output from the EOR gates, due to the delay between the master and slave channels changing period.

7.2 NXT_B_RISE_TIME

The TPU stores the calculated event time for the next rising edge on the slave B channel (the second channel of the CA mode pair) in this parameter. This parameter must not be written by the CPU. This parameter is not recommended for interpretation by the user.

7.3 NXT_B_FALL_TIME

The TPU stores the calculated event time for the next falling edge on the slave B channel (the second channel of the CA mode pair) in this parameter. This parameter must not be written by the CPU. This parameter is not recommended for interpretation by the user.

7.4 HIGH_TIME_PTR

This parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the LSB (that is, the odd address) of the parameter containing the PWM high time. It is used by the TPU as an address pointer to obtain the HIGH_TIME parameter. For example, if the high time is contained in parameter 0 of channel 6 then the value $61 is placed in HIGH_TIME_PTR. Normally the high time parameter resides in slave B channel parameter RAM, but it could be anywhere in PRAM (as a pair) — this allows the output of another TPU function to be used as PWM high time.

7.5 DEAD_TIME

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the total dead time to be subtracted from the high time of the PWM, specified in TCR1 clocks. If no dead time is required, the DEAD_TIME parameter should be set to zero. Since DEAD_TIME specifies the total time to be subtracted from HIGH_TIME, a value of $02 results in a dead time of one TCR1 clock on each PWM transition. Any number in the range zero to $FF is valid, although the value used should normally be even and small.

7.6 RISE_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the master channel LAST_RISE_TIME parameter. For example, if channel 5 is the MCPWM master channel, then the value $54 would be stored in RISE_TIME_PTR.

7.7 FALL_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the master channel LAST_FALL_TIME parameter. For example, if channel 5 is the MCPWM master channel, then the value $56 would be stored in FALL_TIME_PTR.

8 Center-Aligned Mode Slave Type B Parameters

8.1 HIGH_TIME

As stated in the slave A section, HIGH_TIME can reside anywhere in PRAM, but would normally be located in parameter 0 of the slave channel. HIGH_TIME is specified in TCR1 clocks over the range 0 to PERIOD, representing 0 to 100% duty cycle. No tests are performed for HIGH_TIME values outside of this valid range — such values result in an indeterminate output from the EOR gates. Valid value testing, if required, is the responsibility of the CPU. HIGH_TIME can be written at any time by the CPU.

8.2 CURRENT_HIGH_TIME

CURRENT_HIGH_TIME is updated by the slave A channel. It contains the value of HIGH_TIME currently in use. When a second CA mode slave pair is being used to generate a PWM with dead time, HIGH_TIME_PTR should point to the LSB of the CURRENT_HIGH_TIME parameter of the first slave pair (the pair without dead time). This insures coherency in a HIGH_TIME update — in any period, the PWM with dead time is guaranteed to use the same HIGH_TIME value as the PWM without dead time. Since the rate at which CURRENT_HIGH_TIME is updated depends on slave A service time, the stored value cannot be usefully interpreted by the user.

8.3 TEMP_STORAGE

This parameter is used by the TPU for temporary storage. It should not be written by the CPU.

8.4 B_FALL_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the slave A channel NXT_B_FALL_TIME parameter. For example if channel 7 is the slave A channel (non-inverted mode) then the value $74 should be stored in B_FALL_TIME_PTR.

8.5 B_RISE_TIME_PTR

This 8-bit parameter is configured by the host CPU prior to initialization. It contains the PRAM address of the slave A channel NXT_B_RISE_TIME parameter. For example if channel 7 is the slave A channel (non-inverted mode) then the value $72 should be stored in B_RISE_TIME_PTR.

9 Host Interface to MCPWM Function

This section provides information concerning the TPU host interface to the MCPWM function. **Figure 8** 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 8 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]	Operating Mode — Only Used On Slave Channels
00	Edge-Aligned Mode
01	Slave A Type CA Mode
10	Slave B Type CA Mode
11	Slave B Type CA Mode

HSRR[1:0] — 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	Initialize as Slave A Inverted Mode
10	Initialize All Other Slave Modes
11	Initialize as Master

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

10 Configuration of MCPWM Function

The CPU configures the MCPWM function as follows. See **11 Performance and Use of MCPWM Function** for constraints on channel assignment.

Edge-Aligned Mode:

1. Disables all the master and slave MCPWM channels by clearing channel priority bits.
2. Selects the MCPWM function on the master and all the slave channels by writing the MCPWM function number to channel function select bits.
3. Initializes PERIOD, IRQ_RATE, PERIOD_COUNT, RISE_TIME_PTR, and FALL_TIME_PTR in master channel parameter RAM.
4. Initializes PERIOD, HIGH_TIME_PTR, RISE_TIME_PTR and FALL_TIME_PTR in the PRAM of all edge-aligned slaves.
5. Selects edge-aligned mode on each slave channel by clearing the host sequence bits.
6. Sets the interrupt enable bit for the master channel if a periodic CPU interrupt is desired. The interrupt enable bits for all the slave channels should be cleared.
7. Initializes the HIGH_TIME parameter(s) wherever they reside in PRAM.
8. Issues an HSR%11 to the master channel and an HSR%10 to each slave channel.
9. Enables servicing by assigning the same non-zero priority (H, M, or L) to all of the MCPWM channels.

Center-Aligned Mode:

1. Disables all the master and slave MCPWM channels by clearing channel priority bits.
2. Selects the MCPWM function on the master and all the slave channels by writing the MCPWM function number to channel function select bits.
3. Initializes PERIOD, IRQ_RATE, PERIOD_COUNT, RISE_TIME_PTR, and FALL_TIME_PTR in master channel parameter RAM.
4. Initializes PERIOD, HIGH_TIME_PTR, RISE_TIME_PTR and FALL_TIME_PTR in the PRAM of all slave A channels.
5. Initializes B_HIGH_TIME_PTR, and B_RISE_TIME_PTR in the PRAM of all center-aligned slave B channels.
6. Selects the type of each slave via the host sequence bits (Slave A: HSQ =%01, Slave B: HSQ =%1X).
7. Sets the interrupt enable bit for the master channel if a periodic CPU interrupt is desired. The interrupt enable bits for all the slave channels should be cleared.
8. Initializes the HIGH_TIME parameter(s) wherever they reside in parameter RAM.
9. Issues an HSR%11 to the master channel and an HSR%10 to each slave B channel.
10. Issues an HSR%10 to non-inverted PWM slave A channels and an HSR%01 to inverted PWM slave A channels.
11. Enables servicing by assigning the same non-zero priority (H, M, or L) to all of the MCPWM channels.

