MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this document
by TPUPN06/D

Programmable Time Accumulator TPU Function (PTA)

By Richard Soja

1 Functional Overview

This TPU input function measures the high time, low time or period of an input signal over a user defined number of periods, presenting the result to the host CPU in the form of a 32-bit accumulation.

2 Detailed Description

The programmable time accumulator (PTA) function measures either period, high time or low time of an input signal over a programmable number of periods. The number of periods over which the measurement is made is selectable over the range 1 to 255. Four modes of measurement are available, selected via the host sequence bits.

Mode 1 measures total high time over the selected number of periods.

Mode 2 measures total low time over the selected number of periods.

Mode 3 measures total period over the selected number of periods, starting on a rising edge.

Mode 4 measures total period over the selected number of periods, starting on a falling edge.

Figure 1 shows the four operating modes. All four examples in the figure are based on a MAX_COUNT value of seven. Shaded areas in the bar below each waveform indicate which parts of the waveform are actually measured. Lightly shaded areas are part of the next measurement.

The output of the PTA function is a 32-bit result expressed in TCR counts. The user can select either TCR1 or TCR2 as the timebase for the measurement.

The function operates continuously. After the specified number of periods has elapsed, the TPU updates the 32-bit result parameter, generates an interrupt request to the host CPU, and restarts the measurement process.

This function is very similar to the original PPWA function, although it has several major enhancements over that function:

1. 32-bit accumulation instead of 24 bit.
2. Option of high or low time measurement instead of high time only.
3. Option of starting period accumulation on rising or falling edge instead of rising only.
4. Better noise immunity.

The PTA function does not link to other TPU channels at the end of each accumulation. If this feature is required, the period/pulse width accumulation (PPWA) function should be used.

