Position-Synchronized Pulse Generator (PSP) TPU Function

By Sharon Darley

1 Functional Overview

The PSP function can generate an output transition that is referenced to a time previously determined by another channel. Typically, this function is used with a PMM or PMA function on another channel. The PMM/PMA function determines the reference time, and the PSP function generates the output pulse in relation to that reference time. The PSP function has two operating modes: angle-angle and angle-time. In angle-angle mode, the rising and falling edges of the output pulse are determined independently of each other. In angle-time mode, the falling edge of the output pulse is determined in reference to the rising edge.

Up to 15 PSP function channels may operate with a single input reference channel executing a PMA or PMM function. The input channel measures and stores the time period between the flywheel teeth and resets TCR2 when the engine reaches top dead center. The PSP output channel uses the most recent period calculated by the input channel to project output transitions at specific engine degrees. Since the flywheel teeth might be 30 or more degrees apart, a fractional multiply resolves down to the precise angle.

NOTE

The PSP function was developed to work in conjunction with the PMA and PMM functions. The programming notes for these functions (TPUPN15A/D and TPUPN15B/D, respectively) should be consulted when using the PSP function.

2 Detailed Description

The PSP function is typically used in automotive applications with a PMM or PMA function on another channel. An optical or magnetic sensor is connected to both a channel executing the PMM/PMA function and the clock input to TCR2. Connecting the sensor to the TCR2 clock allows engine-cycle position to be mapped into TCR2 counts. The sensor detects the teeth on a flywheel. The flywheel has regularly spaced teeth with the exception of one or more reference points in the form of missing or additional teeth. The PMM or PMA function detects these reference points. The PSP function then generates an output pulse in relation to the desired reference points.

Once initialized and synchronized to the input pulse sequence, a PSP function continually generates output pulses based on the angle and ratio parameters. To generate the output pulse, the CPU updates the angle and ratio parameters. The parameters ANGLE1/RATIO1 and ANGLE2/RATIO2 must be written coherently if both are changed, since they are used together. If changes are made by long word move instructions, coherency is assured. When changes are made with an immediate update request (HSR = 01), host sequence bit 1 determines which parameters are used for pulse generation.

If host sequence bit 1 is zero, all parameters are used coherently and immediately if the pin is low and TCR2 is still less than ANGLE1. If host sequence bit 1 is one, only the falling edge is changed immediately. As each transition is generated, an interrupt request may be asserted. The PSP function has two modes for determining pulse length: angle-angle mode and angle-time mode. Angle-angle mode determines the falling edge of the pulse by angular position, while angle-time mode determines the falling edge of the pulse in reference to the rising edge.

In angle-angle mode (host sequence bit 0 equals zero), the parameters ANGLE1, ANGLE2, RATIO1, and RATIO2 are used to calculate the rising and falling output transitions. ANGLE1 is the reference angle, or tooth number, on the flywheel that determines the rising output transition. ANGLE2 is the reference angle, or tooth number, on the flywheel that determines the falling output transition. The output pulse is generated as follows. When TCR2 (which is clocked by the input pulse train from the flywheel) reaches the position indicated by ANGLE1, the value in RATIO1 is multiplied by the period pointed to by PERIOD_ADDRESS to determine the projected time for the rising edge. (The period referenced by PERIOD_ADDRESS is typically generated by a PMA/PMM function on another channel.) A similar sequence occurs for the falling-edge computation, using RATIO2, when TCR2 matches the value in ANGLE2. Each ratio parameter has an effective range of zero to $(2 – 2^{-8})$ that allows the TPU to project up to two times the previously calculated period. The lower seven bits of these parameters are used in a fractional multiply; hence, the edge can be projected to an accuracy of 1/128 of the input period. Figure 1 illustrates the relationship between the parameters and includes a special case where ANGLE1 equals ANGLE2.

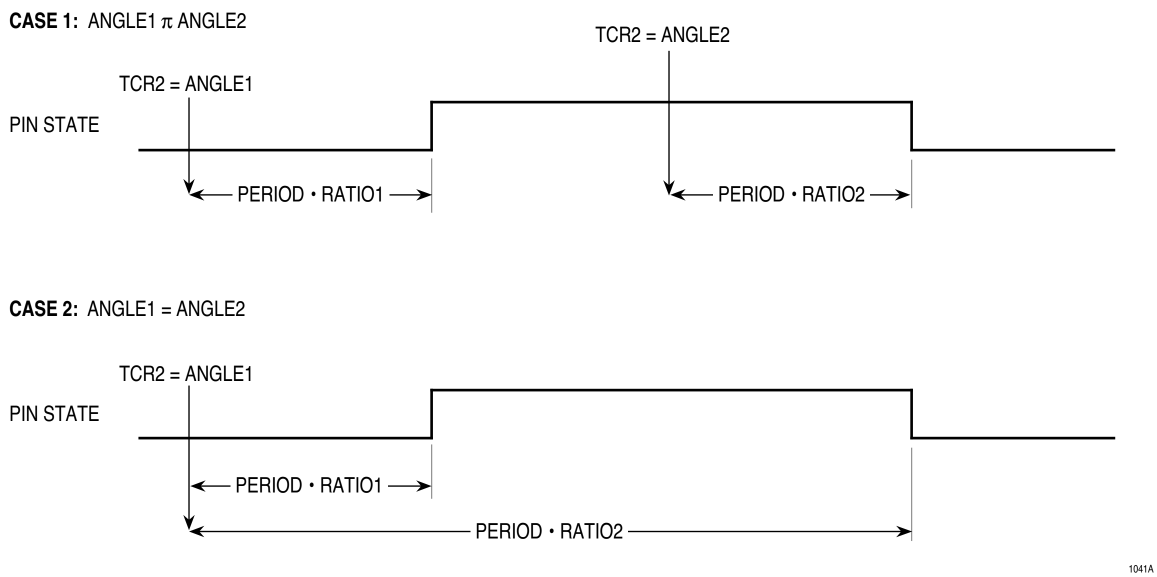

Figure 1 PSP Angle-Angle Mode

In angle-time mode (host sequence bit 0 equals one), ANGLE1, RATIO1, and HIGH_TIME are the parameters used to calculate the rising and falling edges of the output pulse. ANGLE1 is the reference angle, or tooth number, on the flywheel that determines the rising output transition. HIGH_TIME is the time duration of the pulse. The output pulse is generated as follows. When TCR2 (which is clocked by the input pulse train from the flywheel) reaches the position indicated by ANGLE1, the value in RATIO1 is multiplied by the period pointed to by PERIOD_ADDRESS to determine the projected time for the rising edge. (The period referenced by PERIOD_ADDRESS is typically generated by a PMA/PMM function on another channel.) The falling edge is then delayed from the rising edge by the amount of time specified in HIGH_TIME. **Figure 2** illustrates the relationship between these parameters.

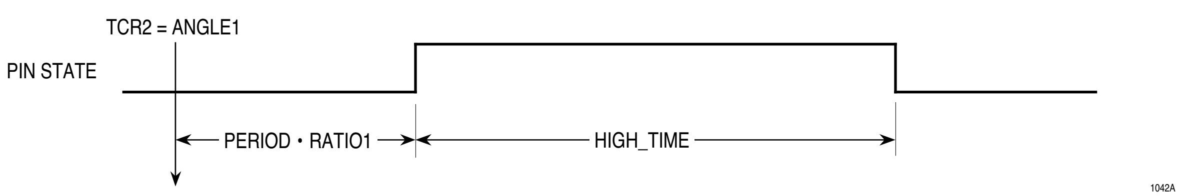

Figure 2 PSP Angle-Time Mode

During normal operation, the TPU generates an output pulse based on the angle and ratio parameters. To control the output pulse, the CPU updates the angle and ratio parameters. The parameter sets ANGLE1/RATIO1 and ANGLE2/RATIO2 must be written coherently (i.e. by long-word writes) if both are changed, since they are used together.

The PSP function uses force mode when an error from the input channel requires the CPU to manipulate the output pin directly. An error occurs if the period is too long to be measured, or if the input channel tracking the flywheel teeth is out of synchronization. In this case, the CPU updates CHANNEL_CONTROL appropriately and issues a host service request, forcing the TPU to assert or negate the output pin and to stop any other pin activity. Reinitializing the time function (to be performed only when the pin is low) causes a return to normal operation.

PSP is often used with the PMA and PMM functions, generating pulses synchronized to the period and position of teeth measured by these functions. Refer to the programming notes for these two functions for examples illustrating the use of PSP with PMA and PMM.