All Modes Continue:

The TPU executes the initialization states of each channel and starts generating the wave forms that are externally gated to produce the desired PWM outputs. The CPU can write new high time parameters at any time.

PERIOD_COUNT is incremented on each service of the master channel (including initialization). When PERIOD_COUNT equals IRQ_RATE, the master channel makes a CPU interrupt service request and resets PERIOD_COUNT to $00.

If the master channel interrupt enable bit is set, the CPU recognizes and services the request.

11 Performance and Use of MCPWM Function

Like all TPU functions, the performance limit of the MCPWM function in a given application depends to some extent on the activity on other TPU channels. This is due to the operation of the scheduler.

Under steady state conditions, all MCPWM channels generate 50% duty cycle output wave forms (the phase delay between the signals generates the variable high time on the output of the EOR gates). This provides maximum service time for all transitions and lets the MCPWM function generate high and low duty cycle PWM consistently under a wide range of TPU operating conditions. Worst case service conditions for the MCPWM function occur when a large change in duty cycle is requested — in the extreme cases of a switch from 0% to 100% or 100% to 0%, the slave channels must produce a single pulse of half the normal period in order to establish the new timing relationship. Thus, in systems where these large duty cycle changes are possible, absolute worst-case timing analysis must be carried out using PERIOD divided by two as the maximum allowable service latency for each channel.

Taking this worst-case condition into account, and assuming a PWM PERIOD of 255 TCR1 clocks with the TCR1 prescaler set to divide by 4 (the fastest possible 8-bit PWM), the following is an approximate TPU loading formula for MCPWM generation:

$$\text{TPU loading (\%)} = 6 + (a * 5.1) + (b * 12.2)$$

where:

a = number of edge-aligned PWM
b = number of center-aligned PWM
The additional 6% is for the master channel.

This formula assumes a very low PRAM collision rate between the CPU and TPU, which is the normal operating condition. However, in applications where the CPU makes very frequent accesses to PRAM, these loading figures do not apply. In this case, and in applications where other TPU functions are running, detailed timing analysis of the TPU system is required.

Since the scheduler assures that the worst-case latencies in any TPU application can be closely estimated, a detailed timing analysis can be performed by following the guidelines given in the TPU reference manual. To perform an analysis, use the MCPWM state timing information in the table below along with the state timing information for any other active TPU functions.

Table 1 Multichannel PWM Function — State Timing

State Number & Name	Max. CPU Clock Cycles	RAM Accesses by TPU
S1 MINIT_MCPWM	20	6
S2 SINIT_MCPWM		
EA Mode slave	18	6
CA Mode slave A	38	8
CA Mode slave B	8	2
S3 S_INV_INIT_MCPWM	40	8
S4 MLH_MCPWM	18	6
S5 MHL_MCPWM	18	6
S6 SHL_MCPWM		
EA Mode slave	16	6
CA Mode slave A	36	8
CA Mode slave B	6	2
S7 SLH_MCPWM		
EA Mode slave	16	6
CA Mode slave A	36	8
CA Mode slave B	6	2

NOTE: Execution times do not include the time slot transition time (TST = 10 or 14 CPU clocks)

11.1 Channel Assignment Restrictions

Due to coherency issues and the operation of the TPU scheduler, the following rules must be adhered to when assigning channels and priorities to ensure correct operation of the MCPWM function:

1. All channels associated with a master channel must be assigned the same priority as that master.
2. The master must be assigned the lowest channel number of all MCPWM channels. In CA mode, the slave A channel of a pair must be assigned a lower channel number than the slave B channel.
3. When two CA mode pairs are used to form an inverter driver (the second pair has dead time and references the CURRENT_HIGH_TIME parameter of the first pair), the pair without dead time must be assigned lower channel numbers than the pair with dead time.

The following example shows application of these rules to a 3-phase H-bridge drive configuration using thirteen TPU channels (six CA mode PWM plus master)

Table 2 Example Multichannel PWM Function

Channel Number	Operation Mode	Function
0	Master	Reference timing channel for all slaves
1	Slave A	PWM1
2	Slave B	PWM1
3	Slave A	PWM1' — as PWM1 but with dead time
4	Slave B	PWM1' — as PWM1 but with dead time
5	Slave A	PWM2
6	Slave B	PWM2
7	Slave A	PWM2' — as PWM2 but with dead time
8	Slave B	PWM2' — as PWM2 but with dead time
9	Slave A	PWM3
10	Slave B	PWM3
11	Slave A	PWM3' — as PWM3 but with dead time
12	Slave B	PWM3' — as PWM3 but with dead time

11.2 Resolution/Frequency Relationship

Unlike most hardware PWM peripherals, the resolution and frequency of the MCPWM function is not fixed and can be tailored to meet application requirements. In general, the two qualities are inversely proportional, i.e., the higher the resolution, the lower the frequency. Also, since PERIOD is freely programmable in TCR1 clocks, the number of bits of resolution does not have to be an integer number. The user could program PERIOD to be $180, to produce an'8.5 bit' PWM with 384 resolution states. With a 16.78-MHz system clock, the minimum TCR1 clock period is 240 ns and the resulting frequency of the PWM output is given by $1 / (\text{PERIOD} * 240 \text{ ns})$. This gives frequencies of 16.4 kHz, 8.2 kHz, and 4.1 kHz for 8-, 9-, and 10-bit resolution PWM respectively. With a 20.97-MHz system clock, the equivalent frequencies are 20.56 kHz, 10.28 kHz and 5.14 kHz.

11.3 Hardware Requirements

The MCPWM function requires one external exclusive OR gate for each PWM output. In addition, it is recommended that pull-up resistors be added to all MCPWM channels except for slave A channels of inverted PWM, which should have pull down resistors. The resistors insure a predictable output from the EOR gates during the time between power up or reset (when all TPU channel pins are in a high-impedance state) and initialization of the MCPWM function. In some circumstances, an external buffer with three-state capability may be required. See **11.10 Stopping the Function** for more information.

11.4 Initialization Timing

The following timing assumes initialization from reset.

In edge-aligned mode, the first rising edge of the PWM outputs from the EOR gates occurs approximately one PWM period after completion of service of the master channel initialization host service request. **Figure 9** shows these relationships.