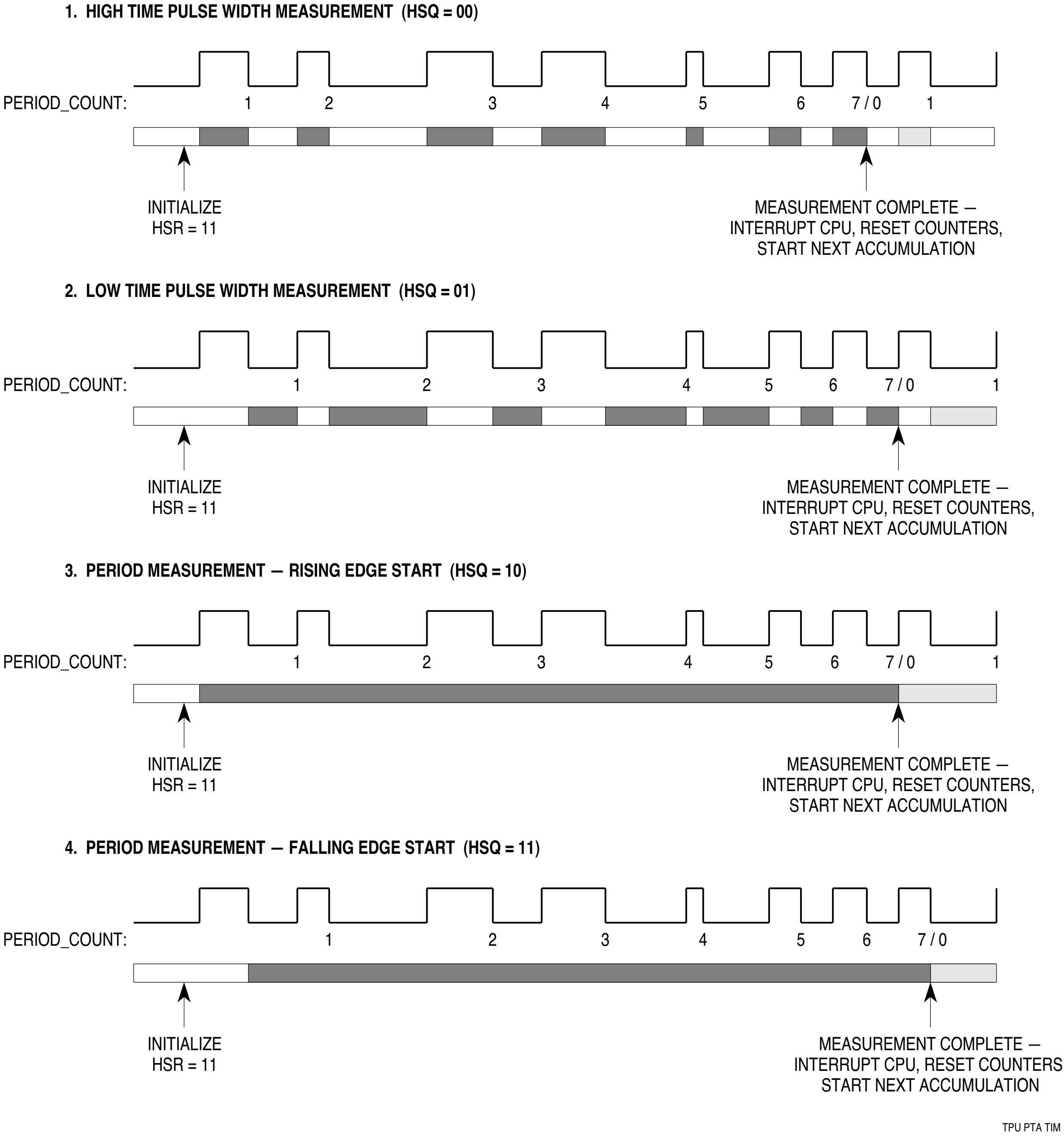


Figure 1 PTA Operating Modes

3 Function Code Size

Total TPU function code size determines what combination of functions can fit in a given ROM or emulation memory microcode space. The code size of the PTA function is:

55 µ instructions + 8 entries = **63 long words**

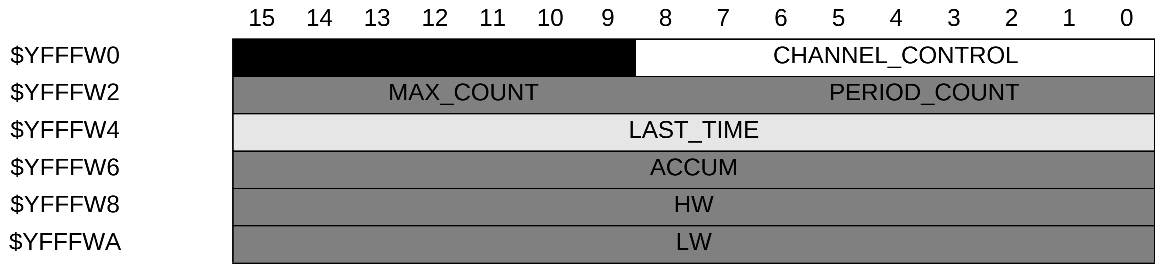
4 Function Parameters

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

Channel 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 2 TPU Channel Parameter RAM CPU Address Map



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$YFFFW0	■							CHANNEL_CONTROL								
$YFFFW2	MAX_COUNT								PERIOD_COUNT							
$YFFFW4	LAST_TIME															
$YFFFW6	ACCUM															
$YFFFW8	HW															
$YFFFWA	LW															

W = Channel Number

Parameter Write Access

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

Figure 3 Parameter RAM Assignment

4.1 CHANNEL_CONTROL

This 9-bit parameter is used during initialization by the TPU to configure the PTA channel. It defines the timebase which is to be used for measurement, and the type of input transition to be detected. The valid options for CHANNEL_CONTROL are shown in the following table. The correct operation of the PTA function is only guaranteed for the values of CHANNEL_CONTROL shown in **Table 1**.

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

CHANNEL_CONTROL must be written by the CPU before initialization. The following table defines the allowable data for this parameter.