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. PSP function code size is:

88 μ instructions + 14 entries = **102 long words**

4 Function Parameters

This section provides detailed descriptions of PSP function parameters stored in channel parameter RAM. **Figure 3** shows TPU parameter RAM address mapping. **Figure 4** shows the parameter RAM assignment used by the PSP 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 3 TPU Channel Parameter RAM CPU Address Map

	15–8	7–0
$YFFFW0	PERIOD_ADDRESS	CHANNEL_CONTROL
$YFFFW2	R2_A2_TEMP	
$YFFFW4	ANGLE_TIME	
$YFFFW6	RATIO_TEMP	
$YFFFW8	RATIO1	ANGLE1
$YFFFWA	RATIO2/HIGH_TIME	ANGLE2/HIGH_TIME

W = Channel number
Parameter Write Access:

- Written by CPU
- Written by TPU
- Written by CPU and TPU
- Unused parameters

Figure 4 PSP Function Parameter RAM Assignment

4.1 CHANNEL_CONTROL

CHANNEL_CONTROL determines the pin state in force mode only, and configures the PSC, PAC, and TBS fields. The PSC field forces the output level of the pin directly without affecting the PAC latches. The PSP function does not use the PAC and TBS fields. The table below defines the allowable data for this parameter.

15–9	8–5	4–2	1–0
NOT USED	TBS	PAC	PSC

Table 1 PSP CHANNEL_CONTROL Options

TBS	PAC	PSC	Action	
8 7 6 5	4 3 2	1 0	Input Channel	Output Channel
		0 1 1 0	— —	Force Pin High Force Pin Low
	0 0 0 1 x x		Do Not Detect Transition Do Not Change PAC	Do Not Change Pin State on Match Do Not Change PAC
1 x x x			Do Not Change TBS	Do Not Change TBS

4.2 PERIOD_ADDRESS

PERIOD_ADDRESS is a byte parameter that is updated by the CPU and contains the address of a period parameter, usually located in a channel executing a PMA/PMM function. In this case, the address should be that of PERIOD_LOW_WORD.

4.3 R2_A2_TEMP

R2_A2_TEMP is used to temporarily store the value of ANGLE2 and RATIO2 during pulse generation, to guarantee coherency of ANGLE1, ANGLE2, RATIO1, and RATIO2. ANGLE2 and RATIO2 are copied to R2_A2_TEMP when a match on ANGLE1 is set up.

4.4 ANGLE_TIME

ANGLE_TIME is updated continuously by the TPU with the TCR1 value. In angle-angle mode, ANGLE_TIME contains the TCR1 value that occurs when TCR2 equals ANGLE1. In angle-time mode, ANGLE_TIME contains the TCR1 value associated with the low-to-high transition.

4.5 RATIO_TEMP

RATIO_TEMP is a temporary variable used by the TPU to store the value of ANGLE1 and RATIO1 during pulse generation. This parameter is used to guarantee coherent access to ANGLE1 and RATIO1.

4.6 RATIO1

RATIO1 contains an 8-bit multiplier for the rising output transition. This parameter is updated by the CPU. The range of RATIO1 is $00 to $FF ($0_{10}$ to 1.99_{10}).

NOTE

If RATIO1 equals zero (or is very small), the rising edge of the pulse is skewed and the amount of skew is dependent upon TPU latency. Updating ANGLE1 to the previous angle and adjusting RATIO1 appropriately eliminates this problem.

4.7 ANGLE1

ANGLE1 is a byte parameter updated by the CPU and contains the reference angle of TCR2 for the rising output transition.

4.8 RATIO2

RATIO2 contains an 8-bit multiplier for the falling output transition. This parameter is updated by the CPU and is used only in angle-angle mode. The range of RATIO2 is $00 to $FF ($0_{10}$ to 1.99_{10}).

NOTE

If RATIO2 equals zero (or is very small), the falling edge of the pulse is skewed and the amount of skew is dependent upon TPU latency. Updating ANGLE2 to the previous angle and adjusting RATIO2 appropriately eliminates this problem.

4.9 ANGLE2

ANGLE2 is a byte parameter updated by the CPU. This parameter contains the reference angle of TCR2 for the falling output transition and is used only in angle-angle mode.

4.10 HIGH_TIME

HIGH_TIME specifies the time duration of the output pulse in angle-time mode. The weighting of the least significant bit of this parameter is defined as equal to the least significant bit of TCR1. This parameter is updated by the CPU and is used only in angle-time mode.

5 Host Interface to Function

This section provides information concerning the TPU host interface to the PSP function. **Figure 5** 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 5 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] — Function Number (Assigned during microcode assembly)

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

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

CH[15:0]	Action Taken
x0	Angle-Angle Mode Output specified by ANGLE1, ANGLE2, RATIO1, RATIO2.
x1	Angle-Time Mode Output pulse specified by ANGLE1, RATIO 1, and HIGH_TIME.
0x	ANGLE1 and ANGLE2/HIGH_TIME are used if pin is low and TCR2 < ANGLE1, else nothing is done until next pulse.
1x	ANGLE2/HIGH_TIME is used when immediate request is issued if pin has not fallen.

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

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

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

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

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

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

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

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

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

6 Function Configuration

The CPU initializes this time function by the following:

1. Writing parameter PERIOD_ADDRESS;
2. Writing parameters ANGLE1/RATIO1 and ANGLE2/RATIO2 (angle-angle mode) or ANGLE1/RATIO1 and HIGH_TIME (angle-time mode);
3. Writing host sequence bit 0 according to mode of operation;
4. Issuing an HSR %10 for initialization; and
5. Enabling channel servicing by assigning a high, middle, or low priority.

The TPU then executes initialization. The CPU should monitor the HSR register (or the channel interrupt) until the TPU clears the service request to 00 before changing any parameters or before issuing a new service request to this channel.

The CHANNEL_CONTROL parameter is not needed for host initialization (HSRR = %10). In this case, microcode forces the channel hardware to match on TCR2, capture TCR1, and force the pin low. The CHANNEL_CONTROL parameter is only used by the Host Service 'Force Mode', (HSRR = %11).

The following general rules apply to updated ANGLE1, RATIO1, ANGLE2, and RATIO2 values during pulse generation:

1. When ANGLE1 is less than or equal to ANGLE2, the new values of these parameters are used coherently to generate pulses scheduled after a special tooth condition that results in TCR2 being reset (to $FFFF).

Figure 6 Updated Parameters When ANGLE1 ≤ ANGLE2

2. When ANGLE1 is greater than ANGLE2, the new values of these parameters are used coherently to generate pulses scheduled after the subsequent falling transition.

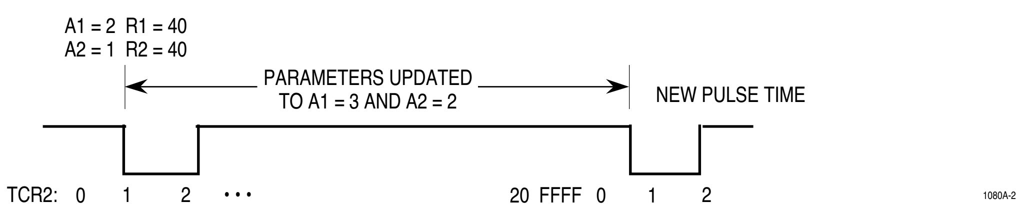

Figure 7 Updated Parameters When ANGLE1 > ANGLE2

3. To cause an immediate update in the output pulse, the parameters should be updated, host sequence bit 1 set appropriately, and an immediate update request issued. If host sequence bit 1 is set to one, only the new ANGLE2 and RATIO2 are used. If host sequence bit 1 is set to zero, the new ANGLE1, ANGLE2, RATIO1, and RATIO2 are used coherently.

If the pin is low when an immediate request is executed, one of the following results:

- If host sequence bit 1 = 1, only the next falling transition is changed; or
- If host sequence bit 1 = 0 and TCR2 is less than ANGLE1, both the rising and falling transitions are changed; or
- Nothing is done.

If the pin is high when an immediate request is executed, one of the following results:

- If host sequence bit 1 = 1, only the next falling transition is changed; or
- Nothing is done.

Following are three special cases of abnormal output pulses:

CASE 1:

In angle-time mode, if the falling edge of the PSP output pulse occurs before the rising edge of

ANGLE1 + 1, then a short stream of pulses will be generated instead of just one pulse. An illustration of this phenomenon is shown in **Figure 8**.

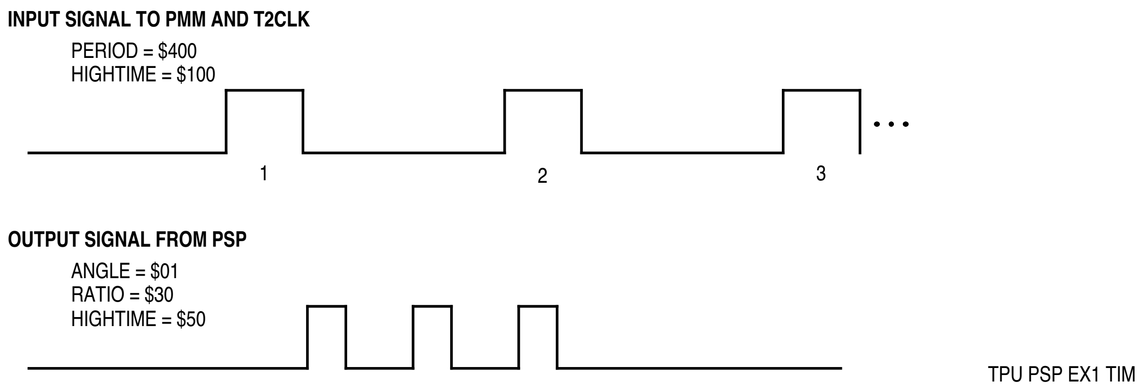

Figure 8 Example for Case 1

This continuous stream is generated because TCR2 continues to match ANGLE1 until the rising edge of the next tooth is reached. To avoid producing a stream of pulses, use the previous angle and a larger value in RATIO1 to determine the rising edge of the output pulse. Thus, in this example, use ANGLE1 = $0 instead of ANGLE1 = $01 and adjust RATIO1 to attain the desired pulse position, or use angle-angle mode.

