应用指南

32位 Cortex™-M0 单片机
NuMicro® 系列

NUC1XX 中怎样使用RTC

目录

1 简介

实时时钟（RTC）单元提供时间和日历信息给用户，RTC时钟源是连接在引脚X32I和X32O的外部32.768KHz晶振，或者从X32I引脚输入的32.768KHz的外部晶振。RTC单元储存时间信息（时分秒）在时间装载寄存器（TLR）中，日期信息（日月年）在日期装载寄存器(CLR)中，数据信息以BCD码表示，该单元提供报警功能，用户可以预先设置报警时间在时间报警寄存器（TAR）和报警日期在日期报警寄存器(CAR)中。

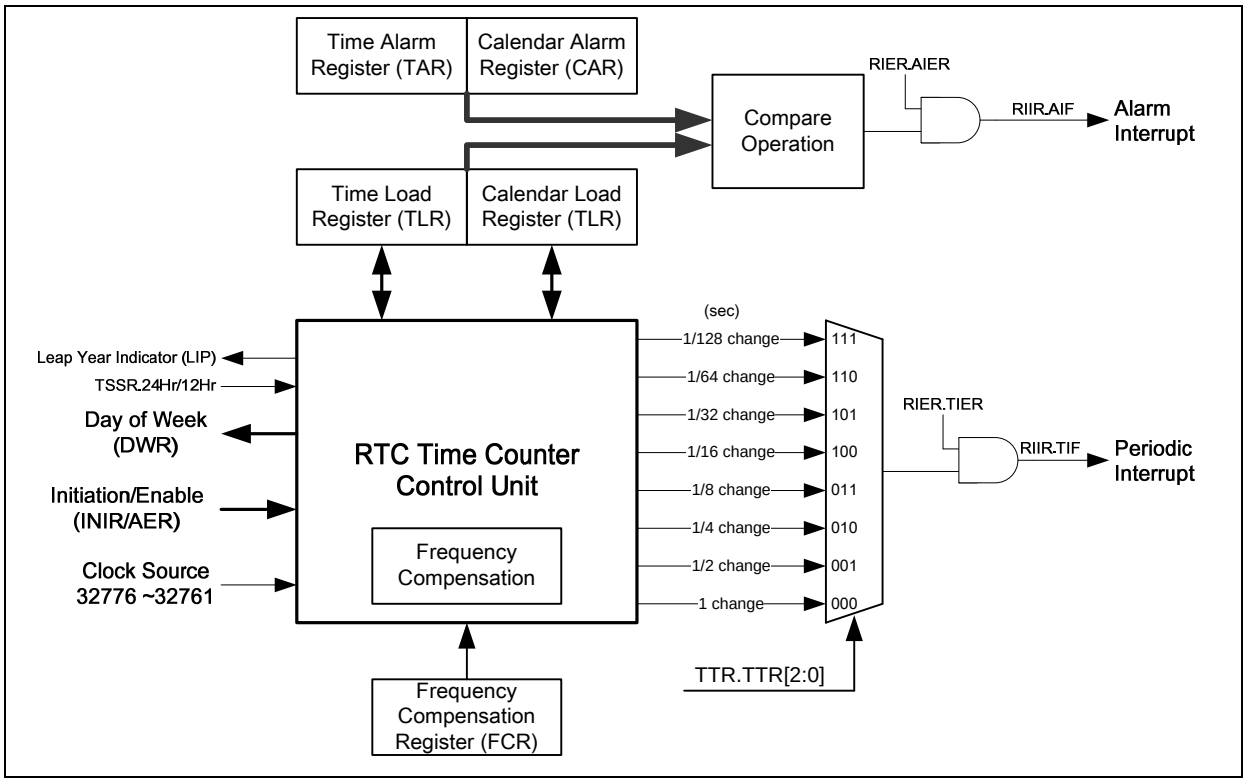
RTC单元支持时间隙和报警中断，中断周期有8种选择：1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2秒及1秒，通过TTR.TTR[2:0]选择，当RTC中的TLR和CLR等于报警设置TAR和CAR时，警报中断标志(RIIR.AIF)置位，如果此时警报中断使能（RIER.AIER=1）那么将产生中断请求。

