PWM Driver Sample Code Reference Guide

V1.00.001

Publication Release Date: Sep. 2011

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

Table of Contents

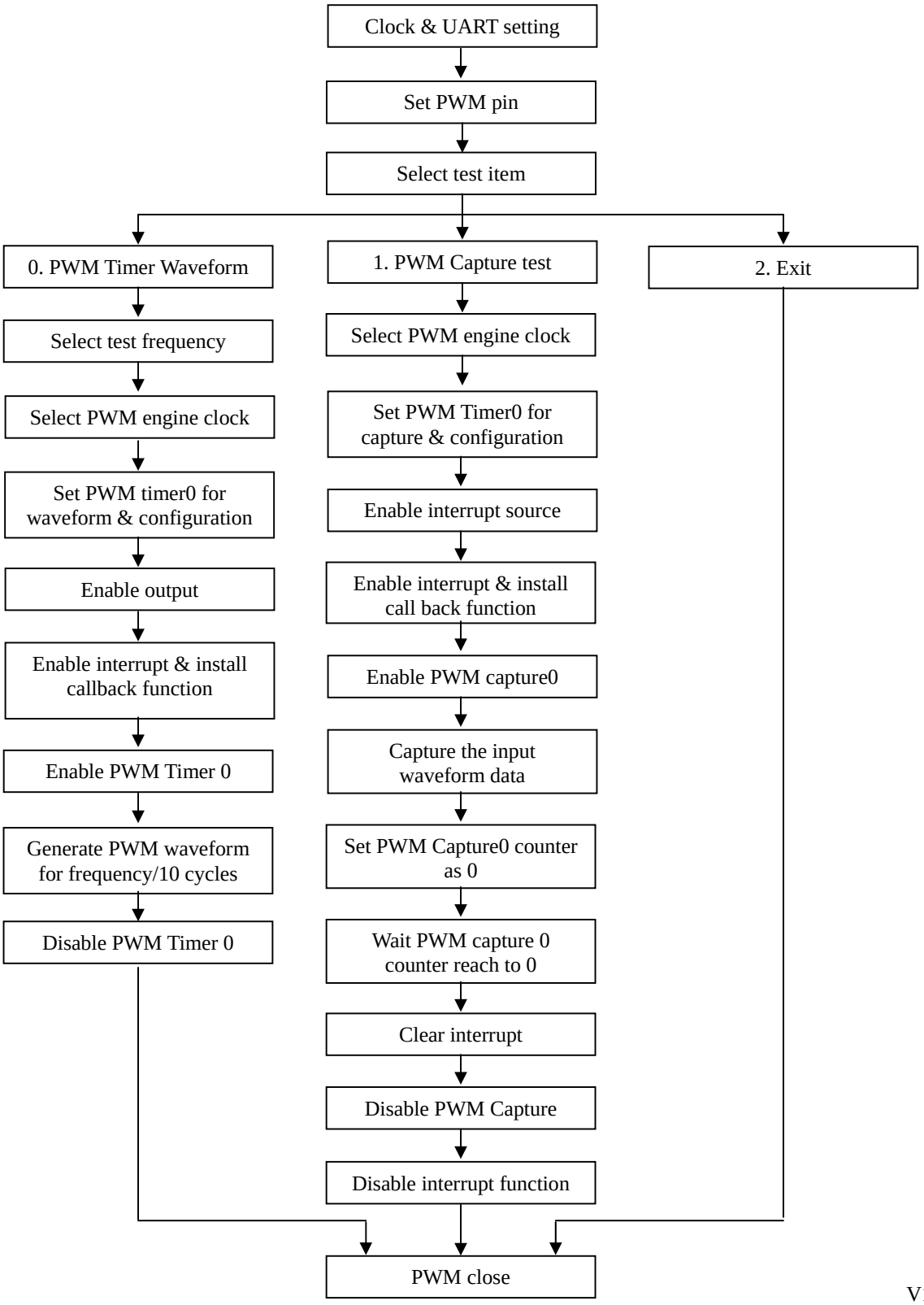
1 Introduction

This sample code will demo PWM IP on ISD9160 chip.