Table 1 PTA CHANNEL_CONTROL Options

TBS	PAC	PSC	Action Taken
8765	**432**	**10**	
0 0 0 0	0 1 1	1 1	Measurement high pulse width with TCR1 timebase
0 0 0 0	0 1 1	1 1	Measurement low pulse width with TCR1 timebase
0 0 0 0	0 0 1	1 1	Measurement period on rising edge with TCR1 timebase
0 0 0 0	0 1 0	1 1	Measurement period on falling edge with TCR1 timebase
0 0 1 0	0 1 1	1 1	Measurement high pulse width with TCR2 timebase
0 0 1 0	0 1 1	1 1	Measurement low pulse width with TCR2 timebase
0 0 1 0	0 0 1	1 1	Measurement period on rising edge with TCR2 timebase
0 0 1 0	0 1 0	1 1	Measurement period on falling edge with TCR2 timebase
1 1 1 1	1 1 1	1 1	No change to channel configuration. Use when previously configured correctly

NOTE: Other values of CHANNEL_CONTROL may result in indeterminate operation.

4.1.1 MAX_COUNT

This 8-bit parameter is written by the CPU before initialization. It determines the number of periods or pulses which are accumulated before the measurement restarts. Any value in the range 0 to 255 is valid. A value of zero or one results in the accumulation of one period or pulse width.

4.1.2 PERIOD_COUNT

This 8-bit parameter is used by the TPU to count the number of input signal periods that have elapsed since the start of the last measurement sequence. When PERIOD_COUNT equals MAX_COUNT, a measurement sequence has been completed. Prior to starting the function with an initialization host service request, the CPU must initialize PERIOD_COUNT to zero.

4.1.3 LAST_TIME

The LAST_TIME parameter is a 16-bit value used by the TPU as temporary storage. It contains the event time of the latest input transition or accumulating match during active measurement of pulse width or period.

4.1.4 ACCUM

ACCUM is a 16-bit value used by the TPU as temporary storage for the incomplete pulse width or period measurement. This parameter is reset to zero by the TPU after each complete measurement sequence, but must be initialized to zero by the CPU prior to issuing the initialization host service request.

4.1.5 HW

This 16-bit parameter is updated by the TPU during the period or pulse width measurement. At the end of a measurement sequence, HW contains the upper word of the 32-bit accumulated result. HW should be cleared to zero by the CPU prior to issuing the initialization host service request. It must also be cleared after the result has been read, before the next accumulation exceeds 16 bits.

4.1.6 LW

This 16-bit parameter is updated by the TPU at the end of the period or pulse width measurement sequence to contain the lower word of the 32-bit accumulated result. LW is not cleared by the TPU during initialization. It must be cleared by the CPU prior to issuing the initialization host service request.

5 Function Initialization

Initialize the PTA function as follows.

1. Disable the intended PTA channel by clearing appropriate priority bits (unnecessary from reset).
2. Select the PTA function on the channel by writing the PTA function number to the channel function select bits.
3. Initialize MAX_COUNT, PERIOD_COUNT, ACCUM, HW, and LW in the channel parameter RAM.
4. Initialize CHANNEL_CONTROL and the channel host sequence bits to select the type of measurement to be made and the measurement timebase.
5. Issue an HSR%11 to initialize the channel and start the accumulation sequence.
6. Enable servicing of the channel by assigning the channel priority (H, M or L).

The TPU then executes the initialization state of the PTA function and starts accumulating period or pulse widths after the first valid start transition.

When the accumulation is complete, the TPU signals the host CPU via an interrupt request. The CPU must then read the 32-bit result from LW and HW (use a long word access for coherency), and reset HW to zero in preparation for the next accumulation. If required, MAX_COUNT can also be updated at this time to alter the number of periods over which the next accumulation will be made.

6 Performance and Use of Function

6.1 Performance

Like all TPU functions, the performance limit of the PTA function in a given application is dependent to some extent on the service time (latency) associated with activity on other TPU channels. This is due to the operational nature of the scheduler. In the case of the PTA function, this limits the maximum frequency and minimum pulse widths of the signal that can be properly measured.

Since the scheduler assures that the worst case latencies in any TPU application can be calculated, it is recommended that the guidelines given in the TPU reference manual are used along with the information given in the PTA function state timing table to perform an analysis on any proposed TPU application that appears to approach the performance limits of the TPU.

