AN2003
应用笔记

UM32x13x RTC 应用手册

版本：V1.0

Unicmicro



广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

本文档的所有部分，其著作产权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

 Copyright © 2023 广芯微电子（广州）股份有限公司

目录

1 摘要 1
2 RTC 简介 1
3 RTC 软件配置 1
3.1 RTC 结构体定义 1
3.1.1 IP 实例结构体 1
3.1.2 功能初始化结构体 2
3.1.3 时间相关结构体 2
3.2 软件配置 2
3.2.1 RTC 时间设置 2
3.2.2 IO 口输出 RTC 信号软件配置 4
3.2.3 时间戳软件设置 4
4 版本维护 6

Unicmicro

 Copyright © 2023 广芯微电子（广州）股份有限公司

1 摘要

本篇应用笔记主要介绍 UM32x13x RTC 配置方法。

本篇应用笔记主要包括：

- RTC 简介
- RTC 软件配置

注：具体功能及寄存器的操作等相关事项请以用户手册为准。

2 RTC 简介

实时时钟（RTC）是一个独立的定时器/计数器，可提供基本的闹钟中断或者长时间的计数服务。闹钟中断通过可配置的实时时钟计数周期实现。

可配合 UM32x13x 用户手册和“Projects\UM32x13x_EVAL\Examples\RTC”中的代码用例进行学习。

3 RTC 软件配置

3.1 RTC 结构体定义

3.1.1 IP 实例结构体

```
typedef struct __RTC_HandleTypeDef
{
  RTC_TypeDef        *Instance;          /**<  registers base address       */
  RTC_InitTypeDef     Init;              /**<  rtc parameters       */
  uint16_t            Stamp_type;        /**<  Stamp_type      */
  void (*RTCISR)(void);                  /**<IRQ handler pointter*/
} RTC_HandleTypeDef;
```

其中 Instance 为指向 RTC_TypeDef 结构体的指针，Init 为功能初始化结构体，Stamp_type 为时间戳类型，(*RTCISR)(void)为中断函数指针。

nicmicro

Copyright © 2023 广芯微电子（广州）股份有限公司

3.1.2 功能初始化结构体

```
typedef struct
{
  s_calendar_obj    set_calendar;      /**<  sec calendar      */
  s_calendar_obj    set_calendar_alarm;    /**<  sec alarm      */
  uint8_t           fout_sel;          /**<  sec output type      */
  uint16_t          adjust;            /**<  sec adjust      */
  uint8_t           adsign;            /**<  sec adsign      */
  uint8_t           prisen;            /**<  sec prisen      */
} RTC_InitTypeDef;
```

其中 set_calendar 为待设置的时间开始时刻，set_calendar_alarm 为闹铃时间设置时刻，fout_sel 为输出类型设置，adjust 为 LTBC 计时调校值，adsign 为 LTBC 计时调校符号，prisen 为精确秒时标虚拟调校使能标志。

3.1.3 时间相关结构体

包括对应功能（实时值或闹铃）年/月/日/星期/时/分/秒的设置。

```
typedef struct
{
  uint8_t  year;    /**<  year value      */
  uint8_t  month;    /**<  month value      */
  uint8_t  day;    /**<  day value      */
  uint8_t  week;     /**<  week value      */
  uint8_t hour;    /**<  hour value      */
  uint8_t min;    /**<  min value      */
  uint8_t sec;     /**<  sec value      */
}s_calendar_obj;
```

3.2 软件配置

3.2.1 RTC 时间设置

1. 对“Init.set_calendar.year/ month/ day/ week/ hour/ min/ sec”进行赋值，例程中调用“HAL_rtc_calendar_data_SetConfig（）”函数进行赋值。

```
HAL_StatusTypeDef HAL_rtc_calendar_data_SetConfig(RTC_HandleTypeDef *hrtc )
{
    /**< Check Null pointer */
  if((hrtc == NULL)||(hrtc->Instance == NULL))
```


Copyright © 2023 广芯微电子（广州）股份有限公司

```
        {
         return HAL_ERROR;
        }
         /**< Config internal  YEAR/MONTH/DDATE/WEEK/HOUR/MIN/SEC data*/
        WRITE_REG(hrtc->Init.set_calendar.year, set_calendar_year);
        WRITE_REG(hrtc->Init.set_calendar.month, set_calendar_month);
         WRITE_REG(hrtc->Init.set_calendar.day, set_calendar_day);
         WRITE_REG(hrtc->Init.set_calendar.week, set_calendar_week);
         WRITE_REG(hrtc->Init.set_calendar.hour, set_calendar_hour);
         WRITE_REG(hrtc->Init.set_calendar.min, set_calendar_min);
         WRITE_REG(hrtc->Init.set_calendar.sec, set_calendar_sec);
         return HAL_OK;
}
```

其中“set_calendar_year/ set_calendar_month…”等为预定义的时间宏，如果实际应用中需要从其他接口传入时间设置值，则在接收后时间后使用“WRITE_REG(hrtc->Init.set_calendar.year, xxx);…”等对“Init.set_calendar”记录时间结构体的值进行更新。

