MOTOROLA
SEMICONDUCTOR
PROGRAMMING NOTE

TPUPN11/D

Period/Pulse-Width Accumulator TPU Function (PPWA)

By Mark Maiolani and Sharon Darley

MOTOROLA
SEMICONDUCTOR
PROGRAMMING NOTE

Order this document
by TPUPN11/D

Period/Pulse-Width Accumulator TPU Function (PPWA)

by Mark Maiolani and Sharon Darley

1 Function Overview

The period/pulse-width accumulator (PPWA) algorithm allows any channel to accumulate the sum in TCR clock counts of up to 255 periods or pulse widths. The hexadecimal value that PPWA returns as the accumulated sum must be multiplied by the appropriate TCR resolution to obtain the actual amount of time in seconds. PPWA has four operating modes, and it can accumulate a 16-bit sum or a 24-bit sum. In the 16-bit modes, it can generate a link to a sequential block of up to eight channels after each accumulation period. In all modes, PPWA can generate an interrupt after each accumulation period.

2 Detailed Description

PPWA accumulates either periods or pulse widths of an input signal. The function is particularly useful in applications requiring frequency multiplication or division. A channel running the PPWA function can link to a channel running the output compare (OC) function to generate an output signal that is proportional to the input signal to the PPWA channel. See Motorola Programming Note TPUPN12/D, *Output Compare TPU Function (OC)*, and the examples at the end of this function note for further explanation.

The user specifies the number of accumulation periods in the parameter MAX_COUNT. During an accumulation, the TPU updates PERIOD_COUNT with the current number of accumulated periods. ACCUM is used as a software accumulator for the lower 16 bits of the period/pulse width, and PPWA_UB is used for the upper eight bits. Once an accumulation is completed (PERIOD_COUNT ≥ MAX_COUNT), the TPU updates PPWA_LW with ACCUM and then initializes ACCUM and PERIOD_COUNT to zero for the next accumulation. However, the TPU does not reinitialize PPWA_UB to zero; the user must do so.

If desired, the user can read the current, uncompleted period/pulse-width measurement at any time in the parameter ACCUM. ACCUM is updated at a periodic rate of 1/(ACCUM_RATE ∗ (resolution of match/capture TCR)) and also at each high-to-low input pin transition. If a running period/pulse-width value is not desired, ACCUM_RATE should be set to $FF. The accuracy of the completed period/pulse-width measurement, assuming the TPU can measure it, depends only on the resolution of the TCR.

PPWA can also serve as a 16-bit software period/pulse-width accumulator with programmable accuracy. ACCUM can be read periodically, and the difference in values represents the accumulated time between reads. For this use, MAX_COUNT should be set to its maximum value ($FF) and the CPU should periodically reset PERIOD_COUNT to zero so that ACCUM functions as a continuous modulo-16 accumulator.

PPWA has four modes of operation that are determined by the bits in the host sequence field. All four modes are continuous in operation. Two of the modes measure pulse periods; the other two measure pulse widths.

The following paragraphs describe each of the four modes. A detailed description of the PPWA algorithm, including a state diagram, is provided for reference at the end of this document.

2.1 Mode 0: Accumulate 24-Bit Periods, No Links

Mode 0 is selected by writing a value of%00 to the host sequence bits. In this mode, when a channel accumulates a programmable number of periods, it generates an interrupt request to notify the CPU. Whenever the current accumulation exceeds 16 bits and ACCUM overflows, the TPU increments PPWA_UB. However, since the TPU does not reset PPWA_UB to zero after an accumulation has completed, the CPU must do so before 2^{16} counts ($FFFF) are again collected in ACCUM in a subsequent accumulation. Otherwise, PPWA_UB will be incorrect for subsequent accumulations. No overflow indication is provided for the case when the sum of the periods exceeds 24 bits ($FFFFFF). CHANNEL_CONTROL should be configured to detect falling edges.

2.2 Mode 1: Accumulate 16-Bit Periods, Links

Mode 1 is selected by writing a value of%01 to the host sequence bits. In this mode, a channel accumulates a programmable number of periods, then generates a link to a sequential block of up to eight channels, and finally generates an interrupt request to notify the CPU. The sum of the periods may be as great as 16 bits ($FFFF). An interrupt and links are also generated if the sum of periods exceeds 16 bits. When using this mode, CHANNEL_CONTROL should be configured to detect falling edges.

The user specifies the first channel of a block of channels to be linked in START_LINK_CHANNEL and the number of channels in the block in LINK_CHANNEL_COUNT. The link to the channels indicated by these parameters happens under one of two conditions: 1) when an accumulation is completed, and 2) when an accumulation exceeds 16 bits. In the latter case, PPWA_LW is set to zero.

2.3 Mode 2: 24-Bit Pulse Widths, No Links

Mode 2 is selected by writing a value of%10 to the host sequence bits. In this mode, once a channel accumulates high-time pulse widths over a programmable number of periods, it generates an interrupt request to notify the CPU. Whenever the current accumulation exceeds 16 bits and ACCUM overflows, the TPU increments PPWA_UB. However, since the TPU does not reset PPWA_UB to zero after an accumulation has completed, the CPU must do so before 2^{16} counts ($FFFF) are again collected in ACCUM during a subsequent accumulation. Otherwise, PPWA_UB will be incorrect for subsequent accumulations. No overflow indication is provided for the case in which the sum of the pulse widths exceeds 24 bits ($FFFFFF). CHANNEL_CONTROL should be configured to detect rising edges.

2.4 Mode 3: 16-Bit Pulse Widths, Links

Mode 3 is selected by writing a value of%11 to the host sequence bits. In this mode, once a channel accumulates the high-time pulse widths over a programmable number of periods, it generates a link to a sequential block of up to eight channels and an interrupt request to notify the CPU. The sum of the pulse widths may be as great as 16 bits ($FFFF). An interrupt and links are also generated if the sum of pulse widths exceeds 16 bits. When using this mode, CHANNEL_CONTROL should be configured to detect rising edges.

The user specifies the first channel of a block of channels to be linked in START_LINK_CHANNEL and the number of channels in the block in LINK_CHANNEL_COUNT. The link to the channels indicated by these parameters happens under one of two conditions: 1) when an accumulation is completed, and 2) when an accumulation exceeds 16 bits. In the latter case, PPWA_LW is set to zero.

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. PPWA function code size is:

38 μ instructions + 5 entries = **43 long words**

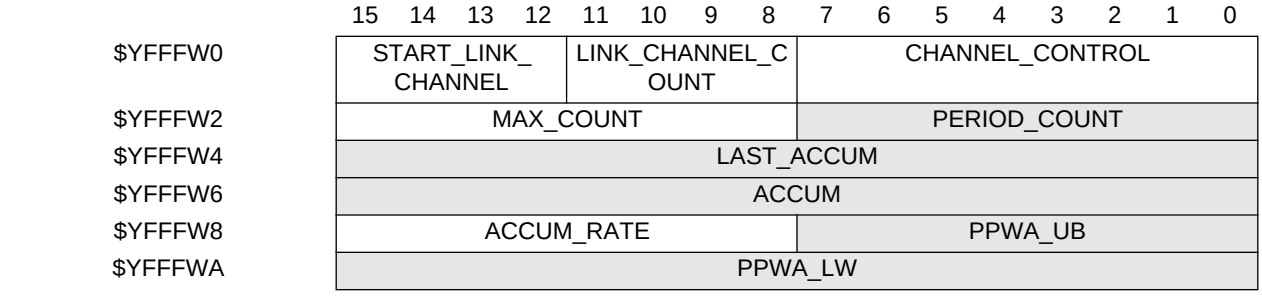
4 Function Parameters

This section provides detailed descriptions of function parameters stored in channel parameter RAM. **Figure 1** shows TPU parameter RAM address mapping. **Figure 2** 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 Number	Base Address	Parameter 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 1 TPU Channel Parameter RAM CPU Address Map



W = Channel number

Parameter Write Access

Written by CPU

Written by TPU

Figure 2 PPWA Function Parameter RAM Assignment

4.1 CHANNEL_CONTROL

CHANNEL_CONTROL is updated by the CPU with configuration data for channel control latches prior to initialization. This data is used to configure the initial input transition to be detected and the match/capture TCR. The following table defines the allowable data for this parameter.

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

Table 1 PPWA CHANNEL_CONTROL Options

TBS	PAC	PSC	Action
8 7 6 5	**4 3 2**	**1 0**	
		1 1	Do Not Force Any State
	0 0 1 0 1 0		Detect Rising Edge – Pulse-Width (High Time) Accumulate Detect Falling Edge – Period Accumulate
0 0 x x 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 1 x x x			**Input Channel** Capture TCR1, Match TCR1 Capture TCR1, Match TCR2 Capture TCR2, Match TCR1 Capture TCR2, Match TCR2 Do Not Change TBS

START_LINK_CHANNEL contains the first channel number of a block of sequential channels for linking in modes 1 and 3. The CPU updates this parameter.

