Часы реального времени

Для работы с временем используют специальный таймер — часы реального времени (англ. real time clock). Очень часто это отдельная микросхема с резервным источником питания, однако в stm32 данный таймер уже встроен. Детальное описание блока, а так же пример работы с ним можно найти в документе AN2811.[1]

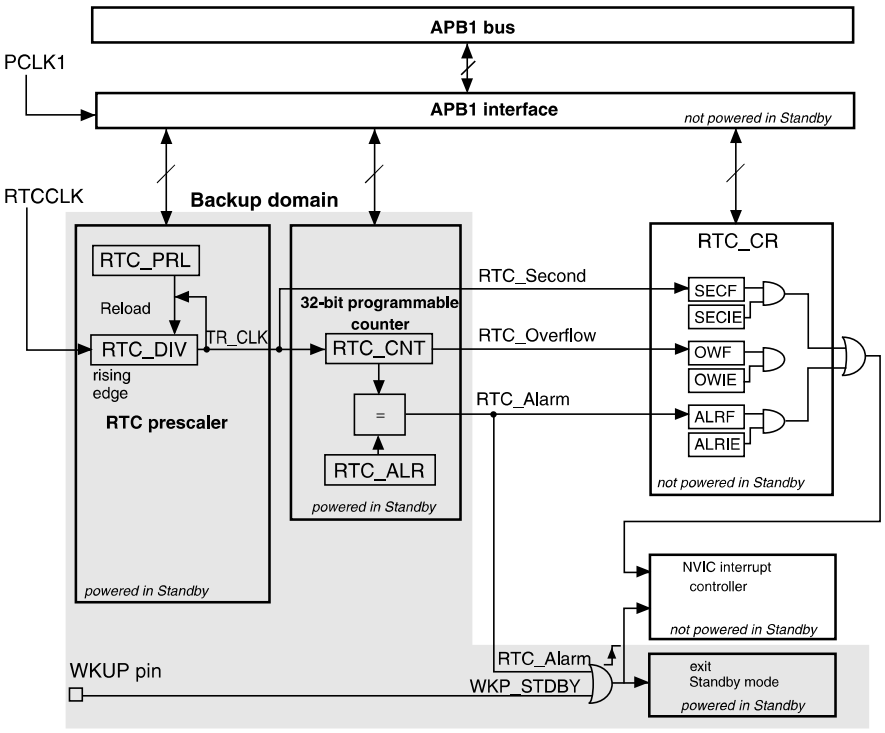
Блок RTC работает от V_{dd}, если достаточное напряжение на нём присутствует. В противном случае блок переключается на ножку V_{bat}, к которой в нашем случае подключена батарейка CR2032. Регистр настроек RTC находится в back-up области, которая так же подключена к резервному источнику питания. По этой причине когда основное питание пропадает, таймер не сбрасывается и продолжает работать.

Тактирование может поступать от одного из трёх источников: внутреннего низкоскоростного RC-резонатора (LSI); внешнего низкоскоростного кварцевого резонатора (LSE); или внешнего высокоскоростного резонатора с делителем (HSE/128).

Внешний низкоскоростной резонатор, с частотой 32768 Гц, предпочтительнее, так как обеспечивает большую точность хода часов. В нашей плате стоит именно такой.

> **Почему именно 32768 Гц?**
>
> В момент, когда цифровая электроника только зарождалась в том виде, в котором мы привыкли видеть её сейчас, не было таких микроконтроллеров, как **stm32f103c8**, но были двоичные счетчики. $32768 = 2^{15}$, легко делится на 2, что в свою очередь делало удобным получать дробные значения секунды. Так и появился кварц с частотой колебаний 32768 Гц.
>
> Кстати говоря, встречаются случаи, когда часы тактируют с частотой 50-60Гц от электросети.

Диаграмма блока приведена ниже.



Структура часов реального времени, изображение из Reference Manual

Блок RTC работает независимо от шины APB1, и использует специальный интерфейс для общения с остальной частью микроконтроллера.

Так как тактовый сигнал у шины APB1 и блока RTC разный, то доступ к регистрам возможен только по возрастающему фронту тактового сигнала часов реального времени. По-этому при работе с регистрами нужно "синхронизировать" действия с блоком RTC.

Перед любой записью в регистры нужно войти в режим настройки, записав единицу в бит `CNF` нижнего регистра настроек. При этом любая запись в регистры RTC возможна только, если предыдущая операция была завершена, о чём сигнализирует флаг `RTOFF` (там должна быть записана `1`).

Алгоритм настройки часов в общем виде:

1. опрос бита `RTOFF` до тех пор пока там не появится `1`;
2. установка бита `CNF`;
3. Запись значений в регистры;
4. сброс бита `CNF`;
5. опрос бита `RTOFF`, чтобы удостовериться, что данные успешно записаны.

