文档版本	V1.0
发布日期	20210525

APT32F102 RTC 应用指南

目录

1 概述

本文介绍了在APT32F102中使用RTC的应用范例。

2. 适用的硬件

该例程使用于 APT32F102x 系列学习板

3. 应用方案代码说明

3.1 RTC 配置

基于 APT32F102 完整的库文件系统，进行配置。

- **硬件配置：**

RTC 模块提供实时的日历和时间信息，包括星期，年、月、日和小时、分钟、秒。

RTC 在所有低功耗模式下均可独立运行，并支持系统唤醒。

- **功能框图**

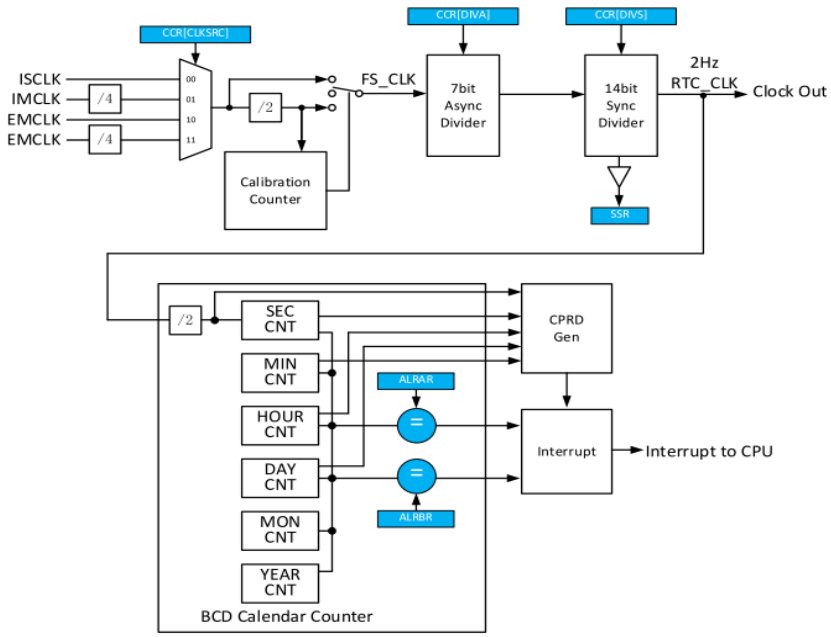


图 3.1.1 功能框图

- **软件配置：**

  可在 apt32f102_initial.c 文件中进行初始化的配置；

  编程要点：

  1. 通过 SYSCON_CONFIG() 选择时钟

  2. 初始化配置 RTC 及设置时间

```
/*************************************************************/
//RTC Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
//RTC CLK=(CLKSRC_EMOSC)/(DIVS+1)/(DIVA+1)/4
//              (ISCLK)/(DIVS+1)/(DIVA+1)/4
//              (IMCLK)/(DIVS+1)/(DIVA+1)/4
void RTC_CONFIG(void)
{
  RTC_RST_VALUE();
  RTC_Stop();                                          //if you want set time data, must be stop RTC clk.
  RTCCLK_CONFIG(2777,124,CLKSRC_IMOSC_4div);         //5.556M /4 /4 /(2777+1) / (124+1) = 1s
  //
  RTC_Function_Config(RTC_24H,CPRD_1S,COSEL_NoCali_1hz);   //Enalbe AlarmA , Enalbe AlarmB , RTC Select 24h , CPRD Select 1s

  RTC_TimeDate_buf.u8Second=0x00;
  RTC_TimeDate_buf.u8Minute=0x59;
  RTC_TimeDate_buf.u8Hour=0X23;                        //24 制
  RTC_TimeDate_buf.u8Day=0X20;
  RTC_TimeDate_buf.u8Month=0X04;
  RTC_TimeDate_buf.u8Year=0X21;
  RTC_TimeDate_buf.u8DayOfWeek=0x02;
  //20 年 5 月 20 日 14 时 42 分 00 秒
  RTC_TIMR_DATR_SET(&RTC_TimeDate_buf);
  //必须打开 ALARMA 设置该时间用于调整小时进位错误问题
  RTC_AlarmA_buf.u8Second=0x59;
  RTC_AlarmA_buf.u8Minute=0x59;
  RTC_AlarmA_buf.u8Hour=0X09;
  RTC_Alarm_TIMR_DATR_SET(Alarm_A,&RTC_AlarmA_buf,