4.2 START_LINK_CHANNEL

LINK_CHANNEL_COUNT specifies how many channels are linked in modes 1 and 3. The CPU updates this parameter.

4.3 LINK_CHANNEL_COUNT

LINK_CHANNEL_COUNT must be a value in the following range: $0 < \text{LINK\_CHANNEL\_COUNT} \leq 8$. No range check of LINK_CHANNEL_COUNT is performed by the TPU. If this parameter does not meet these limits in modes 1 and 3, the results are unpredictable.

4.4 MAX_COUNT

MAX_COUNT contains the number of periods over which the periods or pulse widths are accumulated. The number of accumulation periods is limited to 255. A value of either zero or one results in the accumulation of one period or pulse width. The CPU updates this parameter.

4.5 PERIOD_COUNT

PERIOD_COUNT contains the current number of periods over which the accumulation has accrued. The TPU updates this parameter.

4.6 LAST_ACCUM

LAST_ACCUM contains the time of the last update of ACCUM by the TPU due to a low-to-high input transition (in modes 2 and 3 only), a high-to-low input transition (in all modes), or the periodic update mechanism (see the following discussion of ACCUM_RATE).

4.7 ACCUM

ACCUM contains the current, uncompleted number of accumulated clock periods during the current accumulation period. The TPU updates this parameter as specified by ACCUM_RATE.

4.8 PPWA_UB

PPWA_UB contains the upper eight bits of the last completed accumulation of the period or pulse width. The weighting of the LSB of this parameter is equal to twice the weighting of the MSB of the TCR used for capture.

PPWA_UB is incremented by the TPU whenever ACCUM overflows. This parameter must be initialized to zero by the CPU prior to execution of initialization and after an accumulation of greater than 16 bits.

4.9 PPWA_LW

PPWA_LW contains the lower word of the previous accumulation of periods or pulse widths.

NOTE

Pulses that have a duration or separation that is less than the TPU latency time cannot be measured accurately by the PPWA function. Refer to **7 Performance and Use of Function** for information on calculating worst-case latency.

4.10 ACCUM_RATE

ACCUM_RATE determines the periodic rate at which ACCUM is updated to reflect the period/pulse width. This parameter is significant for two reasons:

1. ACCUM_RATE determines the periodic rate at which the PPWA channel is serviced. Consequently, ACCUM_RATE should be set to a value that does not result in a service rate exceeding performance requirements of other time functions in the user's system. The service rate for the PPWA channel is equal to the following:

$$\frac{1}{(\text{ACCUM\_RATE} \cdot (\text{resolution of the match TCR}))}$$

Therefore, the maximum value ($FF) results in the least frequent service rate.

2. ACCUM_RATE defines the accuracy of ACCUM with respect to the current, uncompleted, period/pulse-width accumulation. In other words, ACCUM_RATE defines how closely ACCUM reflects the actual period or pulse width.

NOTE

This parameter has no effect on the accuracy of PPWA_UB and PPWA_LW. Only the resolution of the capture TCR affects the completed accumulation.

For example, in using ACCUM as a 16-bit software accumulator, the pulse widths accumulate and the CPU reads ACCUM periodically. In addition, the actual cumulative pulse width between reads is greater than or equal to 10 ms. If ACCUM_RATE equals $14 and the resolution of the match/capture TCR equals 2 μs, the error in the pulse-width accumulation calculation over the interval is less than or equal to the error for ACCUM times two:

$$\leq 2 * (\text{ACCUM error})$$
$$\leq 2\ (20)\ (2^{-6} * 10 \text{ seconds} / 1^{-2} * 10$$
$$\leq + 0.80\%$$

The service rate is 25 kHz. The error is obviously more significant if the sampling time is small; therefore, using ACCUM as a continuous accumulator may not be acceptable in all applications.

5 Host Interface to Function

This section provides information concerning the TPU host interface to the function. **Figure 3** 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 3 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] — PPWA 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	Action Taken
00	Accumulate 24-bit periods, no links
01	Accumulate 16-bit periods, link to one to eight channels
10	Accumulate 24-bit pulse widths, no links
11	Accumulate 16-bit pulse widths, link to one to eight channels

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	Initialization
00	No Host Service (Reset Condition)
01	Not implemented
10	Initialize (Init)
11	Not implemented

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	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 configures the PPWA function as follows:

1. Writes CHANNEL_CONTROL, MAX_COUNT, and ACCUM_RATE to parameter RAM;
2. Writes START_LINK_CHANNEL, LINK_CHANNEL_COUNT to parameter RAM if operating in link mode (host sequence bits = x1);
3. Writes host sequence bits according to the mode of operation;
4. Issues an HSR%10 for initialization; and
5. Enables channel servicing by assigning a high, middle, or low priority.

The TPU executes initialization and starts accumulating the periods or pulse widths as specified by the host sequence bits. The CPU should monitor the HSR register until the TPU clears the service request bits to %00 before changing any parameters or before issuing a new service request to this channel.

Whenever the TPU completes an accumulation, an interrupt request is generated, and one or more channels may be linked, depending on the mode of operation and the parameters. If an accumulation of greater than 16 bits occurs, the host CPU must clear PPWA_UB to zero after reading it. (The TPU increments this parameter when ACCUM overflows.) Since ACCUM is initialized to zero by the TPU on completion of an accumulation, no write collision of PPWA_UB should occur since the CPU has sufficient time to write this parameter before the TPU can increment it again.

7 Performance and Use of Function

7.1 Performance

Like all TPU functions, PPWA 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 signals faster than this performance limit are applied, the PPWA function may miss edges and produce results inconsistent with normal operation. The following sections give details to help calculate the actual performance limits of the function, together with the likely results when these limits are exceeded.

7.1.1 TPU Input Filtering

All TPU channels configured as inputs have an associated synchronizer and digital filter that samples pin transitions. These filter out high and low pulse widths less than two system clocks and are only guaranteed to pass pulses with a period greater than four system clocks. This operation must always be considered when operating the TPU with short duration input pulses.

7.1.2 Operational Overview

To understand the performance limitations of the PPWA function, it is necessary to know what TPU microcode is executed at what times. By calculating the time required to execute the microcode and taking into account other TPU latency considerations, one can make an estimate of the fastest reliable operation.

7.1.2.1 PPWA Operations and PPWA State Timing

To simplify the explanation of PPWA timing, this section on PPWA performance describes PPWA function in terms of the operation being performed, such as update processing, rather than in terms of the state being executed. To enable the reader to look up the operation in the PPWA State Timing Table or correlate the operation to the algorithm description later in this programming note, the state is also provided in parentheses.

The PPWA state timing table that follows lists the CPU clock cycles and RAM accesses for each PPWA state, according to the operation being performed. The table is referred to in subsequent paragraphs. The number of clock cycles listed are maximum service times, not including time slot transition times.

Table 2 Period/Pulse-Width Accumulator — State Timing

State	PPWA Operation	CPU Clock Cycles	RAM Accesses
S1 Init	Initialize (All Modes)	6	2
S2 First_H_L	Start Edge - Period Mode (Modes 0 and 1)	6	2
S3 High_Time_Begin	Start Edge - Pulse Mode (Modes 2 and 3)	6	2
S4 Accum_Low_Pin	Period End/Start Mode 0 No 16-bit overflow 16-bit overflow[1] Mode 1[2,3] Pulse End Mode 2 No 16-bit overflow 16-bit overflow[1] Mode 3[2,3] Periodic Update (on match while pin is low)[4] Mode 0 Mode 1	 42 34 + 42 50 42 34 + 42 50 34 26	 10 8 + 10 12 9 7 + 9 11 8 6
S5 Accum_High_Pin	Periodic Update (on match while pin is high)[4] Modes 0 and 2 Modes 1 and 3	 34 26	 8 6

1. Processing of 16-bit overflow requires that S4 be executed twice. Because the microcode instructions branch differently, depending on whether or not 16-bit overflow needs to be processed, is currently being processed, or does not need to be processed, the number of clock cycles varies with each execution of the state. In this case, the first and second executions of S4 require 34 and 42 clock cycles, respectively. In addition, TPU overhead and latency must be taken into account, as well as the possibility of another channel being serviced between the two executions of S4. Refer to **7.2 Overflow and Update Processing** to determine if these timings must be included.
2. Assumes one channel linked. Add two clocks for each additional channel linked.
3. Clock cycles/RAM accesses shown for operation without 16-bit overflow error condition.
4. Periodic update processing occurs in state 4 on a match while the pin is low, or in state 5 on a match while the pin is high; the timing is identical for both cases. (In the latter case, the microcode jumps to the update routine in state 4.)

7.1.2.2 Period Measurement (Modes 0 and 1)

The TPU first performs an initialization sequence after the PPWA function has been configured and enabled by the CPU. After initialization, the first falling edge causes the function to begin start-edge processing (S2). Each subsequent falling edge causes period end/start processing (S4). TPU processing to periodically update the ACCUM parameter, at the rate set in ACCUM_RATE, may also be performed but is not shown.

