To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

H8SX Series

Pulse Width Measurement

Introduction

As well as having an architecture that is upward-compatible with each CPU of the H8/300, H8/300H, and H8S series, so as to inherit a full complement of peripheral functions, the H8SX microcomputer series has a maximum operating frequency of 50 MHz and uses a 32-bit H8SX core CPU as well as an on-chip multiplier/divider to improve performance.

This H8SX series Application Note provides information you may be need during software and hardware design. This is a basic edition that provides operation examples that each use a single H8SX series on-chip peripheral function.

Although the operation of each program, circuit, and other aspects covered by this application note has been checked, make sure that you conduct your own operation checks before actually using the H8SX series.

Contents

1. Overview

The 16-bit timer pulse unit (TPU) of the H8SX series measures the high and low widths (times) of the input pulses.

2. Configuration

Channel 0 of the 16-bit timer pulse unit (TPU) measures the width (high or low) of each pulse input from the input capture input pin (TIOCA0). When the peripheral module clock (Pϕ) is 25 MHz, you can measure a pulse width of up to 163.84 msec[1)] in units of 2.56 µsec (1/Pϕ). Figure 1 is a block diagram illustrating pulse width measurement.

[1)] for a count clock of Pϕ/64

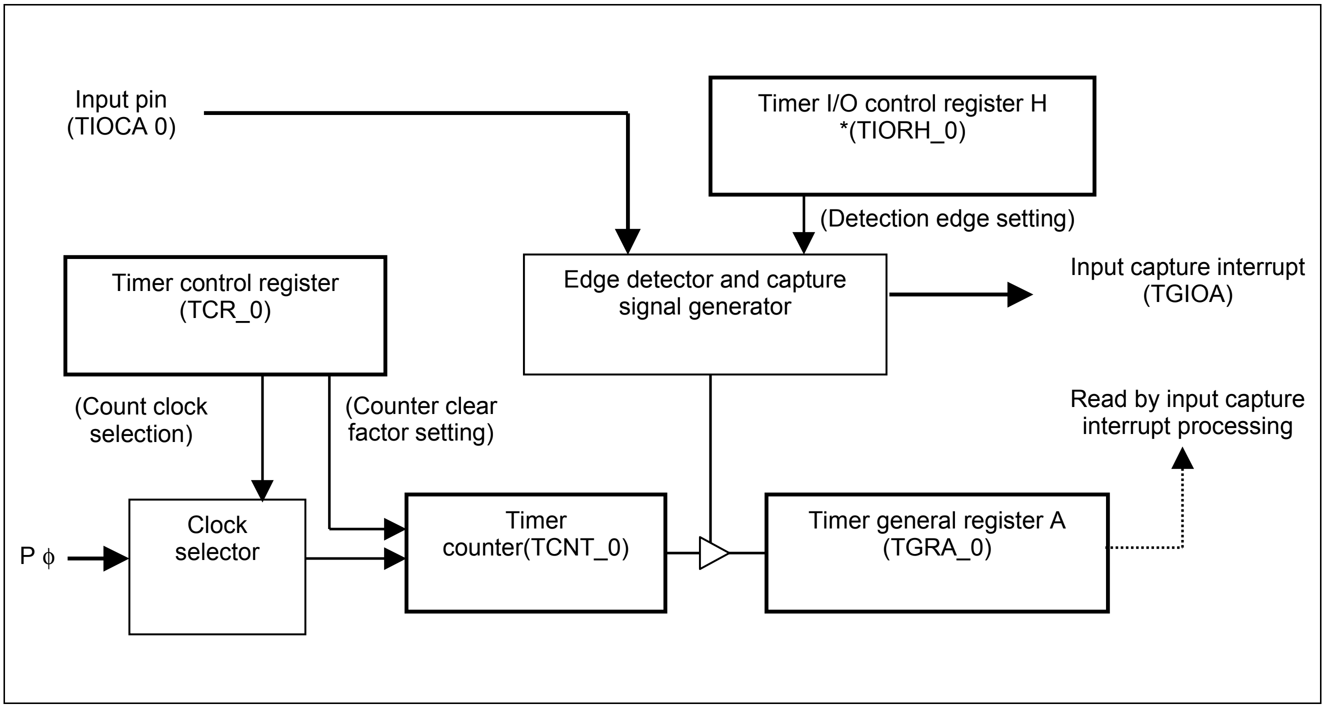


Figure 1 Block Diagram of Pulse Width Measurement

3. Sample Program

3.1 Function

This sample program measures the pulse width (high or low) of each input pulse. The input capture function obtains the timer value for each pulse width. You can calculate the pulse width (time) from the obtained timer value by using the following equation:

pulse-width = timer-value × count-clock

Assume that the count clock is peripheral module (Pϕ)/64. When Pϕ is 25 MHz, the count clock is 2.56 μsec. Figure 2 shows an example of the operation.