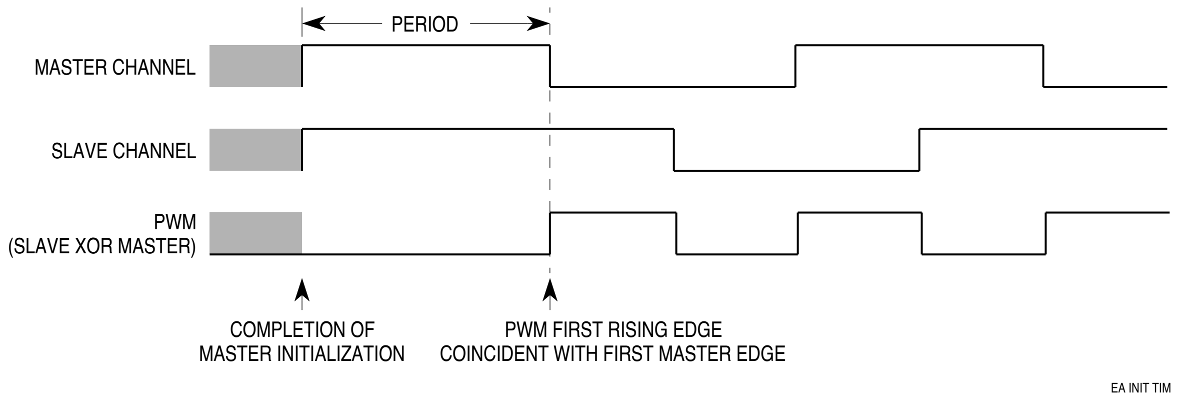

PERIOD
MASTER CHANNEL
SLAVE CHANNEL
PWM (SLAVE XOR MASTER)
COMPLETION OF MASTER INITIALIZATION
PWM FIRST RISING EDGE COINCIDENT WITH FIRST MASTER EDGE
EA INIT TIM

Figure 9 EA Mode Initialization Timing

In center-aligned mode, the first rising edge occurs between one and one-and-a-half PWM periods after completion of service of the master channel initialization host service request, depending on the programmed duty cycle. **Figure 10** shows these relationships.

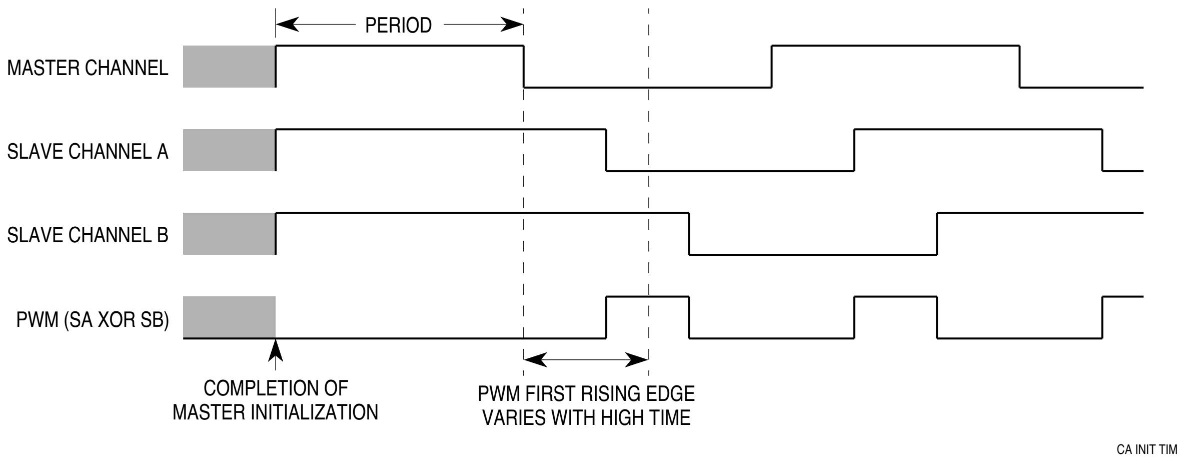

PERIOD
MASTER CHANNEL
SLAVE CHANNEL A
SLAVE CHANNEL B
PWM (SA XOR SB)
COMPLETION OF MASTER INITIALIZATION
PWM FIRST RISING EDGE VARIES WITH HIGH TIME
CA INIT TIM

Figure 10 CA Mode Initialization Timing

11.5 High Time Update Timing

In both edge-aligned and center-aligned modes, when a new HIGH_TIME value is written by the CPU, it takes effect either at the beginning of the next period or the beginning of the period following that, depending on the exact timing of the write as explained below. There is no provision for updating the high time of the period in progress.

When a new high time value is written in edge-aligned mode, if the write takes place before the falling edge of the PWM, the new high time value is used in the next period. If the write takes place after the falling edge of the PWM, the new value is used no later than the second period after the write.

When a new high time value is written in center-aligned mode, if the write takes place before the rising edge of the PWM (non-inverted), the new high time value is used in the next period. If the write takes place after the rising edge, the new value is used no later than the second period after the write.

11.6 Coherent High Time Updates

There are two possible cases when a HIGH_TIME parameter is updated at a random time in relation to PWM waveform and channel servicing. Using the periodic master channel interrupt does not guarantee either case, since the master edge that results in an interrupt is coincident with the start of the PWM period — a CPU write in response to the interrupt could result in either case depending on the current high time and CPU and TPU latencies.

There may be applications where this high time uncertainty is unacceptable. For example, an application may demand that multiple PWM outputs be updated in the same period. MCPWM can support such a requirement in center-aligned mode by using a second master channel solely for periodic interrupt generation. The second master channel runs at twice the speed of the master channel used for actual PWM generation.

To support this capability, set up the second master channel as follows:

1. Second master channel number > last PWM slave channel number
2. PERIOD = (PWM period)/2
3. Interrupts enabled
4. IRQ_RATE = (desired number of PWM periods between interrupts) $*$ 2
5. PERIOD_COUNT initialized to $00
6. Priority same as other MCPWM channels

The initialization HSR for the second master channel should be issued at the same time as that for other MCPWM channels. Channel order is important because the second master channel must be serviced last when multiple MCPWM channels request service simultaneously (see details of scheduler operation in the TPU reference manual). This scheme causes an interrupt request to be made to the CPU in the second half of the PWM period, close to the period mid-point but after other pending MCPWM channels have been serviced. The CPU has the remainder of the PWM period in which to update HIGH_TIME parameters before the next PWM rising edge, thus guaranteeing use of a new high time in the following cycle. **Figure 11** shows an example of this technique where the high times of two PWMs are coherently updated every three periods (IRQ_RATE of second master = 6).

