AN1021
应用笔记

使用 PY32F030/003 微控制器的 RTC 实时时钟模块

前言

RTC(Real Time Clock)实时时钟是记录当前时间的计算机时钟。实时时钟的模块是一个独立的定时器，拥有一组连续计数的计数器。

本应用笔记提供了含有配置 RTC 日历，RTC 闹钟等功能的代码例程。

在本文档中，PY32 仅指表 1 中列出的产品系列。

表 1. 适用产品

类型	产品系列
微型控制器系列	PY32F030、PY32F003

目录

1 RTC 功能简介

PY32 微控制器中的嵌入式实时时钟是一个独立定时器。RTC 模块拥有一组连续的计数器，在相应的软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

2 RTC 应用例程

2.1 RTC 寄存器配置过程

- 查询 RTOFF 位，直到该位变高
- 置位 CNF 位，进入配置模式
- 写一个或者多个 RTC 寄存器
- 清零 CNF 位，退出配置模式
- 查询 RTOFF 位，等待该位变高(检查写操作的结束)

注：仅当 CNF 位被清零，写操作才执行，至少经过 3 个 RTC_CLK 周期才能完成写操作。

2.2 RTC 日历与闹钟

注：1.使用 RTC 模块时，系统时钟不能选择 LSI 或 LSE 时钟源，请用户注意。
2.RTC_CRL 寄存器的 RSF 位默认每间隔 1s 置起。

- 配置 RTC 日历与闹钟的步骤：

步骤	操作
1	初始化底层硬件：RTC 时钟，配置秒中断，闹钟中断等
2	初始化 RTC：配置 RTC 时间基准
3	设置 RTC 初始时间，设置闹钟时间
4	获取当前时间值，设置闹钟事件。

- 日历闹钟配置代码介绍

1. 初始化底层硬件，打开 RTC 代码例程，在 py32f030_hal_msp.c 中，HAL_RTC_MspInit 函数配置了 RTC 时钟，中断等内容。若用户使用 LSE 为 RTC 时钟源则定义 RTC_CLOCK_SOURCE_LSE 即可。

```
#ifdef RTC_CLOCK_SOURCE_LSE
  RCC_OscInitStruct.OscillatorType=
  RCC_OSCILLATORTYPE_LSI|RCC_OSCILLATORTYPE_LSE;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_OFF;
  if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
//    Error_Handler();
  }

  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC;
  PeriphClkInitStruct.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
  if(HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
//    Error_Handler();
  }
```

若用户需要使用 LSI 作为 RTC 的时钟源则定义 RTC_CLOCK_SOURCE_LSI 即可。

```
#elif defined (RTC_CLOCK_SOURCE_LSI)
  RCC_OscInitStruct.OscillatorType=RCC_OSCILLATORTYPE_LSI|
RCC_OSCILLATORTYPE_LSE;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.LSEState = RCC_LSE_OFF;
  if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
```

```
  {
//    Error_Handler();
  }

  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC;
  PeriphClkInitStruct.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;
  if(HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
//    Error_Handler();
  }
```

然后初始化 RTC 总线时钟，配置 RTC 中断优先级，使能 RTC 中断。

```
/*-2- Enable RTC peripheral Clocks */
__HAL_RCC_RTCAPB_CLK_ENABLE();
/* Enable RTC Clock */
__HAL_RCC_RTC_ENABLE();

/*-4- Configure the NVIC for RTC Alarm */
HAL_NVIC_SetPriority(RTC_IRQn, 0, 0);
NVIC_EnableIRQ(RTC_IRQn);

__HAL_RTC_OVERFLOW_ENABLE_IT(hrtc, RTC_IT_OW);
__HAL_RTC_ALARM_ENABLE_IT(hrtc, RTC_IT_ALRA);
__HAL_RTC_SECOND_ENABLE_IT(hrtc, RTC_IT_SEC);
```

2. 在 mian.c 文件中的 main 函数中，我们开始初始化 RTC，配置 RTC 时间基准。

```
RtcHandle.Instance = RTC;
RtcHandle.Init.AsynchPrediv =RTC_AUTO_1_SECOND;
if (HAL_RTC_Init(&RtcHandle) != HAL_OK)
{
    /* Initialization Error */
}
```

3. 打开 main.c 文件，在 RTC_AlarmConfig 函数中我们可以设置当前日历时间。

```
/*-1- Configure the Date */
/* Set Date: Tuesday May 21th 2021 */
sdatestructure.Year = 0x21;
sdatestructure.Month = 0x05;
sdatestructure.Date = 0x21;
sdatestructure.WeekDay = RTC_WEEKDAY_TUESDAY;

if(HAL_RTC_SetDate(&RtcHandle,&sdatestructure, RTC_FORMAT_BCD) != HAL_OK)
{
        /* Initialization Error */
        Error_Handler();
}

/*-2- Configure the Time */
/* Set Time: 12:13:00 */
stimestructure.Hours = 0x12;
stimestructure.Minutes = 0x13;
stimestructure.Seconds = 0x00;

if(HAL_RTC_SetTime(&RtcHandle, &stimestructure, RTC_FORMAT_BCD) != HAL_OK)
{
        /* Initialization Error */
        Error_Handler();
}
```

如果需要闹钟功能也可以在 RTC_AlarmConfig 函数中配置闹钟时间。

```
/*-3- Configure the RTC Alarm peripheral */
/* Set Alarm to 12:13:35
     RTC Alarm Generation: Alarm on Hours, Minutes and Seconds */
salarmstructure.Alarm = RTC_ALARM_A;
salarmstructure.AlarmTime.Hours = 0x12;
salarmstructure.AlarmTime.Minutes = 0x13;
salarmstructure.AlarmTime.Seconds = 0x35;

if(HAL_RTC_SetAlarm_IT(&RtcHandle, &salarmstructure, RTC_FORMAT_BCD) != HAL_OK)
{
      /* Initialization Error */
}
```

4. 在 py32f030_hal_rtc.c 中我们提供了获取当前日期和获取当前时间两个函数，用户可以调用这两个函数进行读取时间日期。下面例程中通过调用这两个函数来打印当前时间。此函数在 main.c 中。

```
static void RTC_TimeShow(void)
{
      RTC_DateTypeDef sdatestructureget;
      RTC_TimeTypeDef stimestructureget;

      /* Get the RTC current Time */
      HAL_RTC_GetTime(&RtcHandle, &stimestructureget, RTC_FORMAT_BIN);
      /* Get the RTC current Date */
      HAL_RTC_GetDate(&RtcHandle, &sdatestructureget, RTC_FORMAT_BIN);
      /* Display time Format : hh:mm:ss */
      printf("%02d:%02d:%02d\r\n", stimestructureget.Hours, stimestructureget.Minutes,
stimestructureget.Seconds);
}
```

5. 闹钟事件动作大家可以通过定义 HAL_RTC_AlarmAEventCallback 函数来设置。在例程中我们是一旦触发闹钟，打印 “RTC_IT_ALRA ”字符来提示当前正在发生闹钟事件。

```
void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc)
{
      printf("RTC_IT_ALRA\r\n");
}
```

3 版本历史

版本	日期	更新记录
V0.1	2021.10.23	初版
V1.0	2022.06.20	修改了应用例程内容

Puya Semiconductor Co., Ltd.