> Запись в регистры производится только после сброса флага `CNF`, и занимает минимум 3 такта RTC.

Тактовый сигнал пропускается через предделитель, для того чтобы задать временное разрешение таймера. Например чтобы получить разрешение в 1 секунду с LSE, в регистр `PRLL` нужно записать `32768 - 1`.

Часы могут генерировать три прерывания: по истечению секунды (`SECIE`); по достижению определённого времени (будильник, `ALRIE`); и по переполнению счётчика (`OWIE`). Запретить (`0`) или разрешить (`1`) их можно через верхний регистр настроек `CRH`.

Счётчик разбит на два регистра по 16 бит `RTC_CNTH` и `RTC_CNTL`.

```
#define GET_CNT()          ((RTC->CNTH << 16) + RTC->CNTL)
#define SET_CNT(x)         { RTC->CNTH = (uint16_t)(x >> 16);        \
                             RTC->CNTL = (uint16_t)(0x0000FFFF & x); }
```

Будильник можно настроить через регистры `RTC_ALRH` и `RTC_ALRL`.

Согласно документу **AN2821**, алгоритм работы с часами будет следующим:

- включаем тактирование RTC и резервных регистров;
- разрешаем доступ к резервным регистрам;
- сбрасываем резервные регистры;
- переключаемся на LSE;
- включаем RTC;
- синхронизируемся с RTC (ждем, пока выполнятся внутренние операции);
- устанавливаем предделитель (задает разрешающую способность);
- задаем время;
- записываем в резервные регистры какую-нибудь константу (сверяя которую, мы будем знать, проводилась ли настройка RTC до этого).

```
// rtc.h
#define BACKUP_REGISTER_VALUE 0xA5A5

void rtc_init(void);
void setTime(uint32_t time);

// rtc.c
void rtc_init(void) {
  if (BKP_ReadBackupRegister(BKP_DR1) != BACKUP_REGISTER_VALUE) {
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP,
                           ENABLE);

    PWR_BackupAccessCmd(ENABLE);

    BKP_DeInit();

    RCC_LSEConfig(RCC_LSE_ON);

    while (RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET) {}

    RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);

    RCC_RTCCLKCmd(ENABLE);

    RTC_WaitForLastTask();
    /* RTC period = RTCCLK/RTC_PR = (32.768 KHz)/(32767+1) */
    RTC_SetPrescaler(32767);

    RTC_WaitForLastTask();
    setTime(0);

    BKP_WriteBackupRegister(BKP_DR1, BACKUP_REGISTER_VALUE);
  } else {
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP,
```

```
                              ENABLE);
    PWR_BackupAccessCmd(ENABLE);
    RCC_RTCCLKCmd(ENABLE);
  }
    RCC_ClearFlag();
}



void setTime(uint32_t time) {
  RTC_WaitForLastTask();
  RTC_SetCounter(time);
  RTC_WaitForLastTask();
}
```

Время храниться ввиде секунд. Для удобства создадим структуру.

```
// rtc.h
typedef struct {
    uint8_t hh;
    uint8_t mm;
    uint8_t ss;
} TIME_t;

// rtc.c
static TIME_t time = { 0, 0, 0, };
```

Счётчик часов реального времени 32-разрядный, поэтому туда можно записывать не просто количество секунд от начала дня (или брать модуль по остатку от 86400 секунд), а использовать **UNIX-время**. Мы однако не будем рассматривать такую реализацию.

Напишем функции установки и считывания времени.

```
void rtc_set_time(const uint8_t hh, const uint8_t mm, const uint8_t ss) {
    uint32_t time = ss + 60 * mm + 3600 * hh;
    RTC_WaitForLastTask();
    RTC_SetCounter(time);
    RTC_WaitForLastTask();
}

TIME_t rtc_get_time(void) {
    uint32_t sec = RTC_GetCounter();
    time.ss = sec % 60;
    time.mm = (sec / 60) % 60;
    time.hh = ((sec / 60) / 60) % 24;
    return time;
}
```

Пока что мы не умеем выводить данные на дисплей, поэтому для проверки просто будем переключать состояние светодиода из одного положения в другое раз в секунду.

```
int main(void) {
    mcu_init(); // --> rtc_init();

    while(1) {
        if (RTC_GetCounter() % 2)
            led_on();
        else
            led_off();
    }
}
```

Наш светодиод должен мигать с частотой 0,5 Гц.

Код урока можно найти на GitHub: **SPL**.

Назад | **Оглавление** | **Дальше**

---

1. Блок RTC в микроконтроллерах серии F1 отличаются от блоков в других сериях, см. документ **AN3371** ↩