Application Note

AN2365/D
Rev. 0, 10/2002

Using the Programmable Time Accumulator TPU Function (PTA) with the MPC500 Family

Stan Ostrum

Metrowerks

This TPU Programming Note is intended to provide simple C interface routines to the programmable time accumulator TPU function (PTA). [1] The routines are targeted for the MPC500 family of devices, but they should be easy to use with any device that has a TPU.

1 Functional Overview

This TPU input function measures the total high time, low time or period of an input signal over a user defined number of periods, and presents 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:

- 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 are based on an accumulation of seven periods, which are numbered below each waveform. Dark shaded areas in the bar below each waveform indicate which parts of the waveform are actually measured. Unshaded areas are not included in the measurement. Lightly shaded areas are part of the next measurement cycle.

The output of the PTA function is a 32-bit result, which is expressed in counts of the selected TPU timebase. The user can select either TCR1 or TCR2 as the timebase for the measurement. The number of TCR counts that have been accumulated multiplied by the period of one TCR count will give the total measurement expressed in seconds.

[1]The information in this Programming Note is based on TPUPN06. It is intended to compliment the information found in that Programming Note.

The PTA 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 process for the next measurement cycle.

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.

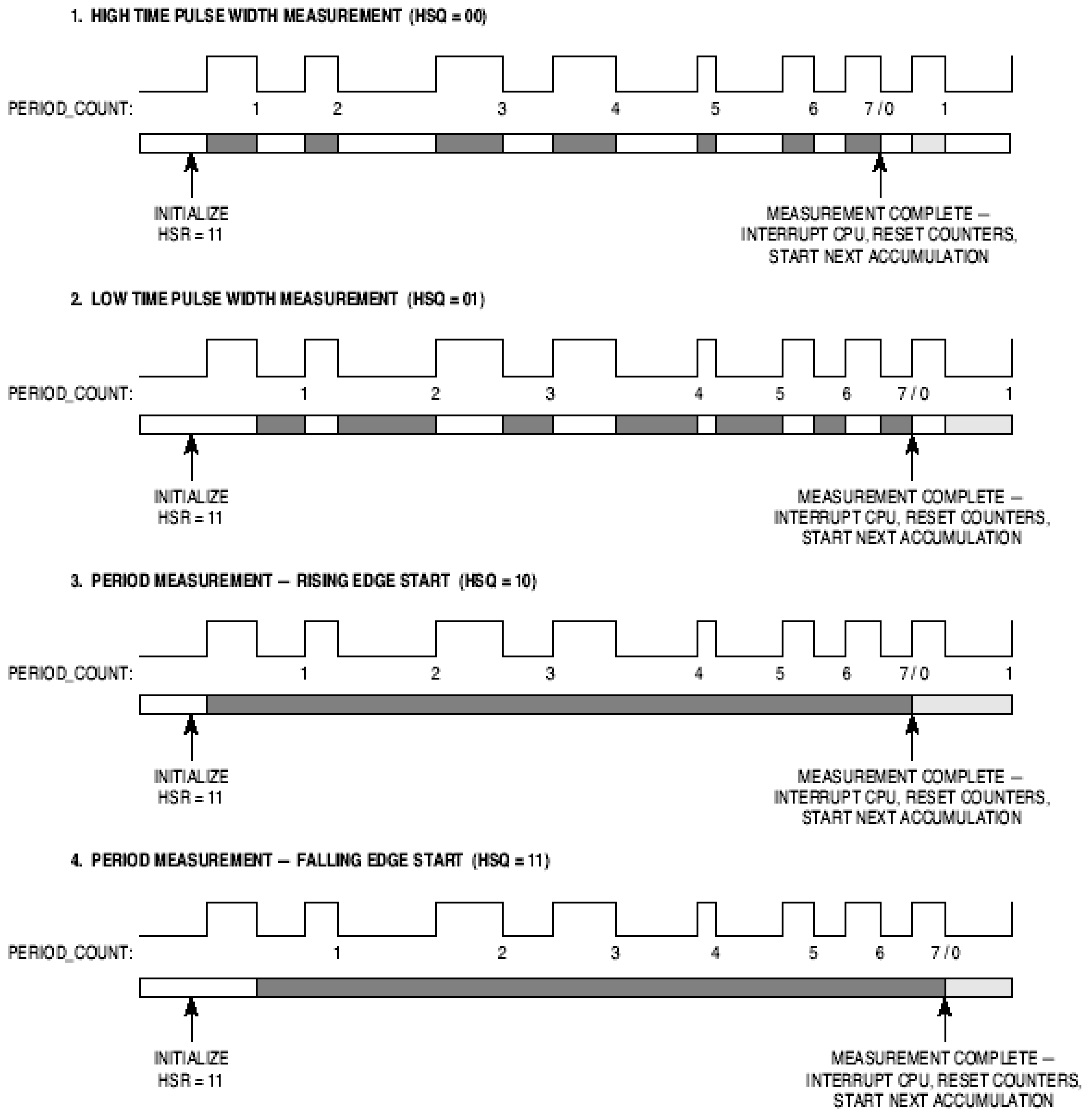
1. HIGH TIME PULSE WIDTH MEASUREMENT (HSQ = 00)
PERIOD_COUNT:
1 2 3 4 5 6 7 / 0 1
INITIALIZE HSR = 11
MEASUREMENT COMPLETE – INTERRUPT CPU, RESET COUNTERS, START NEXT ACCUMULATION
2. LOW TIME PULSE WIDTH MEASUREMENT (HSQ = 01)
PERIOD_COUNT:
1 2 3 4 5 6 7 / 0 1
INITIALIZE HSR = 11
MEASUREMENT COMPLETE – INTERRUPT CPU, RESET COUNTERS, START NEXT ACCUMULATION
3. PERIOD MEASUREMENT – RISING EDGE START (HSQ = 10)
PERIOD_COUNT:
1 2 3 4 5 6 7 / 0 1
INITIALIZE HSR = 11
MEASUREMENT COMPLETE – INTERRUPT CPU, RESET COUNTERS, START NEXT ACCUMULATION
4. PERIOD MEASUREMENT – FALLING EDGE START (HSQ = 11)
PERIOD_COUNT:
1 2 3 4 5 6 7 / 0
INITIALIZE HSR = 11
MEASUREMENT COMPLETE – INTERRUPT CPU, RESET COUNTERS, START NEXT ACCUMULATION