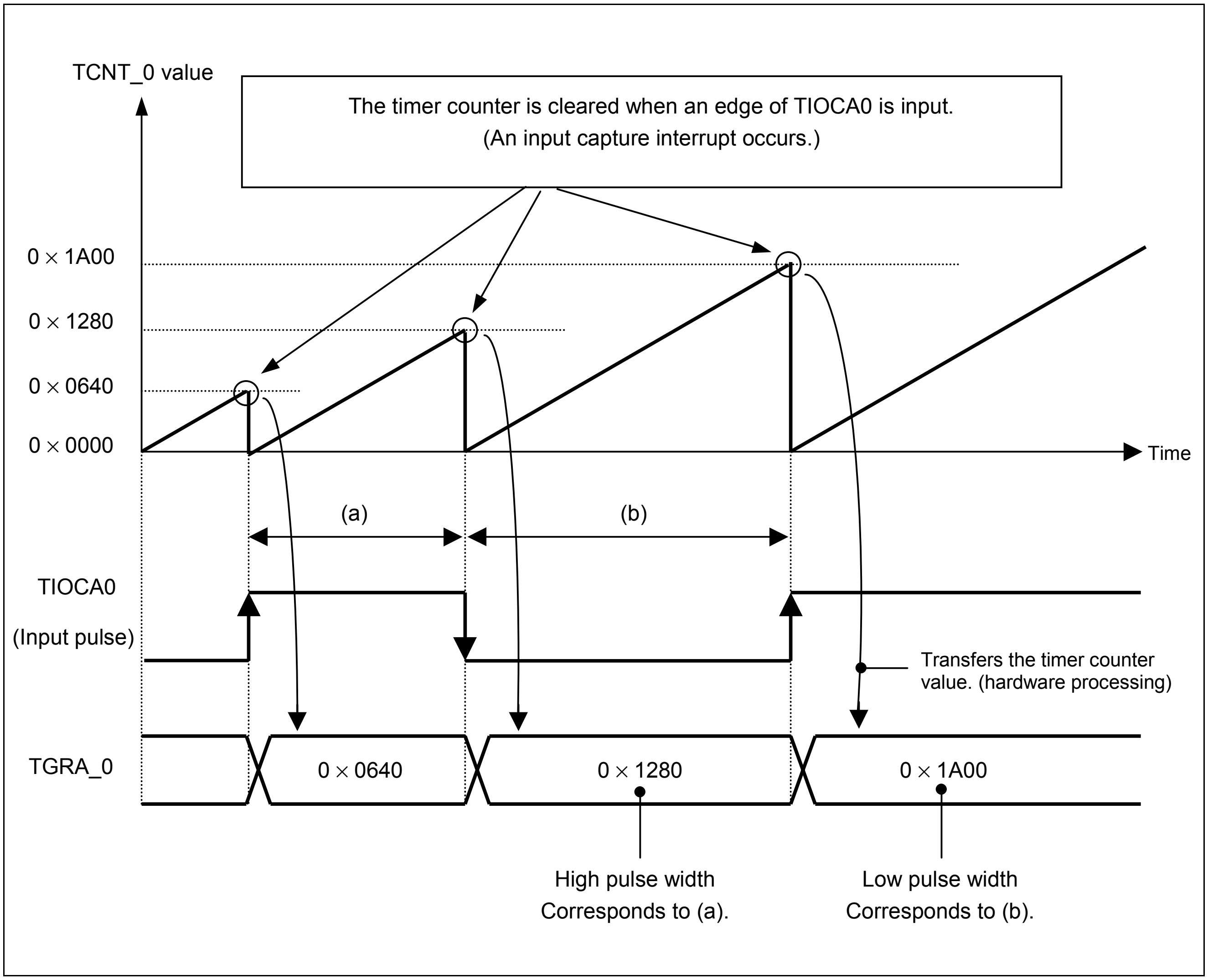


Figure 2 Example of Pulse Width Measurement

When the TPU detects an edge of an input pulse, this function transfers the timer counter value (TCNT_0 value) at that instant to the timer general register (TGRA_0), clears the timer counter value, and then generates an input capture interrupt (TGI0A).