2 RTC性能

- 设有一个时间计数器（时分秒）和日历计数器（年月日）为用户确认时间
- 报警寄存器（年月日时分秒）
- 可选12时或24时制
- 闰年自动补偿
- 星期计数
- 频率补偿寄存器(FCR)
- 所有时间和日期信息以BCD码表示
- 支持周期性时间隙中断，有8种周期选择：1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2秒 及1秒
- 支持报警中断
- 支持唤醒CPU功能

3 RTC框图

RTC框图如下表示:



RTC 框图

4 RTC功能描述

4.1 访问RTC寄存器

由于RTC时钟和系统时钟的差异，当用户对任意RTC寄存器写入新的数据时，寄存器将在2 RTC时钟周期(60us)后被更新，这样程序员必须考虑访问TSSR, TAR及TLR的顺序。

此外，用户必须明白RTC单元不会检测装载的数据是否超出界限，RTC也不会检查DWR和CLR间的合理性。

4.2 RTC初始化

RTC单元上电后，用户必须写数字(0xa5eb1357)到INIR中以复位所有逻辑电路，INIR作为硬件复位电路，一旦INIR被设置为0xa5eb1357，再对INIR寄存器做任何编程都不起作用

4.3 RTC读/写使能

寄存器AER的15-0位被保存作为RTC读/写密码，它被用来避免系统掉电时的干扰，系统上电后AER的15-0位必须被设置为0xa965 ，设置后再过512个RTC时钟周期（约15毫秒）开始起作用，程序员可以通过读RTC使能状态标志位AER.ENF来检查RTC单元是否开始工作。

4.4 频率补偿

RTC的FCR允许软件对输入时钟做数字补偿，时钟输入频率必须在32776Hz和32761Hz之间，在生产过程中，用户可以利用频率计数器测量GPIO引脚上输入的RTC时钟，并将测量的值存储在Flash存储器中，系统第一次上电时可以获得这些数据。下面是高/低频率时钟输入的补偿示例

示例1:

频率计数器测量值： 32773.65Hz (> 32768 Hz)

整数部分: 32773 => 0x8005

FCR.Integer = 0x05 – 0x01 + 0x08 = 0x0c

小数部分: 0.65 x 60 = 39 => 0x27

FCR.Fraction = 0x27

示例2

频率计数器测量值： 32765.27Hz (≦ 32768 Hz)

整数部分: 32765 => 0x7ffd

FCR.Integer = 0x0d – 0x01 – 0x08 = 0x04

小数部分: 0.27 x 60 = 16.2=> 0x10

FCR.Fraction = 0x10

4.5 时间和日历计数器

TLR和CLR用来加载时间和日历，TAR和CAR用来报警，他们都以BCD码表示

4.6 12/24 小时制选择

12/24 时制选择取决于TSSR的第0位

4.7 星期计数器

RTC单元在星期寄存器（DWR）中提供了星期信息，其中定义的值0-6代表了星期日到星期六

4.8 周期时间隙中断

周期时间隙中断有8种周期选择：1/128，1/64，1/32，1/16，1/8，1/4，1/2秒 及1秒，可以通过TTR.TTR[2:0]选择，当周期时间隙中断通过设置RIER.AIER 为1使能后，中断将以TTR寄存器设置的周期产生

4.9 周期时间报警

当RTC计数器TLR和CLR等于报警设置时间TAR和CAR时，报警中断标志(RIIR.AIF)被置位，如果报警中断使能(RIER.AIER=1)则将产生中断请求

4.10 应用注意事项:

1. TAR, CAR, TLR 和CLR 寄存器都是BCD计数器
2. 程序员必须保证装载的值是合理的，例如：不可以装载CLR寄存器201a（年），13（月），00（日）或CLR与DWR不匹配等等
3. 复位值 :

寄存器	复位值
AER	0
CLR	05/1/1 (年/月/日)
TLR	00:00:00 (时 : 分 : 秒)
CAR	00/00/00 (年/月/日)
TAR	00:00:00 (时 :分:秒)
TSSR	1 (24 时制)
DWR	6 (星期六)
RIER	0
RIIR	0
LIR	0