Figure 1. PTA Operating Modes

2.1 PTA C Level API

Rather then controlling the TPU registers directly, the PTA routines in this TPU Programming Note may be used to provide a simple and easy interface. There are 3 routines for controlling the PTA function in 2 files (tpu_pta.h and tpu_pta.c). The tpu_pta.h file should be included in any files that use the routines. This file contains the function prototypes and useful #defines. Each of the routines in tpu_pta.c will be looked at in detail, the routines are:

- Initialization Function:
  — void tpu_pta_init(struct TPU3_tag *tpu, UINT8 channel, UINT8 priority, UINT8 timebase, UINT8 mode, UINT8 max_count);

- Return Value functions:
  - UINT32 tpu_pta_get_accumulation(struct TPU3_tag *tpu, UINT8 channel);
  - UINT8 tpu_pta_get_period_count(struct TPU3_tag *tpu, UINT8 channel);

2.1.1 void tpu_pta_init

This function is used to initialize a channel to run the PTA function. This function has 6 parameters:

- *tpu - This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h.
- channel - This is the TPU channel number of the PTA channel. This parameter should be assigned a value of 0 to 15.
- priority - This is the priority to assign to the channel. This parameter should be assigned a value of TPU_PRIORITY_HIGH, TPU_PRIORITY_MIDDLE or TPU_PRIORITY_LOW. The TPU priorities are defined in mpc500_utils.h.
- timebase – This is the TPU timebase to use for PTA measurements. This parameter should be assigned a value of TPU_PTA_TCR1 or TPU_PTA_TCR2.
- mode – This is the mode to use for PTA measurements. This parameter should be assigned a value of TPU_PTA_HIGH_TIME_ACCUM, TPU_PTA_LOW_TIME_ACCUM, TPU_PTA_PERIOD_ACCUM_RISE, or TPU_PTA_PERIOD_ACCUM_FALL.
- max_count – This is the number of periods or pulses that are accumulated before the measurement restarts. This parameter should be assigned a value in the range 0 to 255. A value of zero or one results in the accumulation of one period or pulse width.

Care should be taken when initializing TPU channels. The TPU’s behavior may be unpredictable if a channel is reconfigured while it is running. The channels should be stopped before they are configured. This is done by setting the channel’s priority to disabled. If the channel is currently being serviced when the priority is set to disable it will continue to service the channel until the state ends. To make sure the channel is not being serviced, you need to wait for the longest state execution time after disabling the channel. All channels are disabled out of reset so the channels can be configured immediately from reset.