Table 1 lists the function allocations of channel 0 of the 16-bit timer pulse unit (TPU).

Table 1 Function Allocation of TPU Channel 0

Type	Name	Function
Register	MSTPCRA	Cancels the TPU module stop mode.
	TSTR	Specifies whether to start or stop the timer count of TPU channel 0.
	TCR_0	Sets the TCNT_0 count clock and counter clear factor.
	TIORH_0	Sets the input source pin and input edge for input capture.
	TGRA_0	Detects the timer counter value when an input capture interrupt occurs
	TIER_0	Enables an interrupt by TGI0A.
	TSR_0	Status flag for occurrence of an input capture interrupt
Input pin	TIOCA0	Input capture pulse input pin
Interrupt	TGI0A	Input capture interrupt

3.2 Function Specifications

The functions to measure the pulse width are given as a sample program. The function specifications are listed below.

(1) Routine for setting pulse width measurement

```
void  capture_set ( void )
```

Argument	Description
None	-

Return value	Description
None	-

(2) Input capture interrupt handler

```
void  inthdr_capture ( void )
```

RAM variable	Type	Description
bPulseHigh	unsigned short	The interrupt handler sets the timer count value for the high input pulse width.
bPulse Low	unsigned short	The interrupt handler sets the timer count value for the low input pulse width.

This function has neither an argument nor return value because it is a TPU0 interrupt handler. Register this interrupt handler in the interrupt vector table.

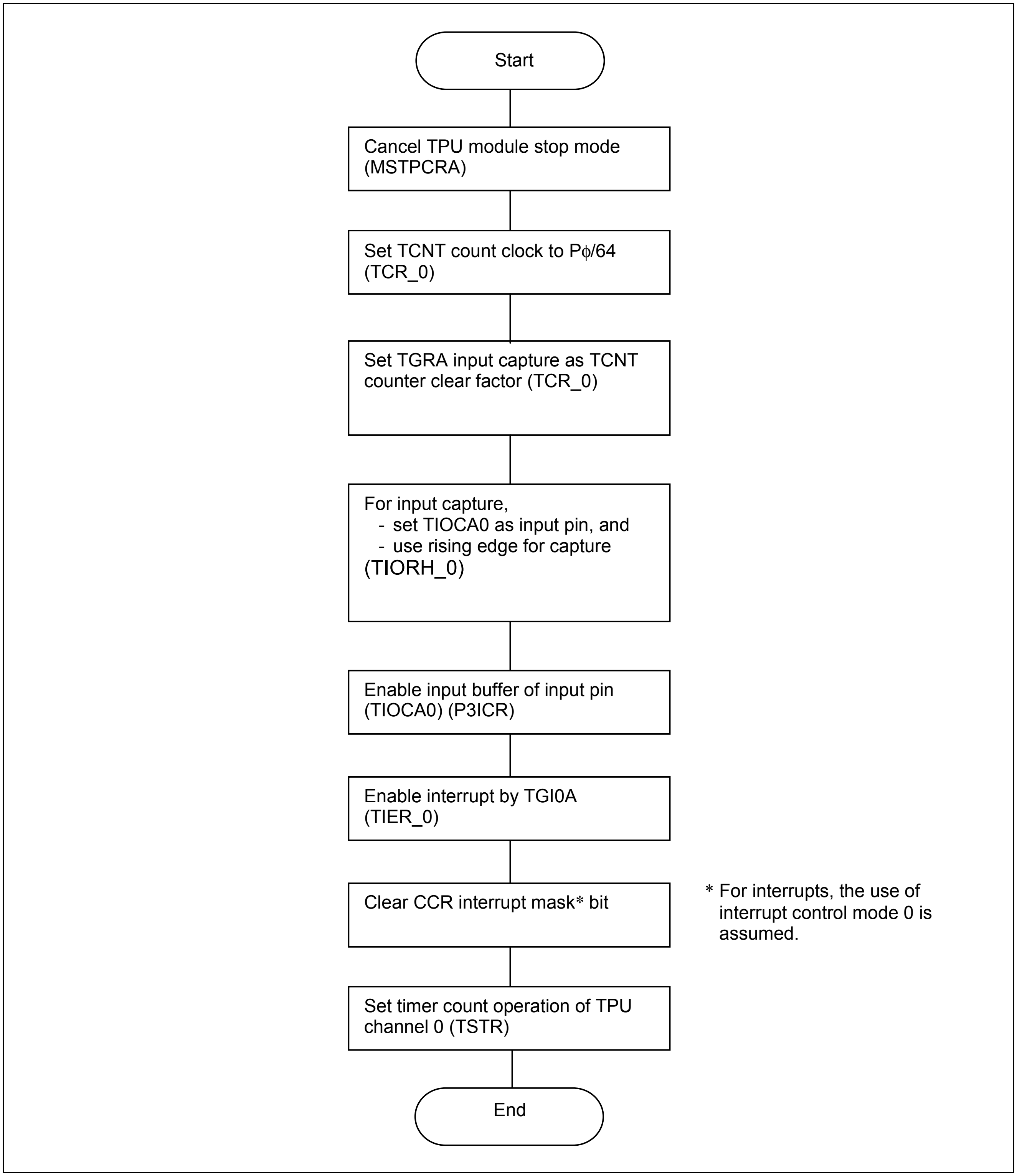
Example)

```
#define P_CLOCK     24                       // Pϕ (MHz)
#define P_TPSC         64                    // Count clock multiplication ratio
extern void  capture_set ( void  );          //  External function reference
declaration
extern unsigned short  bPulseHigh;           //  External function reference
declaration
extern unsigned short  bPulseLow;            //  External function reference
declaration
void   main( void )                          //  Main routine
{
   volatile unsigned long  wTimeHigh;        // Pulse high width time (µsec)
   volatile unsigned long  wTimeLow;         // Pulse low width time (µsec)
   capture_set ( );                          //  Sets the pulse width measurement.
   while(1)
   {
. .
                                             // Calculates the pulse width (in µsec) by
                                             // referencing the interrupt handler set
value.
         wTimeHigh = ((unsigned long)bPulseHigh* P_TPSC)/P_CLOCK;
         wTimeLow  = ((unsigned long)bPulseLow* P_TPSC)/P_CLOCK;
         . . .
   }
}
```