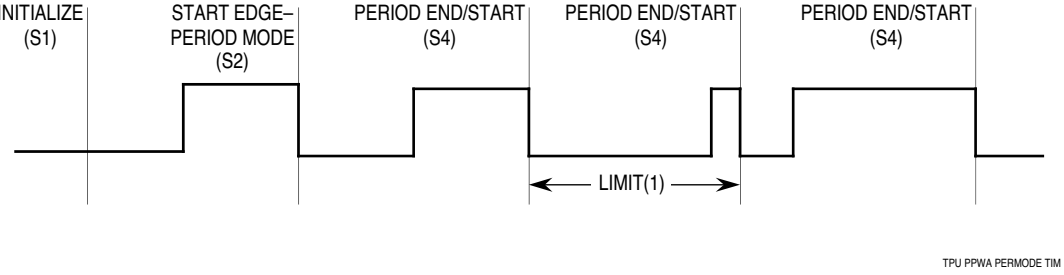


Figure 4 Periodic Mode Timing

When PPWA measures periods, it measures from falling edge to falling edge. Consequently, the input signal limitation is the period between these edges. This limitation is shown as Limit(1) in **Figure 4**. As long as the TPU input filter conditions are met, the duty cycle does not affect this mode of operation.

7.1.2.3 Pulse Measurement (Modes 2 and 3)

The main TPU microcode execution associated with pulse measurement is shown in **Figure 5**. The TPU first performs an initialization sequence after the PPWA function has been configured and enabled. After this, each rising edge initiates start-edge (S3) processing, and each falling edge initiates pulse-end (S4) processing. TPU processing to periodically update the ACCUM total, at the rate set in ACCUM_RATE, may also be performed but is not shown.

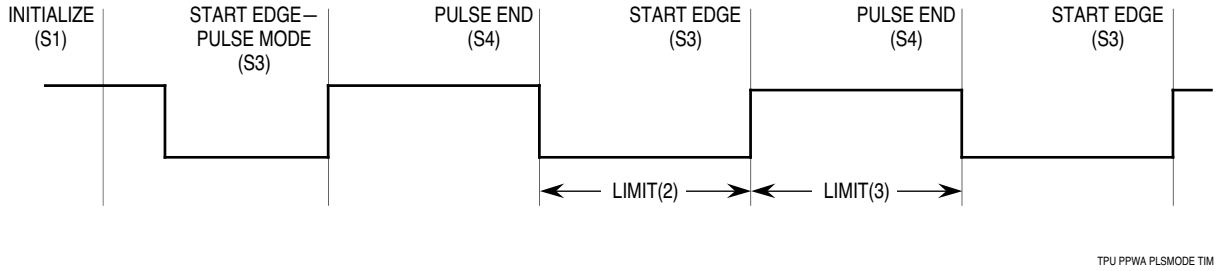


Figure 5 Pulse Mode Operation

As shown in **Figure 5**, when PPWA measures pulse widths, it begins edge processing with the rising edge and completes processing with the falling edge. Here, duty cycle does affect operation, and both the low and high times must have a minimum duration to guarantee correct operation. These are shown as Limit(2) and Limit(3) in **Figure 5**.

7.2 Overflow and Update Processing

As shown in the PPWA State Timing table, edges that generate 16-bit overflows result in additional processing time during period-end and pulse-end processing. The periodic update operation also involves additional processing time. Depending on the PPWA application, either or both of these operations need to be taken into account when calculating maximum PPWA performance. If the PPWA function is known to be used in such a way that periodic updates and/or 16-bit overflows do not occur, then a higher maximum performance can be guaranteed for the function.

If the programmed periodic update period is longer than the longest input period, then update processing will never be performed and does not need to be included in performance calculations. To determine whether overflow processing should be included, the maximum value for the accumulated periods or pulses, PPWA_UB/LW, should be determined. If this value exceeds $FFFF TCR counts, then overflow processing should be included when calculating performance limits. Note that the PPWA State Timing table gives timings for 16-bit overflows occurring in the 24-bit modes only, as an overflow in a16-bit mode is an error condition that does not result in a valid measurement.

7.3 Period Mode Limit Calculation

Figure 6 shows a worst-case example for period accumulation. TPU processing required is shown as shaded areas. The worst-case time slot latencies, shown as WCTSL, are the calculated worst case times from the time that the PPWA channel requests service until the time that it actually begins service. Because the period accumulation terminated by the first falling edge results in a 16-bit overflow, and a periodic update begins service shortly before this edge, both periodic update and overflow processing have to be completed before the next falling edge. Note that only a single worst case time slot latency period is required during the following period. Note also that the extra overflow and update processing determines the minimum period of the following pulse, which may or may not itself generate an overflow or periodic updates.

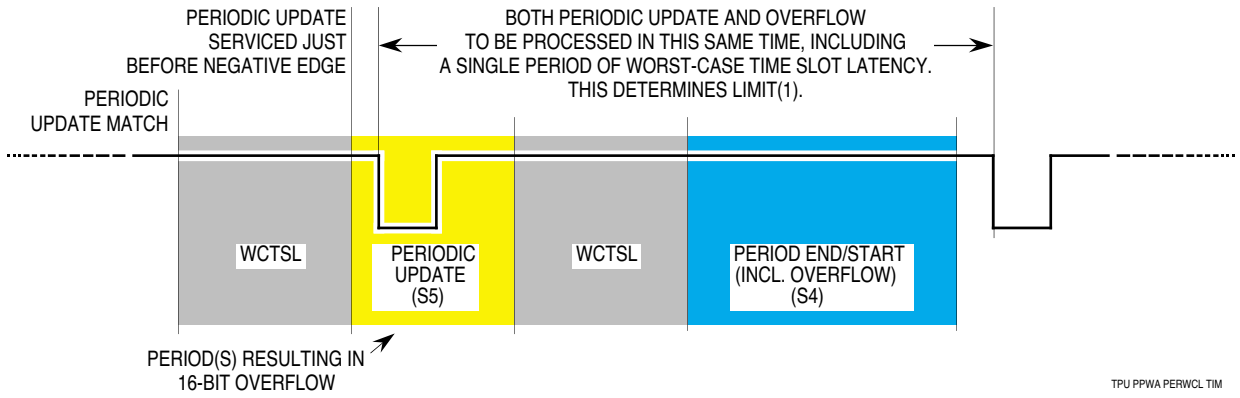


Figure 6 Period Mode Worst Case

For period accumulation, update processing needs to be included in the minimum period/Limit(1) calculations only if T_Update < Max_Period, where Max_Period is the known maximum input period between negative edges, and T_Update is the update period.

Following are calculations to determine Limit 1. Here, T_Update = ACCUM_RATE ∗ TCR timebase.

If T_Update < Max_Period then
Limit(1) = WCTSL + Period End/Start (S4) processing time + Update (S4 or S5) processing time

Else
Limit(1) = WCTSL + Period End/Start (S4) processing time

The maximum possible PPWA_UB/LW result should be calculated to determine whether overflow processing should be included. If the maximum PPWA_UB/LW is greater than $FFFF, then the Period End/Start (S4) timings including overflows should be used in the above calculation.

7.4 Example of Period Mode Limit Calculation

In this example, PPWA is being used with the following configuration:

- 24-bit period accumulation (mode 0)
- MAX_COUNT = 4 (accumulation of four pulses)
- TPU RAM collision rate (RCR) = 0.1. The RAM collision rate is an approximation of the percentage of time the CPU and the TPU try to access the RAM simultaneously.
- ACCUM_RATE = $FF
- Selected TCR period = 238.4 ns
- CPU clock speed = 16.778 MHz
- No other TPU channels operating
- Maximum input period (Max_Period) between falling edges = 200 us.

The following calculations determine the input signal limitations (in this case, limit 1):

1. Determine if periodic updates should be included in the calculation for input signal limitations:

   T_Update = ACCUM_RATE ∗ TCR period = $FF ∗ 238.4 ns = **60.792 µs**

   T_Update is less than Max_Period. Thus, periodic updates need to be included.

2. Determine if overflow processing should be included in the calculation for input signal limitations:

   Maximum possible PPWA_UB/LW = MAX_COUNT ∗ Max_Period = 4 ∗ 200 µs = 800 µs

   Maximum number of TCR counts = 800 µs/238.4 ns = **$0D1C** TCR counts

Maximum possible accumulation count in PPWA_UB/LW is less than $FFFF, so overflow processing is not required.

3. Calculate worst case latency (WCTSL):

WCTSL = time slot transition time + 4 clock NOP = 10 CPU clks + 4 CPU clks = **14 CPU clks**

There is a 10-clock delay between servicing time slots. In addition, there is one four-clock NOP between the time that all of the channels requesting service on a particular priority level are granted service and the next time slot transition.

4. Calculate Period End/Start (S4) processing time (Refer to the PPWA State Timing Table for the number of clock cycles and RAM accesses).