The *tpu_pta_init* function attempts to wait between the disabling of the channels before it starts configuring them, however the actual execution speed of the code will be depend on the specific system. If you are not configuring the channels from reset, then ideally it is best to have the functions disabled before calling this function. TPU channels can be disabled by using the *tpu_disable* function in the mpc500_utils.c file. For example, disabling channel 0 is done like this: tpu_disable(tpu, 0);

2.1.2 UINT32 tpu_pta_get_accumulation

This function returns the current value of the 32-bit accumulation. This function has 2 parameters:

- *tpu – This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h.
- channel – This is the TPU channel number of the PTA channel. This parameter should be assigned a value of 0 to 15.

This function is normally called when the TPU signals the host CPU via an interrupt request that the accumulation is complete. The CPU must execute this call and read the result before the next accumulation has exceeded 16 bits.

2.1.3 UINT8 tpu_pta_get_period_count

This function returns the current value of the 8-bit period count parameter. This parameter is used by the TPU to count the number of input signal periods that have elapsed since the start of the last accumulation sequence. When period_count equals max_count, a measurement sequence has been completed. This function has 2 parameters:

- *tpu – This is a pointer to the TPU3 module to use. It is of type TPU3_tag which is defined in m_tpu3.h.
- channel – This is the TPU channel number of the PTA channel. This parameter should be assigned a value of 0 to 15.

3 Programmable Time Accumulator Examples

The following examples show configuration of the programmable time accumulator function for both period and pulse width measurement. Each example is a C program that shows how to configure and use the PTA interface routines.

3.1 Example 1

3.1.1 Description

This sample program show a simple PTA example that measures the total period over a selected number of periods, starting on a rising edge, of an input signal connected to channel 0 on TPUA. A simple interrupt handler is used to trigger the acquisition of the completed measurement.

The program then runs in an infinite loop, continuing to read back each accumulated measurement from the input on TPUA channel 0.

3.1.2 Program

```
/**************************************************************************/
/* FILE NAME: tpu_pta_example1.c              COPYRIGHT (c)  2002 */
/* VERSION: 1.1                                   All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION: This sample program show a simple PTA example that        */
/* measures the total period over a selected number of periods, starting  */
/* on a rising edge, of an input signal connected to channel 0 on TPUA.   */
/* A simple interrupt handler is used to trigger the acquisition of the   */
/* completed measurement.                                                 */
/*                                                                        */
/* The program is targeted for the MPC555 but should work on any MPC500   */
/* device with a TPU. For other devices the setup routines will also need */
/* to be changed.                                                         */
/*========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Stan Ostrum                         */
/*                                                                          */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                    */
/* ---   -----------  ---------    ---------------------                    */
/* 1.0   Stan Ostrum  03 Sep 02    Initial version of function.             */
/* 1.1   Stan Ostrum  17 Sep 02    Revised after review.                    */
/***************************************************************************/


#include "mpc555.h"         /* Define MPC555 registers, this needs to be  */
                            /* changed if other MPC500 devices are used.  */
#include "mpc500_util.h"    /* Utility routines for using MPC500 devices  */
#include "tpu_pta.h"        /* TPU PTA functions */

UINT32   accumulation;      /* global variable for period/PW accumulation */

void main ()
{
    struct TPU3_tag *tpua = &TPU_A;         /* pointer for TPU routines */

    setup_555();                 /* Metrowerks setup routine for MPC555 */

    /* initialize Prog. Time Accum function with:   */
    /*     - Input signal on TPU A channel 0         */
    /*     - Use TCR1 timebase                       */
    /*     - Measure periods, start with rising edge */
    /*     - Max count value of 10                   */
    /*     - Schedule as high priority in the TPU    */

    tpu_pta_init(tpua, 0, TPU_PRIORITY_HIGH, TPU_PTA_TCR1, \
                 TPU_PTA_PERIOD_ACCUM_RISE, 10);

    tpu_interrupt_enable(tpua, 0);          /* enable PTA interrupts */

    while (1) {                             /* loop and measure */
    }
}

/****************************************************************************
FUNCTION        : ext_Int_Handler
PURPOSE         : Respond to external interrupts from the PTA function.
INPUT NOTES     : All enabled external interrupts jump into this routine.
RETURN NOTES    : If PTA interrupt, get accumulation, clear the interrupt,
                     and quit.
GENERAL NOTES   :
***************************************************************************/

void ext_Int_Handler()
    {
    struct TPU3_tag *tpua = &TPU_A;         /* pointer for TPU routines */
    extern UINT32 accumulation;             /* global accumulation value */

    if ( tpu_check_interrupt(tpua, 0) )     /* test PTA int. status bit */
        {
        accumulation = tpu_pta_get_accumulation(tpua, 0);

        tpu_clear_interrupt(tpua, 0);       /* clear interrupt */
        }
    else
        {
        /* process external interrupt from another source */
        }
    }
```

3.2 Example 2

3.2.1 Description

This sample program show a simple PTA example that measures the total high time over a selected number of periods of an input signal connected to channel 0 on TPUA. A simple interrupt handler is used to trigger the acquisition of the completed measurement.