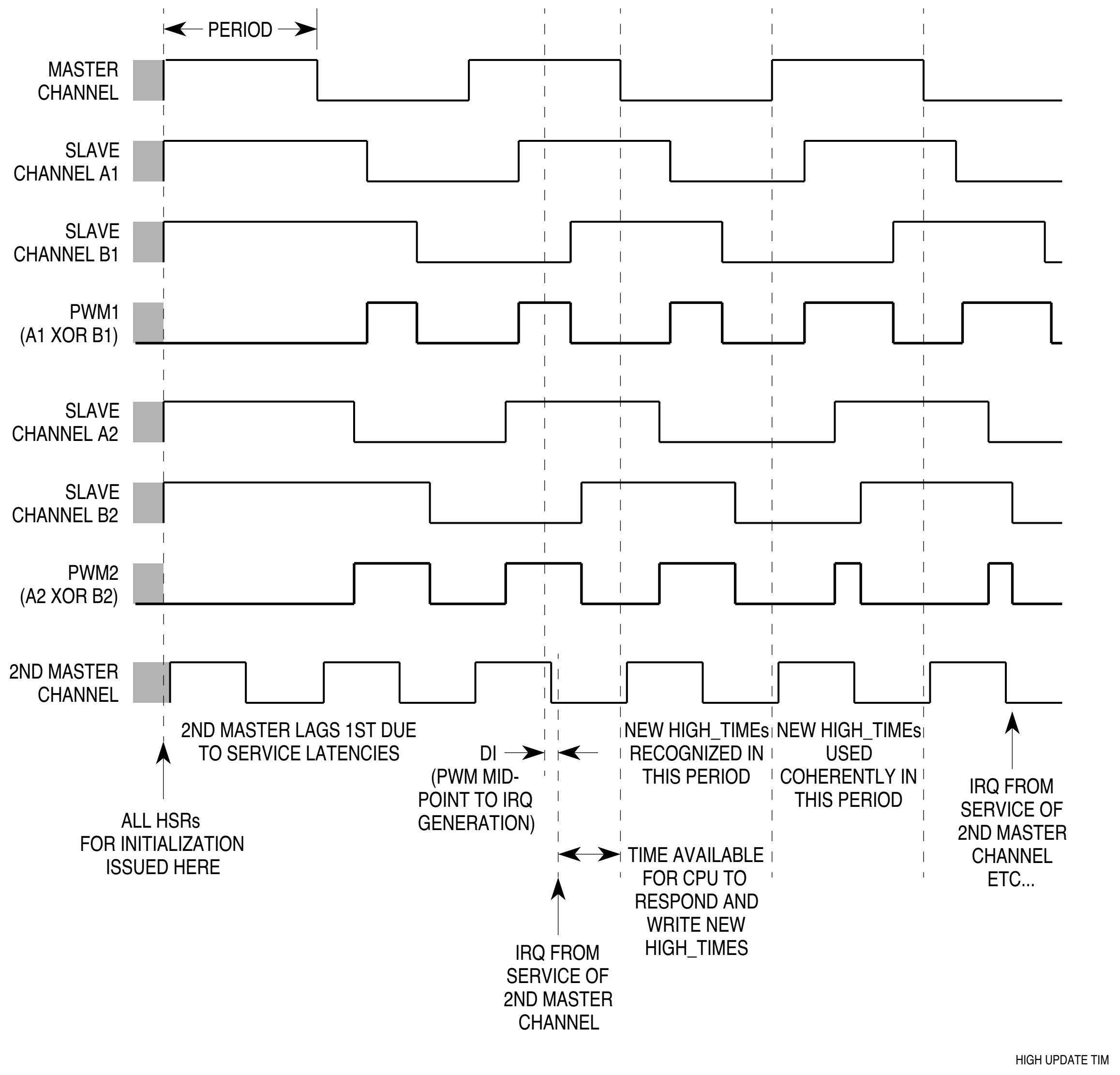

PERIOD
MASTER CHANNEL
SLAVE CHANNEL A1
SLAVE CHANNEL B1
PWM1 (A1 XOR B1)
SLAVE CHANNEL A2
SLAVE CHANNEL B2
PWM2 (A2 XOR B2)
2ND MASTER CHANNEL
2ND MASTER LAGS 1ST DUE TO SERVICE LATENCIES
ALL HSRs FOR INITIALIZATION ISSUED HERE
DI (PWM MID-POINT TO IRQ GENERATION)
IRQ FROM SERVICE OF 2ND MASTER CHANNEL
NEW HIGH_TIMEs RECOGNIZED IN THIS PERIOD
TIME AVAILABLE FOR CPU TO RESPOND AND WRITE NEW HIGH_TIMES
NEW HIGH_TIMEs USED COHERENTLY IN THIS PERIOD
IRQ FROM SERVICE OF 2ND MASTER CHANNEL ETC...
HIGH UPDATE TIM

Figure 11 HIGH_TIME Update Timing

Worst-case delay (DI) from the period mid-point to interrupt generation using this technique occurs when all PWM are set to 0% (all slave channels request service at PWM mid-point). DI can be closely estimated:

DI = Total Worst Case Service Time (WCST) of all MCPWM slave channels +
Worst case service time of second master channel.

For example, if MCPWM is used to generate two centered PWM (four slaves), with a second master channel for coherent high time updates, then:

DI = WCSTA1 + WCSTA2 + WCSTB1 + WCSTB2 + WCSTM2 = 46 + 16 + 46 + 16 + 28 = 152 CPU clocks

The CPU has the remainder of the PWM period to respond to the interrupt and update the high times.

11.7 Effects of DEAD_TIME

When DEAD_TIME has a non-zero value, the programmed HIGH_TIME is reduced by an amount equal to DEAD_TIME TCR1 clocks. This means that the minimum HIGH_TIME value that can cause a non-zero PWM duty cycle is (DEAD_TIME + 1). Similarly, at the upper end of the scale, 100% duty cycle is not possible with a non-zero DEAD_TIME value. If HIGH_TIME is programmed to equal PERIOD, the output has a non-zero low time equal to DEAD_TIME.

11.8 Odd PERIOD/HIGH_TIME/DEAD_TIME Values in CA Mode

In center-aligned mode, the MCPWM function attempts to center ‘real’ high time (HIGH_TIME – DEAD_TIME) within the PWM period. Obviously, with an even PERIOD and an odd high time or with an odd PERIOD and an even high time, exact centering is not possible. In these two cases, full resolution is maintained by making the portion of the high time before period mid-point one TCR1 clock longer than the portion following the period mid-point.

11.9 Changing Period

The MCPWM function is designed primarily for applications where the period of the PWM output is not variable. Since the PWM output is the result of the combination of two independently-running channels, when the CPU changes the PERIOD parameters, one channel must use the new period before the other. This results in temporary loss of synchronization and indeterminate output from the EOR gate. When HIGH_TIME is less than or equal to the new PERIOD, the function returns to normal operation after a maximum of two times the original PERIOD value.