S4 = Num of Clock Cycles + (Num of RAM Accesses ∗ RCR ∗ 2) = 42 + (10 ∗ 0.1 ∗ 2) = **44 CPU clks**

5. Calculate Update (S4 or S5) processing time (Refer to the PPWA State Timing Table for the number of clock cycles and RAM accesses):

Update Processing Time = Num of Clock Cycles + (Num of RAM accesses ∗ RCR ∗ 2)
= 34 + (8 ∗ 0.1 ∗ 2) = **35.6 CPU clks**

6. Substitute previously calculated values into the calculation for the Limit(1) input signal limitation:

Limit 1 = WCTSL + Period End/Start(S4) processing time + Update (S4 or S5) processing time
= 14 + 44 + 35.6 = 93.6 CPU clocks = **5.57 μs**

7.5 Pulse Mode Limit Calculation

Figure 7 and **Figure 8** show the corresponding worst cases for pulse mode operation. **Figure 7** shows the worst case timings to determine the minimum allowable low time (limit(2)), while **Figure 8** shows the corresponding timings for the minimum high time (limit(3)).

During pulse accumulation, processing is required for both rising and falling input edges, thus limiting the minimum low and high times of the input signal. Each falling edge of the input signal must be handled by pulse end processing, as shown in **Figure 7** and detailed in the PPWA State Timing table. This defines the minimum low time (limit(2)). If the update period is less than the maximum input high time, Max_High, then periodic update processing time must also be included. In the following equations, T_Update = ACCUM_RATE ∗ TCR timebase.

If T_Update < Max_High then
Limit(2) = WCTSL + Pulse End (S4) processing time + Update (S5) processing time

Else
Limit(2) = WCTSL + Pulse End (S4) processing time

Determining whether overflow processing should be included in the previous calculation requires calculation of the maximum possible PPWA_UB/LW result. If this result is greater than $FFFF, then the Pulse End (S4) timings including overflows should be used in the previous calculation.

Rising edges are handled by start edge (S3) processing as shown in **Figure 8**. This defines the minimum high time, limit(3). Periodic update processing time must be included if the update period is less than the maximum period between positive edges, Max_Period 2.

If T_Update < Max_Period 2 then
Limit 3 = WCTSL + Start Edge (S3) processing time + Update (S5) processing time

Else
Limit 3 = WCTSL + Start Edge (S3) processing time

As start edge (S3) processing time is unaffected by overflows in the PPWA_UB/LW result, overflows do not affect limit(3) values.

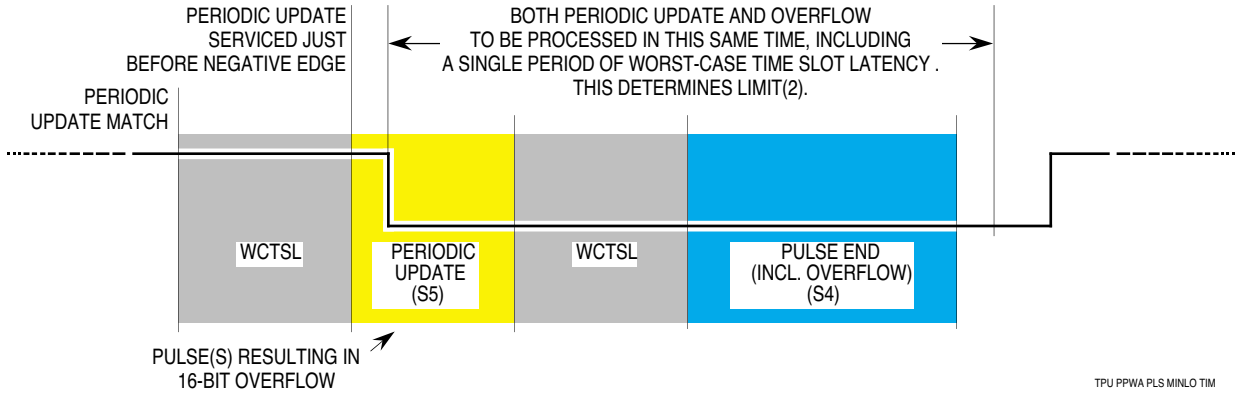


Figure 7 PPWA Minimum Low Time

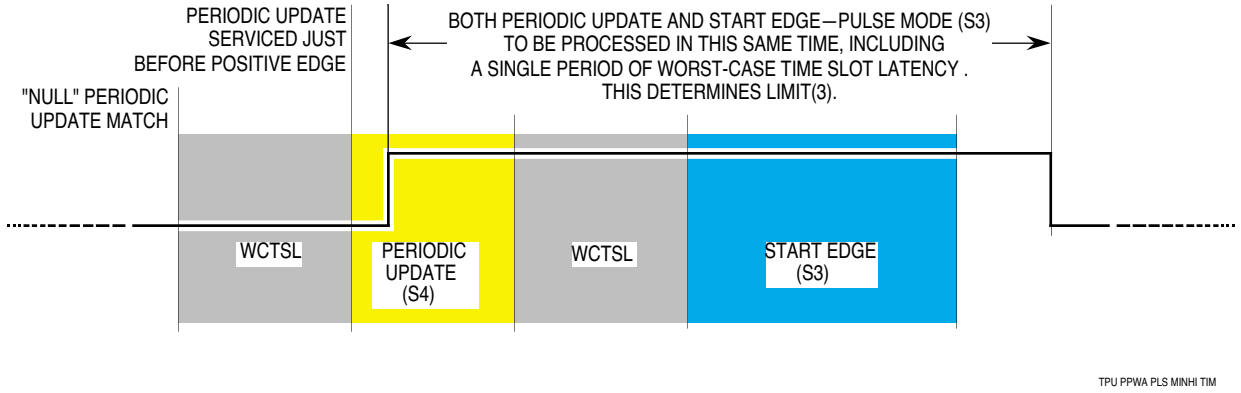


Figure 8 PPWA Minimum High Time

7.6 Example of Pulse Mode Limit Calculations

In this example, PPWA is being used with the following configuration:

- 16-bit pulse width accumulation with links (mode 3)
- PPWA links to two output compare (OC) channels
- MAX_COUNT = 1 (accumulate one pulse width)
- TPU RAM collision rate (RCR) = 0.1
- ACCUM_RATE = $10
- Maximum input period, Max_Period 2, between rising edges = 100 µs
- Maximum input high time, Max_High, = 80 µs
- selected TCR period = 238.4 ns
- PPWA operating at high priority
- 2 OC channels operating at low priority
- CPU clock speed is 16.78 MHz

The following calculations determine the input signal limitations (Limit(2) and Limit(3)):

1. Determine if periodic updates should be included in the Limit(2) and Limit(3) calculations:

T_Update = ACCUM_RATE ∗ TCR timebase = $10 ∗ 238.4 ns = 16 ∗ 238.4 ns = **3.8144 µs**

T_Update < Max_Period 2, and T_Update < Max_High. Thus, periodic updates need to be included in both Limit(2) and Limit(3) calculations.

2. No overflow processing is required in the Limit(2) and Limit(3) calculations, since PPWA is configured in 16-bit mode.
3. Calculate the worst case time-slot latency (WCTSL). Here, it is possible for the OC function to be serviced between two PPWA time slots. Thus, the service time for the OC function must be included in the worst-case time-slot latency calculations along with the standard 10-clock time-slot transition time and 4-clock NOP.

Service time for OC function = Num of clocks + (Num of RAM accesses ∗ RCR ∗ 2)
= 32 + (6 ∗ 0.1 ∗ 2) = 33.2 CPU clocks

WCTSL = TST + NOP + single OC service time + TST
= 10 CPU clks + 4 CPU clks + 33.2 CPU clks + 10 CPU clks = **57.2 CPU clks**

WCTSL is determined by the number of functions running, their types and priorities. There is a 10-clock delay between servicing time slots (shown as TST in calculation). In addition, there is one four-clock NOP between the time that all of the channels requesting service on a particular priority level are granted service and the next time slot transition. The service time shown for the OC function is that of state S2, Offset_Cal, for links other than the first after initialization.

4. Calculate the pulse end (S4) processing time. Refer to the PPWA State Timing Table for the number of clock cycles and RAM accesses. Remember that two clock cycles are added to the value in the table for each additional channel that is linked.

S4 = clock cycles + (Num of RAM accesses ∗ RCR ∗ 2) = 52 + (11 ∗ 0.1 ∗ 2) = **54.2 CPU clks**

5. Calculate the update (S5) processing time:

S5 = clock cycles + (Num of RAM accesses ∗ RCR ∗ 2) = 26 + (6 ∗ 0.1 ∗ 2) = **27.2 CPU clks**

6. Calculate the start edge (S3) processing time:

S3 = clock cycles + (Num of RAM accesses ∗ RCR ∗ 2) = 6 + (2 ∗ 0.1 ∗ 2) = **6.4 CPU clks**

7. Substitute these calculated values into the Limit(2) and Limit(3) calculations:

Limit(2) = WCTSL + S4 + S5 = 57.2 + 54.2 + 27.2 = 138.6 CPU clocks = **8.26 μs**

Limit(3) = WCTSL + S3 + S5 = 57.2 + 6.4 + 27.2 = 90.8 CPU clocks = **5.41 μs**

Minimum pulse low time is 8.26 μs, and the minimum pulse high time is 5.41 μs.

