PMU Driver Sample Code Reference Guide

V1.00.001

Publication Release Date: Sep. 2011

Support Chips:

ISD9160

Support Platforms:

NuvotonPlatform_Keil

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

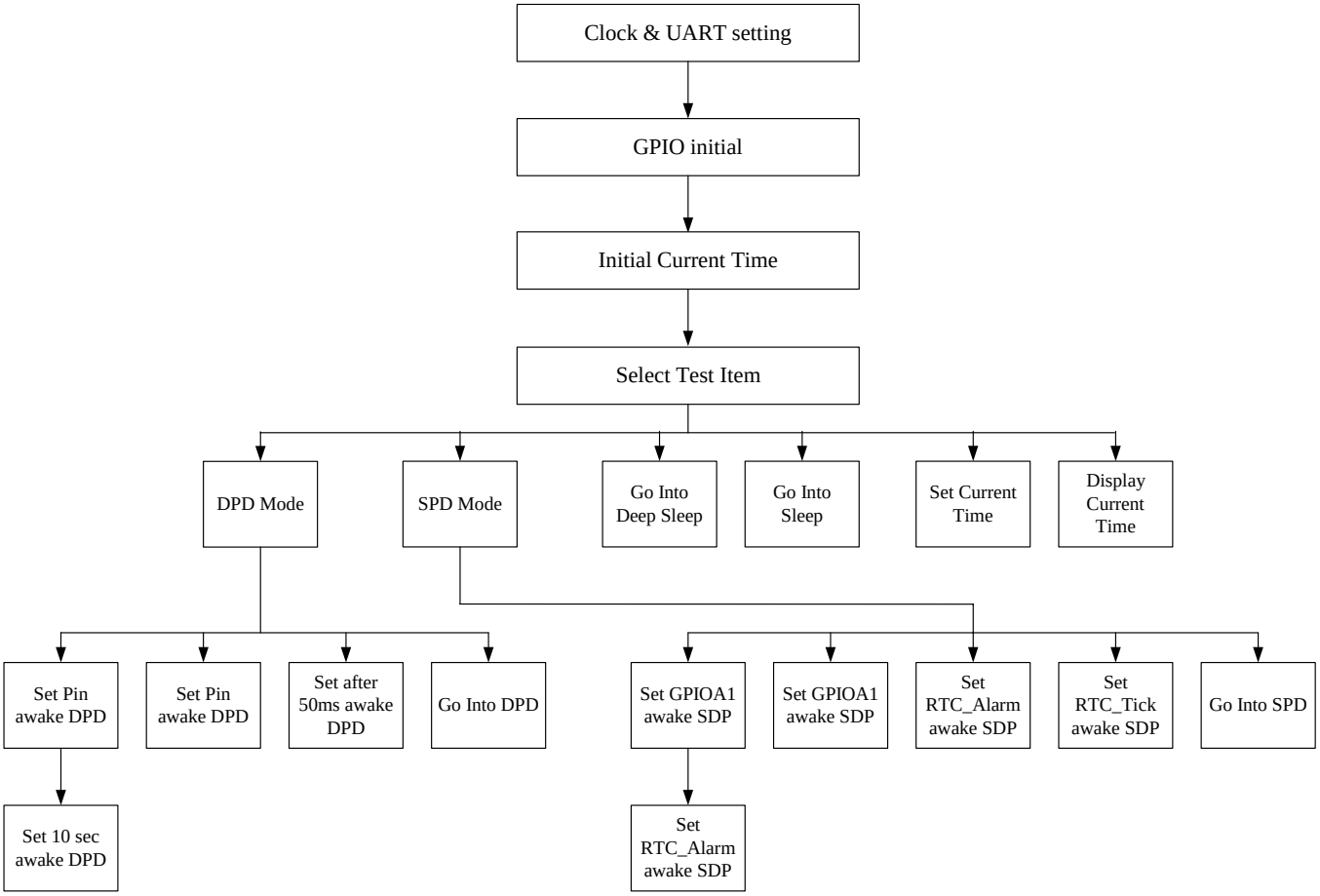
1. PMU Driver Introduction

This sample code will demo PMU IP on ISD9160 chip.