2. 设置时间寄存器，将记录时间的结构体的值一一赋给寄存器，例程调用“HAL_RTC_calendar_SetConfig(RTC_HandleTypeDef *hrtc)”函数对时间寄存器赋值。

```
HAL_StatusTypeDef HAL_RTC_calendar_SetConfig(RTC_HandleTypeDef *hrtc )
{
    /**< Check Null pointer */
   if((hrtc == NULL)||(hrtc->Instance == NULL))
   {
    return HAL_ERROR;
   }
    /**< Config YEAR/MONTH/DDATE/WEEK/HOUR/MIN/SEC register */
   WRITE_REG(hrtc->Instance->WE, 0xACACACAC);
   WRITE_REG(hrtc->Instance->BCDYEAR, hrtc->Init.set_calendar.year);
   WRITE_REG(hrtc->Instance->BCDMONTH, hrtc->Init.set_calendar.month);
   WRITE_REG(hrtc->Instance->BCDDATE, hrtc->Init.set_calendar.day);
   WRITE_REG(hrtc->Instance->BCDWEEK, hrtc->Init.set_calendar.week);
   WRITE_REG(hrtc->Instance->BCDHOUR, hrtc->Init.set_calendar.hour);
   WRITE_REG(hrtc->Instance->BCDMIN, hrtc->Init.set_calendar.min);
   WRITE_REG(hrtc->Instance->BCDSEC, hrtc->Init.set_calendar.sec);
   WRITE_REG(hrtc->Instance->WE, 0xFFFFFFFF);
  return HAL_OK;
}
```


Copyright © 2023 广芯微电子（广州）股份有限公司

3. 设置闹铃时间寄存器。同时间设置一样将记录时间的结构体的值一一赋给寄存器，例程调用“HAL_rtc_calendar_alarm_data_SetConfig(RTC_HandleTypeDef *hrtc)”记录闹铃时间，及HAL_rtc_alarm_SetConfig(RTC_HandleTypeDef *hrtc)对闹铃时间寄存器赋值。

3.2.2 IO 口输出 RTC 信号软件配置

1. IO 口设置为 RTC 输出。

```
__HAL_IO_WRITE_SWITCH(SCU_IO_WRITE_EN);
GPIO_InitStructure.Pin = GPIO_PIN_0;
GPIO_InitStructure.Mult_Type = PA0_MULT_RTC_FOUT;//IO 口设置为 RTC 输出
HAL_GPIO_Init(GPIOA,&GPIO_InitStructure);
__HAL_IO_WRITE_SWITCH(SCU_IO_WRITE_DISEN);
```

2. 设置 Init.fout_sel 输出类型，类型可选以下之一。

```
#define  OUTSEL_1S_SQUARE_WAVE                    (0U)
#define  OUTSEL_Second_timescale_80MS             (1U)
#define  OUTSEL_Second_counter_carry_signal_1S    (2U)
#define  OUTSEL_Sub_counter_carry_signal_1S       (3U)
#define  OUTSEL_Hour_counter_carry_signal_1S      (4U)
#define  OUTSEL_Day_counter_carry_signal_1S       (5U)
#define  OUTSEL_Alarm_clock_matching_signal       (6U)
#define  OUTSEL_128S_SQUARE_WAVE                  (7U)
#define  OUTSEL_Reverse_output_timescale_80MS     (8U)
#define  OUTSEL_Reverse_output_second_counter_carry_signal  (9U)
#define  OUTSEL_Reverse_output_min_counter_carry_signal     (10U)
#define  OUTSEL_Reverse_output_hour_counter_carry_signal    (11U)
#define  OUTSEL_Reverse_output_day_counter_carry_signal     (12U)
#define  OUTSEL_Reverse_Alarm_clock_matching_signal         (13U)
#define  OUTSEL_Reverse_1S_SQUARE_WAVE            (14U)
#define  OUTSEL_rtc_internal_second_time_scale    (15U)
```

3. 调用 HAL_rtc_fout(RTC_HandleTypeDef *hrtc)函数进行输出配置。

3.2.3 时间戳软件设置

1. IO 口设置为时间戳（输入）模式。

```
__HAL_IO_WRITE_SWITCH(SCU_IO_WRITE_EN);
  GPIO_InitStructure.Pin = GPIO_PIN_4;   // PA4 RTC_STAMP0
  GPIO_InitStructure.Mult_Type = PA4_MULT_RTC_STAMP0;
  GPIO_InitStructure.direction = GPIO_INPUT;
```


Copyright © 2023 广芯微电子（广州）股份有限公司

```
HAL_GPIO_Init(GPIOA,&GPIO_InitStructure);
GPIO_InitStructure.Pin = GPIO_PIN_3;     // PD3 RTC_STAMP1
GPIO_InitStructure.Mult_Type = PD3_MULT_RTC_STAMP1;
GPIO_InitStructure.direction = GPIO_INPUT;
HAL_GPIO_Init(GPIOD,&GPIO_InitStructure);
    __HAL_IO_WRITE_SWITCH(SCU_IO_WRITE_DISEN);
```

2. 参考 3.2.1 RTC 时间设置章节设置时间。
3. 设置时间戳中断允许功能。

```
HAL_RTC_IT(&hRTC,RTC_IRQ_ENABLE,STPR0IE);//设置 STAMP0 上升沿触发时间戳
HAL_RTC_IT(&hRTC,RTC_IRQ_ENABLE,STPR1IE);//设置 STAMP1 上升沿触发时间戳
```

4. 调用一下函数，允许时间戳功能。

```
HAL_rtc_stamp0_init(RTC_HandleTypeDef *hrtc)
HAL_rtc_stamp1_init(RTC_HandleTypeDef *hrtc)
```

5. 设置 Deepsleep。
6. 当对应 IO 口出现引发时间戳的边沿出现时会进入 HAL_RTC_IRQHandler 中断服务函数。

Unicmicro

Copyright © 2023 广芯微电子（广州）股份有限公司

4 版本维护

版本	日期	描述
V1.0	2023.09.11	初始版

Unicmicro

Copyright © 2023 广芯微电子（广州）股份有限公司