RTC 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

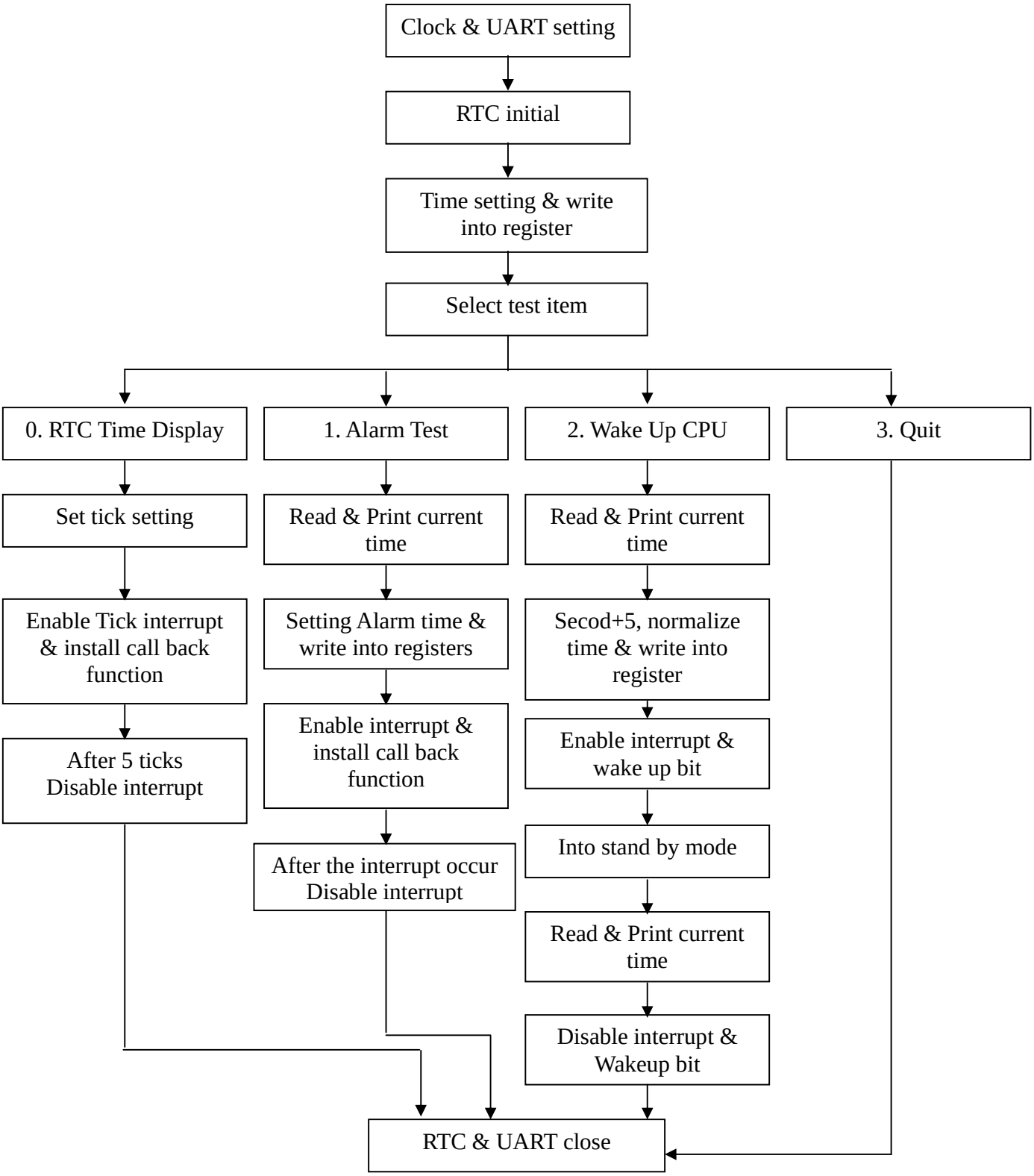
1 Introduction

This sample code will demo RTC IP on ISD9160 chip.