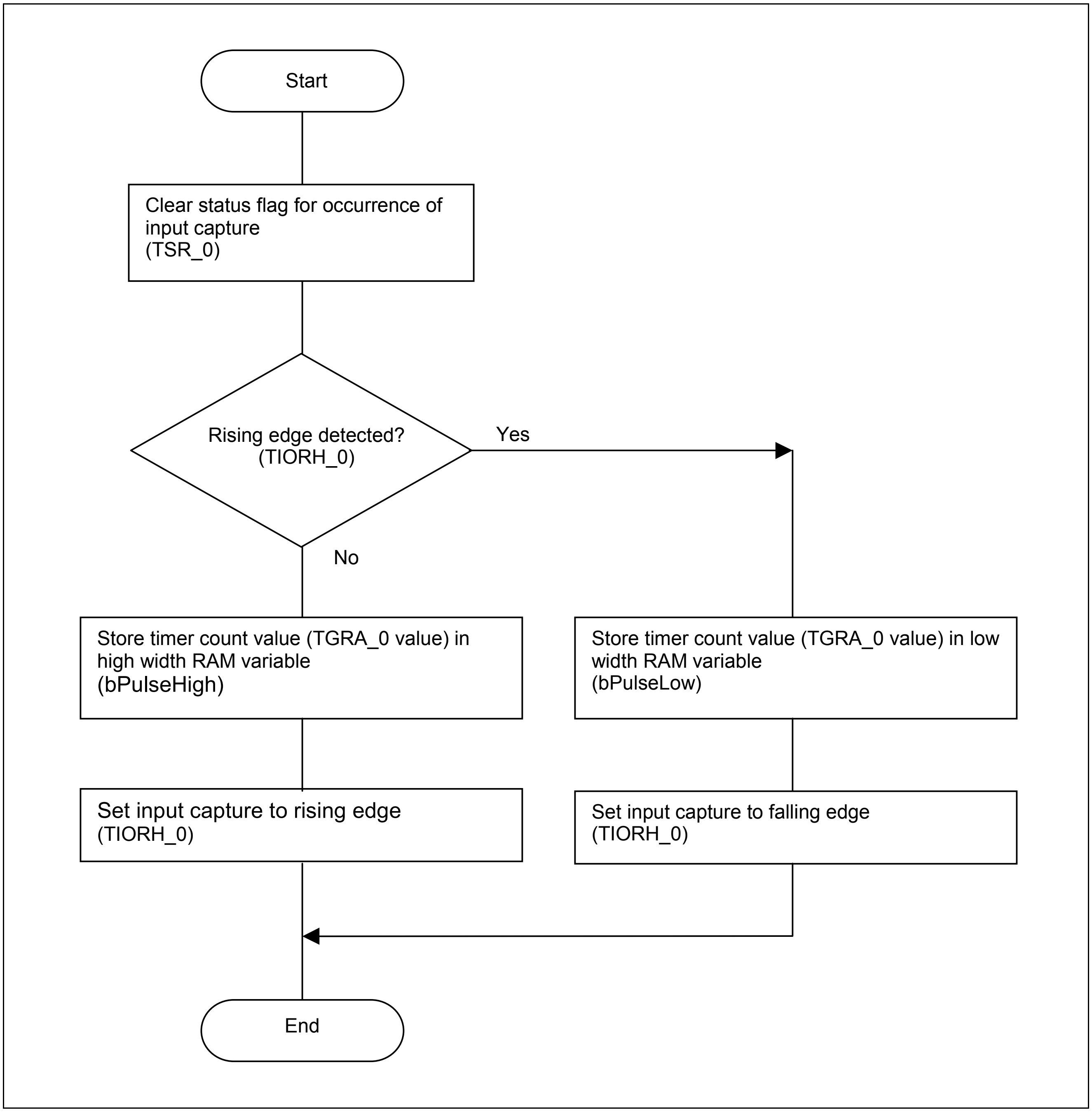
3.3 Flowchart

The processing flow is shown below.

(1) void capture_set (void)



(2) void inthdr_capture (void)



3.4 Program Listing

A listing of the source program is given below. In the following source program, Renesas's standard definition (file automatically generated by High-performance Embedded Workshop: iodefine.h) is used to define the I/O register structure. To specify your own definition, change the I/O register structure in the sample program.

```
/**************************************************************/
/* include file                                               */
/**************************************************************/
#include <machine.h>
#include "iodefine.h"

/**************************************************************/
/* function prototype                                         */
/**************************************************************/
void capture_set( void );

/**************************************************************/
/* interface variable                                         */
/**************************************************************/
unsigned short  bPulseHigh;
unsigned short  bPulseLow;

/**************************************************************/
/* function definition                                        */
/**************************************************************/