TTR	0
PWRTOUT	5555

4. TLR和TAR中，只有两位BCD码用来表示“年”，我们假设两位BCD表示20xy而不是19xy或者21xy.

5 寄存器表

R: 只读, **W**: 只写, **R/W**: 读/写, **C**: 只能写0

寄存器	偏移量	读/写	描述	复位值
RTC_BA = 0x4000_8000				
INIR	RTC_BA+0x000	R/W	RTC 初始化寄存器	0x0000_0000
AER	RTC_BA+0x004	R/W	RTC存取使能寄存器	0x0000_0000
FCR	RTC_BA+0x008	R/W	RTC频率补偿寄存器	0x0000_0700
TLR	RTC_BA+0x00C	R/W	时间装载寄存器	0x0000_0000
CLR	RTC_BA+0x010	R/W	日期装载寄存器	0x0005_0101
TSSR	RTC_BA+0x014	R/W	时制选择寄存器	0x0000_0001
DWR	RTC_BA+0x018	R/W	星期寄存器	0x0000_0006
TAR	RTC_BA+0x01C	R/W	时间报警寄存器	0x0000_0000
CAR	RTC_BA+0x020	R/W	日期报警寄存器	0x0000_0000
LIR	RTC_BA+0x024	R	闰年指示寄存器	0x0000_0000
RIER	RTC_BA+0x028	R/W	RTC中断使能寄存器	0x0000_0000
RIIR	RTC_BA+0x02C	R/C	RTC中断指示寄存器	0x0000_0000
TTR	RTC_BA+0x030	R/W	RTC时隙寄存器	0x0000_0000

6 寄存器描述

6.1 RTC初始化寄存器(INIR)

寄存器	偏移量	读/写	描述	复位值
INIR	RTC_BA+0x000	R/W	RTC初始化寄存器	0x0000_0000

31	30	29	28	27	26	25	24
INIR							
23	22	21	20	19	18	17	16
INIR							
15	14	13	12	11	10	9	8
INIR							
7	6	5	4	3	2	1	0
INIR							INIR/Active

位	描述	
[31:0]	**INIR**	**RTC初始化** 当RTC单元上电时，RTC在复位状态用户必须写一个数字(0xa5eb1357)到INIR以释放所有的逻辑和计数器，INIR就像硬件复位电路
[0]	**Active**	**RTC使能状态**(只读), 0: RTC在复位状态 1: RTC在正常使能状态

6.2 RTC读写使能寄存器(AER)

寄存器	偏移量	读/写	描述	复位值
AER	RTC_BA+0x004	R/W	RTC读写使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							ENF
15	14	13	12	11	10	9	8
AER							
7	6	5	4	3	2	1	0
AER							

位	描述	
[16]	ENF	**RTC寄存器读写使能**(只读) 1 = RTC寄存器读写使能 0 = RTC寄存器读写禁止 当AER[15:0]寄存器装载0xA965后此位置位，128RTC时钟周期后或者AER[15:0]不是0xA965后此位自动清零
[15:0]	AER	**RTC寄存器读写使能密码**(读写) 0xA965 = 使能RTC读写 其他 = 禁止RTC读写

6.3 RTC频率补偿寄存器(FCR)

寄存器	偏移量	读/写	描述	复位值
FCR	RTC_BA+0x008	R/W	频率补偿寄存器	0x0000_0700

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				INTEGER			
7	6	5	4	3	2	1	0
RESERVED		FRACTION					

位	描述	
[11:8]	**INTEGER**	**整数部分** 测量值整数部分 / FCR[11:8] / 测量值整数部分 / FCR[11:8] 32776 / 1111 / 32768 / 0111 32775 / 1110 / 32767 / 0110 32774 / 1101 / 32766 / 0101 32773 / 1100 / 32765 / 0100 32772 / 1011 / 32764 / 0011 32771 / 1010 / 32763 / 0010 32770 / 1001 / 32762 / 0001 32769 / 1000 / 32761 / 0000
[5:0]	**FRACTION**	**小数部分** 公式= (测量值小数部分) x 60 注意:FCR中的数字必须以十六进制表示，参考4.4示例

注意: RTC使能后此寄存器可以被读出

6.4 RTC时间装载寄存器(TLR)

寄存器	偏移量	读/写	描述	复位值
TLR	RTC_BA+0x00C	R/W	时间装载寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED		10HR		1HR			
15	14	13	12	11	10	9	8
RESERVED	10MIN			1MIN			
7	6	5	4	3	2	1	0
RESERVED	10SEC			1SEC			