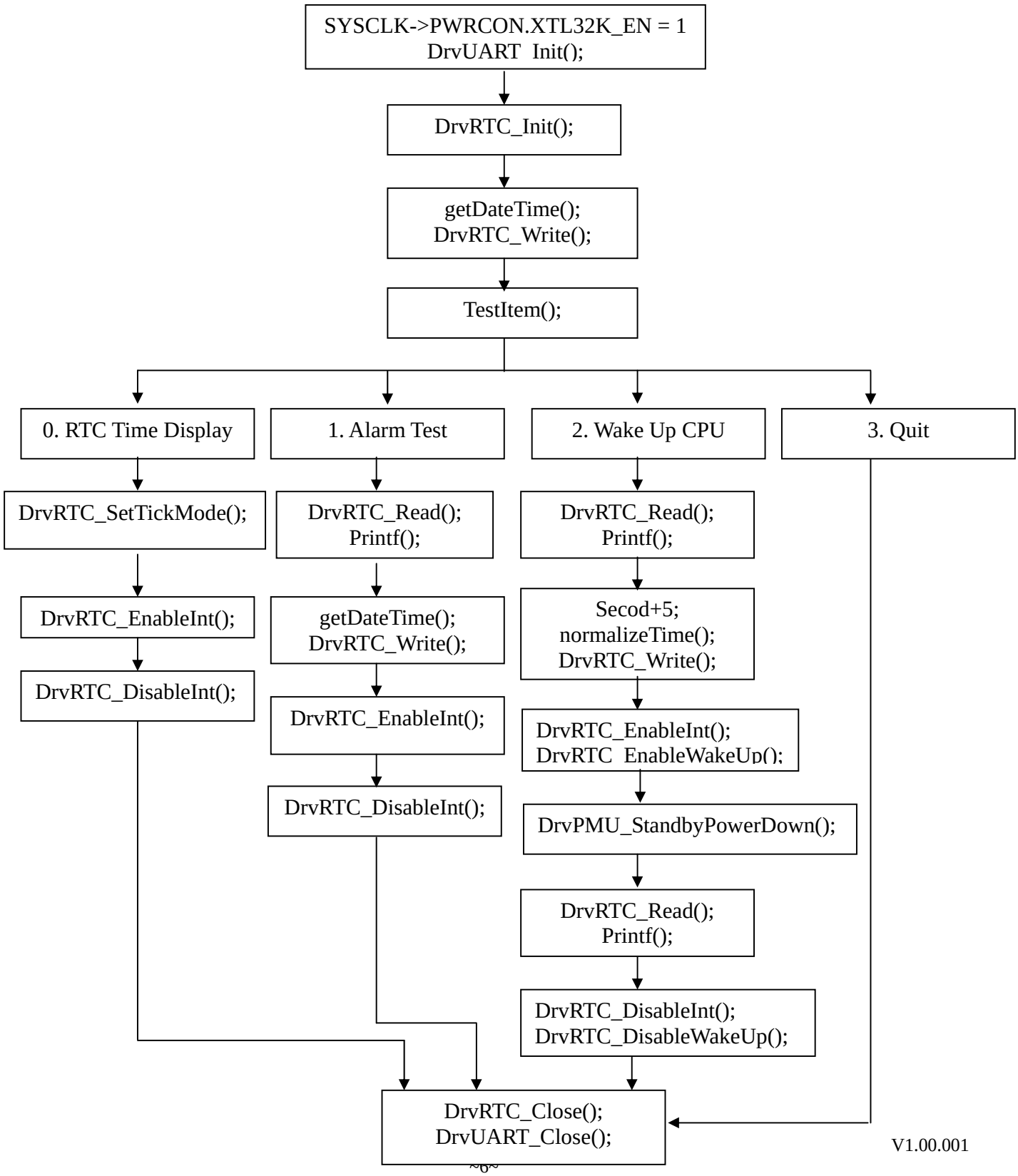
1.1 Feature

- Use 32 KHz crystal oscillator clock source.
- Set Time in main function, and start RTC.
- When RTC time out interrupt occurs, print Time with Semi Host.