void capture_set( void )
{
    P_MSTPCRA.BIT.MSTPA0  = 0;  // reset module-standby for TPU
    P_TPU0.TCR.BIT.TPSC   = 3;  // set TPU0 countup clock source
    P_TPU0.TCR.BIT.CCLR   = 1;  // set TPU0 counter clear cause
    P_TPU0.TIOR.BIT.IOA   = 8;  // set TPU0 input-capture-A by
                                // rising-edge
    P_P3.ICR.BIT.Pn0ICR   = 1;  // set input buffer enable
    P_TPU0.TIER.BIT.TGIEA = 1;  // set TGI0A-interrupt enable
    set_imask_ccr(0);           // clear interrupt mask
    P_TPU.TSTR.BIT.CST0   = 1;  // start TPU0
}

/**************************************************************/
/* interrupt handler definition                               */
/**************************************************************/
#pragma interrupt ( inthdr_capture )
void inthdr_capture( void )
{
    volatile unsigned char  dummy;
    dummy = P_TPU0.TSR.BYTE;    // read TPU0 interrupt status
    P_TPU0.TSR.BIT.TGFA = 0;    // clear TGI0A-interrupt status
    if ( 8 == P_TPU0.TIOR.BIT.IOA )
    {
      bPulseLow   = (unsigned short)P_TPU0.TGRA;
      P_TPU0.TIOR.BIT.IOA = 9;  // change to falling-edge capture
    }
```

```
    else
    {
      bPulseHigh = (unsigned short)P_TPU0.TGRA;
      P_TPU0.TIOR.BIT.IOA = 8;  // change to rising-edge capture
    }
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sept.19.03	—	First edition issued