CASE 2:

In angle-angle mode, if ANGLE2 = ANGLE1 + 1 and RATIO1 is greater than $80, the pulse width will be longer than expected. RATIO1 = $80 occurs at the point when the rising edge of the output pulse is at the rising edge of TCR2 = ANGLE2. The larger the RATIO1 value is, the longer the pulse width will be. At RATIO1 = RATIO2 = $FF, the pulse width will be twice as long as expected. Figure 9 illustrates the case when RATIO1 < $80 and also the case when RATIO1 > $80.

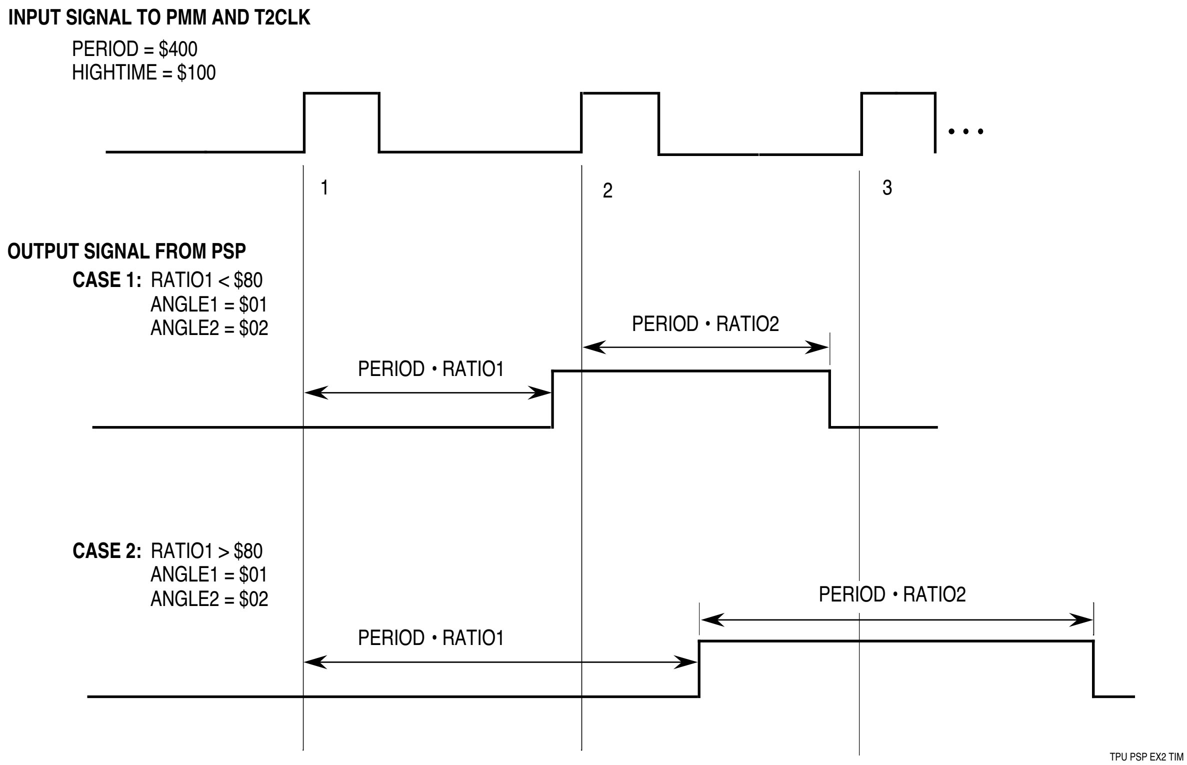

Figure 9 Example for Case 2

This phenomenon occurs because of the way the output signal is generated. First, TCR2 = ANGLE1 matches. Then, the rising edge is calculated. Next, after the low-to-high pin transition takes place, TCR2 = ANGLE2 matches. Then, the falling edge is calculated. The falling edge is offset from the TCR1 time of the TCR2 = ANGLE2 match by the product of the reference period at PERIOD_ADDRESS and RATIO2. The TPU cannot match TCR2 = ANGLE2 and calculate the falling edge until the rising edge has taken place. Thus, if the rising edge occurs after time TCR2 = ANGLE2, the TPU will immediately generate a match, and the reference used will be the TCR1 value just after the rising edge of the pulse, instead of the TCR1 value of ANGLE2. Thus, the falling edge will be referenced to the rising edge, rather than to ANGLE2.

In summary, for values of 0 ≤ RATIO1 ≤ $80, the falling edge is referenced to the sensor tooth at ANGLE1. For values of $80 < RATIO1 ≤ $FF, the falling edge is referenced approximately to the rising edge. If $80 < RATIO1 ≤ $FF, use ANGLE1 = ANGLE2 with a larger RATIO2 value to avoid the extended pulse width.

CASE 3:

If the hightime of an output pulse is so long that the falling edge occurs after the next scheduled rising edge, the next pulse will not be generated.

7 Performance and Use of Function

7.1 Performance

Like all TPU functions, PSP 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. The more TPU channels are active, the more performance decreases. Worst-case latency in any TPU application can be closely estimated. To analyze the performance of an application that appears to approach the limits of the TPU, use the guidelines given in the *TPU Reference Manual* (TPURM/AD) and the information in the following table.

Table 2 PSP State Timing

State Name	Clock Cycles Angle-Angle Mode	RAM Accesses Angle-Angle Mode	Clock Cycles Angle-Time Mode	RAM Accesses Angle-Time Mode
S1 *Init*	16	4	16	4
S2 *Trans_H_L*	30	3	24	5
S3 *Angle1_gt_$8000*	20	4	20	4
S4 *TCR2_eq_Angle1*	50	4	50	3
S5 *Ratio1_gt_$8000*	8	1	8	1
S6 *Trans_L_H* Angle 1 ≠ Angle 2 Angle 1 = Angle 2	 32 70	 4 6	 18 —	 2 —
S7 *Angle1_gt_Angle2*	12	3	—	—
S8 *TCR2_eq_Angle2*	44	3	—	—
S9 *Ratio2_gt_$8000*	10	1	10	1
S10 *Immed_L*	18	5	18	5
S11 *Immed_H* A1 ≠ A2 A1 = A2	 26 76	 3 6	 30 —	 4 —
S12 *Force*	6	1	6	1

7.2 Changing Mode

The host sequence bits are used to select PSP 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

A common application of the PSP function is an angle-based automotive engine control system. A typical system is shown in **Figure 10**. As shown in this illustration, the flywheel has reference points in the form of missing or additional teeth. The PMM/PMA functions can be used to detect these reference points; the PMM function detects missing teeth, and the PMA function detects additional teeth.