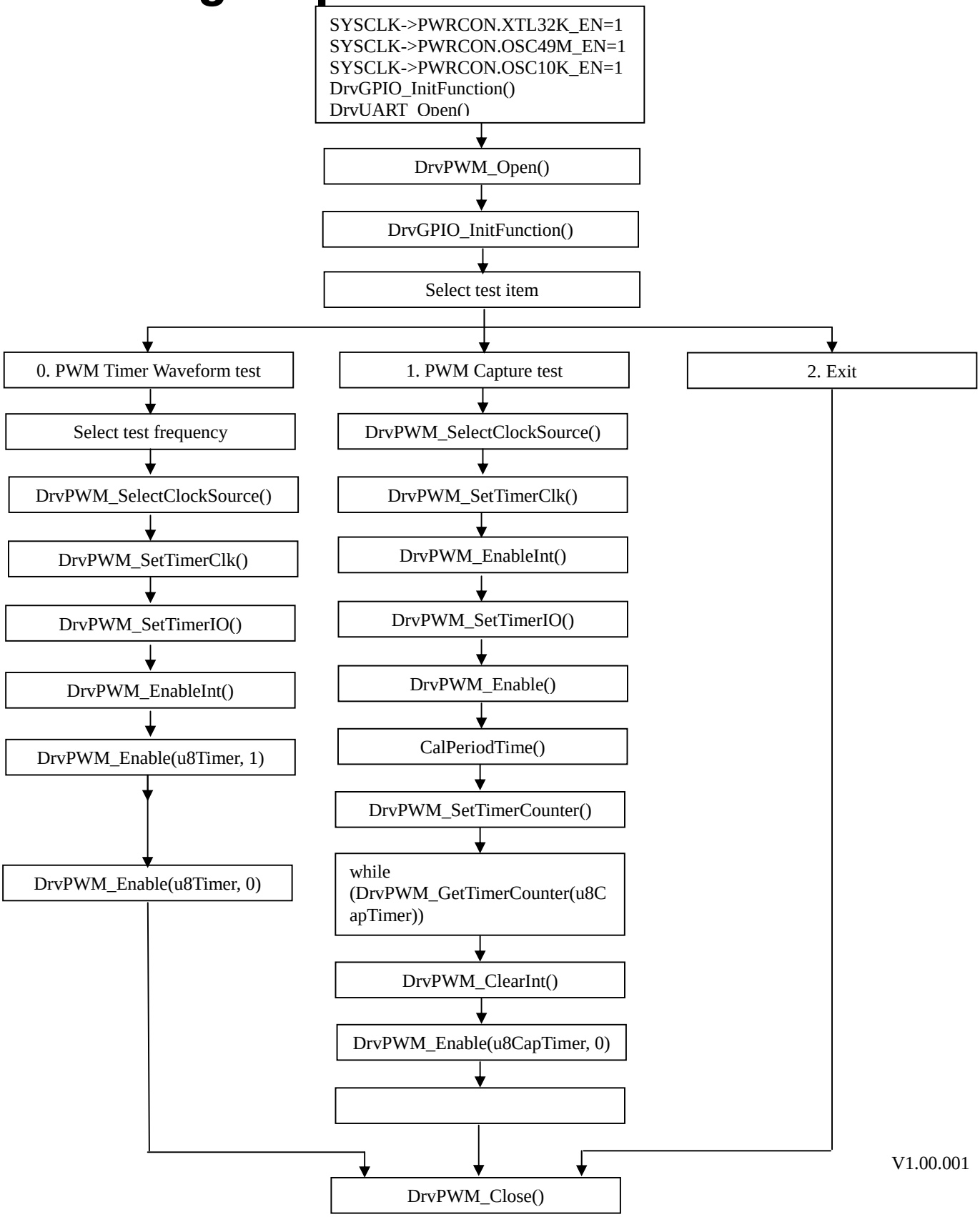
1.1 Feature

- This sample test PWM Timer and capture function.
- Case 0: PWM Timer waveform test. Waveform output to Buzzer. Case 1: PWM Capture test, use PWM Capture 0 to capture the waveform from PA.12.