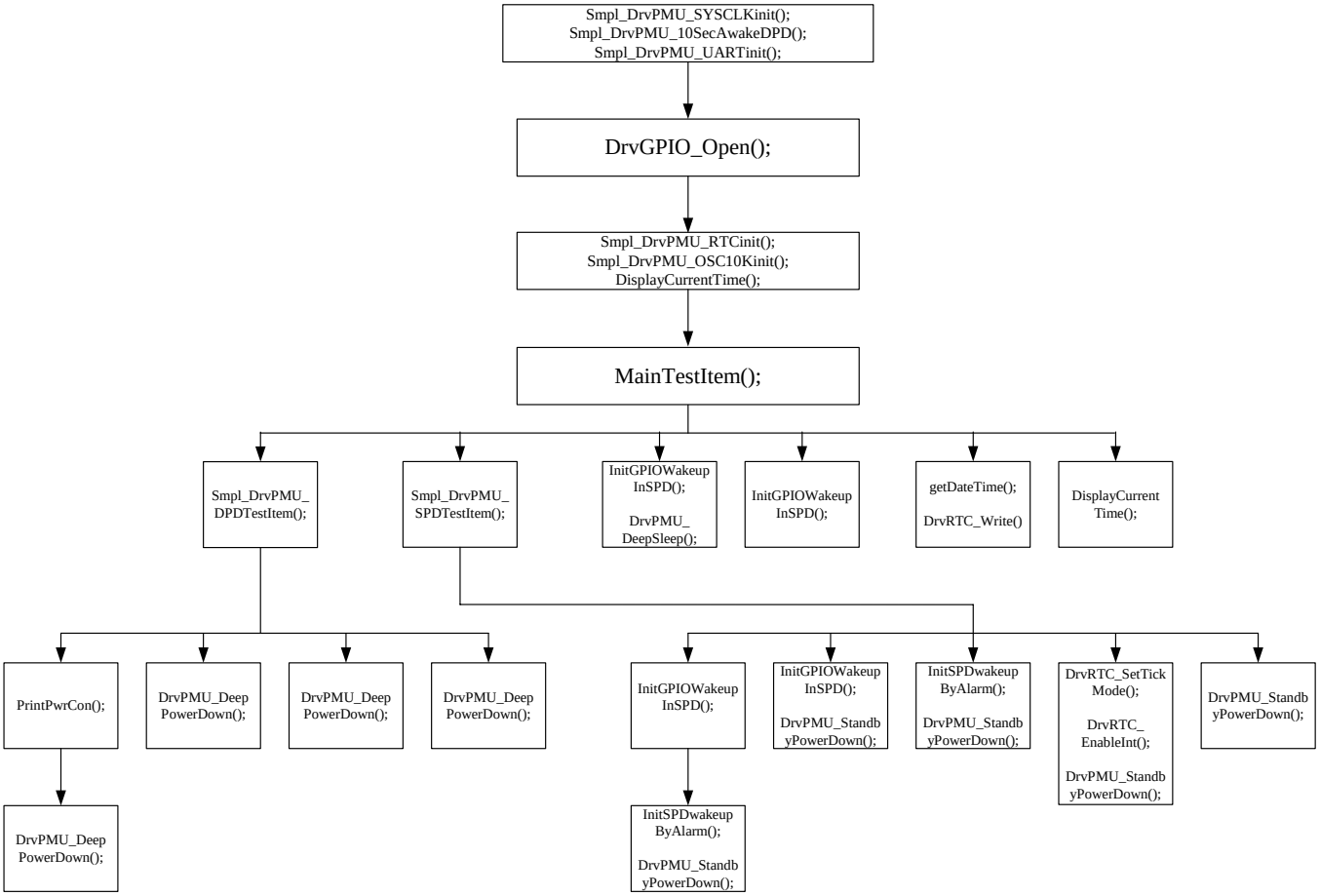
1.1 Feature

- Using Pin and 10sec awake ISD9160 when the chip enters in DPD.
- Using Pin awakes ISD9160 when the chip enters in DPD.
- After 50ms awakes ISD9160 when the chip enters in DPD.
- Directly enters into DPD.
- Using GPIOA1 pin and RTC Alarm awake ISD9160 when the chip enters in SPD.
- Using GPIOA1 pin awakes ISD9160 when the chip enters in SPD.
- Using RTC Alarm awakes ISD9160 when the chip enters in SPD.
- Using RTC Tick awakes ISD9160 when the chip enters in SPD.
- Directly enters into SPD.
- Directly enters into Deep Sleep.
- Directly enters into Sleep.
- Set current time.
- 11. Display current time.