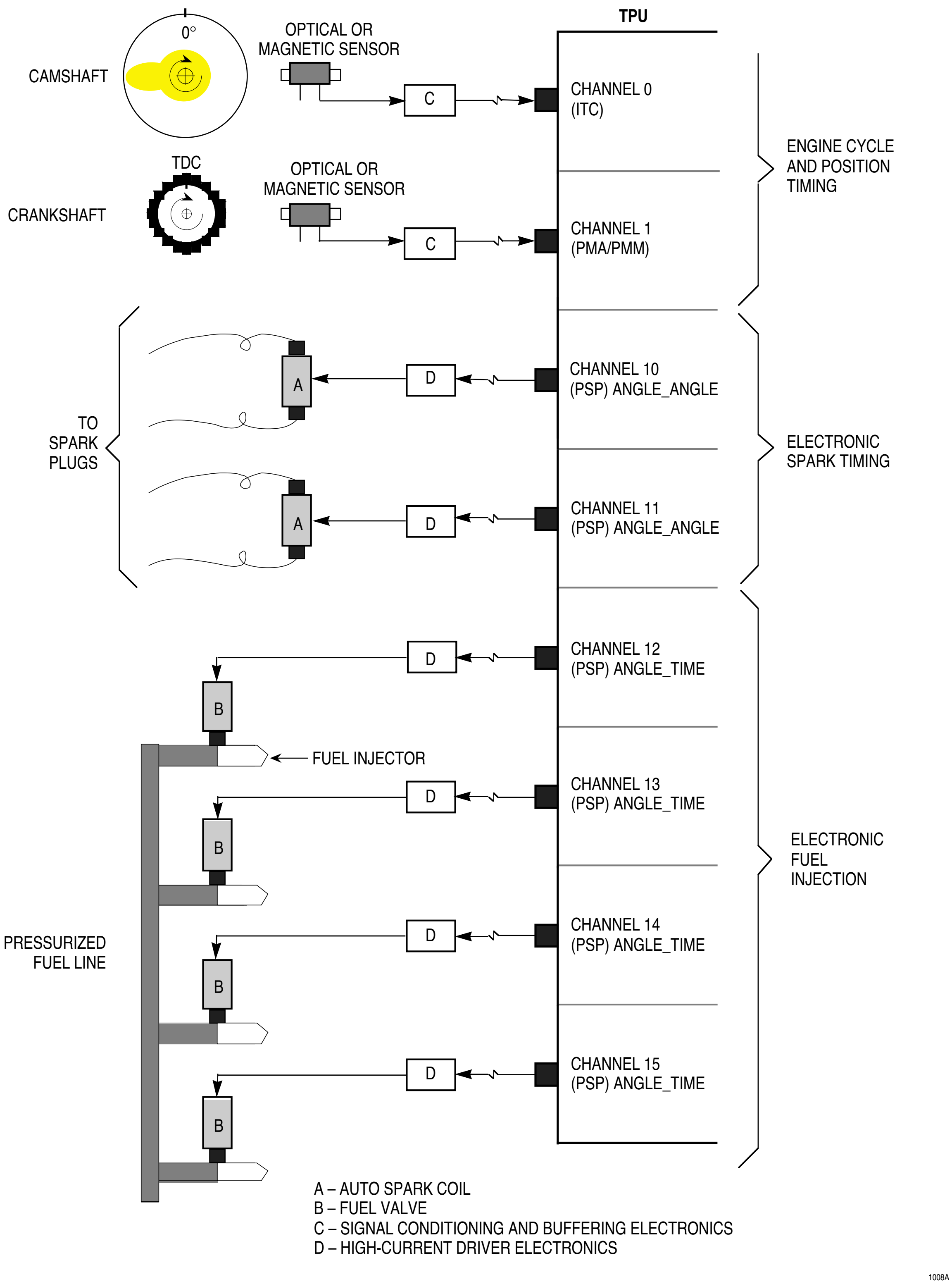

Figure 10 Engine Control Example

In this example, the PSP function is used in conjunction with the PMM function. The PMM function determines when the missing tooth occurs, and the PSP function waits a programmable amount of time before it generates an output pulse. The PSP function has two operating modes, angle-angle and angle-time, and it generates the output pulse based on five parameters: RATIO1, RATIO2, ANGLE1, ANGLE2, and HIGH_TIME. RATIO1 and RATIO2 are 8-bit numbers that represent a decimal multiplier of the period that can range from 0 to 1.99. ANGLE1 and ANGLE2 represent reference angles. A reference angle is simply a tooth number. The teeth are numbered starting with 0 after the missing transition. HIGH_TIME specifies the time duration of the output pulse in angle-time mode. In this example, the PSP function uses the angle-time mode.

The PMM function has two modes: count mode and bank mode. In count mode, timer TCR2 is reset after the number of missing transitions in MAX_MISSING has been counted. In bank mode, timer TCR2 is reset after a missing transition only if BANK_SIGNAL is a non-zero value. In this example, the PMM function uses the count mode.

8.1.2 Hardware Setup

The input pulse train to the PMM channel and to the T2CLK input mimics the flywheel in an automobile engine. In this example, the flywheel has 35 teeth and one missing tooth (36 evenly spaced tooth positions total). See **Figure 11** below for an illustration of this input pulse.

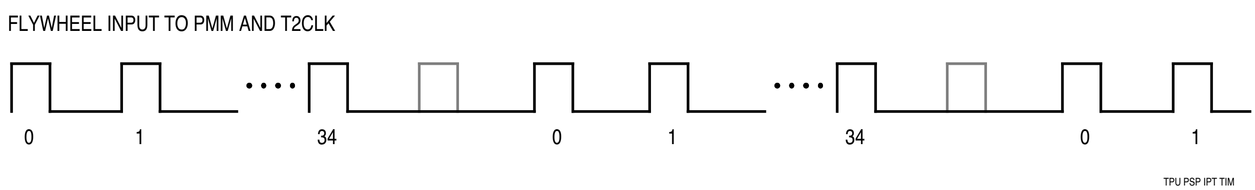

Figure 11 Input Pulse Train

In this example, the PMM function is on channel 4, and the PSP function is on channel 5. The PSP function is not physically connected to any of the other channels, but its parameter PERIOD_ADDRESS points to the PMM parameter PERIOD_LOW_WORD.

8.1.3 Output Waveform

Channel 3 in **Figure 12** shows the output waveform produced by this example.

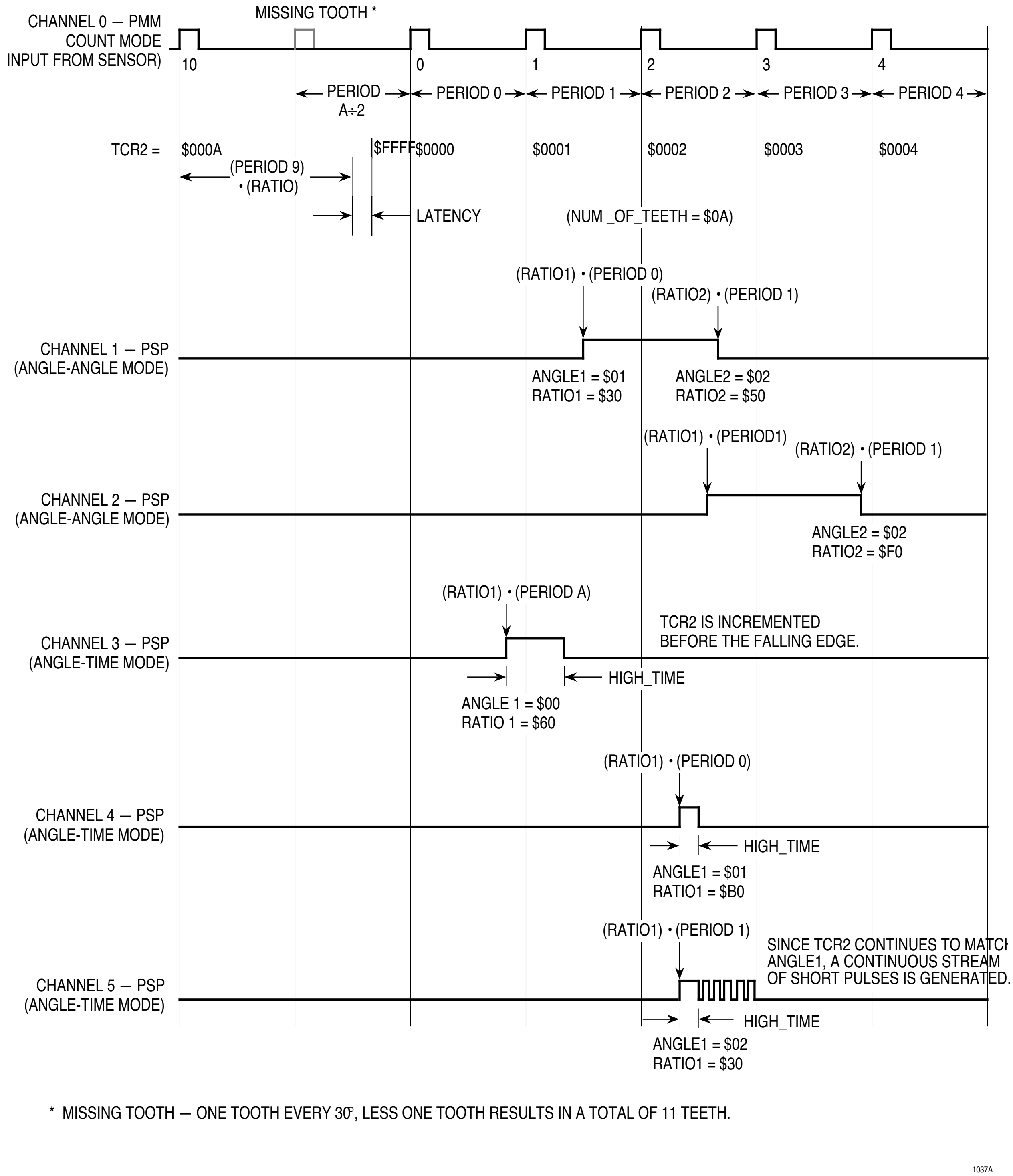

Figure 12 Output Waveform for Example A

8.2 Program Code for CPU32-Based Microcontrollers

This program is a demonstration of how to use the PMM and PSP functions together to generate an output pulse in relation to a “missing tooth.” In this case, the input pulse train to the PMM channel and T2CLK is a series of 35 pulses followed by a 36th missing transition.

This program was assembled using the IASM32 assembler available from P&E Microcomputer Systems with the M68332 In-Circuit Debugger. It was run on an M68332EVS and BCC.