2 Block Diagram



3 Calling Sequence



4 Code Section –Smpl_DrvPWM.c

```
/*---------------------------------------------------------------------------------------------------------*/
/*                                                                                                         */
/* Copyright(c) 2011 Nuvoton Technology Corp. All rights reserved.                                         */
/*                                                                                                         */
/*---------------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include "Driver\DrvGPIO.h"
#include "Driver\DrvPWM.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvUART.h"

/*---------------------------------------------------------------------------------------------------------*/
/* Global variables                                                                                        */
/*---------------------------------------------------------------------------------------------------------*/
int32_t     g_bCapInt = 0;
uint8_t volatile g_u8PWMCount = 1;
uint16_t g_u16Frequency;
static uint32_t s_u32Pulse = 0;

/*---------------------------------------------------------------------------------------------------------*/
/* PWM Timer Callback function                                                                             */
/*---------------------------------------------------------------------------------------------------------*/
void DRVPWM_PwmIRQHandler()
{
    if (s_u32Pulse == 1 * g_u16Frequency /10)
    {
        /*-----------------------------------------------------------------------------------------------------*/
        /* Stop PWM Timer 0 (Recommended procedure method 2)                                                   */
        /* Set PWM Timer counter as 0, When interrupt request happen, disable PWM Time                         */
        /*-----------------------------------------------------------------------------------------------------*/

        DrvPWM_SetTimerCounter(DRVPWM_TIMER0, 0);

    }

    if (s_u32Pulse == 1 * g_u16Frequency /10 + 1)
        g_u8PWMCount = 0;
    s_u32Pulse++;

}
```

```
/*-------------------------------------------------------------------------------------------------------------*/
/* Capture function to calculate the input waveform information                                                */
/* u32Count[4] : Keep the internal counter value when input signal rising / falling                            */
/*                          happens                                                                            */
/*                                                                                                             */
/* time          A       B       C       D                                                                     */
/*                  ___   ___   ___   ___   ___   ___   ___   ___                                              */
/*           ____|      |_|     |_|     |_|     |_|     |_|     |_|     |_|     |_____                        */
/*                                                                                                             */
/* index                0 1    2 3                                                                            */
/*                                                                                                             */
/* The capture internal counter down count from 0x10000, and reload to 0x10000 after                           */
/* input signal falling happens       (Time B/C/D)                                                             */
/*-------------------------------------------------------------------------------------------------------------*/

void CalPeriodTime(uint8_t u8Capture)
{
    uint16_t u32Count[4];
    uint32_t u32i;
    uint16_t u16RisingTime, u16FallingTime, u16HighPeroid, u16LowPeroid, u16TotalPeroid;


    /* Clear the Capture Interrupt Flag (Time A) */
    DrvPWM_ClearCaptureIntStatus(u8Capture, DRVPWM_CAP_FALLING_FLAG);

    /* Wait for Interrupt Flag    (Falling) */
    while (DrvPWM_GetCaptureIntStatus(u8Capture, DRVPWM_CAP_FALLING_FLAG) != 1);

    /* Clear the Capture Interrupt Flag (Time B)*/
    DrvPWM_ClearCaptureIntStatus(u8Capture, DRVPWM_CAP_FALLING_FLAG);

    u32i = 0;


    while (u32i < 4)
    {
        /* Wait for Interrupt Flag (Falling) */
        while(DrvPWM_GetCaptureIntStatus(u8Capture, DRVPWM_CAP_FALLING_FLAG) != 1);

        /* Clear the Capture Interrupt Flag */
        DrvPWM_ClearCaptureIntStatus(u8Capture, DRVPWM_CAP_FALLING_FLAG);

        /* Clear the Capture Rising Interrupt Flag */
        DrvPWM_ClearCaptureIntStatus(u8Capture, DRVPWM_CAP_RISING_FLAG);

        /* Get the Falling Counter Data */
        u32Count[u32i++] = DrvPWM_GetFallingCounter(u8Capture);
```

```
            /* Wait for Capture Rising Interrupt Flag */
            while(DrvPWM_GetCaptureIntStatus(u8Capture, DRVPWM_CAP_RISING_FLAG) != 1);

            /* Clear the Capture Rising Interrupt Flag */