位	描述	
[21:20]	**10HR**	10小时时间数字(0~3)[2]
[19:16]	**1HR**	1小时时间数字(0~9)
[14:12]	**10MIN**	10分钟时间数字(0~5)
[11:8]	**1MIN**	1分钟时间数字(0~9)
[6:4]	**10SEC**	10秒时间数字(0~5)
[3:0]	**1SEC**	1秒时间数字(0~9)

注意:

1. TLR是一个BCD码数字计数器，RTC不会检查装载的值
2. 合理值的范围在括号中列出

6.5 RTC日期装载寄存器(CLR)

寄存器	偏移量	读/写	描述	复位值
CLR	RTC_BA+0x010	R/W	日期装载寄存器	0x0005_0101

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
10YEAR				1YEAR			
15	14	13	12	11	10	9	8
RESERVED			10MON	1MON			
7	6	5	4	3	2	1	0
RESERVED		10DAY		1DAY			

位	描述	
[23:20]	**10YEAR**	10年日期数字(0~9)
[19:16]	**1YEAR**	1年日期数字(0~9)
[12]	**10MON**	10月日期数字 (0~1)
[11:8]	**1MON**	1月日期数字(0~9)
[5:4]	**10DAY**	10天日期数字(0~3)
[3:0]	**1DAY**	1天日期数字(0~9)

注意:

1. TLR是一个BCD码计数器，RTC将不会检测装载的数据
2. 合理值的范围在括号中列出

6.6 RTC时制选择寄存器(TSSR)

寄存器	偏移量	读/写	描述	复位值
TSSR	RTC_BA+0x014	R/W	时制选择寄存器	0x0000_0001

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							24hr/12hr

位	描述

<table>
<tr>
<td>[0]</td>
<td>24hr/12hr</td>
<td>
24时/ 12时制选择
显示了TLR和TAR是24时制还是12时制
1 = 选择24时制
0 = 选择12时制，带有上午下午显示
<table>
<tr><th>24时制</th><th>12时制</th><th>24时制</th><th>12时制
(下午时间+ 20)</th></tr>
<tr><td>00</td><td>12(AM12)</td><td>12</td><td>32(PM12)</td></tr>
<tr><td>01</td><td>01(AM01)</td><td>13</td><td>21(PM01)</td></tr>
<tr><td>02</td><td>02(AM02)</td><td>14</td><td>22(PM02)</td></tr>
<tr><td>03</td><td>03(AM03)</td><td>15</td><td>23(PM03)</td></tr>
<tr><td>04</td><td>04(AM04)</td><td>16</td><td>24(PM04)</td></tr>
<tr><td>05</td><td>05(AM05)</td><td>17</td><td>25(PM05)</td></tr>
<tr><td>06</td><td>06(AM06)</td><td>18</td><td>26(PM06)</td></tr>
<tr><td>07</td><td>07(AM07)</td><td>19</td><td>27(PM07)</td></tr>
<tr><td>08</td><td>08(AM08)</td><td>20</td><td>28(PM08)</td></tr>
<tr><td>09</td><td>09(AM09)</td><td>21</td><td>29(PM09)</td></tr>
<tr><td>10</td><td>10(AM10)</td><td>22</td><td>30(PM10)</td></tr>
<tr><td>11</td><td>11(AM11)</td><td>23</td><td>31(PM11)</td></tr>
</table>
</td>
</tr>
</table>

6.7 RTC星期寄存器(DWR)

寄存器	偏移量	读/写	描述	复位值
DWR	RTC_BA+0x018	R/W	星期寄存器	0x0000_0006

31	30	29	28	27	26	25	24
RESERVED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED					**DWR**		

位	描述	
[2:0]	**DWR**	**星期寄存器** 值 / 星期 0 / 星期日 1 / 星期一 2 / 星期二 3 / 星期三 4 / 星期四 5 / 星期五 6 / 星期六

6.8 RTC时间报警寄存器(TAR)

寄存器	偏移量	读/写	描述	复位值
TAR	RTC_BA+0x01C	R/W	时间报警寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED		10HR		1HR			
15	14	13	12	11	10	9	8
RESERVED	10MIN			1MIN			
7	6	5	4	3	2	1	0
RESERVED	10SEC			1SEC			