Table 2 PTA Function State Timing

State Number and Name	Max. CPU Clock Cycles	RAM Accesses by TPU
S0 INIT_PTA	6	1
S1 POS_TRANS0_PTA		
High time accumulate	54	6
Low time accumulate	54	6
Period accumulate — Rising	16	2
Period accumulate — Falling	16	2
S2 POS_TRANS1_PTA		
High time accumulate	18	2
Low time accumulate	12	1
Period accumulate — Rising	44	6
Period accumulate — Falling	44	6
S3 NEG_TRANS0_PTA		
High time accumulate	50	6
Low time accumulate	50	6
Period accumulate — Rising	16	2
Period accumulate — Falling	16	2
S4 NEG_TRANS1_PTA		
High time accumulate	12	1
Low time accumulate	22	2
Period accumulate — Rising	44	6
Period accumulate — Falling	44	6

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

6.2 Usage Notes and Restrictions

6.2.1 Clearing HW

At the end of the accumulation sequence, the 32-bit result of the PTA function is resident in HW and LW and the function issues an interrupt request to the CPU. The CPU must read the result and clear HW to zero before the next accumulation has exceeded 16 bits. This is because HW is updated during the measurement process, unlike LW, which is accumulated in ACCUM and then copied into LW at the end of the measurement.

6.2.2 Maximum Accumulation

The PTA function allows a maximum accumulation of 32 bits of the selected TCR clock. If HW overflows during the accumulation, an interrupt is generated to the CPU, but the function continues to run normally. Investigation of the parameters by the CPU may reveal that an overflow has occurred, but under some circumstances it may be difficult to tell this condition from a valid termination of a measurement sequence. For this reason, the prescaler of the selected timebase should be set to ensure that the longest measurement under worst case conditions does not exceed 32 bits.

6.2.3 Reading the Incomplete Accumulation

Under some circumstances, it may be advantageous to get an approximation of how far the active accumulation has progressed. This can be achieved by reading HW and ACCUM (LW) coherently, then reading PERIOD_COUNT to determine the number of periods over which the partial accumulation has been made. Note that PERIOD_COUNT may have an error of one with respect to the accumulated value read, and that the partial accumulation can be up to one pulse width/period or $8000 TCR clocks (if pulse width/period very long) less than the real value at the instant of reading.

6.2.4 Changing Modes

Avoid changing measurement modes while the function is running. The correct way to change modes is to stop the function by clearing the channel priority bits, then follow the procedure outlined under **5 Function Initialization** to restart the function in the new mode.

6.2.5 Noise Immunity

The PTA function is designed to filter out individual pulses which are too short to be measured correctly. These will not cause anomalous results in any of the measurement modes. However, repetitive noise on the input signal can cause anomalous results and also increased TPU activity, leading to an overall reduction in system performance. For this reason, every effort should be made to present the TPU with a noise free signal. Guaranteed minimum measurable pulse width or period can be determined by calculating worst-case latency for the PTA function. Refer to **6 Performance and Use of Function** for more information.

7 Host Interface to Function

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

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

Figure 4 TPU Address Map

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

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

CH	Interrupt Enable
0	Channel interrupts disabled
1	Channel interrupts enabled

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

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

CFS[4:0] — PTA 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	Operating Mode
00	High Time Accumulate
01	Low Time Accumulate
10	Period Accumulate — Rising
11	Period Accumulate — Falling

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	No effect
10	No effect
11	Initialize Function

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

8 Examples

The following two examples contain CPU32 code needed to initialize the PTA function for low pulse measurements and period on rising edge measurements using the TCR1 timebase. The code includes the initialization and use of basic interrupt handlers for these measurement modes. The assembled code was executed on a BCC board.