The program then runs in an infinite loop, continuing to read back each accumulated measurement from the input on TPUA channel 0.

3.2.2 Program

```
/***************************************************************************/
/* FILE NAME: tpu_pta_example2.c              COPYRIGHT (c)  2002 */
/* VERSION: 1.0                                   All Rights Reserved     */
/*                                                                        */
/* DESCRIPTION: This sample program show a simple PTA example that        */
/* measures the total high time over a selected number of periods of an   */
/* input signal connected to channel 0 on TPUA.                           */
/* A simple interrupt handler is used to trigger the acquisition of the   */
/* completed measurement.                                                 */
/*                                                                          */
/* The program is targeted for the MPC555 but should work on any MPC500    */
/* device with a TPU. For other devices the setup routines will also need  */
/* to be changed.                                                           */
/*==========================================================================*/
/* HISTORY           ORIGINAL AUTHOR: Stan Ostrum                          */
/*                                                                          */
/* REV      AUTHOR      DATE       DESCRIPTION OF CHANGE                   */
/* ---   -----------  ---------    ---------------------                   */
/* 1.0   Stan Ostrum  16 Sep 02    Initial version of function.            */
/****************************************************************************/

#include "mpc555.h"          /* Define MPC555 registers, this needs to be  */
                             /* changed if other MPC500 devices are used.  */
#include "mpc500_util.h"     /* Utility routines for using MPC500 devices  */
#include "tpu_pta.h"         /* TPU PTA functions */

UINT32   accumulation;       /* global variable for period/PW accumulation */

void main ()
{
    struct TPU3_tag *tpua = &TPU_A;         /* pointer for TPU routines */

    setup_555();    Metrowerks setup routine for MPC555 */

    /* Initialize Prog. Time Accum function with:   */
    /*     - Input signal on TPU A channel 0         */
    /*     - Use TCR1 timebase                       */
    /*     - Measure high times                      */
    /*     - Max count value of 15                   */
    /*     - Schedule as high priority in the TPU    */

    tpu_pta_init(tpua, 0, TPU_PRIORITY_HIGH, TPU_PTA_TCR1, \
                 TPU_PTA_HIGH_TIME_ACCUM, 15);

    tpu_interrupt_enable(tpua, 0);          /* enable PTA interrupts */

    while (1) {                             /* loop and measure */
    }
}
```

The external interrupt handler code that is used for this example is shown in Example 1 above.

3.3 Function State Timing

When calculating the worst case latency for the TPU, the execution time of each state of the TPU is needed. The state timings for each of the four modes of the PTA function are shown below in Table 1. The states used by the C interface functions are shown in Table 2.

Table 1. 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 or 14 CPU clocks)

Table 2. PTA API Function State Usage

PTA API Function	State Uses
tpu_pta_init	S1
tpu_pta_get_accumulation	none
tpu_pta_get_period_count	none

3.4 Function Code Size

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

55 µ instructions + 8 entries = 63 long words

4 Notes on Use and Performance of the PTA Function

4.1 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.

4.2 Usage Notes and Restrictions

4.2.1 Short Measurement Cycles

If the frequency of the PTA input signal is too high and the number of periods in each measurement cycle is too low, the CPU may not have enough time to process the interrupt at the end of the cycle and read the accumulated result before the end of the next measurement cycle. Additionally, the CPU must be able to execute the API call to read the accumulated result before the next accumulation has exceeded 16 bits. This is due to the fact that the call to read the accumulated result clears the high word of the 32-bit accumulation. If this does not occur, then the next measurement result will be invalid.

4.2.2 Maximum Accumulation

The PTA function allows a maximum accumulation of 32 bits of the selected TCR timebase. If the accumulation overflows, an interrupt is generated to the CPU, but the function continues to run normally. Investigation of the returned 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 long- est measurement under worst case conditions does not exceed 32 bits.

4.2.3 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 calling the API function tpu_pta_get_accumulation to get the partial measurement and then calling tpu_pta_get_period_count to determine the number of periods over which the partial accumulation has been made. Note that the returned period count value 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 timebase clocks (if pulse width/period very long) less than the real value at the instant of reading.p

4.3 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 Section 3.3, “Function State Timing” for more information.

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

freescale™
semiconductor

AN2365/D