NV32F100x RTC 编程示例

第一章 各部分模块简介

库函数列表

void RTC_Init(RTC_ConfigType *pConfig);
RTC 模块的初始化配置（时钟、分频、模值、使能中断等）
void RTC_DeInit(void);
复位 RTC 模块
void RTC_SetClock(uint16_t u16Clock_Number, uint16_t u16Presalcer);
RTC 模块时钟选择及分频系数设置
void RTC_ClrFlags(void);
清除 RTC 模块中断标志位 RTIF
uint8_t RTC_GetFlags(void);
获取 RTC 模块中断标志位 RTIF
void RTC_DisableInt(void);
禁止 RTC 中断，使用轮询
void RTC_EnableInt(void);
使能 RTC 中断
void RTC_Isr(void);
中断服务程序
void RTC_SetCallback(RTC_CallbackType pfnCallback);
设置 RTC 中断回调函数

1.1 模块初始化

RTC 状态和控制寄存器(RTC_SC)：

字段	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-14 RTCLKS	实时时钟源选择 该读/写字段选择 RTC 预分频器的时钟源输入。更改时钟源会将预分频器和 RTCCNT 计数器清零。复位会将 RTCLKS 清除为 00。 00 外部时钟源 01 实时时钟源为 1kHz（LPOCLK）。 10 内部参考时钟（ICSIRCLK）。 11 总线时钟。
13-11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10-8 RTCPS	实时时钟预分频器选择 该读/写字段为时钟源选择基于二进制或基于十进制的分频值。更改预分频器值会将预分频

	器和 RTCCNT 计数器清零。复位会将 RTCPS 清除为 000。 000 关闭。 001 如果 RTCLKS=x0,它为 1;如果 RTCLKS=x1,它为 128。 010 如果 RTCLKS=x0,它为 2;如果 RTCLKS=x1,它为 256。 011 如果 RTCLKS=x0,它为 4;如果 RTCLKS=x1,它为 512。 100 如果 RTCLKS=x0,它为 8;如果 RTCLKS=x1,它为 1024。 101 如果 RTCLKS=x0,它为 16;如果 RTCLKS=x1,它为 2048。 110 如果 RTCLKS=x0,它为 32;如果 RTCLKS=x1,它为 100。 111 如果 RTCLKS=x0,它为 64;如果 RTCLKS=x1,它为 1000。
7 RTIF	实时中断标志 该状态位指示 RTC 计数器寄存器已达到 RTC 模数寄存器中的值。写入逻辑 0 无效。写入逻辑 1 会将该位清零并清除实时中断请求。复位会将 RTIF 清除为 0。 0 RTC 计数器未达到 RTC 模数寄存器中的值。 1 RTC 计数器已达到 RTC 模数寄存器中的值。
6 RTIE	实时中断使能 该读/写位使能实时中断。如果 RTIE 置位，那么在 RTIF 置位时会生成中断。复位会将 RTIE 清除为 0。 0 实时中断请求禁用。使用软件轮询。 1 实时中断请求使能。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 RTCO	实时计数器输出 该读/写位使能实时计数器把切换输出到引脚上。如果该位置位，那么在 RTC 计数器溢出时，将切换 RTCO 至引脚。 0 实时计数器输出禁用。 1 实时计数器输出使能。
3-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

预分频器周期表:

RTCPS	32768 Hz OSC 时钟源预分频器周期 (RTCLKS=00)	LPO 时钟(1kHz)源预分频器周期 (RTCLKS=01)	内部参考时钟(32.768kHz)源预分频器周期 (RTCLKS=10)	总线时钟(8MHz)源预分频器周期 (RTCLKS=11)
000	关闭	关闭	关闭	关闭
001	30.5176μs	128ms	30.5176μs	16μs
010	61.0351μs	256ms	61.0351μs	32μs
011	122.0703μs	512ms	122.0703μs	64μs
100	244.1406μs	1024ms	244.1406μs	128μs
101	488.28125μs	2048ms	488.28125s	256μs
110	976.5625μs	100ms	976.5625μs	12.5μs
111	1.9531ms	1s	1.9531ms	125μs

函数名	RTC_Init
函数原形	RTC_Init(RTC_ConfigType *pConfig)
功能描述	以配置结构体 pConfig 来初始化 RTC
输入参数	配置结构体 RTC_ConfigType
输出参数	无
返回值	无
先决条件	无
函数使用实例	先设置配置结构体，RTC_Init(&RTC_Config)

```
/*****************************************************************************//*!
*
* @初始化 RTC 模块.
*
* @输入        pConfig         指向配置结构体
*
* @无返回
*
*****************************************************************************/
void RTC_Init(RTC_ConfigType *pConfig)
{
    uint16_t      u16Clocksource, u16Prescler;
    uint16_t      u16ModVal;
    u16Clocksource =0;
    u16Prescler      =0;
    u16ModVal      =0;

    SIM->SCGC |= SIM_SCGC_RTC_MASK; //选通 RTC 模块时钟

    u16ModVal    = pConfig->u16ModuloValue;//设置 RTC 模数
    RTC_SetModulo(u16ModVal);

    if (pConfig->bRTCOut)
    {
        RTC->SC= RTC_SC_RTCO_MASK; //实时计数器输出使能
    }
    if (pConfig->bInterruptEn)
    {
         NVIC_EnableIRQ(RTC_IRQn);//使能 RTC 的 IRQ 中断
         RTC_EnableInt();
    }
    else
    {
        NVIC_DisableIRQ(RTC_IRQn);//禁止 RTC 的 IRQ 中断
    }
```

```
    if (pConfig->bFlag)
    {
        RTC_ClrFlags();//实时中断标志置位，RTC 计数器已达到模数寄存器中的值
    }
    u16Clocksource = pConfig->bClockSource;
    u16Prescler    = pConfig->bClockPresaler;

    RTC_SetClock(u16Clocksource,u16Prescler );//实时时钟源及预分频器选择
}
```