8.1 Example A: Low Pulse Measurement

This example shows how to initialize the PTA function for low pulse measurement and how to implement an interrupt handler. The channel generates an interrupt service request after each low pulse is measured. On every interrupt the handler logs all the PTA parameter RAM to BCC external RAM, between addresses $6000 and $6FFF. In addition, the interrupt handler also ensures the HW result if the function is cleared to zero after it is logged. The data log pointer is a word maintained at address $5FFE. In this example, the PTA function has been assembled as function number $F, but the function number can be different, depending on the application.

8.2 Parameter RAM Content

After the CPU has initialized the parameter RAM for low pulse measurement, and before the host service request is issued, the parameter RAM content should be as follows:

	15							8	7							0
$YFFF10	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
$YFFF12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
$YFFF16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$YFFF18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$YFFF1A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After the PTA function generates an interrupt service request to indicate that a low pulse measurement has been made, the parameter RAM content should be as follows:

	15							8	7							0
$YFFF10	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
$YFFF12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
$YFFF16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
$YFFF18	High Word of Measured Low Pulse															
$YFFF1A	Low Word of Measured Low Pulse															

8.3 Program Listing

```
TPRAM0          equ $ffffff00    TPU Channel 0 parameter RAM base address
TMCR            equ $fffffe00    TPU Module Configuration
TICR            equ $fffffe08    TPU Interrupt Configuration
CIER            equ $fffffe0a    TPU Channel Interrupt Enable
CFSR3           equ $fffffe12    TPU Channel Function Select Register 3
HSQR1           equ $fffffe16    TPU Host Sequence Register 1
HSRR1           equ $fffffe1a    TPU Host Service Request Register 1
CPR0            equ $fffffe1c    TPU Channel Priority Register 0
CPR1            equ $fffffe1e    TPU Channel Priority Register 1
CISR            equ $fffffe21    TPU Channel Interrupt Status
log_count       equ $5FFE        Data log counter.

* Channel Function assignments
PTA            equ TPRAM0

* PTA Parameter RAM constants
channel_control equ $000F        Detect rising/falling edges, capture and match on
TCR1.
max_count      equ 1             Number of periods to accumulate.

          section.text
* Set up data log pointer
          move.w #$6000,log_count

          clr.l d0
          movec d0,vbr
          move.w #$04C0,TMCR     Ensure TPU is in emulation mode, 240 ns res.

* First set up PTA channel configuration
          clr.l CPR0             Ensure scheduler disabled before setting parameters
          move.w #$000F,CFSR3    PTA on Chan 0.

* Now initialize PTA parameter RAM on Channel 0
*       Addr offset      Parameter Name         Parameter Position
*           00           channel_control            9 LSbits
*           02           max_count                  MS byte
*           02           period_count               LS byte
*           04           last_time                  Word
*           06           accum                      Word
*           08           HW                         Word
*           0A           LW                         Word

          move.w #channel_control,PTA   Timebase select
          move.w #(max_count<<8),PTA+2 Number of periods to accumulate
          clr.w PTA+6                   TPU doesn't clear Accum.
          clr.l PTA+8                   TPU doesn't initialize HW:LW Result.

* Host sequence bits 0,1 dictate Pulse-High/Low or Period-Rising/Falling measurement.
          move.w #$0001,HSQR1    Low Pulse measurement.

* Now set up and enable PTA interrupt
          ori.w #1,TMCR          Set IARB to non-zero value, and
          move.w #$0640,TICR     TPU intr level 6, Base vector num=$40.
          move.l #PTASRV,$40*4   Set up PTA vector (Base=$40, channel 0)
          move.w #$0001,CIER     Enable chan 0 interrupt (PMM)
          move.w #$2500,SR       Drop intr level below TPU's

* Send Host service request and start PTA
          move.w #$0003,HSRR1    Host service init%11
```

```
          move.w #$0003,CPR1      Enable scheduler for PTA

          bra *                   TPU should be running now, CPU in idle loop

**********************************************************************
* Interrupt function: PTASRV                                         *
* Description: Entered when bit 0 of CISR register is set.          *
*              This happens when the PTA function has               *
*              completed the accumulated measurement.               *
*              PTA Parameter RAM on channel 0 is logged             *
*              and whole 32-bit result is cleared if low            *
*              word of result is non-zero.                          *
*Input conditions: bit 0 of CISR is'1'.                             *
*                  HW:LW holds measurement result.                  *
*Output conditions: bit 0 of CISR is'0'                             *
*                   (provided CPU interrupt latency is not          *
*                   exceeded).                                      *
*                   HW:LW is cleared to zero.                       *
**********************************************************************

PTASRV     equ *                  PTA interrupt service routine
          bclr #0,CISR+1          Clear PTA status flag

log_all   bsr ptalog              Log data
          tst.w PTA+$A            If LW Result < > 0 Then
          beq log_ex
log_clr   clr.l PTA+8             clear whole 32-bit result, since TPU doesn't.

log_ex    equ *
          rte                     and return

**********************************************************************
* Subroutine: ptalog                                                 *
* Description: If data space is available, 8 words of               *
*              channel 0 parameter are logged in next 8             *
*              word locations.                                      *
* Input conditions: log_count points to next free data log          *
*                   location, or is greater than or equal           *
*                   to the limit if no data space                   *
*                   available.                                      *
* Output conditions: log_count incremented by 16, or is             *
*                    greater than or equal to the limit if          *
*                    no more data space available.                  *
*                    CPU registers a2, a3, d2 changed               *
*                                                                   *
**********************************************************************

ptalog    move.w log_count,a2
          cmpa #$7000,a2         If log limit reached then
          bge ptalogex           do nothing, else
          move.w #TPRAM0,a3      Get start address of required data
          moveq #7,d2
loop      move.w (a3)+,(a2)+     and log parameters.
          dbra d2,loop
          move.w a2,log_count    Save next log position
ptalogex rts                    and return
         end
```