  //闹钟时间 7 时 00 分 00 秒,周末闹铃
  Alarm_Second_Compare_EN,Alarm_Minute_Compare_EN,Alarm_Hour_Compare_EN,Alarm_DataOrWeek_Compare_DIS,Alarm_data_selecte);
  RTC_AlarmB_buf.u8Second=0x02;
  RTC_AlarmB_buf.u8Minute=0x01;
  RTC_AlarmB_buf.u8Hour=0X20;                                                    //24 制
```

```
    RTC_AlarmB_buf.u8WeekOrData=0X22;
    RTC_Alarm_TIMR_DATR_SET(Alarm_B,&RTC_AlarmB_buf,
    //闹钟时间 7 时 00 分 00 秒,周末闹铃
    Alarm_Second_Compare_EN,Alarm_Minute_Compare_EN,Alarm_Hour_Compare_EN,Alarm_DataOrWeek_Compare_EN,Alarm_data_selecte);
    //RTC_Int_Enable(ALRA_INT);
    //RTC_Int_Enable(ALRB_INT);
    //RTC_Int_Enable(CPRD_INT);
    //RTC_Int_Enable(RTC_TRGEV0_INT);
    //RTC_Int_Enable(RTC_TRGEV1_INT);
    RTC_Vector_Int_Enable();
    //RTC_WakeUp_Enable();
    RTC_Start();
    //RTC EVT
    //RTC_TRGSRC0_Config(RTC_EVTRG_TRGSRC0_CPRD,RTC_ESYNxOE_EN,1);
    //RTC_TRGSRC1_Config(RTC_EVTRG_TRGSRC1_CPRD,RTC_ESYNxOE_EN,2);
    //RTC_TRGSRC0_SWFTRG();                    //RTC_TRGEV0    SW SET
    //RTC_TRGSRC1_SWFTRG();                    //RTC_TRGEV1    SW SET
}
```

代码说明：

RTC_RST_VALUE()------用于复位寄存器初值

RTC_Stop()-----用于停止 RTC，设置 RTC 到初始化模式

RTCCLK_CONFIG()-------用于 RTC 时钟源选择。

RTC_Function_Config()-----用于 RTC 功能配置

RTC_TIMR_DATR_SET();---用于设置初始时间

RTC_Alarm_TIMR_DATR_SET();---用于设置闹钟时间

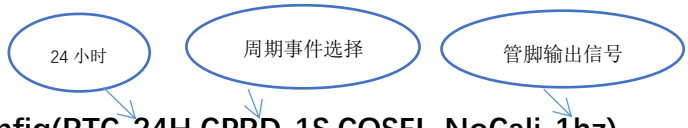
RTC_Int_Enable();----用于清除中断

RTC_Vector_Int_Enable();-----*用于开启 RTC 中断*

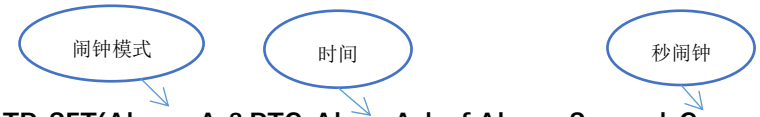
RTC_Start();---*用于开启 RTC*

函数参数说明：

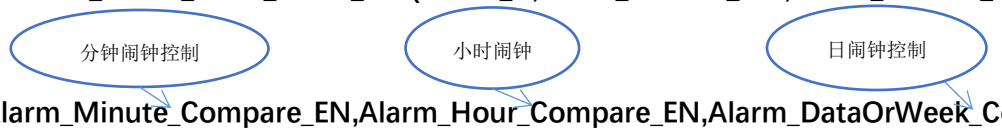
RTCCLK_CONFIG(2777,124,CLKSRC_IMOSC_4div);