8.2.1 Initialization

Channel 4 is configured as a PMM channel, and channel 5 is configured as a PSP function. PMM is in count mode, and PSP is in angle-time mode.

```
TPUMCR        equu    $fffe00
TICR          equ     $fffe08
CIER          equ     $fffe0a
CFSR0         equ     $fffe0c
CFSR1         equ     $fffe0e
CFSR2         equ     $fffe10
CFSR3         equ     $fffe12
HSQR0         equ     $fffe14
HSQR1         equ     $fffe16
HSRR0         equ     $fffe18
HSRR1         equ     $fffe1a
CPR0          equ     $fffe1c
CPR1          equ     $fffe1e
PRAM4_0       equ     $ffff40
PRAM4_1       equ     $ffff42
PRAM4_2       equ     $ffff44
PRAM4_3       equ     $ffff46
PRAM4_4       equ     $ffff48
PRAM4_5       equ     $ffff4a
PRAM5_0       equ     $ffff50
PRAM5_1       equ     $ffff52
PRAM5_2       equ     $ffff54
PRAM5_3       equ     $ffff56
PRAM5_4       equ     $ffff58
PRAM5_5       equ     $ffff5a
        org     $4000               ;begin program at memory location $4000
        move.w  #$00cb,(CFSR2).l    ;Channel function select field (Note: function numbers
                                    ;may vary for different mask sets)
        move.w  #$0700,(HSQR1).l    ;HSQR1, PSP in angle-time mode
                                    ;PMM in count mode
        move.w  #$0f00,(CPR1).l     ;CPR1, high priority to both channels
```

8.2.2 PMM Initialization for Channel 4

```
        move.w  #$0004,(PRAM4_0).l;Channel control reg., detect rising edge
        move.w  #$0122,(PRAM4_1).l;MAX_MISSING = 1, NUM_OF_TEETH = 34
        move.w  #$a022,(PRAM4_3).l;RATIO=$a0, TCR2_MAX_VALUE=34
```

8.2.3 PSP Initialization for Channel 5

```
        move.w  #$4a01,(PRAM5_0).l ;PERIOD_ADDRESS points to
                                   ;PERIOD_LOW_WORD of PMM
                                   ;Force pin high in force mode
        move.w  #$6000,(PRAM5_4).l ;RATIO1 = $60, ANGLE1 = $00
        move.w  #$0200,(PRAM5_5).l ;HIGH_TIME = $200
start
        move.w  #$0900,(HSRR1).l    ;host service initialization request for channels 4 and 5
finish
        bra     finish
```

8.3 Program Code for CPU16-Based Microcontrollers

This program was assembled using the IASM16 assembler from P&E Microcomputer Systems with the ICD16 In-Circuit Debugger. It was run on an MC68HC16Y1 EVB.

8.3.1 Initialization

Channel 4 is configured as a PMM channel, and channel 5 is configured as a PSP function. PMM is in count mode, and PSP is in angle-time mode.

```
TPUMCR          equu    $fe00
TICR            equ     $fe08
CIER            equ     $fe0a
CFSR0           equ     $fe0c
CFSR1           equ     $fe0e
CFSR2           equ     $fe10
CFSR3           equ     $fe12
HSQR0           equ     $fe14
HSQR1           equ     $fe16
HSRR0           equ     $fe18
HSRR1           equ     $fe1a
CPR0            equ     $fe1c
CPR1            equ     $fe1e
PRAM4_0         equ     $ff40
PRAM4_1         equ     $ff42
PRAM4_2         equ     $ff44
PRAM4_3         equ     $ff46
PRAM4_4         equ     $ff48
PRAM4_5         equ     $ff4a
PRAM5_0         equ     $ff50
PRAM5_1         equ     $ff52
PRAM5_2         equ     $ff54
PRAM5_3         equ     $ff56
PRAM5_4         equ     $ff58
PRAM5_5         equ     $ff5a
```

The following code is included to set up the reset vector ($00000 – $00006). It may be changed for different systems.

```
        ORG     $0000                   ;put the following reset vector information
                                        ;at address $00000 of the memory map
        DW      $0000                   ;zk=0, sk=0, pk=0
        DW      $0200                   ;pc=200 -- initial program counter
        DW      $3000                   ;sp=3000 -- initial stack pointer
        DW      $0000                   ;iz=0 -- direct page pointer
        org     $0400                   ;begin program at memory location $0400
```

The following code initializes and configures the system including the Software Watchdog and System Clock. It was written to be used with an EVB.

```
INITSYS:                                ;give initial values for extension registers
                                        ;and initialize system clock and COP
        LDAB    #$0F
        TBEK                            ;point EK to bank F for register access
        LDAB    #$00
        TBXK                            ;point XK to bank 0
        TBYK                            ;point YK to bank 0
        TBZK                            ;point ZK to bank 0
        TBSK
        LDD     #$0003                  ;at reset, the CSBOOT block size is 512K.
        STD     CSBARBT                 ;this line sets the block size to 64K since
                                        ;that is what physically comes with the EVB16
        LDAA    #$7F                    ;w=0, x=1, y=111111
        STAA    SYNCR                   ;set system clock to 16.78 MHz
        CLR     SYPCR                   ;turn COP (software watchdog) off,
                                        ;since COP is on after reset
```

```
         lds     #$f000

**** MAIN PROGRAM ****
         ldab    #$0f
         tbek                      ;use bank $0f for parameter RAM
         ldd     #$00cb
         std     CFSR2             ;Channel function select field (Note: function numbers
         ldd     #$0700            ;may vary for different mask sets)
         std     HSQR1             ;HSQR1, PSP in angle-time mode, PMM in count mode
         ldd     #$0f00
         std     CPR1              ;CPR1, high priority to both channels
```

8.3.2 PMM Initialization for Channel 4

```
         ldd     #$0004
         std     PRAM4_0    ;Channel control reg., detect rising edge
         ldd     #$0122
         std     PRAM4_1    ;MAX_MISSING = 1, NUM_OF_TEETH = 34
         ldd     #$a022
         std     PRAM4_3    ;RATIO=$a0, TCR2_MAX_VALUE=34
```

8.3.3 PSP Initialization for Channel 5

```
         ldd     #$4a01
         std     PRAM5_0           ;PERIOD_ADDRESS points to
                                   ;PERIOD_LOW_WORD of PMM
                                   ;Force pin high in force mode
         ldd     #$6000
         std     PRAM5_4           ;RATIO1 = $60, ANGLE1 = $00
         ldd     #$0200
         std     PRAM5_5           ;HIGH_TIME = $200
start
         ldd     #$0900
         std     HSRR1             ;host service initialization request for channels 4 and 5
finish
         bra     finish
```

9 Function Algorithm

If periods longer than $8000 are to be measured before generating a transition (HIGH_TIME > $8000, or period ∗ RATIO1 > $8000), the following method is used:

1. A match on (current time + $8000) is set, the PAC is set for no change on match, and the remaining period to be measured is stored in RATIO_TEMP.
2. When the match event occurs, a match on the value stored in RATIO_TEMP is then set and PAC configured such that the pin changes when the new match event occurs.

When the next angle to be matched is less than the last angle matched, the following steps are used:

1. A match on last angle plus $8000 is set; and
2. A match on the next angle is set.

The following paragraphs summarize the PSP time function. This 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.

PROCEDURE Next state(state){
    Switch(state){
        State *Trans_H_L*
            Set TBS to match TCR1, capture TCR1
            Set PAC to high to low

Assert channel flag0
Negate channel flag1
State *ANGLE1_gt_8000*
Set TBS to match TCR2, capture TCR1
Set PAC to no change
Assert channel flag1
State *TCR2_eq_ANGL1*
Set TBS to match TCR2, capture TCR1
Set PAC to no change
Negate channel flag0
Negate channel flag1
State *RATIO1_gt_8000*
Set TBS to match TCR1, capture TCR1
Set PAC to no change
Negate channel flag0
Assert channel flag1
State *Trans_L_H*
Set TBS to match TCR1, capture TCR1
Set PAC to low to high
Assert channel flag0
Negate channel flag1
State *ANGL1_gt_ANGL2*
Set TBS to match TCR2, capture TCR1
Set PAC to no change
Assert channel flag0
Assert channel flag1
State *TCR2_eq_ANGL2*
Set TBS to match TCR2, capture TCR1
Set PAC to no change
Negate channel flag0
Negate channel flag1
State *RATIO2_gt_8000*
Set TBS to match TCR1, capture TCR1
Set PAC to no change
Negate channel flag0
Assert channel flag1
}
}

The individual states of the PSP time function are described in detail in the following paragraphs.