7.7 Single Fast Pulse Handling

The PPWA limits calculated above are all based on the PPWA function measuring sustained repetitive high speed signals. Pulses or periods faster than these limits may be measured if they do not occur repetitively, but are instead separated by pulses/periods much slower than the calculated limits.

7.8 Typical Error Results

When the PPWA function attempts to measure input signals that do not meet the limitations, the resultant measurement may not be as expected. In period accumulation mode, applying a signal which has faster periods than the minimum calculated as Limit 1 may result in edges being missed by the function. This will result in two or more periods being recognized as a single period, as shown in **Figure 9**.

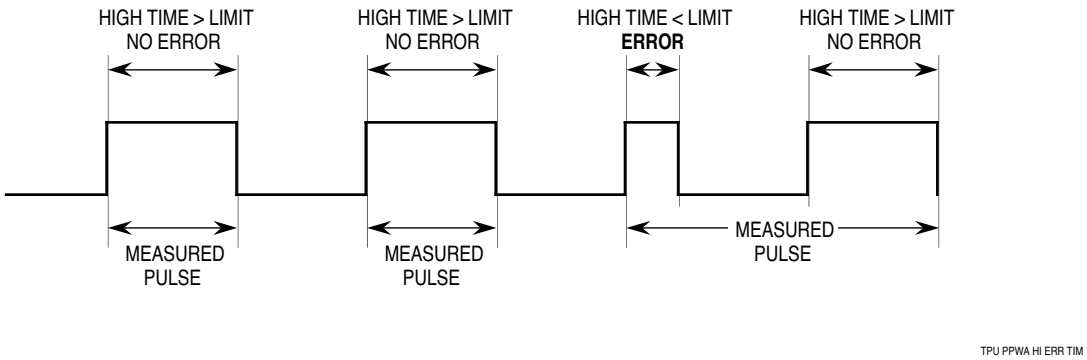


Figure 9 Period Mode Error Example

In pulse accumulation mode, attempting to measure a signal with a high time less than the specified minimum, Limit(3), may result in the falling edge being missed. This will return a pulse measurement from the first rising edge to the first captured falling edge as shown in **Figure 10**.

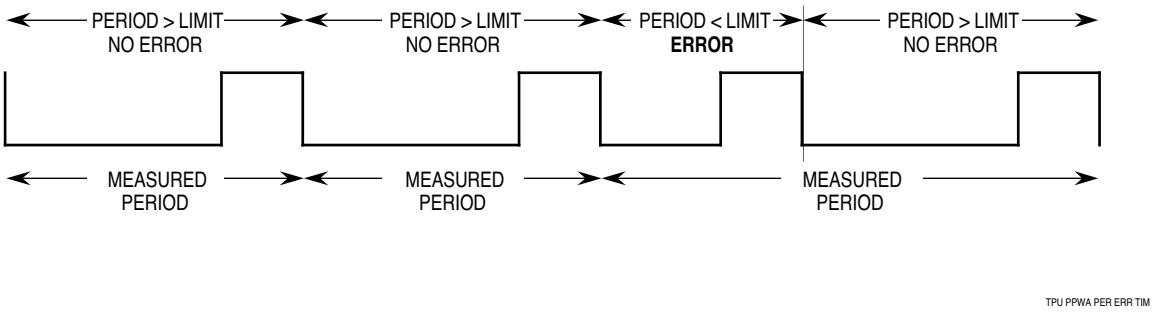


Figure 10 High Error Example

Attempting to measure a signal that has a low time less than Limit(2) can have either of two results:

1. The following pulse may be missed, with no measurement made, as in **Figure 11a**.
2. The next measurement may be made from the falling edge at the start of the short low time to the next recognized falling edge, as in **Figure 11b**.

Extra edges may be missed if the input signal has more than one consecutive violation of a timing limitation. An example is shown in **Figure 11c**, where a fast high time follows a fast low time. In this case, the falling edge after the fast high pulse is missed, so the pulse measurement continues until the next falling edge.

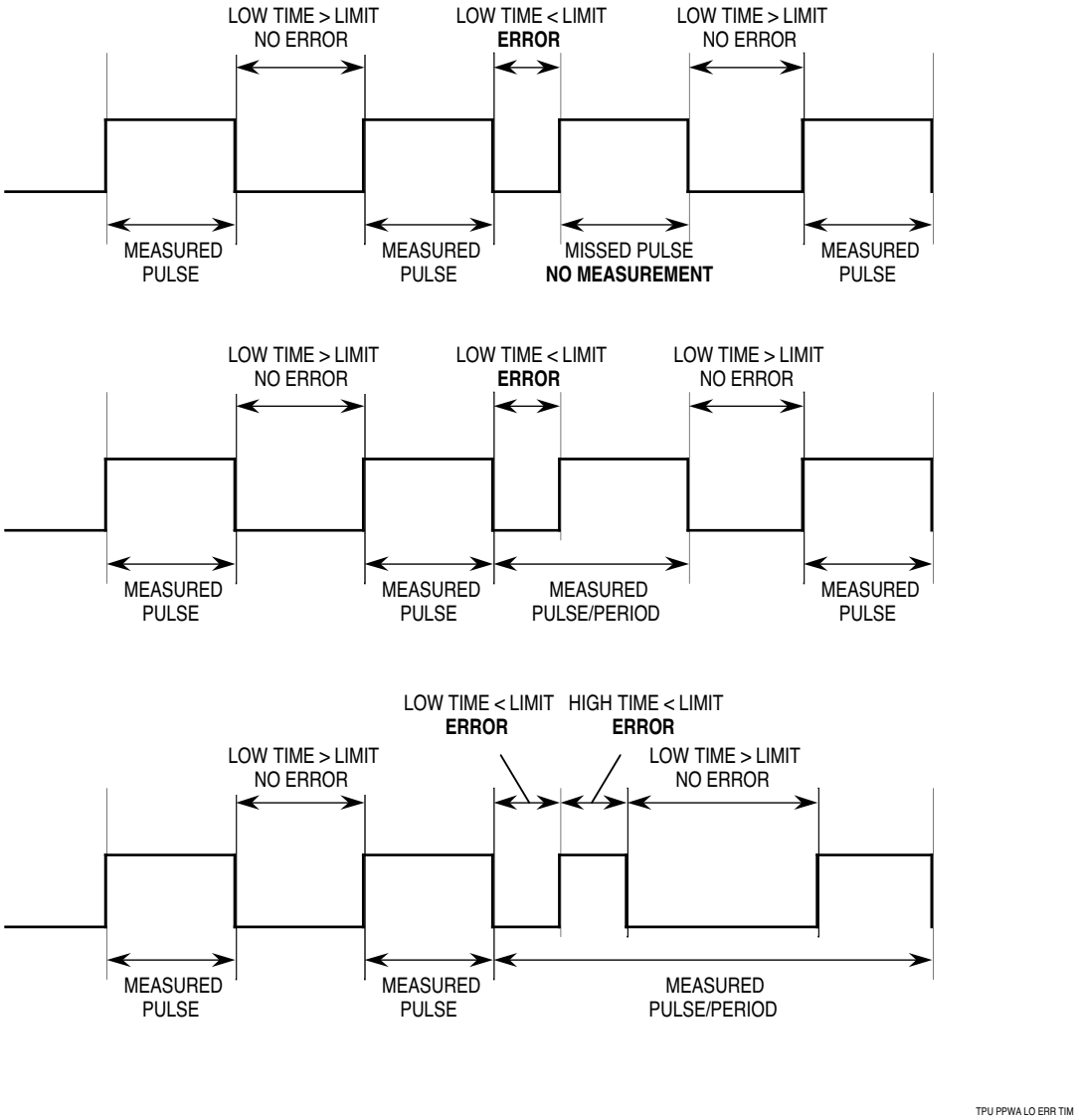


Figure 11 Low-Time Error Examples

7.9 Changing Mode

The host sequence bits are used to select PPWA function operating mode. Change host sequence bit values only when the function is stopped or disabled (channel priority bits = %00). Disabling the channel before changing mode avoids conditions that cause indeterminate operation.

8 Function Examples

8.1 Example A

8.1.1 Description

This example uses the PPWA function with the output compare (OC) function to multiply PPWA input frequency by two. The PPWA function repeatedly accumulates one input period and then generates a link to the OC function. The OC function scales the accumulated period and then generates the scaled output. The PPWA function uses channel 0, and the OC function uses channel 3. Both outputs are 50% duty cycle.

8.1.2 Initialization

The PPWA function is assigned to channel 0 so that it has higher priority than the OC function. It is set up for mode 1, accumulate 16-bit periods with links. Its parameters are initialized as follows:

START_LINK_CHANNEL is set to 3, since the OC function is on channel 3.

LINK_CHANNEL_COUNT is set to 1, since there is one channel in the link block.