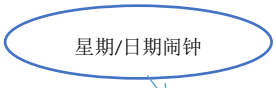

```
RTC_Function_Config(RTC_24H,CPRD_1S,COSEL_NoCali_1hz)
```




```
RTC_Alarm_TIMR_DATR_SET(Alarm_A,&RTC_AlarmA_buf,Alarm_Second_Compare_EN,
```




```
Alarm_Minute_Compare_EN,Alarm_Hour_Compare_EN,Alarm_DataOrWeek_Compare_DIS,
```




```
Alarm_data_selecte);
```

3.2 RTC 时钟配置

```
volatile    uint8_t R_RTC_BUF[10];
/************************************************************************/
//main
/************************************************************************/
int main(void)
{
   //delay_nms(5000);
   APT32F102_init();
  while(1)
   {
          //SYSCON_IWDCNT_Reload();
          //.....
          RTC_TIMR_DATR_Read(&RTC_TimeDate_buf);
          if(RTC_TimeDate_buf.u8Second != R_RTC_BUF[0])
          {
                R_RTC_BUF[0] = RTC_TimeDate_buf.u8Second;
                UARTTxByte(UART0,0X0A);
                UARTTxByte(UART0,RTC_TimeDate_buf.u8Hour);
                UARTTxByte(UART0,RTC_TimeDate_buf.u8Minute);
                UARTTxByte(UART0,RTC_TimeDate_buf.u8Second);
          }
    }
}
```

- **串口打印输出数据：**

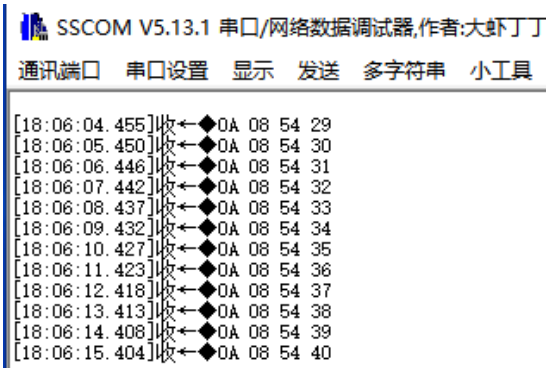


图 3.2.1 时间数据输出

3.3 RTC 定时

系统时钟为内部主频 48MHz,内部 IMOSC 5.556Mhz 作为 RTC 的时钟。定时 1S 翻转一次 PA0.12。

```
/*************************************************************/
//GPIO Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_Init(GPIOA0,12,0);                         //PA0.12 as output
    GPIO_Write_High(GPIOA0,12);                     //PA0.12 output High
}

/*************************************************************/
//RTC Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
//RTC CLK=(CLKSRC_EMOSC)/(DIVS+1)/(DIVA+1)/4
//              (ISCLK)/(DIVS+1)/(DIVA+1)/4
//              (IMCLK)/(DIVS+1)/(DIVA+1)/4
void RTC_CONFIG(void)
{
    RTC_RST_VALUE();
    RTC_Stop();                                     //if you want set time data, must be stop RTC clk.
    RTCCLK_CONFIG(2777,124,CLKSRC_IMOSC_4div);      ////5.556M/4/4/(2777+1)/(124+1)=1S
    RTC_Function_Config(RTC_24H,CPRD_1S,COSEL_NoCali_1hz);  //Enalbe AlarmA , Enalbe AlarmB , RTC Select 24h , CPRD Select 1s
   RTC_Int_Enable(CPRD_INT);
```

```
RTC_Vector_Int_Enable();
RTC_Start();
}

volatile U8_T BitRtcIO;
/*****************************************************************/
//RTC Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void RTCIntHandler(void)
{
    // ISR content ...
 if((RTC->MISR&ALRA_INT)==ALRA_INT)                    //ALRAR
 {
        RTC->ICR=ALRA_INT;
        RTC->KEY=0XCA53;
        RTC->CR=RTC->CR|0x01;
        RTC->TIMR=(0x10<<16)|(0x00<<8)|(0x00);          //Hour bit6->0:am 1:pm
        while(RTC->CR&0x02);//busy  判断 TIMR DATR ALRAR ALRBR 数据写完
        RTC->CR &= ~0x1;
 }
 else if((RTC->MISR&ALRB_INT)==ALRB_INT)               //ALRBR
 {
        RTC->ICR=ALRB_INT;
 }
 else if((RTC->IMCR&CPRD_INT)==CPRD_INT)               //CPRD
 {
        RTC->ICR=CPRD_INT;

        if(!BitRtcIO)
        {
            BitRtcIO    = 1;
            GPIO_Write_Low(GPIOA0,12);
        }
        else
        {
            BitRtcIO    = 0;
            GPIO_Write_High(GPIOA0,12);
        }
 }
 else if((RTC->IMCR&RTC_TRGEV0_INT)==RTC_TRGEV0_INT)
 {
        RTC->ICR=RTC_TRGEV0_INT;
```

```
    }
    else if((RTC->IMCR&RTC_TRGEV1_INT)==RTC_TRGEV1_INT)
    {
        RTC->ICR=RTC_TRGEV1_INT;
    }
}
```

