Университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа №4
по дисциплине
«Встроенные системы»

Работу выполнил:
студент группы P3410
Глушков Дима

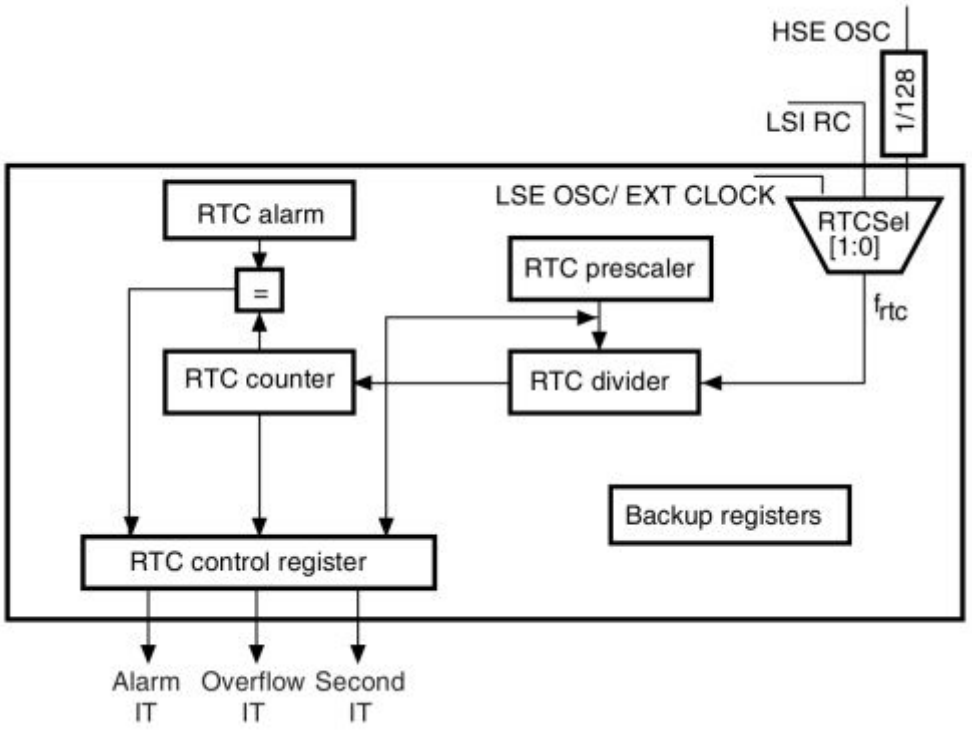
Санкт-Петербург
2020

Введение

Данная лабораторная предназначена для получения практическихнавыков работы с RTC, а так же использования UARTдля передачи и приема полученного времени и даты.

Теория о RTC

RTC(Real Time Clock) -специальнаямикросхема, которую тактирует внутренний низкоскоростной RC-резонатор (LSI)или внешний низкоскоростной кварцевый резонатор (LSE);



Для определения первого включения устройства используются резервные регистры (англ. backup registers, которые так же, как и RTC, используют резервное питание и сохраняют свое содержимое после отключения основного. В STM32f10x доступно целых 16 бит данных (использовать их вы можете и для собственных нужд, не только в связке с часами).

У STM32 RTC обладает следующими возможностями:

- Автоматическое пробуждение во всех режимах энергосбережения
- Независимый BCD таймер счетчик. Отсчет времени и календарь реализованы аппаратно, с возможностью настройки сигнализирующих прерываний.

- Программный флаг пробуждения с возможностью вызова прерывания.
- Два 32-х разрядных регистра, в которых хранятся секунды, минуты, часы(в 12 часовом или 24 часовом формате), день недели, день месяца, месяц и год. Секунды хранятся в двоичном формате, все остальное в двоично-десятичном.
- Компенсация длинны месяца(а также високосного года) выполняется автоматически. Также возможна компенсация летнего времени.
- Два 32-х разрядных регистра программируемых сигнализирующих прерываний.
- Наличие калибровки для компенсации отклонения кварцевого резонатора.
- После сброса область RTC защищена от случайной записи.
- До тех пор пока напряжение питания RTC остается в рабочем диапазоне, он будет работать в не зависимости от режима работы(Run mode, low-power mode or under reset).

Чтобы RTC начали работать, их нужно подключить к источнику тактирования, и это может быть:

- внутренний низкоскоростной RC-резонатор (LSI);
- внешний низкоскоростной кварцевый резонатор (LSE);
- внешний генератор с предделителем (1/128).