2 Block Diagram



3 Calling Sequence



4 Code Section –Smpl_DrvRTC.c

```
/*---------------------------------------------------------------------------------------------------------*/
/*                                                                                                         */
/* Copyright(c) 2011 Nuvoton Technology Corp. All rights reserved.                                         */
/*                                                                                                         */
/*---------------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>


#include "ISD9xx.h"
#include "Driver/DrvUART.h"
#include "Driver/DrvRTC.h"
#include "Driver/DrvGPIO.h"
#include "Driver/DrvSYS.h"
#include "Driver/DrvPMU.h"

/*---------------------------------------------------------------------------------------------------------*/
/* Global variables                                                                                        */
/*---------------------------------------------------------------------------------------------------------*/
volatile uint32_t    g_u32TICK = FALSE;
volatile int32_t     g_bAlarm    = FALSE;
static uint8_t u8Item= 0;

/*---------------------------------------------------------------------------------------------------------*/
/* RTC Tick Callback function                                                                              */
/*---------------------------------------------------------------------------------------------------------*/
void DrvRTC_TickISR(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
      \n",sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay,sCurTime.u32cHour,sCurTime.u3
      2cMinute,sCurTime.u32cSecond);
      g_u32TICK++;
}
```

```
/*---------------------------------------------------------------------------------------------------------*/
/* RTC Alarm Callback function                                                                             */
/*---------------------------------------------------------------------------------------------------------*/
void DrvRTC_AlarmISR(void)
{

    printf("Alarm!!\n");

    g_bAlarm = TRUE;
}
/*-----------------------------------------------*/
/*    define date and time                       */
/*-----------------------------------------------*/

int32_t getDateTime( E_DRVRTC_TIME_SELECT type , S_DRVRTC_TIME_DATA_T *sInitTime)
{

    //Year
    printf ("enter year :\n ");
    scanf    ("%d",&sInitTime->u32Year );
    while(sInitTime->u32Year-2000 > 99)
    {
      printf("Error! Please input year again: \n");
      scanf    ("%d",&sInitTime->u32Year );
    }
    printf ("year = %d\n", sInitTime->u32Year);

    // Month
    printf ("enter month (1-12) :\n ");
    scanf    ("%d", &sInitTime->u32cMonth);
    while( (sInitTime->u32cMonth==0 )    || (sInitTime->u32cMonth>12)    )
    {
      printf("Error! Please input month again: \n");
      scanf    ("%d",&sInitTime->u32cMonth );
    }
    printf ("month = %d\n", sInitTime->u32cMonth);

    //Day
    printf ("enter day (1-31) :\n ");
    scanf    ("%d", &sInitTime->u32cDay );
    while( (sInitTime->u32cDay==0 )    || (sInitTime->u32cDay>31)    )
    {
      printf("Error! Please input day again: \n");
      scanf    ("%d",&sInitTime->u32cDay );
    }
    printf ("day = %d\n", sInitTime->u32cDay);
```

```
// Day of week & Time Scale
    switch(type)
    {
        case DRVRTC_CURRENT_TIME:
        {
            //Day of week
            printf ("enter day of week (0-6) : \n");
            scanf    ("%d", &sInitTime->u32cDayOfWeek );
            while (sInitTime->u32cDayOfWeek >6)
            {
                printf("Error! Please input Day of Week again: \n");
                scanf    ("%d", &sInitTime->u32cDayOfWeek );
            }
            printf ("day = %d\n", sInitTime->u32cDayOfWeek);

            // Time scale
            printf("enter time scale (0:12hour / 1:24hour) \n");
            scanf ("%d",&sInitTime->u8cClockDisplay );
            while (sInitTime->u8cClockDisplay>1)
            {
                printf("Error! Please input time scale again: \n");
                scanf    ("%d",&sInitTime->u8cClockDisplay );
            }
            printf ("time scale = %d\n", sInitTime->u8cClockDisplay);


            //12Hour
            if (sInitTime->u8cClockDisplay == DRVRTC_CLOCK_12 )
            {
                printf ("enter Am or Pm (1:Am 2:Pm): \n");
                scanf ("%d", &sInitTime->u8cAmPm);
                while (sInitTime->u8cAmPm >2)
                {
                    printf("Error! Please input Am or Pm again: \n");
                    scanf    ("%d", &sInitTime->u8cAmPm );
                }

                if ( sInitTime->u8cAmPm==1)
                printf("Am \n");
                else
                printf("Pm \n");

                printf ("\nenter hour (1-12): \n");
                scanf    ("%d", &sInitTime->u32cHour );
                while ((sInitTime->u32cHour <1) ||    (sInitTime->u32cHour >12))
                {
                    printf("Error! Please input hour again: \n");