            DrvPWM_ClearCaptureIntStatus(u8Capture, DRVPWM_CAP_RISING_FLAG);

            /* Get the Rising Counter Data */
            u32Count[u32i++] = DrvPWM_GetRisingCounter(u8Capture);
        }

        u16RisingTime = u32Count[1];

        u16FallingTime = u32Count[0];

        u16HighPeroid = u32Count[1] - u32Count[2];

        u16LowPeroid = 0x10000 - u32Count[1];

        u16TotalPeroid = 0x10000 - u32Count[2];

        printf("Test Result:\nRising Time = %d, Falling Time = %d.\nHigh Period = %d, Low Period = %d,
        Total Period = %d.\n\n", u16RisingTime, u16FallingTime, u16HighPeroid, u16LowPeroid,
        u16TotalPeroid);
}


void SysTimerDelay(uint32_t us)
{
        SysTick->LOAD = us * 48; /* Assume the internal 48M RC used */
        SysTick->VAL    =   (0x00);
        SysTick->CTRL = (1 << SYSTICK_CLKSOURCE) | (1<<SYSTICK_ENABLE);

        /* Waiting for down-count to zero */
        while((SysTick->CTRL & (1 << 16)) == 0);
}

/*---------------------------------------------------------------------------------------------------------*/
/* PWM Capture Callback function                                                                           */
/*---------------------------------------------------------------------------------------------------------*/
void DRVPWM_CapIRQHandler()
{

        g_bCapInt = 1;
}
```

```
/*---------------------------------------------------------------------------------------------------------*/
/*    Main Function                                                                                        */
/*---------------------------------------------------------------------------------------------------------*/
int32_t main (void)
{
      S_DRVPWM_TIME_DATA_T sPt;
      STR_UART_T sParam;
      uint8_t u8Item;
      uint8_t u8Timer, u8CapTimer;
      int32_t i32Loop = 1;
      int32_t i32TestLoop = 1;


      UNLOCKREG();

      SYSCLK->PWRCON.XTL32K_EN = 1;
      SYSCLK->PWRCON.OSC49M_EN = 1;
      SYSCLK->PWRCON.OSC10K_EN = 1;

      /* Waiting for Xtal stalble */
      SysTimerDelay(5000);

      /* Set UART Pin */
      DrvGPIO_InitFunction(FUNC_UART0);

      /* UART Setting */
      sParam.u32BaudRate        = 115200;
      sParam.u8cDataBits        = DRVUART_DATABITS_8;
      sParam.u8cStopBits        = DRVUART_STOPBITS_1;
      sParam.u8cParity          = DRVUART_PARITY_NONE;
      sParam.u8cRxTriggerLevel= DRVUART_FIFO_1BYTES;

      /* Set UART Configuration */
      DrvUART_Open(UART_PORT0,&sParam);

      /* Enable PWM clock */
      DrvPWM_Open();

      /* Set PWM pins */
      DrvGPIO_InitFunction(FUNC_PWM01);

      UNLOCKREG();
      DrvSYS_SetHCLKSource(0);
      LOCKREG();
```

```
printf("+------------------------------------------------------------------------+\n");
printf("|                        PWM Driver Sample Code                          |\n");
printf("|                                                                        |\n");
printf("+------------------------------------------------------------------------+\n");
printf("    This sample code will use PWM0 to drive Buzzer or capture    the signal\n from PA.12.\n");



    while (i32Loop)
    {
        printf("\n    Please Select Test Item\n");
        printf("       0 : PWM Timer Waveform Test\n");
        printf("       1 : PWM Caputre Test\n");
        printf("       2 : Exit\n\n");

        u8Item = getchar();

        switch (u8Item)
        {
            case '0':
            {
                uint8_t u8ItemOK;

                printf("\nPWM Timer Waveform Test. Waveform output(PWM0) to Buzzer\n");

                i32TestLoop = 1;

                printf("Select Test Item\n");
                printf(" 1: Do (256Hz)\n");
                printf(" 2: Re (287Hz)\n");
                printf(" 3: Mi (322Hz)\n");
                printf(" 4: Fa (341Hz)\n");
                printf(" 5: Sol(383Hz) \n");
                printf(" 6: La (430Hz)\n");
                printf(" 7: Si (483Hz)\n");
                printf(" 0: Exit\n");

                while (i32TestLoop)
                {
                    u8ItemOK = 1;
                    u8Item = getchar();

                    switch (u8Item)
                    {
                        case '1':
                            g_u16Frequency = 256;
                            break;
```

```
            case '2':
                g_u16Frequency = 287;
                break;
            case '3':
                g_u16Frequency = 322;
                break;
            case '4':
                g_u16Frequency = 341;
                break;
            case '5':
                g_u16Frequency = 383;
                break;
            case '6':
                g_u16Frequency = 430;
                break;
            case '7':
                g_u16Frequency = 483;
                break;
            case '0':
                i32TestLoop = 0;
                break;
            default:
                u8ItemOK = 0;
                break;

        }

        if (i32TestLoop && u8ItemOK)