11.10 Stopping the Function

Like all TPU functions, MCPWM function can be disabled by clearing the priority bits of all MCPWM channels. However, this procedure leaves EOR gate output at an indeterminate level, even when HIGH_TIME is programmed to 0 or 100% before the bits are cleared. This is due to the phase relationship between the two channels forming the PWM output and to independent servicing of each channel. In most applications the best method of switching off the PWM is to program HIGH_TIME to 0 or 100% and leave the function running. When the function must be stopped cleanly there are two solutions:

1. Place an external buffer with three-state capability on the output of the EOR gate and control it with an MCU output pin.
2. Reset the MCU — this returns all TPU pins to their original high-impedance condition.

11.11 Interrupts

Interrupt service requests are generated by the MCPWM master channel every IRQ_RATE periods. Interrupts are enabled or disabled by setting or clearing the interrupt enable bit for the master channel. The slave A channels can also generate interrupt requests, but these are meaningless to the user and should be disabled by clearing the interrupt enable bits for all slave channels.

Since interrupt request generation is performed by microcode software when the relevant master channel edge is serviced, there is a delay between when the master channel edge transition occurs and the interrupt request signal is asserted. This delay is variable — it depends on the service latency of the master channel in a particular application. The time between successive interrupt requests can vary and does not exactly equal the IRQ_RATE periods. Worst-case variation (WCV) about IRQ_RATE periods are derived as follows:

WCV = (Worst case master channel service latency) –(Best case master channel service latency)

It follows that there is a spread of interrupt periods from

$$\{(IRQ\_RATE * PERIOD) – WCV\} \text{ to } \{(IRQ\_RATE * PERIOD) + WCV\}.$$

11.12 Combining EA and CA Modes

Since master channel operation is identical for both edge-aligned and center-aligned modes, it is possible to generate both types of PWM in a system using only one master channel. However, the periods of all PWM that refer to a common master must be identical.

11.13 Using MCPWM as a Periodic Timer

The periodic update interrupt feature of the MCPWM function makes it suitable for use as a periodic interrupt timer (PIT) for the CPU. To use the function as a PIT, only one channel, programmed as a master, is required. When no slave channels are referencing the master channel, maximum PERIOD parameter value can be increased to $8000 TCR1 clocks. The combination of PERIOD and the 8-bit IRQ_RATE parameters allows a wide variety of interrupt periods to be programmed. The maximum interrupt period can be greater than one minute; the minimum period is obtained by programming IRQ_RATE to one and setting PERIOD to a minimum value determined from the state timing table and other TPU system activity.

12 Multichannel PWM Examples

Although complex, the MCPWM function is relatively easy to configure and control. The following examples show configuration for both center-aligned and edge-aligned modes. 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 wave form.

12.1 Example A

12.1.1 Description

Edge-aligned mode. Generate two PWM with a period of 256 TCR1 clocks, one with a duty cycle of 50%, the other 25%. Use channel 1 as the master and channels 2 and 3 as the slaves. Generate a periodic interrupt every five PWM periods.

12.1.2 Initialization

Disable channels by clearing priority bits. Select MCPWM function by programming the function select registers. Enable interrupts on channel 1 and disable interrupts on channels 2 and 3. Load the parameter RAM of the three channels as shown below. Write HSQ = %00 to both channels 2 and 3. Issue HSR = %11 to channel 1 and HSR = %10 to channels 2 and 3 to initialize and start the function. Write the priority bits of all three channels to the same non-zero value.

Table 3 Channel 1 (Master) Parameter RAM

	15							8								0	
$YFFF10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PERIOD
$YFFF12	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	IRQ_RATE
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF18	x	x	x	x	x	x	x	x	0	0	0	1	0	1	0	0	$XX14
$YFFF1A	x	x	x	x	x	x	x	x	0	0	0	1	0	1	1	0	$XX16

PERIOD = $100, IRQ_RATE = $5

Table 4 Channel 2 (Slave) Parameter RAM

	15							8								0	
$YFFF20	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PERIOD
$YFFF22	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	HIGH_TIME 1
$YFFF24	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF26	x	x	x	x	x	x	x	x	0	0	1	0	0	0	1	0	$XX22
$YFFF28	x	x	x	x	x	x	x	x	0	0	0	1	0	1	0	0	$XX14
$YFFF2A	x	x	x	x	x	x	x	x	0	0	0	1	0	1	1	0	$XX16

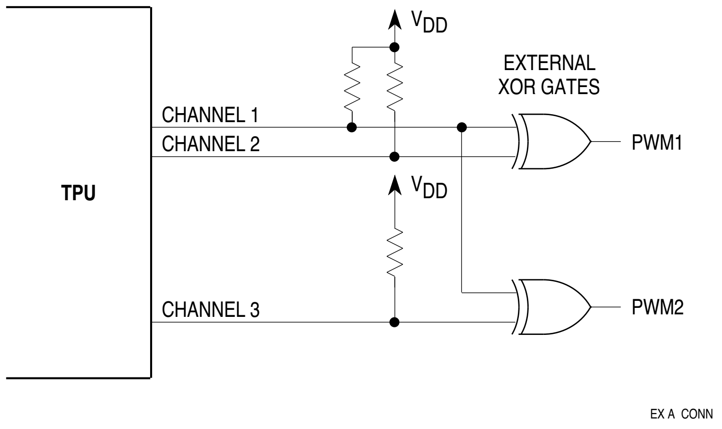
PERIOD = $100, HIGH_TIME_PTR ⇒ HIGH_TIME 1 (= $80), RISE/FALL_TIME_PTRs ⇒ channel1