Для работы с RTC нужно знать особенности программной реализации:
- Для работы с датой и временем стандартная библиотека предоставляет структуры данных
    - `RTC_TimeTypeDef` - для глобальной переменной со значением времени.
    - `RTC_DateTypeDef` - для глобальной переменной со значением даты.
- Для получения текущего времени и даты библиотека HAL предоставляет функции
    - `HAL_RTC_GetTime (RTC_HandleTypeDef *hrtc, RTC_TimeTypeDef *sTime, uint32_t Format)` - функция получения времени.

- `HAL_RTC_GetDate (RTC_HandleTypeDef *hrtc, RTC_DateTypeDef *sDate, uint32_t Format)` - функция получения даты.

- Для установки времени и датытакже предоставляется 2 функции основной их особенность является параметр формата
  - `RTC_FORMAT_BIN` - двоичный
  - `RTC_FORMAT_BCD` - двоично-десятичный

Пример вывода в двух форматах

Вывод при двоичном формате записи:

```
0.115:    1: Hour = 14 Minutes = 0 Seconds = 23
0.138:    1: WeekDay = 7 Date = 19 Month = 12 Year = 20
```

Вывод при двоично-десятичном формате записи:

```
0.114:    2: Hour = 20 Minutes = 0 Seconds = 35
0.137:    2: WeekDay = 7 Date = 25 Month = 18 Year = 32
```

Представленные выше результаты объясняются следующим образом: Например, рассмотрим вывод значения Year. В первом случае десятичное число 20 будет записано как 0x14, а при чтении обратится обратно в 20 (16^1 * 1 + 16^0 * 4). Во втором же случае десятичное число 20 будет записано как 0x20, а при чтении обратится в 32 (16^ * 2 + 16^0 * 0). Аналогично объясняются изменения других значений.

Установка времени (по аналогии с установкой даты)

Для управления датой и времени используется файл rtc.c и описанные выше структуры `RTC_TimeTypeDef` и `RTC_DateTypeDef`. Для установки времени изменим переменные Hours, Minutes, Seconds с помощью следующих функций:

```
HAL_RTC_SetTime (RTC_HandleTypeDef *hrtc, RTC_TimeTypeDef
*sTime, uint32_t Format)
HAL_RTC_SetDate (RTC_HandleTypeDef *hrtc, RTC_DateTypeDef
*sDate, uint32_t Format)
```

Передача даты и времени по UART

```
692.228:  UART time: 14:26:4
1349.597: UART date: 19-12-20
```

Вывод

В результате выполнения данной лабораторной работы познакомился с основными способами установки и получения даты и времени(RTC), в том числе и структурами используемых для этих действий. Так же научился передавать полученные значения по UART.

Приложение А. Исходный код программы

Вывод времени при использовании разных форматов хранения

```
RTC_TimeTypeDef time;
RTC_DateTypeDef date;
char str_for_date[100];
char str_for_time[100];
SDK_TRACE_Timestamp(PRINT, 0);
HAL_RTC_GetTime(&hrtc, &time, RTC_FORMAT_BIN);
HAL_RTC_GetDate(&hrtc, &date, RTC_FORMAT_BIN);
SDK_TRACE_Timestamp(PRINT, 1);
sprintf(str_for_time, "1: Hour = %d Minutes = %d Seconds =
%d", time.Hours, time.Minutes, time.Seconds);
sprintf(str_for_date, "1: WeekDay = %d Date = %d Month = %d
Year = %d", date.WeekDay, date.Date, date.Month, date.Year);
SDK_TRACE_Timestamp(PRINT, 0);
SDK_TRACE_Print(str_for_time);
SDK_TRACE_Print(str_for_date);
SDK_TRACE_Timestamp(PRINT, 1);
SDK_TRACE_Timestamp(PRINT, 0);
HAL_RTC_GetTime(&hrtc, &time, RTC_FORMAT_BCD);
HAL_RTC_GetDate(&hrtc, &date, RTC_FORMAT_BCD);
SDK_TRACE_Timestamp(PRINT, 1);
sprintf(str_for_time, "2: Hour = %d Minutes = %d Seconds =
%d", time.Hours, time.Minutes, time.Seconds);
sprintf(str_for_date, "2: WeekDay = %d Date = %d Month = %d
Year = %d", date.WeekDay, date.Date, date.Month, date.Year);
SDK_TRACE_Timestamp(PRINT, 0);
SDK_TRACE_Print(str_for_time);
SDK_TRACE_Print(str_for_date);
SDK_TRACE_Timestamp(PRINT, 1);
```