CHANNEL_CONTROL is set to $0B. This sets the TBS field to capture and match TCR1, the PAC field to detect falling edge for period accumulate, and the PSC field to not force any state.

MAX_COUNT is the number of periods per accumulation. In this case it is one.

ACCUM_RATE determines how often ACCUM is updated during accumulations. Since a running accumulation is not desired, this parameter is set to the slowest rate possible, $FF.

The host sequence field bits are set to %01, accumulate 16-bit periods with links. The host service request bits are set to %10, initialization.

Load parameter RAM as shown.

Table 3 PPWA Channel Parameter RAM

$FFFF00	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	1
$FFFF02	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
$FFFF04	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF06	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF08	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$FFFF0A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

The OC function is set up on channel 3 in continuous pulse mode. Its parameters are initialized as follows:

CHANNEL_CONTROL is set to $8A. This captures and matches TCR1, forces the pin low on a match, and forces the initial pin level low. It configures the pin level response such that the input wave and output wave are in phase.

RATIO is an 8-bit fractional number between 0 and $FF used to scale the value indicated by REF_ADDR2 to form the output pulse hightime. Here, REF_ADDR2 points to PPWA_LW. Thus, for the input PPWA channel, PPWA_LW represents a period accumulation, while for the output OC channel, it represents the unscaled output pulse hightime. The following equation relates the accumulation value in PPWA_LW, the desired output period, and RATIO:

$$T_o = 2 * A_i * (RATIO / 255) \text{ seconds}$$

In this equation, A_i is the accumulation value in PPWA_LW multiplied by the resolution of the timer for the input channel, and T_o is the period of the output pulse multiplied by the resolution of the timer for the output channel. The factor of 2 is included since the OC channel scales the value in PPWA_LW to be the output pulse hightime instead of the output pulse period. Thus, the scaled output pulse hightime must be multiplied by two in order to calculate the output pulse period.

Solving this equation for RATIO yields the following:

$$RATIO = (T_o / A_i) * (255 / 2)$$

In this example, the desired ratio of the output period to the input period is 1/2. Since the value accumulated in PPWA_LW (A_i)represents one input period, T_o/A_i is also 1/2. Thus, RATIO = 1/2 * 255/2 = (approximately) 64 = $40.

REF_ADDR1 points to a synchronization reference value used whenever a link is received that is not the first link after initialization. Here, it points to LAST_ACCUM.

REF_ADDR2 points to a reference value used in calculating the output pulse hightime. Here, it points to PPWA_LW.

REF_ADDR3 points to a synchronization reference value used when the first link service request is serviced after initialization.

The host sequence field bits are initialized to %00, matches and pulses scheduled. The host service request bits are set to %11, initialization for the continuous mode.

Load parameter RAM as shown.

Table 4 OC Channel Parameter RAM

$FFFF30	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
$FFFF32	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF34	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
$FFFF36	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0
$FFFF38	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF3A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

8.2 Example B

8.2.1 Description

This example also uses the PPWA function with the OC function to multiply the PPWA input frequency by two. In this example, however, the PPWA function repeatedly accumulates two input pulse high times and then generates a link to the OC function. The OC function scales the accumulated high times and then generates the scaled output. The PPWA function uses channel 0, and the OC function uses channel 3. Both outputs are 50% duty cycle.

8.2.2 Initialization

The PPWA function is assigned to channel 0 so that it has higher priority than the OC function. It is set up for mode 3, accumulate 16-bit pulse widths with links. Its parameters are initialized as followed:

START_LINK_CHANNEL is set to 3, since the OC function is on channel 3.

LINK_CHANNEL_COUNT is set to 1, since there is one channel in the link block.

CHANNEL_CONTROL is set to $07. This sets the TBS field to capture and match TCR1, the PAC field to detect rising edge for pulse accumulation, and the PSC field to not force any state.

MAX_COUNT is the number of pulse high times per accumulation. In this case it is two.

ACCUM_RATE determines how often ACCUM is updated during accumulations. Since a running accumulation is not desired, this parameter is set to the slowest rate possible, $FF.

The host sequence field bits are set to %11, accumulate 16-bit pulse widths with links. The host service request bits are set to %10, initialization.

Load parameter RAM as shown.

Table 5 PPWA Channel Parameter RAM

$FFFF00	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	1
$FFFF02	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
$FFFF04	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF06	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF08	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$FFFF0A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

The OC function is set up on channel 3 in continuous pulse mode. Its parameters are initialized as follows:

CHANNEL_CONTROL is set to $008A. This captures and matches TCR1, forces the pin low on a match, and forces the initial pin level low.

RATIO is an 8-bit fractional number between 0 and $FF used to scale the value indicated by REF_ADDR2 to form the output pulse hightime. Here, REF_ADDR2 points to PPWA_LW. Thus, for the input PPWA channel, PPWA_LW represents a period accumulation, while for the output OC channel, it represents the unscaled output pulse hightime. The following equation relates the input pulse hightime accumulation in PPWA_LW, the desired output period, and RATIO:

$$T_o = 2 * A_i * (RATIO/255) \text{ seconds}$$

In this equation, A_i is the accumulation value in PPWA_LW multiplied by the resolution of the timer for the input channel, and T_o is the period of the output pulse multiplied by the resolution of the timer for the output channel. The factor of 2 is included since the OC channel scales the value in PPWA_LW to be the output pulse hightime instead of the output pulse period. Thus, the scaled output pulse hightime must be multiplied by two in order to calculate the output pulse period.

Solving this equation for RATIO yields the following:

$$RATIO = (T_o / A_i) * (255/2)$$

In this example, the desired ratio of the output period to the input period is 1/2. Thus, the desired ratio of the output period to a single input pulse high time is 1/1. Since the value accumulated in PPWA_LW represents two input high times, A_i equals the accumulation of two input pulse high times, and T_o/A_i equals 1/2. Thus, RATIO = $T_o/A_i * 255/2 = 1/2 * 255/2 =$ (approximately) 64 = $40.

REF_ADDR1 points to a synchronization reference value used whenever a link is received that is not the first link after initialization. Here, it points to LAST_ACCUM.

REF_ADDR2 points to a reference value used in calculating the output pulse hightime. Here, it points to PPWA_LW.

REF_ADDR3 points to a synchronization reference value used when the first link service request is serviced after initialization.

The host sequence field bits are initialized to %0x, matches and pulses scheduled. The host service request bits are set to %11, initialization for the continuous mode.

Load parameter RAM as shown.

Table 6 OC Channel Parameter RAM

$FFFF30	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
$FFFF32	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF34	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
$FFFF36	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0
$FFFF38	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF3A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

8.3 Example C

8.3.1 Description

This example uses the PPWA function with the OC function to divide PPWA input frequency by two. The PPWA function repeatedly accumulates eight input periods and then generates a link to the OC function. The OC function scales the accumulated period and then generates the scaled output waveform. The PPWA function uses channel 0, and the OC function uses channel 3. Both outputs are 50% duty cycle.

8.3.2 Initialization

The PPWA function is assigned to channel 0 so that it has higher priority than the OC function. It is set up for mode 1, accumulate 16-bit periods with links. Its parameters are initialized as followed:

START_LINK_CHANNEL is set to 3, since the OC function is on channel 3.

LINK_CHANNEL_COUNT is set to 1, since there is one channel in the link block.

CHANNEL_CONTROL is set to $0B. This sets the TBS field to capture and match TCR1, the PAC field to detect falling edge for period accumulate, and the PSC field to not force any state.

MAX_COUNT is the number of periods per accumulation. In this case it is eight.

ACCUM_RATE determines how often ACCUM will be updated during accumulations. Since a running accumulation is not desired, this parameter is set to the slowest rate possible, $FF.

The host sequence field bits are set to %01, accumulate 16-bit periods with links. The host service request bits are set to %10, initialization.

Load parameter RAM as shown.

Table 7 PPWA Channel Parameter RAM

$FFFF00	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	1
$FFFF02	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$FFFF04	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF06	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF08	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$FFFF0A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

The OC function is set up on channel 3 in continuous pulse mode. Its parameters are initialized as follows:

CHANNEL_CONTROL is set to $8A. This captures and matches TCR1, forces the pin low on a match, and forces the initial pin level low.

RATIO is an 8-bit fractional number between 0 and $FF used to scale the value indicated by REF_ADDR2 to form the output pulse hightime. Here, REF_ADDR2 points to PPWA_LW. Thus, for the input PPWA channel, PPWA_LW represents a period accumulation, while for the output OC channel, it represents the unscaled output pulse hightime. The following equation relates the input pulse hightime accumulation in PPWA_LW, the desired output period, and RATIO:

$$T_o = 2 * A_i * (RATIO/255) \text{ seconds}$$

In this equation, A_i is the accumulation value in PPWA_LW multiplied by the resolution of the timer for the input channel, and T_o is the period of the output pulse multiplied by the resolution of the timer for the output channel. The factor of 2 is included since the OC channel scales the value in PPWA_LW to be the output pulse hightime instead of the output pulse period. Thus, the scaled output pulse hightime must be multiplied by two in order to calculate the output pulse period.