9.1 State 1: *Init*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 10xxxx.

Match Enable: Disable

Summary:

This state is entered as the result of an HSR%10 for initialization. In this state, the channel latches are configured, and the following occurs for mode 0:

— If ANGLE1 – TCR2 > $8000, generate match on ANGLE2 + $8000 (next state: *Angle1_gt_$8000)*;
— Else generate match on ANGLE1 (next state: *TCR2_eq_Angle1*)

In mode 1, channel latches are configured, and the following occurs:

— If ANGLE1 – TCR2 > $8000, generate match on ANGLE1 + $8000 (next state: *Angle1_gt_$8000*);
— Else generate match on ANGLE1 (next state: *TCR2_eq_Angle1*)

Algorithm:

```
Set pin low
                                  /* The TPU accesses ANGLE1 and RATIO1 coherently */
                                               /* The TPU prepares first transition */
If ANGLE1 – TCR2 ≤ $8000 then {
    RATIO_TEMP = RATIO1, ANGLE1
    R2_A2_TEMP = RATIO2, ANGLE2                                  /* or HIGH_TIME */
    Generate match at time = ANGLE1
                                                /* next state (TCR2_eq_ANGL1) */
}
Else {
    If host sequence bit 0 = 1
    Then {                                              /* Mode 1 HIGH_TIME mode */
        Generate match at time = $8000 + ANGLE1
                                               /* next state(ANGLE1_gt_8000) */
    }
    Else {                                            /* Mode 0 ANGLE_ANGLE mode */
        Generate match at time = $8000 + ANGLE2
                                               /* next state(ANGLE1_gt_8000)*/
    }
}
```

9.2 State 2: *Trans_H_L*

9.3 State 3: *Angle1_gt_$8000*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001001.

Match Enable: Enable

State 2 Summary:

This state is entered when a high-to-low pin transition results from a match event involving TCR1. In this state the following occurs for mode 0:

— If ANGLE1 ≤ ANGLE2 generate match on ANGLE2 + $8000 (next state: *Angle1_gt_$8000*)
— Else generate match on ANGLE1 (next state: *TCR2_eq_Angle1*)

In mode 1, the following occurs:

— If ANGLE1 – TCR2 > $8000, generate match on ANGLE1 + $8000 (next state: *Angle1_gt_$8000*);
— Else generate match on ANGLE1 (next state: *TCR2_eq_Angle1*)

State 3 Summary:

This state is entered when no pin transition results due to a match event with TCR2. In angle-angle mode, this state is entered when ANGLE2 is greater than ANGLE1. In this state a match event on ANGLE1 is set up.

Algorithm (States 2 and 3):

```
If channel flag1 = 0 then {
                                                     /* Trans_H_L (state 2) */
    Assert interrupt request
    OLD_ANGLE2 = RATIO_TEMP                                       /* ANGLE2 */
    If host sequence bit 0 = 1 then {                             /* Mode 1 */
```

```
        If ANGLE1 – TCR2 ≤ $8000 then {
            RATIO_TEMP = RATIO1, ANGLE1
            R2_A2_TEMP = RATIO2, ANGLE2
            Generate match at time = ANGLE1
                                                    /* next state (TCR2_eq_ANGL1) */
        }
        Else {
            Generate match at time = $8000 + ANGLE1
                                                    /* next state (ANGLE1_gt_8000) */
        }
    }
    Else {                                                              /* Mode 0 */
        If ANGLE1 ≤ OLD_ANGLE2 then {
            Generate match at time = $8000 + OLD_ANGLE2
                                                    /* next state (ANGLE1_gt_8000) */
        }
        Else {                                  /* ANGLE1 > OLD_ANGLE2 and MODE 0 */
            RATIO_TEMP = RATIO1, ANGLE1
            R2_A2_TEMP = RATIO2, ANGLE2
                                  /* TCR2 has passed ANGLE1 and ANGLE1 > ANGLE2
                                     If TCR2 = FFFF, 0, 1 then TCR2 < OLD_ANGLE2
                                        If TCR2 = FFFF a match on ANGLE1 will be
                                           delayed a whole turn of TCR2, therefore
                                                generate an immediate match */
            If ((TCR2 – OLD_ANGLE2) < 0) and
            ((TCR2 + 1 – OLD_ANGLE2) < $8000)Then {
                Generate match at time = TCR2
            }                                       /* next state(TCR2_eq_ANGL1) */
            Else {
                Generate match at time = RATIO_TEMP                    /* ANGLE1 */
            }                                       /* next state(TCR2_eq_ANGL1) */
}
Else {
                              /* channel flag1 = 1: ANGLE1_gt_8000 (state 3) */
    RATIO_TEMP = RATIO1, ANGLE1
    R2_A2_TEMP = RATIO2, ANGLE2                                   /* or HIGH_TIME */
    Generate match at time = ANGLE1
                                                    /* next state (TCR2_eq_ANGL1) */
}
```

9.4 State 4: *TCR2_eq_Angle1*

9.5 State 5: *Ratio1_gt_$8000*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001000.

Match Enable: Enable

State 4 Summary:

This state is entered when no pin transition results due to a match event with TCR2. During this state the TPU calculates the offset from the occurrence of ANGLE1, or the TCR1 time of ANGLE1, to the rising transition of the pulse. This calculation is accomplished with the multiplication of the reference period (at PERIOD_ADDRESS) by RATIO1. If the offset is ≤ $8000, a match event is set up with the offset from the TCR1 time of ANGLE1 (next state: *Trans_L_H*). Else a match event is set up with the offset + $8000 from the TCR1 time of ANGLE1 (next state: *Ratio1_gt_$8000*).

State 5 Summary:

This state is entered when no pin transition results due to a match event with TCR1. In this state, a match event on the previous match value + $8000 is set up. When the match event occurs, a low-to-high pin transition results.

Algorithm (States 4 and 5):

```
If (channel flag1 = 0) then {
                                                        /* TCR2_eq_ANGL1 (state 4) */
    If host sequence bit 0 = 0 then ANGLE_TIME = ERT
    RESULT = RATIO_TEMP ∗ (PERIOD_ADDRESS) / 128
                                  /* RATIO1 byte of RATIO_TEMP used for multiplication */
    If (RESULT ≤ $8000) Then {
        Generate match at time = ERT + RESULT
                                                        /* next state (Trans_L_H) */
    }
    Else {                                              /* (RESULT > $8000) */
        Generate match at time = ERT + RESULT – $8000
                                                        /* next state (RATIO1_gt_$8000) */
    }
Else {
                                  /* channel flag1 = 1, RATIO1_gt_$8000 (state 5) */
    Generate match at time = MATCH_REGISTER + $8000
                                                        /* next state (Trans_L_H) */
}
```

9.6 State 6: *Trans_L_H*

9.7 State 7: *Angle1_gt_Angle2*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001011.

Match Enable: Enable

State 6 Summary:

This state is entered when a low-to-high pin transition results after a match event involving TCR1. In this state the following occurs for mode 0:

— If ANGLE1 = ANGLE2, calculate offset as described in *TCR2_eq_Angle2* and generate match (next state: see state *TCR2_eq_Angle2*)
— If ANGLE1 > ANGLE2, generate match on ANGLE1 + $8000 (next state: *Angle1_gt_Angle2*)
— Else generate match on ANGLE2 (next state: *TCR2_eq_Angle2*).

In mode 1, the following occurs:

— If HIGH_TIME > $8000, generate match on low-to-high transition time + HIGH_TIME + $8000 (next state: *Ratio2_gt_$8000*)
— Else generate match on low-to-high transition time + HIGH_TIME (next state: *Trans_H_L*).

State 7 Summary:

This state is entered, in angle-angle mode, when no pin transition results after a match with TCR2. In this state a match event on ANGLE2 is set up.

Algorithm (States 6 and 7):

```
If (host sequence bit 0 = 1) then {
                                          /* state = Trans_L_H (state 6) for Mode 1 */
    ERT = MATCH_REGISTER
    ANGLE_TIME = ERT
                                                        /* Store transition time */
```

```
        Assert interrupt request
        RESULT = R2_A2_TEMP                                        /* HIGH_TIME */
        If (RESULT ≤ $8000) then {
            Generate match at time = ERT + RESULT
                                                        /* next state (Trans_H_L) */
        }
        Else {                                               /* RESULT > $8000 */
            Generate match at time = ERT + RESULT – $8000
                                                   /* next state (RATIO2_gt_$8000) */
        }
    }
    Else {                                     /* (host sequence bit 0 = 0) Mode 0 */
        If (channel flag1 = 0) then {
                                                   /* Trans_L_H state for Mode 0 */
            Assert interrupt request
                  /* RATIO_TEMP = RATIO1, ANGLE1 and R2_A2_TEMP = RATIO2, ANGLE2 */
            If (RATIO_TEMP = R2_A2_TEMP) Then {                /* ANGLE1 = ANGLE2 */
    RATIO_TEMP = R2_A2_TEMP
                ERT = ANGLE_TIME
                RESULT = RATIO_TEMP ∗ @
                  (PERIOD_ADDRESS) / 128
                            /* RATIO2 byte of RATIO_TEMP used for multiplication*/
                TCR1_TEMP = TCR1
                ELAPSED_TIME = TCR1_TEMP – ANGLE_TIME
                MATCH_ABS = ANGLE_TIME + RESULT
                Goto COMPUTE FALLING EDGE
            }
            Else {                                            /* ANGLE1 - ANGLE2 */
                If (RATIO_TEMP > R2_A2_TEMP)
                    Then {                                    /* ANGLE1 > ANGLE2 */
                    Generate match at time = $8000 +
                      RATIO_TEMP                              /* $8000 + ANGLE1 */
                                                  /* next state(ANGL1_gt_ANGL2) */
                }
                Else {                                        /* ANGLE1 < ANGLE2 */
                    If ((TCR2 – RATIO_TEMP < 0) and
                     ((TCR2 + 1 – RATIO_TEMP < $8000) Then {
                                              /* see the dual case description
                                                        in the TRANS_H_L state */
                    RATIO_TEMP = R2_A2_TEMP
                Generate match at time = TCR2
            }
            Else {
                RATIO_TEMP = R2_A2_TEMP
                Generate match at time = RATIO_TEMP
            }
                                                   /* next state (TCR2_eq_ANGL2) */
        }
    Else {                    /* Channel flag1 = 1, state = ANGL1_gt_ANGL2 (state 7) */
        RATIO_TEMP = R2_A2_TEMP
        Generate match at time = RATIO_TEMP
                                                   /* next state (TCR2_eq_ANGL2) */
    }
}
```