        {
            s_u32Pulse = 0;
            g_u8PWMCount = 1;

            /* PWM Timer property */
            sPt.u8Mode = DRVPWM_TOGGLE_MODE;
            sPt.u32Frequency = g_u16Frequency;

            /* High Pulse peroid : Total Pulse peroid = 50 :100 */
            sPt.u8HighPulseRatio = 50;
            sPt.i32Inverter = 0;
            u8Timer = DRVPWM_TIMER0;

            /* Select PWM engine clock */
            DrvPWM_SelectClockSource(DRVPWM_HCLK);

            /* Set PWM Timer0 Configuration */
            DrvPWM_SetTimerClk(u8Timer, &sPt);
```

```
                        /* Enable Output for PWM Timer0 */
                        DrvPWM_SetTimerIO(u8Timer, 1);



                    /* Enable Interrupt Sources of PWM Timer0 and install call back function */
                        DrvPWM_EnableInt(u8Timer, 0, DRVPWM_PwmIRQHandler);

                        /* Enable the PWM Timer 0 */
                        DrvPWM_Enable(u8Timer, 1);


                        while (g_u8PWMCount);


/*---------------------------------------------------------------------------------------------------------*/
/* Stop PWM Timer0 (Recommended procedure method 2)                                                        */
/* Set PWM Timer counter as 0, When interrupt request happen, disable PWM Timer                           */
/* Set PWM Timer counter as 0 in Call back function                                                        */
/*---------------------------------------------------------------------------------------------------------*/

                        /* Disable the PWM Timer 0 */
                        DrvPWM_Enable(u8Timer, 0);
                    }
                }

                break;
            }
            case '1':
            {

                        printf("PWM Capture Test\n");
                        printf("Use PWM Capture 0 to capture the Waveform from PA.12 \n");

                /*---------------------------------------------------------------------------------------------------------*/
                /* Set the PWM Capture 3 for capture function                                                              */
                /*---------------------------------------------------------------------------------------------------------*/

                /* PWM Timer property for Capture */
                sPt.u8Mode = DRVPWM_TOGGLE_MODE;

                /* Set the proper frequency to capture data (Less than the input data)*/
                sPt.u32Frequency = 100;

                /* High Pulse peroid : Total Pulse peroid = 50 : 100 (Set a non-zero value) */
                sPt.u8HighPulseRatio = 50;
```

```
                sPt.u32Duty = 0x10000;              /* Set the counter to the maximum value */
                sPt.i32Inverter = 0;
                u8CapTimer = DRVPWM_CAP0;

                /* Select PWM engine clock */
                DrvPWM_SelectClockSource(DRVPWM_HCLK);

                /* Set PWM Timer 0 for Capture */
                DrvPWM_SetTimerClk(u8CapTimer, &sPt);

                /* Enable Interrupt Sources of PWM Capture0 */
                DrvPWM_EnableInt(u8CapTimer, DRVPWM_CAP_FALLING_INT,
                DRVPWM_CapIRQHandler);

                /* Enable Input function for PWM Capture0    */
                DrvPWM_SetTimerIO(u8CapTimer, 1);

                /* Enable the PWM Capture0 */
                DrvPWM_Enable(u8CapTimer, 1);

                /* Capture the Input Waveform Data */
                CalPeriodTime(u8CapTimer);


/*---------------------------------------------------------------------------------------------------------*/
/* Stop PWM Timer 0 (Recommended procedure method 1)                                                       */
/* Set PWM Timer counter as 0, When PWM internal counter reaches to 0, disable PWM Timer*/
/*---------------------------------------------------------------------------------------------------------*/

                /* Set PWM Capture0 counter as 0 */
                DrvPWM_SetTimerCounter(u8CapTimer, 0);

                /* Wait PWM Capture0 Counter reach to 0 */
                while (DrvPWM_GetTimerCounter(u8CapTimer));

                /* Clear the PWM Capture0 Interrupt */
                DrvPWM_ClearInt(u8CapTimer);

                /* Disable PWM Capture0 */
                DrvPWM_Enable(u8CapTimer, 0);

                /* Disable Input function for PWM Capture0 */
                DrvPWM_SetTimerIO(u8CapTimer, 0);

                break;
            }
```

```
                case '2':
                {
                    i32Loop = 0;
                    break;
                }
            }
        }

        printf("PWM sample is complete.\n");

        DrvPWM_Close();

        return 0;
```

5 Execution Environment Setup and Result

- Prepare a ISD9160 board.
- Compile the sample code.
- Console window show result of PWM timer waveform and capture test.

6 Revision History

Version	Date	Description
V1.00.01	Sep. 2011	Created