2. Block Diagram



3. Calling Sequence



4. Code Section –Smpl_DrvPMU.c

```
/*---------------------------------------------------------------------------------------------------------*/
/*
*/
/* Copyright(c) 2011 Nuvoton Technology Corp. All rights reserved.
*/
/*
*/
/*---------------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include "Driver\DrvUART.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvRTC.h"
#include "Driver\DrvOSC.h"
#include "Driver\DrvPMU.h"
#include "Driver\DrvPWM.h"
#include "Driver\DrvI2S.h"
#include "ISD9xx.h"

/*---------------------------------------------------------------------------------------------------------*/
/* Global variables
*/
/*---------------------------------------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------------------------------------*/
/* Define functions prototype
*/
/*---------------------------------------------------------------------------------------------------------*/


void SysTimerDelay(uint32_t us)
{
    SysTick->LOAD = us * 22; /* Assume the internal 22MHz 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);
}
```

```
void Smpl_DrvPMU_SYSCLKinit(void)
{
	UNLOCKREG();
	SYSCLK->CLKSEL0.HCLK_S = 0; /* Select HCLK source as OSC48 */
	SYSCLK->CLKDIV.HCLK_N    = 9;   /* Select no division                  */
	SYSCLK->CLKSEL0.OSCFSel = 1;    /* 1= 32MHz, 0=48MHz */
	SYSCLK->PWRCON.XTL32K_EN = 1;
	SYSCLK->PWRCON.OSC10K_EN = 1;
	/* Enable RTC Clock */
	SYSCLK->APBCLK.RTC_EN =1;
	SYSCLK->APBCLK.I2S_EN     = 1;
	SYSCLK->APBCLK.ANA_EN     = 1;
	LOCKREG();
}

void Smpl_DrvPMU_10SecAwakeDPD(void)
{
	if(SYSCLK->DPDSTATE.DPD_STATE_RD != 0){
		// We have state from DPD, increment until 100 then exit
		if(SYSCLK->DPDSTATE.DPD_STATE_RD != 100){
			SYSCLK->DPDSTATE.DPD_STATE_WR =
SYSCLK->DPDSTATE.DPD_STATE_RD + 1;

	DrvPMU_DeepPowerDown(DPDWAKEUPMODE_PINOSC10KWAKEUP,DPDWAKETIME_
100ms);
		}
	}
}

void Smpl_DrvPMU_UARTinit(void)
{
	STR_UART_T sParam;
	/* 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;

	DrvUART_Open(UART_PORT0,&sParam);
}
```

```
void Smpl_DrvPMU_RTCinit(void)
{
    uiTmp = RTC->INIR;
    if(    RTC->INIR == 0){
        /* RTC Initialize if it is a "cold" boot (i.e. not from SPD) */
        DrvRTC_Init();
       uiTmp = RTC->INIR;
    }

    // Clear interrupt sources: If we have been in SPD mode, WIC is preserved.
    GPIOA->ISRC = GPIOA->ISRC;
    RTC->RIIR.AI            = 1;
    RTC->RIIR.TI            = 1;
    RTC->RIER.TIER              = 0;
    RTC->RIER.AIER              = 0;
}

void Smpl_DrvPMU_OSC10Kinit(void)
{
    SYS->OSC10K.TM_REG = 0x0;
    SYS->OSC10K.TRIM_CLK =0;
    SYS->OSC10K.TRIM_CLK =1;
    SYS->OSC10K.TRIM_CLK =0;
    printf ("TM_REG %x\n", SYS->OSC10K.TM_REG);
}

void PrintPwrCon(void)
{
    printf("ICER register %08X\n",    NVIC->ICER[0]);
    printf("PFLAG register %02X\n", M32(&SYSCLK->PFLAGCON));
    M32(&SYSCLK->PFLAGCON) = 0xff;
    printf("RSTSRC register %02X\n", M32(&SYS->RSTSRC));
    M32(&SYS->RSTSRC) = 0xff;
    printf("PWRCON register %08X\n", M32(&SYSCLK->PWRCON));
    printf("XTLEN=%d, OSC49=%d OSC10=%d STOP=%d SBPD=%d DPD=%d\n",
            SYSCLK->PWRCON.XTL32K_EN,
            SYSCLK->PWRCON.OSC49M_EN,
            SYSCLK->PWRCON.OSC10K_EN,