- **代码说明：**

RTCCLK_CONFIG(2777,124,CLKSRC_IMOSC_4div); ----用于配置 RTC 时钟

RTC_Function_Config(RTC_24H,CPRD_1S,COSEL_NoCali_1hz); ----用于配置 RTC 周期事件

RTC_Int_Enable(CPRD_INT); ----用于使能 RTC 周期中断

RTC_Vector_Int_Enable(); ----用于开启中断

- **输出波形：**

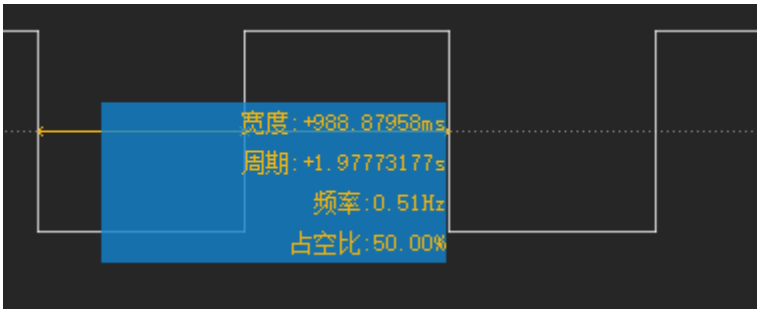


图 3.3.1 定时翻转 IO

3.4 使用外部晶振

- 硬件配置：

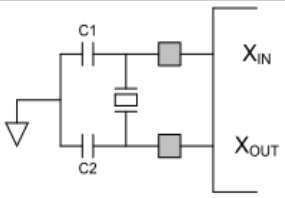
外接晶振（低频模式）	–			32.768		KHz

图 3.4.1 硬件接法

C1/C2 电容范围 20-30pF

外部晶振：32.768KHZ

- **外接晶振引脚：**

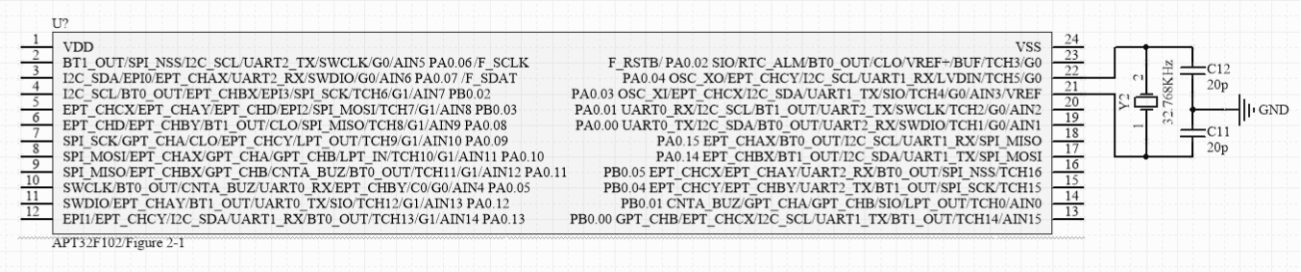


图 3.4.2 外部晶振电路

- 编程要点：

1. 配置 SYSCON_CONFIG 函数，选择对应的外部时钟 ENDIS_EMOSC
2. 配置 RTC_CONFIG 函数。

```
/*************************************************************/

//syscon Functions

//EntryParameter:NONE

//ReturnValue:NONE

/*************************************************************/

void SYSCON_CONFIG(void)

{

//------SYSTEM CLK AND PCLK FUNTION--------------------------/

    SYSCON_RST_VALUE();                                          //SYSCON all register clr

    //SYSCON_General_CMD(ENABLE,ENDIS_ISOSC);                   // ISOSC    内部振荡器

    EMOSC_OSTR_Config(0XAD,0X1f,EM_LFSEL_EN,EM_FLEN_EN,EM_FLSEL_10ns);

   SYSCON_General_CMD(ENABLE,ENDIS_EMOSC);

    //

   SYSCON_HFOSC_SELECTE(HFOSC_SELECTE_48M);                     //HFOSC selected 48MHz

    SystemCLK_HCLKDIV_PCLKDIV_Config(SYSCLK_HFOSC,HCLK_DIV_1,PCLK_DIV_1,HFOSC_48M);

//------------    WDT FUNTION    -------------------------------/

      SYSCON_IWDCNT_Config(IWDT_TIME_2S,IWDT_INTW_DIV_7);        //WDT TIME 1s,WDT alarm interrupt time=1s-1s*1/8=0.875S

      //SYSCON_WDT_CMD(ENABLE);

    SYSCON_WDT_CMD(DISABLE);                                     //enable WDT

      SYSCON_IWDCNT_Reload();                                    //reload WDT
```

```
    //IWDT_Int_Enable();
//------------    WWDT FUNTION    --------------------------------/
    WWDT_CNT_Load(0xFF);
    WWDT_CONFIG(PCLK_4096_DIV0,0xFF,WWDT_DBGDIS);
    WWDT_Int_Config(DISABLE);
    //WWDT_CMD(ENABLE);
//------------    CLO    --------------------------------/
    //SYSCON_CLO_CONFIG(CLO_PA02);
    //SYSCON->OPT1=(SYSCON->OPT1&0XFFFF8000)|(0X01<<12)|(0X04<<8)|(0x00<<4);
//------------    LVD FUNTION    --------------------------------/
   SYSCON_LVD_Config(DISABLE_LVDEN,INTDET_LVL_3_3V,RSTDET_LVL_1_9V,DISABLE_LVD_INT,INTDET_POL_fall);
  //LVD_Int_Enable();
//------------ EVTRG function --------------------------------/
    //SYSCON->EVTRG=0X00|0x01<<20;                                    //SYSCON_trgsrc0