Solving this equation for RATIO yields the following:

$$RATIO = (T_o / A_i) * (255/2)$$

In this example, the desired ratio of the output period to the input period is 2/1. Since the accumulated value in PPWA_LB represents eight input periods, $T_o/A_i = 2/1 * 1/8 = 1/4$. Thus, $T_o/A_i * 255/2 = 1/4 * 255/2$ = (approximately) 32 = $20.

REF_ADDR1 points to a synchronization reference value used whenever a link is received that is not the first link after initialization. Here, it points to LAST_ACCUM.

REF_ADDR2 points to a reference value used in calculating the output pulse hightime. Here, it points to PPWA_LW.

REF_ADDR3 points to a synchronization reference value used when the first link service request is serviced after initialization.

The host sequence field bits are initialized to %0x, matches and pulses scheduled. The host service request bits are set to %11, initialization for the continuous mode.

Load parameter RAM as shown.

Table 8 OC Channel Parameter RAM

$FFFF30	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
$FFFF32	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF34	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
$FFFF36	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0
$FFFF38	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
$FFFF3A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

9 Function Algorithm

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 Motorola Freeware bulletin board. Refer to *Using the TPU Function Library and TPU Emulation Mode* (TPUPN00/D) for detailed instructions regarding downloading and compiling microcode.

The PPWA function consists of five states. For clarity, reference is made in the state descriptions to internal channel flags 0 and 1. Although the CPU has no access to the channel flags, a description of their use aids in understanding the following state descriptions.

Flag0 is used as an internal state indicator and is negated in modes 2 and 3 whenever a low-to-high input transition occurs. It is asserted only in modes 2 and 3 whenever a high-to-low input transition occurs. Flag0 always remains negated in modes 0 and 1.

Flag1 is used as an internal indication in modes 1 and 3 of an overflow of ACCUM. It is asserted when an overflow of ACCUM occurs and negated when MAX_COUNT periods are counted.

9.1 STATE 1: *INIT*

This state is entered as a result of an HSR%10. The channel latches are configured using CHANNEL_CONTROL, ACCUM is initialized to zero, and PIN_CTRL is copied into CHANNEL_CONTROL.

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

Configure the channel latches via CHANNEL_CONTROL
Initialize ACCUM to 0
Clear flag1 and assert flag0
Clear all service requests

9.2 STATE 2: *FIRST_H_L*

This state is entered in modes 0 and 1 for the first high-to-low input transition after *Init* is executed; LAST_ACCUM is updated with the time of the transition. This state is also entered in modes 0 and 1 when a match event occurs for the periodic update of ACCUM while the pin is low. This case happens only when a match is set up for accumulation at ACCUM_RATE and the pin goes low prior to the match.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001001
Match Enable: Disable

If (TDL = 1) then {
    Update LAST_ACCUM with ERT

```
            Setup next match time = ERT + ACCUM_RATE
            Configure pin for high-to-low transition
            Clear flag0
        }
        Else, negate MRL
```

9.3 STATE 3: HIGH_TIME_BEGIN

This state is entered in modes 2 and 3 after a low-to-high input transition. In this state, LAST_ACCUM is updated with the time of the transition.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001011
Match Enable: Disable

```
        If (TDL = 1) then {
            Update LAST_ACCUM with ERT
            Setup next match time = ERT + ACCUM_RATE
            Configure pin for high-to-low transition
            Clear flag0
}
Else, negate MRL
```

9.4 STATE 4: ACCUM_LOW_PIN

This state is entered in all modes after a high-to-low input transition or when a match event occurs for the periodic update of ACCUM. In this state, the period/pulse width is accumulated in ACCUM while PERIOD_COUNT is less than MAX_COUNT. PPWA_UB is incremented for overflows of ACCUM. When PERIOD_COUNT is greater than or equal to MAX_COUNT, PPWA_LW is updated with ACCUM, and PERIOD_COUNT and ACCUM are initialized to zero. An interrupt request is then made to the CPU, and/or a link is generated to a block of one or more TPU channels.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 00100x
Match Enable: Disable

```
    Update LAST_ACCUM_TEMP with LAST_ACCUM
                                        /* LAST_ACCUM_TEMP is a temporary register */
    Update LAST_ACCUM with ERT
    If (host sequence bit 1 = 1) then {
        Assert flag0
        Configure pin for low-to-high transition
        Goto UPDATE_ACCUM
    }
SETUP_ACCUM_TIME
    Setup next match time = ERT + ACCUM_RATE
    Configure pin for high-to-low transition
    Clear flag0
    Negate MRL
UPDATE_ACCUM
    Update ACCUM = ACCUM + LAST_ACCUM – LAST_ACCUM_TEMP
    If MRL = 0 then {                                          /* TDL = 1 */
        If (ACCUM > 16 bits) then {
            Clear flag0, flag1
            Clear MRL
            Increment PPWA_UB
            If (host sequence bit 0 = 1) then {
                Link to channels START_LINK_CHANNEL to
                [START_LINK_CHANNEL + LINK_CHANNEL_COUNT – 1]
```

```
        }
    Else {                                                  /* ACCUM ≤ 16 bits */
        PERIOD_COUNT = PERIOD_COUNT + 1
        Clear MRL, TDL
        If (PERIOD_COUNT ≥ MAX_COUNT) then {
            PPWA_LW = ACCUM
            PERIOD_COUNT = 0
            ACCUM = 0
            Generate interrupt request
            If flag1 = 1 then clear flag1
            Else {
                Clear flag1
                If (host sequence bit 0 = 1) then {
                    Link to channels START_LINK_CHANNEL to
                    [START_LINK_CHANNEL + LINK_CHANNEL_COUNT – 1
                }
            }
        }
    }
Else {                                                      /* MRL = 1 */
    If TDL = 1 then {
        If (ACCUM > 16 bits) then {
            Clear flag0, flag1
            Clear MRL
            Increment PPWA_UB
            If (host sequence bit 0 = 1) then {
                Link to channels START_LINK_CHANNEL to
                [START_LINK_CHANNEL + LINK_CHANNEL_COUNT – 1]
        }
    }
    Else {                                                  /* ACCUM ≤ 16 bits */
        PERIOD_COUNT = PERIOD_COUNT +1
        Clear MRL, TDL
        If (PERIOD_COUNT ≥ MAX_COUNT) then {
            PPWA_LW = ACCUM
            PERIOD_COUNT = 0
            Generate interrupt request
            If flag1 = 1 then clear flag1
            Else {
                Clear flag1
                If (host sequence bit 0 = 1) then {
                    Link to channels START_LINK_CHANNEL to
                    [START_LINK_CHANNEL + LINK_CHANNEL_COUNT – 1
                }
            }
        }
    }
}
Else {                                                      /* TDL = 0 */
    If ACCUM > 16 bits) then {
    Clear flag0, flag1
    Clear MRL
    Increment PPWA_UB
    If (host sequence bit 0 = 1) then {
        Link to channels START_LINK_CHANNEL to
```

```
                [START_LINK_CHANNEL + LINK_CHANNEL_COUNT - 1]
            }
        }
    }
    }
```

9.5 STATE 5: ACCUM_HIGH_PIN

This state is entered in all modes when a match event occurs for the periodic update of ACCUM, while the pin is high. In this state, the period/pulse width is accumulated in ACCUM, and the next match, which results in another update of ACCUM, is set up. Also, PPWA_UB is incremented for overflows of ACCUM. If an overflow of ACCUM occurs in this state for modes 1 or 3, a link is generated to a block of TPU channels.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001010
Match Enable: Disable

```
Update LAST_ACCUM_TEMP with LAST_ACCUM
                                    /* LAST_ACCUM_TEMP is a temporary register */
Update LAST_ACCUM with ERT
Goto SETUP_ACCUM_TIME
```

The following tables show the PPWA state transitions listing the service request sources and channel conditions from current state to next state. **Figure 12** illustrates the flow of PPWA states in modes 0 and 1; **Figure 13** illustrates the flow of PPWA states in modes 2 and 3.

Table 9 PPWA State Transition Tables
Modes 0 and 1

Current State	HSR	M/TSR	LSR	Pin	Flag0	Next State
Any State	10	—	—	—	—	S1 *Init*
S1 *Init*	00	1	—	0	0	S2 *First_H_L*
S2 *First_H_L*	00 00	1 1	— —	0 1	0 0	S4 *Accum_Low_Pin* S5 *Accum_High_Pin*
S4 *Accum_Low_Pin*	00 00	1 1	— —	0 1	0 0	S4 *Accum_Low_Pin* S5 *Accum_High_Pin*
S5 *Accum_High_Pin*	00 00	1 1	— —	0 1	0 0	S4 *Accum_Low_Pin* S5 *Accum_High_Pin*
Unimplemented Conditions	11 00	— 0	— 1	— —	— —	— —

NOTES:
1. Conditions not specified are “don't care.”
2. LSR = Link service request
   HSR = Host service request
   M/TSR = Either a match or capture (transition) service request occurred (M/TSR = 1) or neither occurred (M/TSR = 0).