Table 5 Channel 3 (Slave) Parameter RAM

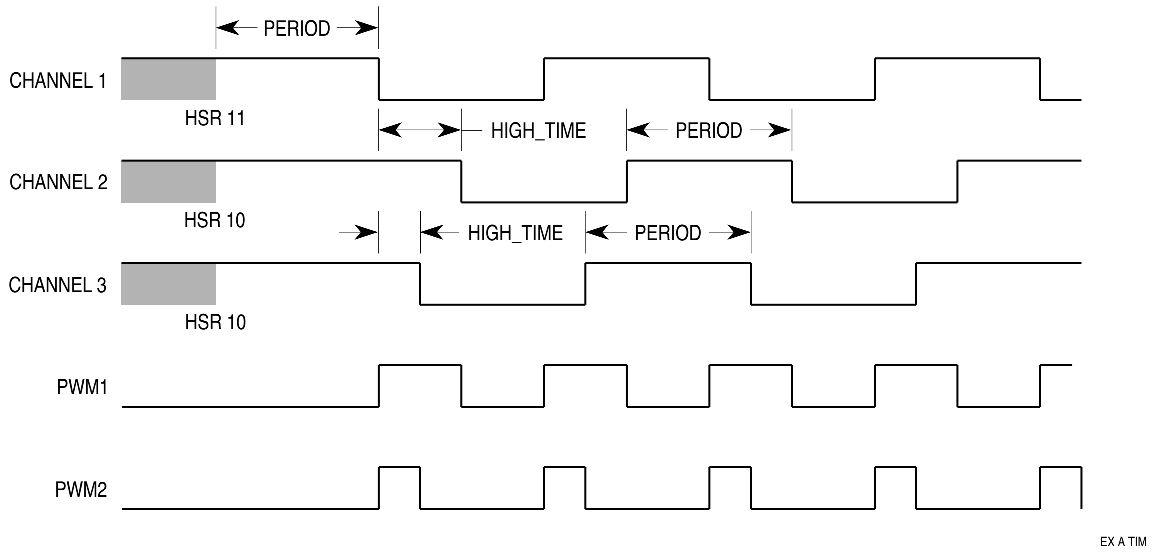
	15							8								0	
$YFFF30	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PERIOD
$YFFF32	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	HIGH_TIME 2
$YFFF34	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF36	x	x	x	x	x	x	x	x	0	0	1	1	0	0	1	0	$XX32
$YFFF38	x	x	x	x	x	x	x	x	0	0	0	1	0	1	0	0	$XX14
$YFFF3A	x	x	x	x	x	x	x	x	0	0	0	1	0	1	1	0	$XX16

PERIOD = $100, HIGH_TIME_PTR ⇒ HIGH_TIME 2 (= $40), RISE/FALL_TIME_PTRs ⇒ channel1

12.1.3 Schematic



12.1.4 Output Waveforms



12.2 Example B

12.2.1 Description

Center-aligned mode. Generate a pair of PWM outputs with a period of 256 TCR1 clocks, sharing a common HIGH_TIME and suitable for driving an inverter. The required 'dead time' for the driving devices is equivalent to 2 TCR1 clocks (⇒ DEAD_TIME = 4). Use channel 1 as the master and channels 2, 3, 4 and 5 as the slaves. Generate a periodic interrupt every 10 PWM periods. Initialize with a duty cycle of 50%.

12.2.2 Initialization

Disable channels by clearing priority bits. Select MCPWM function by programming the function select registers. Enable interrupts on channel 1 and disable interrupts on the other channels. Load the parameter RAM of the five channels as shown below. Write HSQ = %01 to channels 2 and 4 and HSQ = %11 to channels 3 and 5. Issue HSR = %11 to channel 1, HSR = %10 to channels 3, 4, and 5 and HSR = %01 to channel 2 to initialize and start the function. Write the priority bits of all five channels to the same non-zero value.

Table 6 Channel 1 (Master) Parameter RAM

	15							8								0	
$YFFF10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PERIOD
$YFFF12	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	IRQ_RATE
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF18	x	x	x	x	x	x	x	x	0	0	0	1	0	1	0	0	$XX14
$YFFF1A	x	x	x	x	x	x	x	x	0	0	0	1	0	1	1	0	$XX16

PERIOD = $100, IRQ_RATE = $A

Table 7 Channel 2 (Inverted Slave A) Parameter RAM

	15							8								0	
$YFFF20	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PERIOD
$YFFF22	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF24	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF26	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	$0031
$YFFF28	x	x	x	x	x	x	x	x	0	0	0	1	0	1	1	0	$XX16
$YFFF2A	x	x	x	x	x	x	x	x	0	0	0	1	0	1	0	0	$XX14

PERIOD = $100, DEAD_TIME = 0, HIGH_TIME_PTR ⇒ channel3, RISE/FALL_TIME_PTRs ⇒ channel1

Table 8 Channel 3 (Slave B) Parameter RAM

	15							8								0	
$YFFF30	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	HIGH_TIME
$YFFF32	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	CURR_HIGH_TIME
$YFFF34	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF36	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF38	x	x	x	x	x	x	x	x	0	0	1	0	0	0	1	0	$XX22
$YFFF3A	x	x	x	x	x	x	x	x	0	0	1	0	0	1	0	0	$XX24

PERIOD = $100, HIGH_TIME = $80, B_RISE/FALL_TIME_PTRs ⇒ channel2

Table 9 Channel 4 (Slave A) Parameter RAM

	15							8								0	
$YFFF40	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PERIOD
$YFFF42	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF44	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF46	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	1	$0433
$YFFF48	x	x	x	x	x	x	x	x	0	0	0	1	0	1	0	0	$XX14
$YFFF4A	x	x	x	x	x	x	x	x	0	0	0	1	0	1	1	0	$XX16

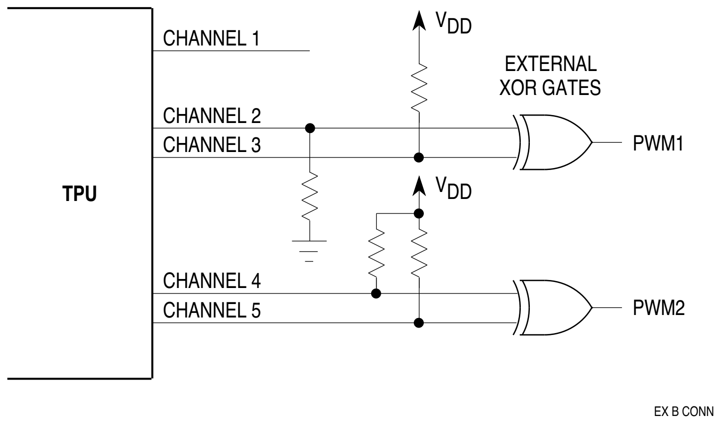
PERIOD = $100, DEAD_TIME = 4, HIGH_TIME_PTR ⇒ channel3, RISE/FALL_TIME_PTRs ⇒ channel1