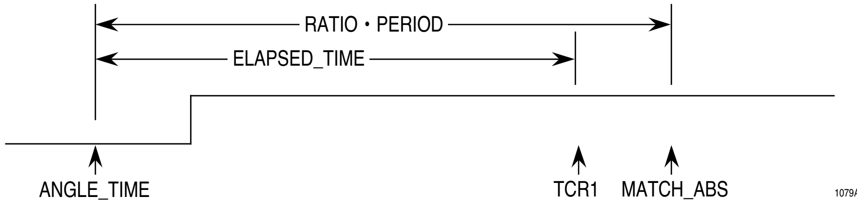

Figure 13 ANGLE1 = ANGLE2 Low to High Transition

Figure 13 illustrates the following process:

If ELAPSED_TIME ≥ RATIO2 ∗ PERIOD then generate immediate match
Else
 If MATCH_ABS – TCR1 > $8000 then generate match on MATCH_ABS – $8000
 Else generate match on MATCH_ABS

9.8 State 8: *TCR2_eq_Angle2*

9.9 State 9: *Ratio2_gt_$8000*

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

Match Enable: Enable

State 8 Summary:

This state is entered, in angle-angle mode, when no pin transition results after a match event with TCR2. During this state the TPU calculates the offset from the occurrence of ANGLE2, or the TCR1 time of ANGLE2, to the falling transition of the pulse. This calculation is accomplished by multiplying the reference period (at PERIOD_ADDRESS) by RATIO2. If the offset ≤ $8000, a match event is set up with the offset from the TCR1 time of ANGLE2 (next state: *Trans_H_L*). Else a match event is set up with the offset + $8000 from the TCR1 time of ANGLE2 (next state: *Ratio2_gt_$8000*).

State 9 Summary:

This state is entered when no pin transition results due to a match event with TCR1. In this state, a match event on the previous match value + $8000 is set up. When the match event occurs, a high-to-low pin transition results.

Algorithm (States 8 and 9):

```
If (channel flag1 = 0) then {
                                              /* TCR2_eq_ANGL2 (state 8) */
    RESULT = RATIO_TEMP(RATIO2) ∗ @(PERIOD_ADDRESS) / 128
    TCR1_TEMP = TCR1
    ELAPSED_TIME = TCR1_TEMP – ANGLE_TIME
    MATCH_ABS = ANGLE_TIME + RESULT
    Goto COMPUTE FALLING EDGE
}
Else {                       /* channel flag1 = 1, state: RATIO2_gt_8000 */
    Generate match at time = MATCH_REGISTER + $8000
                                              /* next state(Trans_H_L) */
}
```

9.10 State 10: *Immed_L*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 01xx0x.

Match Enable: Disable

Summary:

This state is entered as the result of an HSR for an immediate update (01) when the pin is low. This state causes an immediate update (recomputation) of the pulse if specific conditions (see **5 Host Interface to Function**) are met.

Algorithm:

```
If (host sequence bit 1 = 1) Then {                    /* change the falling time */
    R2_A2_TEMP = RATIO2, ANGLE2                        /* or HIGH_TIME */
}
Else {                                   /* change rising and falling times coherently */
    If (TCR2 < ANGLE1) Then {                          /* generate a match on ANGLE1 */
        Generate match at time = ANGLE1
                                                       /* access parameters coherently */
        RATIO_TEMP = ratio1, ANGLE1
        R2_A2_TEMP = RATIO2, ANGLE2                    /* or HIGH_TIME */
        Generate match at time = ANGLE1
                                                       /* next state (TCR2_eq_ANGL1) */
    }
    Else {
                                                       /* It's too late - do nothing */
    }
}
```

9.11 State 11: *Immed_H*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 01xx1x.

Match Enable: Disable

Summary:

This state is entered as the result of an HSR for an immediate update (01) when the pin is high. This state causes an immediate update (recomputation) of the falling edge of the pulse if specific conditions (see **5 Host Interface to Function**) are met.

Algorithm:

```
If (host sequence bit 1 = 0) Then {
    Do nothing                       /* immediate high service request is to be ignored */
}
Else If (host sequence bit 0 = 0) Then {                       /* MODE 0 */
                                                       /* access parameters coherently */
    NEW_A1 = ANGLE1
    NEW_A2 = ANGLE2
    R2_A2_TEMP = NEW_R2, NEW_A2
    RATIO_TEMP = NEW_R2, NEW_A2
                                                       /* update RATIO_TEMP */
    If (NEW_A2 – TCR2 < $8000) Then {
                                                       /* generate a match on ANGLE2 */
        Generate match at time = RATIO_TEMP
                                                       /* next state(TCR2_eq_ANGL2) */
    }
                                                       /* TCR2 > NEW_ANGLE2 */
    Else If (NEW_A1 > NEW_A2) Then {
        Generate match on OLD_A1 + $8000
                                                       /* next state (ANGL1_gt_ANGL2) */
    }
    Else If (NEW_A1 < NEW_A2) Then {
```

```
                                                /* immediate match */
        Generate match on TCR2
                                                /* next state (TCR2_eq_ANGL2) */
    }
    Else If (NEW_A1 = NEW_A2) Then {
                                                /* treat like A1 = A2 */
        RESULT = RATIO2 * (PERIOD_ADDRESS)/128
        TCR1_TEMP = TCR1
        ELAPSED_TIME = TCR1_TEMP – ANGLE_TIME
        MATCH_ABS = ANGLE_TIME + RESULT
        Goto COMPUTE FALLING EDGE
    }
}
Else {          /* host sequence bit 0 = 1: HIGH_TIME MODE */
    R2_A2_TEMP = RATIO2, ANGLE2                 /* HIGH TIME */
    If (MRL = 1 and Channel flag0 = 1) Then {
                                                /* Match on low to high detected */
        Goto state Trans_L_H_ Mode1
    }
    Else {
                                                /* sample TCR1 */
        TCR1_TEMP = TCR1
        ELAPSED_TIME = TCR1_TEMP – ANGLE_TIME
        MATCH_ABS = ANGLE_TIME + R2_A2_TEMP
    }
COMPUTE FALLING EDGE:
    If (ELAPSED_TIME ≤ R2_A2_TEMP) Then {
        Generate immediate match on TCR1
                                                /* next state (Trans_H_L) */
    }
    Else If (TCR1_TEMP – MATCH_ABS < $8000)Then
    {                       /* generate match on ANGLE_TIME + HIGH_TIME – 8000 */
        Generate match at time = MATCH_ABS – 8000
                                                /* next state (RATIO2_gt_8000) */
    }
    Else {                  /* generate match on ANGLE_TIME + HIGH_TIME */
        Generate match at time = MATCH_ABS
                                                /* next state (Trans_H_L) */
    }
}
```

9.12 State 12: *Force_Mode*

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 11xxxx.

Match Enable: Disable

Summary:

This state is entered as the result of an HSR %11 in response to an error condition from the input channel. The pin is set as indicated by CHANNEL_CONTROL. Further service requests due to match/transition events are ignored.

Algorithm:

```
Configure channel latches via CHANNEL_CONTROL
Disable service request (PAC is set to no change by CHANNEL_CONTROL)
```

The table below shows the PSP state transitions listing the service request sources and channel conditions from current state to next state. **Figure 14** illustrates the flow of PSP states for angle-angle mode, and **Figure 15** illustrates the flow of PSP states for angle-time mode. Each figure includes initialization and force modes.