            SYSCLK->PWRCON.STOP,
            SYSCLK->PWRCON.STANDBY_PD,
            SYSCLK->PWRCON.DEEP_PD);

    printf("PIN_ENB=%d, DPD_10K=%d TIMER=%d PIN_WAKE=%d TIMER_WAKE=%d
POI_WAKE=%d TIMER=%d\n",
```

```
                SYSCLK->PWRCON.PIN_ENB,
                SYSCLK->PWRCON.OSC10K_ENB,
                SYSCLK->PWRCON.TIMER_SEL,
                SYSCLK->PWRCON.PIN_WAKE,
                SYSCLK->PWRCON.TIMER_WAKE,
                SYSCLK->PWRCON.POI_WAKE,
                SYSCLK->PWRCON.TIMER_SEL_RD);

}

static void MainTestItem (void)
{
    printf("\n\n");
    PrintPwrCon();
    printf("+------------------------------------------------------------+\n");
    printf("|                  PMU Sample Program                        |\n");
    printf("+------------------------------------------------------------+\n");
    printf("| [0] Deep Power Down Mode                                   |\n");
    printf("| [1] Standby Power Down Mode                                |\n");
    printf("| [2] Deep Sleep                                             |\n");
    printf("| [3] Sleep                                                  |\n");
    printf("| [4] Set Current Time                                       |\n");
    printf("| [5] Display Current                                        |\n");
    printf("+------------------------------------------------------------+\n");
    printf("Select Test Item : \n>");
}

void Smpl_DrvPMU_DPDTestItem(void)
{
    uint8_t u8Option;

    while(1)
    {
        printf("\n\n");
        printf("+-----------------------------------------------------------------+\n");
        printf("|              Deep Power Down Test Item                          |\n");
        printf("+-----------------------------------------------------------------+\n");
        printf("| [0] Deep Power Down (pin or wake in 10 sec)                     |\n");
        printf("| [1] Deep Power Down can wakeup by WAKEUP-PIN                    |\n");
        printf("| [2] Deep Power Down can wakeup by OSC10K:50ms                   |\n");
        printf("| [3] Deep Power Down can only wakeup by POR                      |\n");
        printf("| [4] Exit to Main Menu                                           |\n");
        printf("+-----------------------------------------------------------------+\n");
        printf("Select Test Item : \n>");
        u8Option = getchar();
        printf("    [%c]\n",u8Option);
```

```
            switch(u8Option)
            {
                case '0': //Deep Power Down
                    PrintPwrCon();
                    printf("Going into DPD wake by pin or in 10 sec\n");
                    SYSCLK->DPDSTATE.DPD_STATE_WR = 1;
                    while(UART0->FSR.TE != 1);
                    DrvPMU_DeepPowerDown(DPDWAKEUPMODE_PINOSC10KWAKEUP,
                    DPDWAKETIME_100ms);
                  break;

                case '1': //DPD wakeup by PIN&POR
                    printf("Going into DPD, wake by pin\n");
                    while(UART0->FSR.TE != 1);
                    DrvPMU_DeepPowerDown(DPDWAKEUPMODE_PINWAKEUP,
                    DPDWAKETIME_100ms);
                  break;

                case '2': //DPD wakeup by OSC10kHz:50ms & POR