Table 10 Channel 5 (Slave B) Parameter RAM

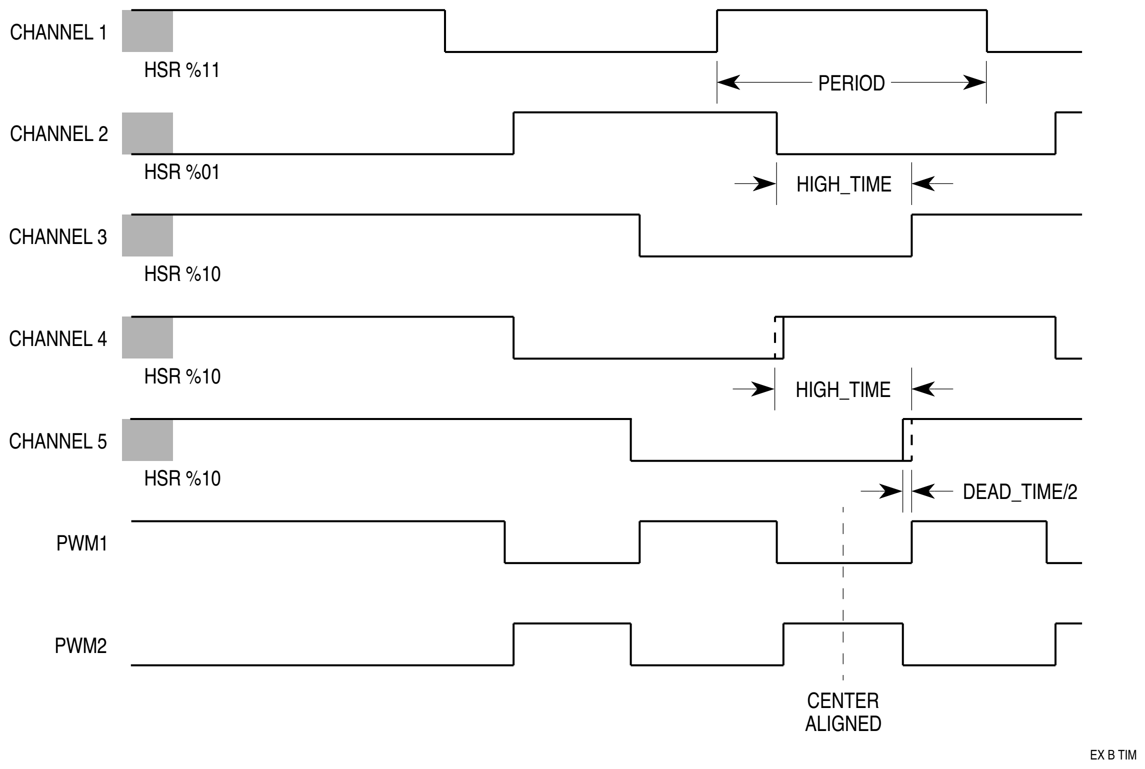
	15							8								0	
$YFFF50	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF52	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF54	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF56	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
$YFFF58	x	x	x	x	x	x	x	x	0	1	0	0	0	1	0	0	$XX44
$YFFF5A	x	x	x	x	x	x	x	x	0	1	0	0	0	0	1	0	$XX42

PERIOD = $100, B_RISE/FALL_TIME_PTRs ⇒ channel4

12.2.3 Schematic



12.2.4 Output Waveforms



13 Multichannel PWM Algorithm

The transition times of all MCPWM channels are defined by the following equations:

Master Channel All Modes

MLH = LMHL + PERIOD
MHL = LMLH + PERIOD

Slave Channels — Edge-Aligned Mode

SLH = LMHL + PERIOD + HIGH_TIME
SHL = LMLH + PERIOD + HIGH_TIME

Slave Channels — Center-Aligned Mode

Non-inverted PWM

If HIGH_TIME > DEAD_TIME:

SALH = LMHL + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2
SAHL = LMLH + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2
SBLH = LMHL + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2 + (HIGH_TIME – DEAD_TIME)
SBHL = LMLH + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2 + (HIGH_TIME – DEAD_TIME)

If HIGH_TIME ≤ DEAD_TIME:

SALH = LMHL + PERIOD + PERIOD/2
SAHL = LMLH + PERIOD + PERIOD/2
SBLH = LMHL + PERIOD + PERIOD/2
SBHL = LMLH + PERIOD + PERIOD/2

Inverted PWM

If HIGH_TIME > DEAD_TIME:

SALH = LMLH + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2
SAHL = LMHL + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2
SBLH = LMHL + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2 + (HIGH_TIME – DEAD_TIME)
SBHL = LMLH + PERIOD + {PERIOD – (HIGH_TIME – DEAD_TIME)}/2 + (HIGH_TIME – DEAD_TIME)

If HIGH_TIME ≤ DEAD_TIME:

SALH = LMLH + PERIOD + PERIOD/2
SAHL = LMHL + PERIOD + PERIOD/2
SBLH = LMHL + PERIOD + PERIOD/2
SBHL = LMLH + PERIOD + PERIOD/2

where:

MLH = Time of next master channel rising transition.
MHL = Time of next master channel falling transition.
LMLH = Time of last master channel rising transition.
LMHL = Time of last master channel falling transition.
SLH = Time of next slave channel rising transition.
SHL = Time of next slave channel falling transition.
SALH = Time of next slave A channel rising transition.
SAHL = Time of next slave A channel falling transition.
SBLH = Time of next slave B channel rising transition.
SBHL = Time of next slave B channel falling transition.

The following description is provided as a guide only, to aid understanding of the function. The exact sequence of operations in microcode may be different from that shown, in order to optimize speed and code size. TPU microcode source listings for all functions in the TPU function library can be downloaded from the Freescale Freeware bulletin board. Refer to *Using the TPU Function Library and TPU Emulation Mode* (TPUPN00/D) for detailed instructions on downloading and compiling microcode.

The multichannel PWM function consists of seven states, which operate as described below. For clarity, reference is made to internal channel flag0 in the following descriptions. This is an internal TPU control bit that is not available to the user.

13.1 STATE1 — MINIT_MCPWM

This state, entered as a result of an HSR %11 issued by the CPU, configures the channel as an MCPWM master and starts generation of the master output wave form.

```
The channel pin is configured as an output
Pin state is set to high
The pin is configured to toggle on match
TPU internal channel flag0 is asserted
The latest TCR1 time is captured and stored in LAST_RISE_TIME
PERIOD_COUNT is incremented
If PERIOD_COUNT equals IRQ_RATE then
    An interrupt request is generated
    PERIOD_COUNT is reset to zero
A match is scheduled for time LAST_RISE_TIME + PERIOD
The state ends
```