                    scanf    ("%d", &sInitTime->u32cHour );
                }
            }
```

```
            else
            {
                //24 Hour
                printf ("enter hour (0-24): \n");
                scanf    ("%d", &sInitTime->u32cHour );
                while    (sInitTime->u32cHour >23)
                {
                    printf("Error! Please input hour again: \n");
                    scanf    ("%d", &sInitTime->u32cHour );
                }
            }

            break;
        }

        case DRVRTC_ALARM_TIME:
        {
            if(RTC->TSSR.HR24 == DRVRTC_CLOCK_12 )
            {
                printf ("\nenter hour (1-12): \n");
                scanf    ("%d", &sInitTime->u32cHour );
                while ((sInitTime->u32cHour <1) ||    (sInitTime->u32cHour >12))
                {
                    printf("Error! Please input hour again: \n");
                    scanf    ("%d", &sInitTime->u32cHour );
                }
            }
            else
            {
                printf ("enter hour (0-24): \n");
                scanf    ("%d", &sInitTime->u32cHour );
                while    (sInitTime->u32cHour >23)
                {
                    printf("Error! Please input hour again: \n");
                    scanf    ("%d", &sInitTime->u32cHour );
                }
            }
            break;
        }
    }
    printf ("hour = %d\n", sInitTime->u32cHour);
    //Minute
    printf ("enter minute (0-59) : \n");
    scanf    ("%d", &sInitTime->u32cMinute );
    while (sInitTime->u32cMinute >59)
    {
        printf("Error! Please input Minute again: \n");
        scanf    ("%d", &sInitTime->u32cMinute );
    }
```

```
        printf ("minute = %d\n", sInitTime->u32cMinute);

        //Second
        printf ("enter second (0-59) : \n");
        scanf    ("%d", &sInitTime->u32cSecond );
        while (sInitTime->u32cSecond >59)
        {
                printf("Error! Please input Second again: \n");
                scanf    ("%d", &sInitTime->u32cSecond );
        }

        printf ("second = %d\n", sInitTime->u32cSecond);


        return E_SUCCESS;
}

void normalizeTime (S_DRVRTC_TIME_DATA_T *sPt)
{

        while    (sPt->u32cSecond >= 60)
        {
                sPt->u32cSecond = sPt->u32cSecond - 60;
                sPt->u32cMinute = sPt->u32cMinute + 1;
        }
        while (sPt->u32cMinute >= 60)
        {
                sPt->u32cMinute = sPt->u32cMinute - 60;
                sPt->u32cHour      = sPt->u32cHour + 1;
        }
        while (sPt->u32cHour >= 24)
        {
                sPt->u32cHour      = sPt->u32cHour - 24;
                sPt->u32cDay       = sPt->u32cDay + 1;
        }
        while (sPt->u32cDay >= 31)
        {
                sPt->u32cDay       = sPt->u32cDay - 31;
                sPt->u32cMonth     = sPt->u32cMonth + 1;
        }
        while (sPt->u32cMonth >= 12)
        {
                sPt->u32cMonth     = sPt->u32cMonth - 12;
                sPt->u32Year       = sPt->u32Year + 1;
        }

}
```

```
static void TestItem (void)
{
    printf("\n\n");
    printf("+------------------------------------------------------------+\n");
    printf("|                          RTC Sample Program      |\n");
    printf("+------------------------------------------------------------+\n");
    printf("| [0] Time Display Test                                       |\n");
    printf("| [1] Alarm Test                                              | \n");
    printf("| [2] Wake up CPU                                             |\n");
    printf("+------------------------------------------------------------+\n");
    printf("| [3] Quit                                                    |\n");
    printf("+------------------------------------------------------------+\n");
    printf("Select key : \n");
}


void SysTimerDelay(uint32_t us)
{
    SysTick->LOAD = us * 48;
    SysTick->VAL   =  (0x00);
    SysTick->CTRL = (1 << SYSTICK_CLKSOURCE) | (1<<SYSTICK_ENABLE);

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

/*---------------------------------------------------------------------------------------------------------*/
/* RTC Test Sample                                                                                         */
/* Test Item                                                                                               */
/*   1. Time Display Test                                                                                  */
/*       Use RTC Tick interrupt to display time every one second.                                          */
/*   2. Alarm Test                                                                                         */
/*       Get the current and alarm after 10 seconds                                                        */
/*---------------------------------------------------------------------------------------------------------*/
int32_t main()
{


    S_DRVRTC_TIME_DATA_T sInitTime;
    int32_t bLoop = TRUE;



    UNLOCKREG();
    SYSCLK->PWRCON.XTL32K_EN = 1;
    LOCKREG();
```