1.2 设置函数回调

函数名	RTC_SetCallback
函数原形	RTC_SetCallback(RTC_CallbackType pfnCallback)
功能描述	设置 RTC 中断回调函数的入口
输入参数	中断回调函数地址
输出参数	无
返回值	无
先决条件	无
函数使用实例	RTC_SetCallback(RTC_Task)

```
/*****************************************************************************//*!
*
* @设置 RTC 模块的函数回调
*
* @输入          pfnCallback      指向回调函数地址
*
* @无返回
*
*****************************************************************************/
void RTC_SetCallback(RTC_CallbackType pfnCallback)
{
  RTC_Callback[0] = pfnCallback;
}
```

1.3 出厂到默认状态

函数名	RTC_DeInit
函数原形	RTC_DeInit(void)
功能描述	复位 RTC 模块到初始化之前
输入参数	无
输出参数	无
返回值	无
先决条件	无
函数使用实例	RTC_DeInit()

```
/*****************************************************************************//*!
*
* @复位 RTC 模块到默认状态
*
* @无返回
*
*****************************************************************************/
void RTC_DeInit(void)
{
    NVIC_DisableIRQ(RTC_IRQn);//禁止 RTC 中断
    RTC->MOD = 0;//装载模数值为 0

    while(RTC->MOD);//确保模数器中值为 0
    if(RTC_GetFlags())
    {
        RTC_ClrFlags();//写入 1 清零该位并清楚实时中断请求
    }
    /* 禁用 RTC */
    RTC->SC = 0;
    while(RTC->SC);
    SIM->SCGC &= ~SIM_SCGC_RTC_MASK;//禁用 RTC 的时钟
}
```

1.4 中断服务程序

函数名	RTC_Isr
函数原形	RTC_Isr(void)
功能描述	RTC 中断服务函数
输入参数	无
输出参数	无
返回值	无
先决条件	无
函数使用实例	RTC_Isr()

```
/*******************************************************************************//*!
*
* @为 RTC 模块提供中断服务程序 ISR
*
* @无返回
*
*******************************************************************************/
void RTC_Isr(void)
{
    RTC_ClrFlags();
    if (RTC_Callback[0])
    {
        RTC_Callback[0]();
    }
}
```

第二章 样例程序

2.1 RTC 计数中断控制 LED 闪烁

```
/*****************************************************************************
*
* @使用 RTC 模块控制 LED 闪烁.
*
*****************************************************************************/
#include "common.h"
#include "rtc.h"
#include "uart.h"
#include "sysinit.h"
int main (void);
void RTC_Task(void);

uint8_t i=0;
int main (void)
{
    uint8_t u8Ch;
    uint16_t u16ModuloValue;
    RTC_ConfigType   sRTCConfig;
    RTC_ConfigType   *pRTC_Config=&sRTCConfig;


    /* 系统初始化 */
    sysinit();
    printf("\nRunning the RTC_demo project.\n");
    LED0_Init();   //初始化 LED

    /* 配置 RTC 的中断频率为 1HZ */
    u16ModuloValue = 0x09;//模值为 10
    pRTC_Config->u16ModuloValue = u16ModuloValue; //装载值到模数寄存器中
    pRTC_Config->bInterruptEn     = RTC_INTERRUPT_ENABLE;       // 使能中断
    pRTC_Config->bClockSource     = RTC_CLKSRC_1KHZ;            // 选取时钟源为 1KHZ
    pRTC_Config->bClockPresaler = RTC_CLK_PRESCALER_100;      //分频数为 100
    RTC_SetCallback(RTC_Task);//设置 RTC 回调函数
    RTC_Init(pRTC_Config);//初始化 RTC 模块
    while(1)
    {
      u8Ch = UART_GetChar(TERM_PORT);
      UART_PutChar(TERM_PORT, u8Ch);
```

```
    }
}

/*****************************************************************************//*!
*
* @RTC 任务子函数，闪烁 LED
*
* @无返回
*
*****************************************************************************/
void RTC_Task(void)
{
    i=i+1;

  if(i%3==2)
  {
      LED0_On();
      LED1_Off();
      LED2_Off();
   }
   else if(i%3==1)
   {
      LED0_Off();
      LED1_On();
      LED2_Off();
    }
   else
    {
      LED0_Off();
      LED1_Off();
      LED2_On();
    }
}
```

本例程通过操作 RTC 模块，完成一个 LED 灯固定频率的闪烁，而且较为精准；为开发者尽快了解 NV32F100 的 RTC 模块提供了一个框架。

该样例工程在 nv32_pdk\build\keil\NV32\RTC_demo 下