13.2 STATE2 — SINIT_MCPWM

This state, entered as a result of an HSR %10 issued by the CPU, configures the channel as an MCPWM slave and starts generation of the slave output wave form.

```
The channel pin is configured as an output
Pin state is set to high
The pin is configured to toggle on match
TPU internal channel flag0 is negated
If this is an edge-aligned mode slave (indicated by Host Sequence bits = 00) then:
    a match is scheduled for time LAST_RISE_TIME + PERIOD + HIGH TIME
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is a center-aligned mode slave A (indicated by Host Sequence bits = 01) then:
    If HIGH_TIME ≥ DEAD_TIME a match is scheduled for time:
        LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2
    else a match is scheduled for time:
        LAST_RISE_TIME + PERIOD + PERIOD/2
    If HIGH_TIME ≥ DEAD_TIME the following time is stored in NXT_B_FALL_TIME:
        LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2 + (HIGH_TIME –
        DEAD_TIME)
    else the following time is stored in NXT_B_FALL_TIME:
        LAST_RISE_TIME + PERIOD + PERIOD/2
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is a center-aligned mode slave B (indicated by Host Sequence bits = 1X) then:
    a match is scheduled for the time contained in the location pointed to by B_FALL_TIME_PTR
    The state ends
```

13.3 STATE3 — S_INV_INIT_MCPWM

This state, entered as a result of an HSR %01 issued by the CPU, configures the channel as an MCPWM inverted slave and starts generation of the inverted slave output wave form. This state should only be entered on a slave A channel in center-aligned mode, an HSR %01 should not be issued to an edge-aligned slave or a slave B channel.

```
The channel pin is configured as an output
Pin state is set to high
The pin is configured to toggle on match
TPU internal channel flag0 is negated
```

```
If HIGH_TIME ≥ DEAD_TIME a match is scheduled for time:
    LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2
Else a match is scheduled for time:
    LAST_RISE_TIME + PERIOD + PERIOD/2
If HIGH_TIME ≥ DEAD_TIME the following time is stored in NXT_B_FALL_TIME:
    LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2 + (HIGH_TIME –
    DEAD_TIME)
Else the following time is stored in NXT_B_FALL_TIME:
    LAST_RISE_TIME + PERIOD + PERIOD/2
HIGH_TIME is stored in CURRENT_HIGH_TIME
The state ends
```

13.4 STATE4 — MLH_MCPWM

This state, entered as a result of a match with the pin high and flag0 set (master), continues generation of the master output wave form and periodic interrupts.

```
The event time of the last match is stored in LAST_RISE_TIME
PERIOD_COUNT is incremented
If PERIOD_COUNT equals IRQ_RATE then:
    An interrupt request is generated
    PERIOD_COUNT is reset to zero
A match is scheduled for time LAST_RISE_TIME + PERIOD
The state ends
```

13.5 STATE5 — MHL_MCPWM

This state, entered as a result of a match with the pin low and flag0 set (master), continues generation of the master output wave form and periodic interrupts.

```
The event time of the last match is stored in LAST_FALL_TIME
If PERIOD_COUNT equals IRQ_RATE then:
    An interrupt request is generated
    PERIOD_COUNT is reset to zero
A match is scheduled for time LAST_FALL_TIME + PERIOD
The state ends
```

13.6 STATE6 — SHL_MCPWM

This state, entered as a result of a match with the pin low and flag0 clear (slave), continues generation of the slave output wave form.

```
If this is an edge-aligned mode slave (indicated by Host Sequence bits = 00) then:
    A match is scheduled for time LAST_FALL_TIME + PERIOD + HIGH_TIME
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is a non-inverted center-aligned mode slave A (indicated by Host Sequence bits = 01) then:
    If HIGH_TIME ≥ DEAD_TIME a match is scheduled for time:
        LAST_FALL_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2
    Else a match is scheduled for time:
        LAST_FALL_TIME + PERIOD + PERIOD/2
    If HIGH_TIME ≥ DEAD_TIME the following time is stored in NXT_B_RISE_TIME:
        LAST_FALL_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2 +
        (HIGH_TIME – DEAD_TIME)
    Else the following time is stored in NXT_B_RISE_TIME:
        LAST_FALL_TIME + PERIOD + PERIOD/2
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is an inverted center-aligned mode slave A (indicated by Host Sequence bits = 01) then:
    If HIGH_TIME ≥ DEAD_TIME a match is scheduled for time:
            LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2
    Else a match is scheduled for time:
            LAST_RISE_TIME + PERIOD + PERIOD/2
    If HIGH_TIME ≥ DEAD_TIME the following time is stored in NXT_B_FALL_TIME:
            LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2 +
            (HIGH_TIME – DEAD_TIME)
    Else the following time is stored in NXT_B_FALL_TIME:
            LAST_RISE_TIME + PERIOD + PERIOD/2
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is a center-aligned mode slave B (indicated by Host Sequence bits = 1X) then:
    a match is scheduled for the time contained in the location pointed to by B_RISE_TIME_PTR
    The state ends
```

13.7 STATE7 — SLH_MCPWM

This state, entered as a result of a match with the pin high and flag0 clear (slave), continues generation of the slave output wave form.

```
If this is an edge-aligned mode slave (indicated by Host Sequence bits = 00) then:
    A match is scheduled for time LAST_RISE_TIME + PERIOD + HIGH_TIME
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is a non-inverted center-aligned mode slave A (indicated by Host Sequence bits = 01) then:
    If HIGH_TIME ≥ DEAD_TIME a match is scheduled for time:
        LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2
    Else a match is scheduled for time:
        LAST_RISE_TIME + PERIOD + PERIOD/2
    If HIGH_TIME ≥ DEAD_TIME the following time is stored in NXT_B_FALL_TIME:
        LAST_RISE_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2 + (HIGH_TIME –
        DEAD_TIME)
    Else the following time is stored in NXT_B_FALL_TIME:
        LAST_RISE_TIME + (2 ∗ PERIOD) – PERIOD/2
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is an inverted center-aligned mode slave A (indicated by Host Sequence bits = 01) then:
    If HIGH_TIME ≥ DEAD_TIME a match is scheduled for time:
        LAST_FALL_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2
    Else a match is scheduled for time:
        LAST_FALL_TIME + PERIOD + PERIOD/2
    If HIGH_TIME ≥ DEAD_TIME the following time is stored in NXT_B_RISE_TIME:
        LAST_FALL_TIME + PERIOD + {PERIOD – (HIGH_TIME –DEAD_TIME)}/2 + (HIGH_TIME –
        DEAD_TIME)
    Else the following time is stored in NXT_B_RISE_TIME:
        LAST_FALL_TIME + (2 ∗ PERIOD) – PERIOD/2
    HIGH_TIME is stored in CURRENT_HIGH_TIME
    The state ends
If this is a center-aligned mode slave B (indicated by Host Sequence bits = 1X) then:
    a match is scheduled for the time contained in the location pointed to by B_FALL_TIME_PTR
    The state ends
```

NOTES

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

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

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