                    printf("Going into DPD for 50msec\n");
                    while(UART0->FSR.TE != 1);
                    DrvPMU_DeepPowerDown(DPDWAKEUPMODE_OSC10KWAKEUP,
                    DPDWAKETIME_50ms);
                  break;

                case '3': //DPD only wakeup by POR
                    printf("Going into DPD\n");
                    while(UART0->FSR.TE != 1);
                    DrvPMU_DeepPowerDown(DPDWAKEUPMODE_ONLYPORWAKEUP,
                    DPDWAKETIME_100ms);
                  break;

                case '4':
                    return;

                default:
                    printf("Wrong Item\n");
                  break;
            }
        }
}
```

```
void Smpl_DrvPMU_SPDTestItem(void)
{
    uint8_t u8Option;

    while(1)
    {
        printf("\n\n");
        printf("+-------------------------------------------------------------------------+\n");
        printf("|                      Standby Power Down Test Item                       |\n");
        printf("+-------------------------------------------------------------------------+\n");
        printf("| [0] SPD can wakeup by GPIOA[1]    &    RTC:ALARM      |\n");
        printf("| [1] Standby Power Down can wakeup by GPIOA[1]            |\n");
        printf("| [2] Standby Power Down can wakeup by RTC:ALARM           |\n");
        printf("| [3] Standby Power Down can wakeup by RTC:TICK            |\n");
        printf("| [4] Standby Power Down can only wakeup by POR            |\n");
        printf("| [5] Exit to Main Menu                                    |\n");
        printf("+-------------------------------------------------------------------------+\n");
        printf("Select Test Item : \n>");
        u8Option = getchar();
        printf("    [%c]\n",u8Option);

        switch(u8Option)
        {
            case '0': //SPD can wakeup by GPIOA[1]    &    RTC:ALARM    &    POR
                InitGPIOWakeupInSPD();
                ///////////////Call from RTC driver
                InitSPDwakeupByAlarm();
                while(UART0->FSR.TE != 1);
                DrvPMU_StandbyPowerDown();
               break;

            case '1': //SPD can awakeup by GPIO & POR
                InitGPIOWakeupInSPD();
                while(UART0->FSR.TE != 1);
                DrvPMU_StandbyPowerDown();
            break;

           case '2': //SPD wakeup by RTC Alarm & POR
                ///////////////Call from RTC driver
                InitSPDwakeupByAlarm();
                while(UART0->FSR.TE != 1);
                DrvPMU_StandbyPowerDown();
               break;
```

```
                case '3': //SPD wakeup by RTC Tick & POR
                    //Set Tick setting
                    DrvRTC_SetTickMode(DRVRTC_TICK_1_SEC);
                    //Enable RTC Tick Interrupt and install tick call back function
                    DrvRTC_EnableInt(DRVRTC_TICK_INT, Smpl_DrvPMU_TickISR);
                    DrvPMU_StandbyPowerDown();
                  break;

                case '4': //SPD only wakeup POR
                    DrvPMU_StandbyPowerDown();
                  break;

                case '5':
                    return;

                default:
                    printf("Wrong Item\n");
                  break;
            }
        }
}

void Smpl_DrvPMU_SPDGPABCallback(uint32_t u32GpaStatus, uint32_t u32GpbStatus)
{
    //DrvGPIO_DisableInt(GPA, 1);
    printf("GPIO Wake up from Standby-Power-Down Mode\n");

}

void InitGPIOWakeupInSPD(void)
{
    // Clear interrupts.
    GPIOA->ISRC = GPIOA->ISRC;
    ////INT from GPA pin1 as an example wake up
    DrvGPIO_Open(GPA, 1, IO_INPUT);
    DrvGPIO_SetIntCallback(Smpl_DrvPMU_SPDGPABCallback);
    DrvGPIO_SetDebounceTime(3, DBCLKSRC_HCLK);
    DrvGPIO_EnableDebounce(GPA, 1);
    DrvGPIO_EnableInt(GPA, 1, IO_BOTH_EDGE, MODE_EDGE);
}
```

```
/*_-----------------------------------------*/
/*     define date and time                */
/*-----------------------------------------*/
S_DRVRTC_TIME_DATA_T getDateTime()
{
        S_DRVRTC_TIME_DATA_T sInitTime;

        printf ("enter year :\n ");
        scanf    ("%d",&sInitTime.u32Year );
        printf ("year = %d\n", sInitTime.u32Year);

        printf ("enter month (1-12) :\n ");
        scanf    ("%d", &sInitTime.u32cMonth);
        printf ("month = %d\n", sInitTime.u32cMonth);

        printf ("enter day (1-31) :\n ");
        scanf    ("%d", &sInitTime.u32cDay );
        printf ("day = %d\n", sInitTime.u32cDay);

        printf ("enter hour (0-23) : \n");
        scanf    ("%d", &sInitTime.u32cHour );
        printf ("hour = %d\n", sInitTime.u32cHour);

        printf ("enter minute (0-59) : \n");
        scanf    ("%d", &sInitTime.u32cMinute );
        printf ("minute = %d\n", sInitTime.u32cMinute);

        printf ("enter second (0-59) : \n");
        scanf    ("%d", &sInitTime.u32cSecond );
        printf ("second = %d\n", sInitTime.u32cSecond);

        printf ("enter day of week (0-6) : \n");
        scanf    ("%d", &sInitTime.u32cDayOfWeek );
        printf ("day = %d\n", sInitTime.u32cDayOfWeek);

        sInitTime.u8cClockDisplay = DRVRTC_CLOCK_24;

        return sInitTime;
}
```

```
void DisplayCurrentTime(E_DRVRTC_TIME_SELECT eTime)
{
    S_DRVRTC_TIME_DATA_T sCurTime;

    /* Get the currnet time */
    DrvRTC_Read(eTime, &sCurTime);
    printf("Time:%d/%02d/%02d
%02d:%02d:%02d\n",sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay,sCurTime.u32cH
our,sCurTime.u32cMinute,sCurTime.u32cSecond);
}

void Smpl_DrvPMU_AlarmISR(void)
{
    S_DRVRTC_TIME_DATA_T sCurTime;

    /* Get the currnet time */
    DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);
    printf("Current Time:%d/%02d/%02d %02d:%02d:%02d
",sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay,sCurTime.u32cHour,sCurTime.u32c
Minute,sCurTime.u32cSecond);
    printf("Alarm!\n");
}

void InitSPDwakeupByAlarm(void)
{
    S_DRVRTC_TIME_DATA_T sInitTime;

    DisplayCurrentTime(DRVRTC_CURRENT_TIME);
  check_time:
    DrvRTC_Read(DRVRTC_CURRENT_TIME, &sInitTime);
    // Set alaram to wakeup in 10 seconds.
    if(sInitTime.u32cSecond < 49)
        sInitTime.u32cSecond += 10;
    else{
        sInitTime.u32cSecond = sInitTime.u32cSecond + 10 - 60;
        if( sInitTime.u32cMinute < 59 )
            sInitTime.u32cMinute += 1;
        else
            // Too lazy, wait a few seconds...
            goto check_time;
    }