    //SYSCON->EVPS=0X00;
   //SYSCON->IMER =EM_EVTRG0_ST;
//------------    SYSCON Vector    --------------------------------/
    //SYSCON_Int_Enable();                                             //SYSCON VECTOR
    //SYSCON_WakeUp_Enable();                                          //Enable WDT wakeup INT
}
/*****************************************************************/
//RTC Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void RTC_CONFIG(void)
{
   RTC_RST_VALUE();                                    //恢复默认值
    RTC_Stop();                                        //设置保护寄存器关闭 if you want set time data, must be stop RTC clk.
```

```
//RTC_ALM_IO_SET(Alarm_A_pulse_output);              //PA0.2   1pulse =1 rtc clk
RTCCLK_CONFIG(2047,3,CLKSRC_EMOSC);              //32.768K/4/(2047+1)/(3+1)=1S      硬件上有 1/4 分频
RTC_Function_Config(RTC_24H,CPRD_1S,COSEL_NoCali_1hz);      //Enalbe AlarmA , Enalbe AlarmB , RTC Select 24h , CPRD Select 1s
RTC_TimeDate_buf.u8Second=0x00;
RTC_TimeDate_buf.u8Minute=0x53;
RTC_TimeDate_buf.u8Hour=0X08;                                        //24 制
RTC_TimeDate_buf.u8Day=0X29;
RTC_TimeDate_buf.u8Month=0X05;
RTC_TimeDate_buf.u8Year=0X20;
RTC_TimeDate_buf.u8DayOfWeek=0x04;
RTC_TIMR_DATR_SET(&RTC_TimeDate_buf);                                //20 年 5 月 20 日 14 时 42 分 00 秒
RTC_AlarmA_buf.u8Second=0x59;                                        //必须打开 ALARMA 设置该时间用于调整小时进位错误问题
RTC_AlarmA_buf.u8Minute=0x59;
RTC_AlarmA_buf.u8Hour=0X09;
RTC_Alarm_TIMR_DATR_SET(Alarm_A,&RTC_AlarmA_buf,
Alarm_Second_Compare_EN,Alarm_Minute_Compare_EN,Alarm_Hour_Compare_EN,Alarm_DataOrWeek_Compare_DIS,Alarm_data_selecte);
RTC_AlarmB_buf.u8Second=0x02;
RTC_AlarmB_buf.u8Minute=0x01;
RTC_AlarmB_buf.u8Hour=0X20;
RTC_AlarmB_buf.u8WeekOrData=0X22;
RTC_Alarm_TIMR_DATR_SET(Alarm_B,&RTC_AlarmB_buf,
Alarm_Second_Compare_EN,Alarm_Minute_Compare_EN,Alarm_Hour_Compare_EN,Alarm_DataOrWeek_Compare_EN,Alarm_data_selecte);
RTC_Int_Enable(ALRA_INT);
//RTC_Int_Enable(ALRB_INT);
RTC_Int_Enable(CPRD_INT);
//RTC_Int_Enable(RTC_TRGEV0_INT);
//RTC_Int_Enable(RTC_TRGEV1_INT);
RTC_Vector_Int_Enable();
RTC_WakeUp_Enable();
```

```
    RTC_Start();

    //RTC EVT

    //RTC_TRGSRC0_Config(RTC_EVTRG_TRGSRC0_CPRD,RTC_ESYNxOE_EN,1);

    //RTC_TRGSRC1_Config(RTC_EVTRG_TRGSRC1_CPRD,RTC_ESYNxOE_EN,2);

    //RTC_TRGSRC0_SWFTRG();                          //RTC_TRGEV0      SW SET

    //RTC_TRGSRC1_SWFTRG();                          //RTC_TRGEV1      SW SET

}
```