Установка даты и времени

```
RTC_TimeTypeDef m_time = {0};
RTC_DateTypeDef m_date= {0};
/* USER CODE END 0 */
RTC_HandleTypeDefhrtc;
/* RTC initfunction */
voidMX_RTC_Init(void)
{
  /** Initialize RTC Only*/
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType  = RTC_OUTPUT_TYPE_OPENDRAIN;
  if(HAL_RTC_Init(&hrtc) != HAL_OK)
```

```
  {
    Error_Handler();
  }

  m_date.WeekDay = RTC_WEEKDAY_THURSDAY;
  m_date.Month = RTC_MONTH_DECEMBER;
  m_date.Date = 0x19;
  m_date.Year = 0x20;
  if(HAL_RTC_SetDate(&hrtc, &m_date, RTC_FORMAT_BCD) !=
HAL_OK)
  {
    Error_Handler();
  }
  m_time.Hours= 14;
  m_time.Minutes= 26;
  m_time.Seconds= 04;
  if(HAL_RTC_SetTime(&hrtc, &m_time, RTC_FORMAT_BIN) !=
HAL_OK)
  {
    Error_Handler();
  }

}

RTC_TimeTypeDef time;
RTC_DateTypeDef date;
char str_for_date[100];
char str_for_time[100];
SDK_TRACE_Timestamp(PRINT, 0);
HAL_RTC_GetTime(&hrtc, &time, RTC_FORMAT_BIN);
HAL_RTC_GetDate(&hrtc, &date, RTC_FORMAT_BIN);
SDK_TRACE_Timestamp(PRINT, 1);

sprintf(str_for_time, "1: Hour = %d Minutes = %d Seconds =
%d", time.Hours, time.Minutes, time.Seconds);
sprintf(str_for_date, "1: WeekDay = %d Date = %d Month = %d
Year = %d", date.WeekDay, date.Date, date.Month, date.Year);

SDK_TRACE_Timestamp(PRINT, 0);
SDK_TRACE_Print(str_for_time);
SDK_TRACE_Print(str_for_date);
SDK_TRACE_Timestamp(PRINT, 1);
```

Листинг получения даты и времени и отправки данных по UART

```
RTC_TimeTypeDef time;
RTC_DateTypeDef date;
uint8_t trans_str_date[64];
uint8_t trans_str_time[64];
```

```
uint8_t time_receiving_in_USART3[1000];
uint8_t date_receiving_in_USART3[1000];
HAL_RTC_GetTime(&hrtc, &time, RTC_FORMAT_BIN);
snprintf(trans_str_time, 63, "%d:%d:%d\n", time.Hours,
time.Minutes, time.Seconds);
int len = (int)(sizeof trans_str_time / sizeof
trans_str_time[0]);
for(int i=0; i<len-1; i++)
{
  if(HAL_UART_Transmit(&huart2, &trans_str_time[i], 1, 10) !=
HAL_OK)
  {
    Error_Handler();
  }
  while(HAL_UART_Receive(&huart3,
&time_receiving_in_USART3[i], 1, 10) != HAL_OK){}
}

SDK_TRACE_Timestamp(PRINT, 0);
SDK_TRACE_Print("%s%s", "UART time:",
time_receiving_in_USART3);
SDK_TRACE_Timestamp(PRINT, 1);
HAL_RTC_GetDate(&hrtc, &date, RTC_FORMAT_BIN);
snprintf(trans_str_date, 63, "%d-%d-%d\n", date.Date,
date.Month, date.Year);
len = (int)(sizeof trans_str_date / sizeof
trans_str_date[0]);
for(int i=0; i<len-1; i++)
{
  if(HAL_UART_Transmit(&huart2, &trans_str_date[i], 1, 10) !=
HAL_OK)
  {
    Error_Handler();
  }
  while(HAL_UART_Receive(&huart3,
&date_receiving_in_USART3[i], 1, 10) != HAL_OK){}
}
SDK_TRACE_Timestamp(PRINT, 0);
SDK_TRACE_Print("%s%s", "UART date:",
date_receiving_in_USART3);
SDK_TRACE_Timestamp(PRINT, 1);
```