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

DrvUART_Init(115200);

/* Set UART Configuration */
if(1)
{
    /* RTC Initialize */
    DrvRTC_Init();

    /* Time Setting */
    getDateTime(DRVRTC_CURRENT_TIME, &sInitTime);

    /* Initialization the RTC timer */
    if (DrvRTC_Write(DRVRTC_CURRENT_TIME, &sInitTime) != E_SUCCESS)
    {
        printf("RTC Open Fail!!\n");
        return FALSE;
    }
    while(bLoop)
    {
        TestItem();
        u8Item = getchar();

        switch(u8Item)
        {
            case '0':
            {

                printf("\n0. RTC Time Display Test (Exit after 5 seconds)\n");

                /* Set Tick setting */
                DrvRTC_SetTickMode(DRVRTC_TICK_1_SEC );

                /* Enable RTC Tick Interrupt and install tick call back function */
                DrvRTC_EnableInt (DRVRTC_TICK_INT, DrvRTC_TickISR );

                g_u32TICK = 0;

                while(g_u32TICK < 5);

                /* Disable RTC Tick Interrupt */
                DrvRTC_DisableInt(DRVRTC_TICK_INT);

                break;
            }
```

```
case '1':
{
    S_DRVRTC_TIME_DATA_T sCurTime;

    printf("\n1. DrvRTCAlarm Test \n");
    g_bAlarm = FALSE;

    /* Get the currnet time */
    DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);
    printf("Current Time:%d/%02d/%02d %02d:%02d:%02d    day of week: %02d
    \n",sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay,sCurTime.u32
    cHour,sCurTime.u32cMinute,sCurTime.u32cSecond,
    sCurTime.u32cDayOfWeek);

    /* The alarm time setting */
    printf("\nSetting the Alarm Time: \n");
    getDateTime( DRVRTC_ALARM_TIME, &sInitTime);
    DrvRTC_Write(DRVRTC_ALARM_TIME,&sInitTime);

    /* Install the call back function and enable the alarm interrupt)*/
    DrvRTC_EnableInt (DRVRTC_ALARM_INT, DrvRTC_AlarmISR );
    printf("\nWaiting for alarm... \n");

    while(!g_bAlarm);

    DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);
    printf("Current Time:%d/%02d/%02d
    %02d:%02d:%02d\n",sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32c
    Day,sCurTime.u32cHour,sCurTime.u32cMinute,sCurTime.u32cSecond);

    /* Disable Alarm INT */
    DrvRTC_DisableInt(DRVRTC_ALARM_INT);

    break;
}
case '2':
{
    /* Get the currnet time */
    DrvRTC_Read(DRVRTC_CURRENT_TIME, &sInitTime);
    printf("\nCurrent Time:%d/%02d/%02d
    %02d:%02d:%02d\n",sInitTime.u32Year,sInitTime.u32cMonth,sInitTime.u32c
    Day,sInitTime.u32cHour,sInitTime.u32cMinute,sInitTime.u32cSecond);

    /* Set the Alarm time */
    printf("\nWake up CPU after 5 second\n");
    sInitTime.u32cSecond = sInitTime.u32cSecond + 5;
    normalizeTime(&sInitTime);
    DrvRTC_Write(DRVRTC_ALARM_TIME, &sInitTime);
```

```
                        /* Install the call back function and enable the alarm interrupt)*/
                        DrvRTC_EnableInt (DRVRTC_ALARM_INT, DrvRTC_AlarmISR );

                        /* Enable the Wakeup bit */
                        DrvRTC_EnableWakeUp();

                        /* Into stand by power down mode */
                        DrvPMU_StandbyPowerDown();

                        printf("Wake up CPU\n");

                        /* Get the current time */
                        DrvRTC_Read(DRVRTC_CURRENT_TIME, &sInitTime);
                        printf("\nCurrent Time:%d/%02d/%02d
                        %02d:%02d:%02d\n",sInitTime.u32Year,sInitTime.u32cMonth,sInitTime.u32c
                        Day,sInitTime.u32cHour,sInitTime.u32cMinute,sInitTime.u32cSecond);

                        /* Disable Alarm INT */
                        DrvRTC_DisableInt(DRVRTC_ALARM_INT);

                        /* Disable Wakeup bit */
                        DrvRTC_DisableWakeUp();

                        break;
                    }
                    case '3':
                        bLoop = FALSE;
                        break;
                    default:
                        printf("Wrong Item\n");
                        break;
                }

            }
            /* Disable RTC Clock */
            DrvRTC_Close();

            /* Disable UART Clock */
            DrvUART_Close(UART_PORT0);

            return TRUE;
        }
        else
            return FALSE;

}
```

5 Execution Environment Setup and Result

- Prepare a ISD9160 board.
- Compile the sample code.
- Console window show result of RTC display time, alarm test and wake up CPU function.

6 Revision History

Version	Date	Description
V1.00.01	Sep. 2011	Created