    DrvRTC_Write(DRVRTC_ALARM_TIME,&sInitTime);
    printf("Alarm ");
    DisplayCurrentTime(DRVRTC_ALARM_TIME);
    DrvRTC_EnableInt(DRVRTC_ALARM_INT, Smpl_DrvPMU_AlarmISR);
}
```

```
void Smpl_DrvPMU_TickISR(void)
{
    printf("Tick Int!\n");
    /* Disable RTC Tick Interrupt */
    DrvRTC_DisableInt(DRVRTC_TICK_INT);
}


/*---------------------------------------------------------------------------*/
/* Using UART for testing PMU Samples                                        */
/* Test Item                                                                 */
/* It sends the messages to HyperTerminal.                                   */
/*---------------------------------------------------------------------------*/

int32_t main(void)
{
    uint8_t u8Item;
    S_DRVRTC_TIME_DATA_T sInitcTime;
    uint32_t uiTmp;

    // Initial SYSCLK
    Smpl_DrvPMU_SYSCLKinit();

    // Timer for 10 Second awake DPD if function has been enabled
    Smpl_DrvPMU_10SecAwakeDPD();

    //Initial UART
    Smpl_DrvPMU_UARTinit();

    printf("\nBOOT\n");

    // Set GPA1 to input
    DrvGPIO_Open(GPA, 1, IO_INPUT);


    //Initial RTC
    Smpl_DrvPMU_RTCinit();
```

```
    if((M32(&SYSCLK->PFLAGCON)==0) && (M32(&SYS->RSTSRC)==0x43)){
        // Indicates a Boot from DPD or POI rather than SPD
        if(SYSCLK->PWRCON.POI_WAKE)
            printf("DPD WakeUp of device was requested with a power-on reset.\n");
        if(SYSCLK->PWRCON.TIMER_WAKE)
            printf("DPD WakeUp of device was requested with Timer count of the 10KHz
oscillator.\n");
        if(SYSCLK->PWRCON.PIN_WAKE)
            printf("DPD WakeUp of device was requested with WAKEUP-PIN.\n");
    }
    PrintPwrCon();

    // Only enable clocks we need in SLEEP modes
    M32(&SYSCLK->CLKSLEEP)=0x20 + 0x01;

    Smpl_DrvPMU_OSC10Kinit();

    DisplayCurrentTime(DRVRTC_CURRENT_TIME);

    while(1)
    {
        MainTestItem();
        u8Item = getchar();
        switch(u8Item)
        {
            case '0':
                Smpl_DrvPMU_DPDTestItem();
              break;

            case '1':
                Smpl_DrvPMU_SPDTestItem();
              break;

            case '2': //Deep sleep
                InitGPIOWakeupInSPD();
                SYSCLK->CLKSEL0.HCLK_S = 0;
                SYSCLK->PWRCON.OSC49M_EN =0;
                SysTimerDelay(5000);
                DrvPMU_DeepSleep();
              break;

            case '3': //Sleep
                InitGPIOWakeupInSPD();
                SCB->SCR = 0;
                __wfi();
              break;
```

```
                case '4':
                    printf("Please Input Current time\n");
                    sInitcTime = getDateTime();
                    DrvRTC_Write(DRVRTC_CURRENT_TIME,&sInitcTime);
                  break;

                case '5':
                    DisplayCurrentTime(DRVRTC_CURRENT_TIME);
                  break;

                default:
                    printf("Wrong Item\n");
                  break;

            }
        }
}
```

5. Execution Environment Setup and Result

- Prepare a ISD9160 board.
- Compile the sample code.
- Console window show result of ACMP.

6. Revision History

Version	Date	Description
V1.00.01	Sep. 2011	Created