8.4 Example B: Rising Edge Period Measurement

This example shows how to initialize the PTA function for period measurement on each rising edge, and how to set up and implement an interrupt handler. On every interrupt service request, the handler logs all the PTA parameter RAM to BCC external RAM, between addresses $6000 and $6FFF. In addition, the interrupt handler also ensures the HW result of the function is cleared to zero after it is logged. The data log pointer is a word maintained at address $5FFE. In this example, the PTA function has been assembled as function number $F, but the function number can be different, depending on the application.

8.5 Parameter RAM Content

After the CPU has initialized the parameter RAM for period measurement on rising edge, and before the host service request is issued, the parameter RAM content should be as follows:

	15							8	7							0
$YFFF10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
$YFFF12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
$YFFF16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$YFFF18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$YFFF1A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After the PTA function generates an interrupt service request to indicate that a period measurement has been made, the parameter RAM content should be as follows:

	15							8	7							0
$YFFF10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
$YFFF12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
$YFFF14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
$YFFF16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
$YFFF18	High Word of Measured Period															
$YFFF1A	Low Word of Measured Period															

8.6 Program Listing

```
TPRAM0          equ $ffffff00   TPU Channel 0 parameter RAM base address
TMCR            equ $fffffe00   TPU Module Configuration
TICR            equ $fffffe08   TPU Interrupt Configuration
CIER            equ $fffffe0a   TPU Channel Interrupt Enable
CFSR3           equ $fffffe12   TPU Channel Function Select Register 3
HSQR1           equ $fffffe16   TPU Host Sequence Register 1
HSRR1           equ $fffffe1a   TPU Host Service Request Register 1
CPR0            equ $fffffe1c   TPU Channel Priority Register 0
CPR1            equ $fffffe1e   TPU Channel Priority Register 1
CISR            equ $fffffe21   TPU Channel Interrupt Status
log_count       equ $5FFE       Data log counter.

* Channel Function assignments
PTA            equ TPRAM0

* PTA Parameter RAM constants
```

```
channel_control equ $0007        Detect only rising edges, capture and match on TCR1.
max_count       equ 1            Number of periods to accumulate.

          section.text

* Set up data log pointer
          move.w #$6000,log_count

          clr.l d0
          movec d0,vbr           Necessary for BDM downloads.
          move.w #$04C0,TMCR     Ensure TPU is in emulation mode, 240 ns res.

* First set up PTA channel configuration

          clr.l CPR0             Ensure scheduler disabled before setting parameters
          move.w #$000F,CFSR3    PTA on Chan 0.

* Now initialize PTA parameter RAM on Channel 0
*       Addr offset      Parameter Name         Parameter Position
*           00           channel_control            9 LSbits
*           02           max_count                  MS byte
*           02           period_count               LS byte
*           04           last_time                  Word
*           06           accum                      Word
*           08           HW                         Word
*           0A           LW                         Word

          move.w #channel_control,PTA          Timebase select
          move.w #(max_count<<8),PTA+2         Number of periods to accumulate
          clr.w PTA+6                          TPU doesn't clear Accum.

          clr.l PTA+8                          TPU doesn't initialize HW:LW Result.

* Host sequence bits 0,1 dictate Pulse-High/Low or Period-Rising/Falling
* measurement.