代码说明：

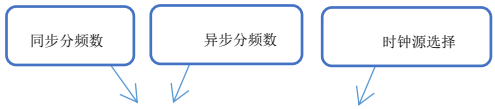
EMOSC_OSTR_Config(); ----用于配置外部时钟

SYSCON_General_CMD(); ----用于开启外部时钟使能

RTCCLK_CONFIG(); ----用于配置 RTC 时钟源

函数参数说明：

EMOSC_OSTR_Config(0XAD,0X1f,EM_LFSEL_EN,EM_FLEN_EN,EM_FLSEL_10ns);



RTCCLK_CONFIG(2047,3,CLKSRC_EMOSC);

- **串口打印输出数据**

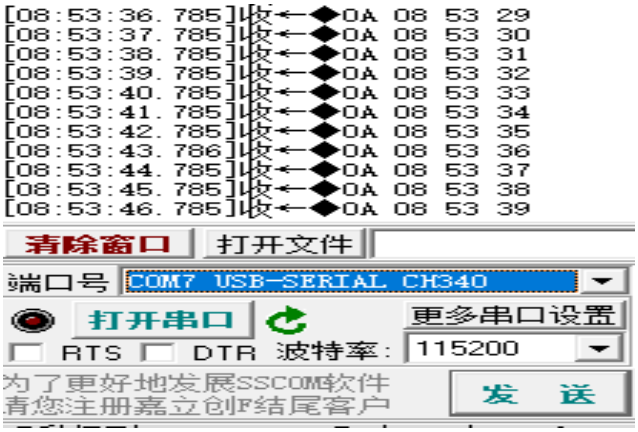


图 3.4.3 RTC 时间

4. 程序下载和运行

1．将目标板与仿真器连接，分别为 VDD SCLK SWIO GND

2. 程序编译后仿真运行

3. 通过查看图 3.2.1、图 3.3.1、图 3.4.3 数据验证