位	描述	
[21:20]	**10HR**	报警设置10小时数字(0~3)[2]
[19:16]	**1HR**	报警设置1小时数字(0~9)
[14:12]	**10MIN**	报警设置10分钟数字 (0~5)
[11:8]	**1MIN**	报警设置1分钟数字(0~9)
[6:4]	**10SEC**	报警设置10秒钟数字(0~5)
[3:0]	**1SEC**	报警设置1秒钟数字 (0~9)

注意:

1. TLR 是 BCD码计数器，RTC不会检查装载的数据
2. 合理值已经在括号中列出.
3. RTC单元使能后该寄存器可读.

6.9 RTC日期报警寄存器(CAR)

寄存器	偏移量	读/写	描述	复位值
CAR	RTC_BA+0x020	R/W	日期报警寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
10YEAR				1YEAR			
15	14	13	12	11	10	9	8
RESERVED			10MON	1MON			
7	6	5	4	3	2	1	0
RESERVED		10DAY		1DAY			

位	描述	
[23:20]	**10YEAR**	报警设置10年日期数字(0~9)
[19:16]	**1YEAR**	报警设置1年日期数字 (0~9)
[12]	**10MON**	报警设置10月日期数字(0~1)
[11:8]	**1MON**	报警设置1月日期数字(0~9)
[5:4]	**10DAY**	报警设置10天日期数字(0~3)
[3:0]	**1DAY**	报警设置1天日期数字(0~9)

注意:

1. TLR 是 BCD码计数器，RTC不会检查装载的数据
2. 合理值已经在括号中列出.
3. RTC单元使能后该寄存器可读

6.10 RTC闰年指示寄存器(LIR)

寄存器	偏移量	读/写	描述	复位值
LIR	RTC_BA+0x024	R	RTC闰年指示寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							LIR

Bits	Descriptions	
[0]	LIR	**闰年指示寄存器**(只读). 1 = 表示这一年是闰年 0 = 表示这一年不是闰年

6.11 RTC中断使能寄存器(RIER)

寄存器	偏移量	读/写	描述	复位值
RIER	RTC_BA+0x028	R/W	RTC中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED						**TIER**	**AIER**

位	描述	
[1]	**TIER**	**时间隙中断使能** 1 = RTC时间隙中断使能 0 = RTC时间隙中断禁止
[0]	**AIER**	**报警中断使能** 1 = RTC报警中断使能 0 = RTC报警中断禁止

6.12 RTC中断指示寄存器(RIIR)

寄存器	偏移量	读/写	描述	复位值
RIIR	RTC_BA+0x02C	R/C	RTC中断指示寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						TI	AI

位	描述	
[1]	TIF	**RTC时间隙中断指示标志** 当RTC时间隙中断使能(RIER.TIER=1)，RTC单元将按照TTR[2:0]选择的周期不停设置TIF为高，此位通过软件写1清零
[0]	AIF	**RTC报警中断指示标志** 当RTC报警中断使能(RIER.AIER=1)，一旦RTC实时计数器TLR和CLR达到报警设置寄存器TAR和CAR的值，RTC单元将设置AIF为高，此位通过软件写1清零

6.13 RTC时间隙寄存器(TTR)

寄存器	偏移量	读/写	描述	复位值
TTR	RTC_BA+0x030	R/C	RTC时间隙寄存器	0x0000_0000

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					TTR[2:0]		

位	描述	
[2:0]	TTR	**时间隙寄存器** 周期时隙中断请求的RTC时隙 TTR[2:0] / 时隙(秒) 0 / 1 1 / 1/2 2 / 1/4 3 / 1/8 4 / 1/16 5 / 1/32 6 / 1/64 7 / 1/128 注意: RTC使能后该寄存器可读取