          move.w #$0002,HSQR1                  Period measurement on rising edges.

* Now set up and enable PTA interrupt

          ori.w #1,TMCR          Set IARB to non-zero value, and
          move.w #$0640,TICR     TPU intr level 6, Base vector num=$40.
          move.l #PTASRV,$40*4   Set up PMM vector (Base=$40, channel 0)
          move.w #$0001,CIER     Enable chan 0 interrupt (PMM)
          move.w #$2500,SR       Drop intr level below TPU's intr level

* Send Host service request and start PTA
          move.w #$0003,HSRR1    Host service init%11
          move.w #$0003,CPR1     Enable scheduler for PTA

          bra *                  TPU should be running now!

************************************************************
* Interrupt function: PTASRV                               *
* Description: Entered when bit 0 of CISR register is set *
*             This happens when the PTA function has      *
*             completed the accumulated measurement.      *
*             PTA Parameter RAM on channel 0 is logged    *
*             and whole 32-bit result is cleared if low   *
*             word of result is non-zero.                 *
*Input conditions: bit 0 of CISR is'1'.                  *
```

```
*                  HW:LW holds measurement result.         *
*Output conditions: bit 0 of CISR is'0'                   *
*                   (provided CPU interrupt latency is not *
*                   exceeded).                             *
*                   HW:LW is cleared to zero.              *
************************************************************
PTASRV    equ *                 PTA interrupt service routine
          bclr #0,CISR+1        Clear PTA status flag

log_all   bsr ptalog            Log data
          tst.w PTA+$A          If LW Result < > 0 Then
          beq log_ex
log_clr   clr.l PTA+8           clear whole 32-bit result, since TPU doesn't.

log_ex    equ *
          rte                   and return

************************************************************
* Subroutine: ptalog                                       *
* Description: If data space is available, 8 words of      *
*              channel 0 parameter are logged in next 8    *
*              word locations.                             *
* Input conditions: log_count points to next free data log *
*                   location, or is greater than or equal *
*                   to the limit if no data space          *
*                   available.                             *
* Output conditions: log_count incremented by 16, or is    *
*                    greater than or equal to the limit if *
*                    no more data space available.         *
*                    CPU registers a2, a3, d2 changed      *
*                                                          *
************************************************************
ptalog    move.w log_count,a2
          cmpa #$7000,a2        If log limit reached then
          bge ptalogex          do nothing, else
          move.w #TPRAM0,a3     Get start address of required data
          moveq #7,d2
loop      move.w (a3)+,(a2)+    and log parameters.
          dbra d2,loop
          move.w a2,log_count   Save next log position
ptalogex rts                    and return

          end
```

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 on downloading and compiling microcode.

The programmable time accumulator function consists of five states. The function uses matches in addition to transition detection to measure the desired signal property. Using matches in this way extends the maximum period and pulse widths that can be successfully measured beyond the $FFFF TCR count limit. For clarity, the following description refers to an internal flag that is not available to the user. Flag0 tracks the pin state from the previous channel service, to ensure correct startup in the various measurement modes. Flag0 is used in conjunction with the new pin state, shared match/transition flag and the host service request bits to determine which of the PTA function states are executed.