Table 3 PSP State Transition Table

Current State	HSR	M/T	LSR	Pin	Flag0	Flag1	Next State
Any State	11	—	—	—	—	—	S 12 *Force_Mode*
Any State	10	—	—	—	—	—	S1 *Initialization*
Any State	01	—	—	1	—	—	S11 *Immed_H*
Any State	01	—	—	0	—	—	S10 *Immed_L*
S1 *Init*	00 00 00	1 1 1	— — —	0 0 1	1 0 1	1 0 0	S3 *Angle1_gt_$8000* S4 *TCR2_eq_Angle1* *S6 Trans_L_H*
S2 *Trans_H_L*	00 00 00	1 1 1	— — —	0 0 1	1 0 1	1 0 0	S3 *Angle1_gt_$8000* S4 *TCR2_eq_Angle1* *S6 Trans_L_H*
S3 *Angle1_gt_$8000*	00 00	1 1	— —	0 1	0 1	0 0	S4 *TCR2_eq_Angle1* S6 *Trans_L_H*
S4 *TCR2_eq_Angle1*	00 00	1 1	— —	0 1	0 1	1 0	S5 *Ratio1_gt_$8000* S6 *Trans_L_H*
S5 *Ratio1_gt_$8000*	00	1	—	1	1	0	S6 *Trans_L_H*
S6 *Trans_L_H*	00 00 00 00	1 1 1 1	— — — —	0 1 1 1	1 1 0 0	0 1 0 1	S2 *Trans_H_L* S7 *Angle1_gt_Angle2* *S8 TCR2_eq_Angle2* *S9 Ratio2_gt_$8000*
S7 *Angle1_gt_Angle2*	00 00	1 1	— —	1 0	0 1	0 0	S8 *TCR2_eq_Angle2* S2 *Trans_H_L*
S8 *TCR2_eq_Angle2*	00 00	1 1	— —	1 0	0 1	1 0	S9 *Ratio2_gt_$8000* S2 *Trans_H_L*
S9 *Ratio2_gt_$8000*	00	1	—	0	1	0	S2 *Trans_H_L*
S10 *Immed_L*	00	1	—	0	0	0	S4 *TCR2_eq_Angle1*
S11 *Immed_H*	00 00 00 00	1 1 1 1	— — — —	0 1 1 1	1 1 0 0	0 1 0 1	S2 *Trans_H_L* S7 *Angle1_gt_Angle2* *S8 TCR2_eq_Angle2* *S9 Ratio2_gt_$8000*
Unimplemented Conditions	00 00	0 —	— 1	— —	— —	— —	— —

NOTES:

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

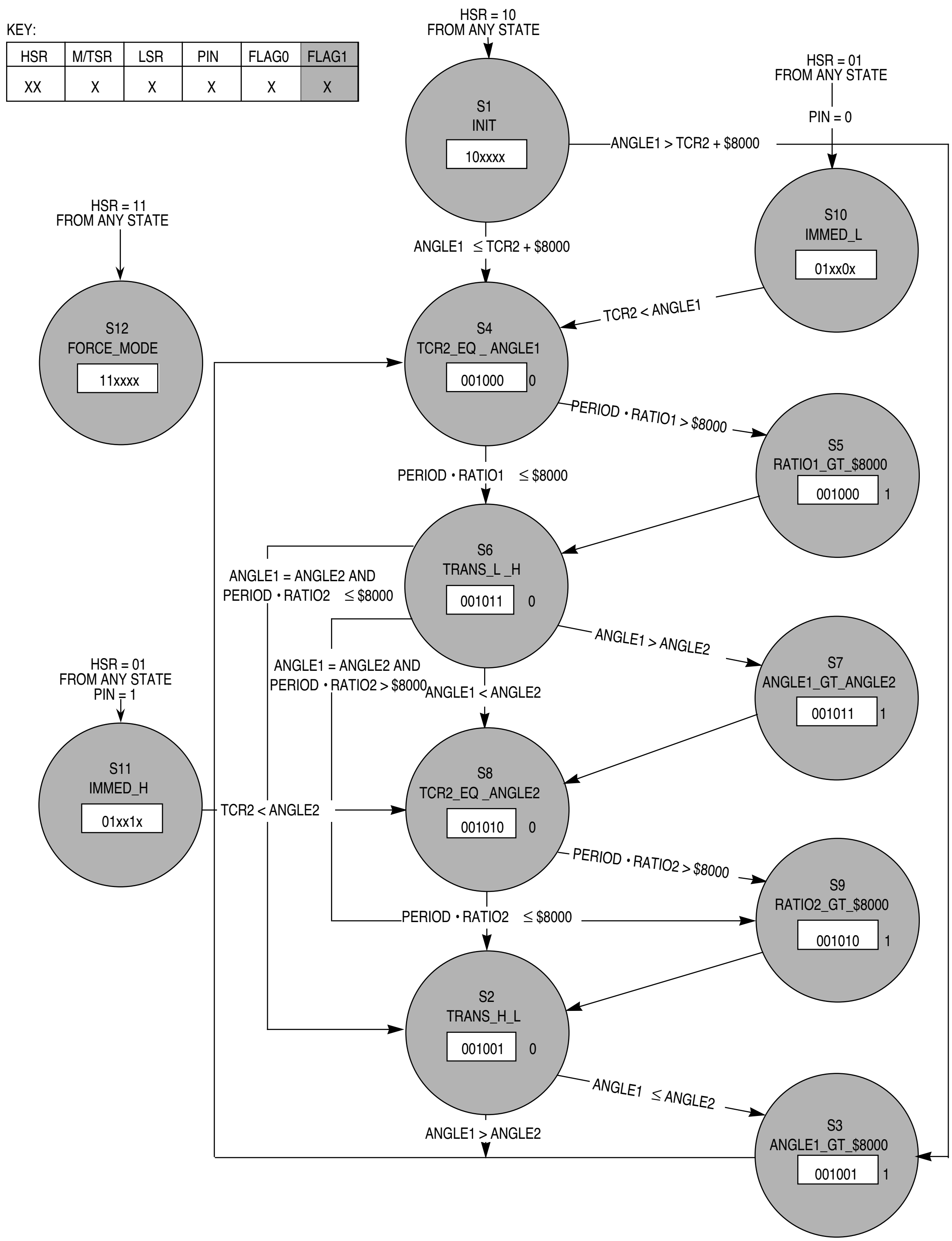

Figure 14 PSP Angle-Angle Mode State Flowchart

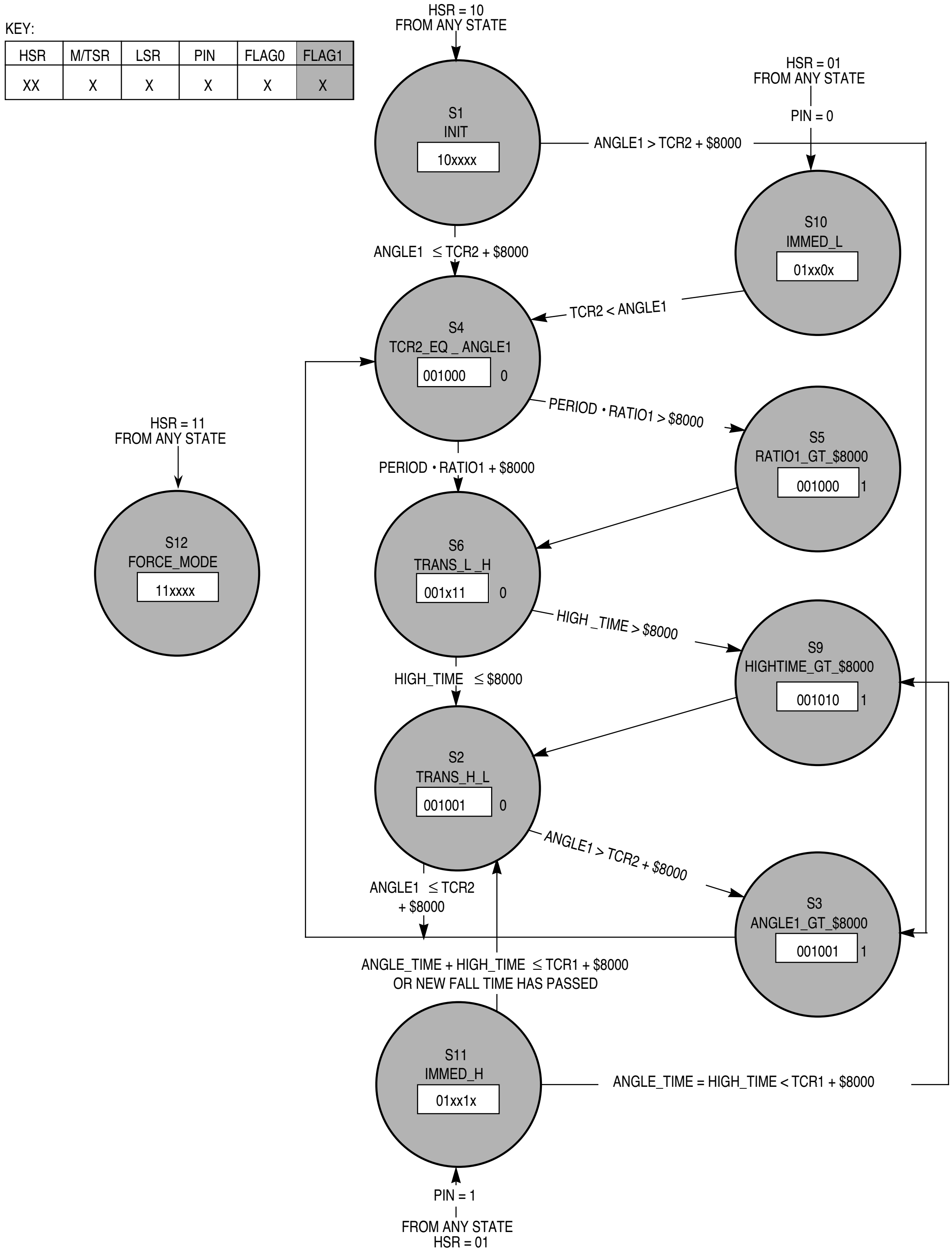

Figure 15 PSP Angle-Time Mode State Flowchart

NOTES

NOTES

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