7 示例代码

```
/*---------------------------------------------------------------------------------------*/
/*                                                                                       */
/* Copyright(c) 2009 Nuvoton Technology Corp. All rights reserved.                       */
/*                                                                                       */
/*---------------------------------------------------------------------------------------*/
#include <stdio.h>

#include "NUC1xx.h"
#include "Driver/DrvUART.h"
#include "Driver/DrvRTC.h"
#include "Driver/DrvGPIO.h"
#include "Driver/DrvSYS.h"
/*---------------------------------------------------------------------------------------*/
/* Global variables                                                                      */
/*---------------------------------------------------------------------------------------*/
volatile uint32_t  g_u32TICK = FALSE;
volatile int32_t   g_bAlarm  = FALSE;


/*---------------------------------------------------------------------------------------*/
/* 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.u32cMinute
,sCurTime.u32cSecond);
        g_u32TICK++;
}

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

        printf("Alarm!!\n");

        g_bAlarm = TRUE;
```

```
}


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("+------------------------------------------------------------+\n");
    printf("| [2] Quit                                                   |\n");
    printf("+------------------------------------------------------------+\n");
    printf("Select key : \n");
}


/*---------------------------------------------------------------------------------------------*/
/* 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;
        uint8_t u8Item;

        STR_UART_T sParam;

        /* 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 Pin */
        DrvGPIO_InitFunction(FUNC_UART0);

        /* Set UART Configuration */
```

```
if(DrvUART_Open(UART_PORT0,&sParam) == E_SUCCESS)
{
        /* RTC Initialize */
        DrvRTC_Init();

        /* Time Setting */
        sInitTime.u32Year           = 2009;
        sInitTime.u32cMonth         = 1;
        sInitTime.u32cDay           = 19;
        sInitTime.u32cHour          = 13;
        sInitTime.u32cMinute        = 20;
        sInitTime.u32cSecond        = 0;
        sInitTime.u32cDayOfWeek = DRVRTC_MONDAY;
        sInitTime.u8cClockDisplay = DRVRTC_CLOCK_24;

        /* Initialization the RTC timer */
        if(DrvRTC_Open(&sInitTime) !=E_SUCCESS)
        {
                printf("RTC Open Fail!!\n");
                return FALSE;
        }

        while(bLoop)
        {
                TestItem();
                u8Item = getchar();

                switch(u8Item)
                {
                        case '0':
                        {
                                S_DRVRTC_TICK_T sTick;

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

                                /* Set Tick property */
                                sTick.ucMode = DRVRTC_TICK_1_SEC;
                                sTick.pfnTickCallBack = DrvRTC_TickISR;

                                /* Set Tick setting */
                                DrvRTC_Ioctl(0,DRVRTC_IOC_SET_TICK_MODE,
                                            (uint32_t)&sTick,0);

                /* Enable RTC Tick Interrupt and install tick call back function */
                                DrvRTC_Ioctl(0,DRVRTC_IOC_ENABLE_INT,
(uint32_t)DRVRTC_TICK_INT,0);
```

```
                g_u32TICK = 0;

                NVIC_EnableIRQ(RTC_IRQn);
                while(g_u32TICK < 5);

                /* Disable RTC Tick Interrupt */
                DrvRTC_Ioctl(0,DRVRTC_IOC_DISABLE_INT,
                (uint32_t)DRVRTC_TICK_INT,0);
                break;
        }
        case '1':
        {
                S_DRVRTC_TIME_DATA_T sCurTime;
        printf("\n1. DrvRTC Alarm Test (Alarm after 10 seconds)\n");

                g_bAlarm = FALSE;

                /* Get the currnet time */
                DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);


                /* Set Alarm call back function */
                sCurTime.pfnAlarmCallBack = DrvRTC_AlarmISR;

                printf("Current Time:%d/%02d/%02d %02d:%02d:%02d\n",
sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay,sCurTime.u32cHour,sCurTime.u32cMinute
,sCurTime.u32cSecond);

                /* The alarm time setting */
                sCurTime.u32cSecond = sCurTime.u32cSecond + 10;

/* Set the alarm time (Install the call back function and enable the alarm interrupt)*/
                DrvRTC_Write(DRVRTC_ALARM_TIME,&sCurTime);

                NVIC_EnableIRQ(RTC_IRQn);

                while(!g_bAlarm);

                printf("Current Time:%d/%02d/%02d %02d:%02d:%02d\n"
,sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay,sCurTime.u32cHour,sCurTime.u32cMinut
e,sCurTime.u32cSecond);

                break;
        }
        case '2':
                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;

}
```

8 修订历史

版本	日期	页	描述
A01	2010-2-26		初次发布

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.