9.1 State 0: INIT_PTS

This state, entered as a result of HSR %11, configures the channel.
Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 11XXXX

The channel is configured with the contents of the CHANNEL_CONTROL parameter
If pulse measurement has been selected via the host sequence bits, flag0 is set, otherwise it is cleared
Negate MRL, TDL, LSL
The state ends

9.2 State 1: POS_TRANS0_PTA

This state is entered as a result of a positive transition or a match when the pin is one and flag0 is zero.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001X10

```
If Pulse Measurement Then
    If Match Then                                          {High Pulse assumed}
        ACCUM_TIME
        NEXT_MATCH
    Else                                          {Low Pulse, Transition assumed}
        Check_Pin
        If Pin = 1 Then
            Flag0:= 1
            ACCUM_TIME
            CHECK_COUNT
        Endif
    Endif
    Else                                                   {Period Measurement}
        If Match Then                  {Possible pending match from previous mode}
            Clear Match Flag
        Else
            Clear Transition Flag
            Flag0:= 1
            NEXT_MATCH
    Endif
Endif
```

9.3 State 2: POS_TRANS1_PTA

This state is entered as a result of a positive transition or a match when the pin is one and flag0 is one.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001X11

```
If Pulse Measurement Then
    If Match Then
        Clear Match Flag
    Else
        Check_Pin
        If High Pulse Then
            If Pin = 1 Then
            Flag0:= 0
            NEXT_MATCH
            Endif
        Endif
    Endif
Else                                                       {Period Measurement}
    If Match Then
```

```
        ACCUM_TIME
        NEXT_MATCH
    Else
        ACCUM_TIME
        CHECK_COUNT
        NEXT_MATCH
    Endif
Endif
```

9.4 State 3: NEG_TRANS0_PTA

This state is entered as a result of a negative transition or a match when the pin is zero and flag0 is zero.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001X00

```
If Pulse Measurement Then
    If Match Then                                              {Low Pulse assumed}
        ACCUM_TIME
        NEXT_MATCH
    Else                                          {High Pulse, Transition assumed}
                               {High Pulse Match is Invalid State and not processed here}
        Check_Pin
        If Pin = 0 Then
            Flag0:= 1
            ACCUM_TIME
            CHECK_COUNT
        Endif
    Endif
Else                                         {Period Measurement, Transition assumed}
    If Match Then{                           {Possible pending match from previous mode}
        Clear Match Flag
    Else
        Flag0:= 1
        NEXT_MATCH
        Clear Transition Flag
    Endif
Endif
```

9.5 State 4: NEG_TRANS1_PTA

This state is entered as a result of a negative transition or a match when the pin is zero and flag0 is one.

Condition: HSR1, HSR0, M/TSR, LSR, Pin, Flag0 = 001X01

```
If Pulse Measurement Then
    If Match Then
        Clear Match Flag
    Else
        Check_Pin
        If Low Pulse Then
            If Pin = 0 Then
            Flag0:= 0
            NEXT_MATCH
            Endif
        Endif
    Endif
Else                                                       {Period Measurement}
```

```
    If Match Then
        ACCUM_TIME
        NEXT_MATCH
    Else
        ACCUM_TIME
        CHECK_COUNT
        NEXT_MATCH
    Endif
Endif
```

9.6 Pseudocode Subroutines

ACCUM_TIME

```
ACCUM:= ACCUM + (EVENT_TIME – LAST_TIME)
If Overflow Then
    HW:= HW + 1
    If Overflow Then
        Generate Interrupt to CPU
    Endif
Endif
```

NEXT_MATCH

```
LAST_TIME:= EVENT_TIME
Schedule Match at (EVENT_TIME + Max)
```

CHECK_COUNT

```
PERIOD_COUNT:= PERIOD_COUNT + 1
If PERIOD_COUNT ≥ MAX_COUNT Then
    LW:= ACCUM
    Generate Interrupt to CPU
    ACCUM:= 0
    PERIOD_COUNT:= 0
Endif
```

Check_Pin

```
    Get new pin level and coherently clear transition flag
```

NOTES

NOTES

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

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.

MOTOROLA