Table 10 PPWA State Transition Tables
Modes 2 and 3

Current State	HSR	M/T	LSR	Pin	Flag0	Next State
Any State	10	—	—	—	—	S1 *Init*
S1 *Init*	00	1	—	1	0	S3 *High_Time_Begin*
S3 *High_Time_Begin*	00 00	1 1	— —	1 0	0 0	S5 *Accum_High_Pin* S4 *Accum_Low_Pin*
S4 *Accum_Low_Pin*	00	1	—	1	1	S3 *High_Time_Begin*
S5 *Accum_High_Pin*	00 00	1 1	— —	0 1	0 0	S4 *Accum_Low_Pin* S5 *Accum_High_Pin*
Unimplemented Conditions	11 00	— 0	— 1	— —	— —	— —

NOTES:

1. Conditions not specified are “don't care.”
2. LSR = Link service request
   HSR = Host service request
   M/TSR = Either an output compare (match) or input capture (transition) service request occurred (M/TSR = 1) or neither occurred (M/TSR = 0).

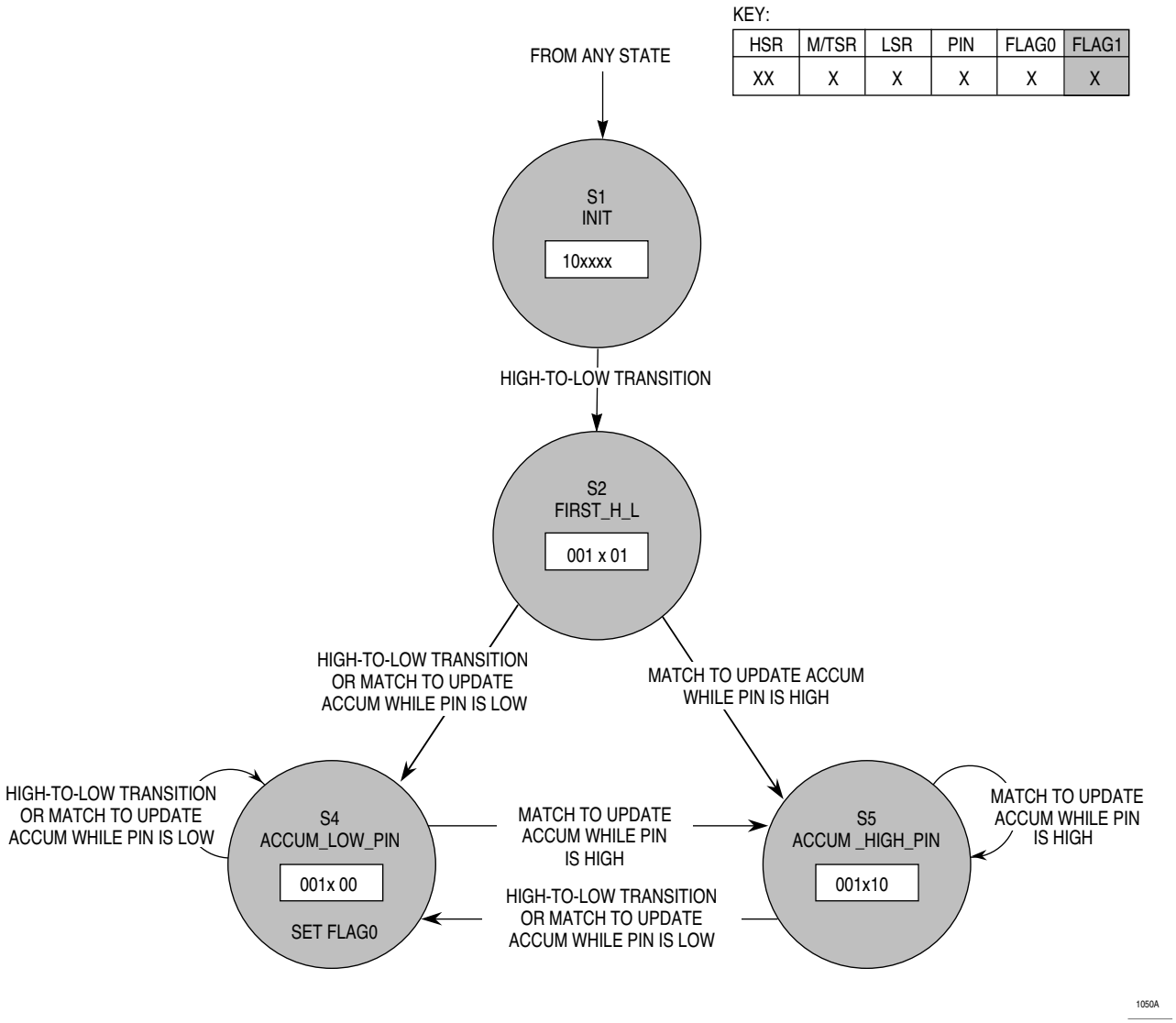


Figure 12 PPWA Modes 0 and 1 State Flowchart

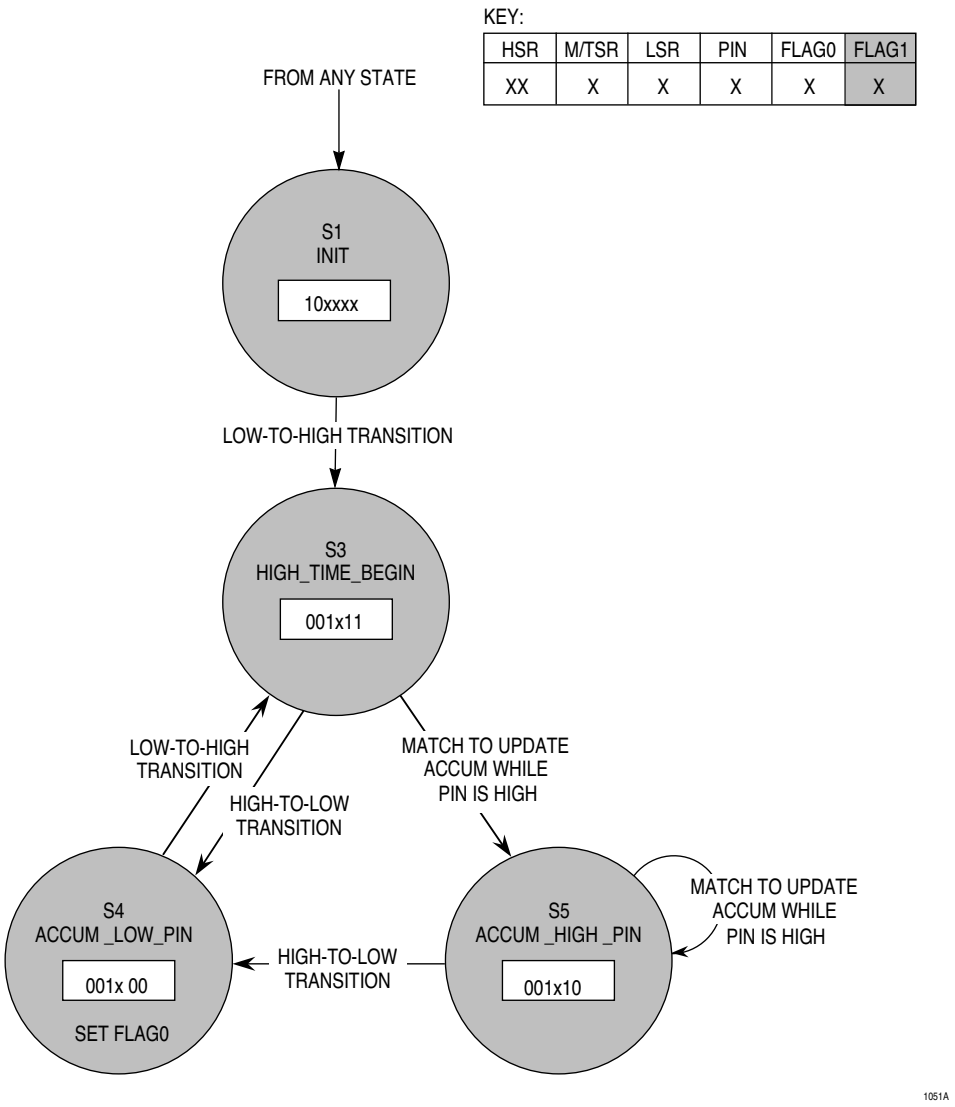


Figure 13 PPWA Modes 2 and 3 State Flowchart

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 5405, Denver Colorado 80217. 1-800-441-2447, (303) 675-2140
Mfax™: RMFAX0@email.sps.mot.com - TOUCHTONE (602) 244-6609, U.S. and Canada Only 1-800-774-1848
INTERNET: http://Design-NET.com
JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC,
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298
Mfax is a trademark of Motorola, Inc.