Современные 32-разрядные ARM-микроконтроллеры серии STM32: часы реального времени RTC

Олег Вальпа (г. Миасс, Челябинская обл.)

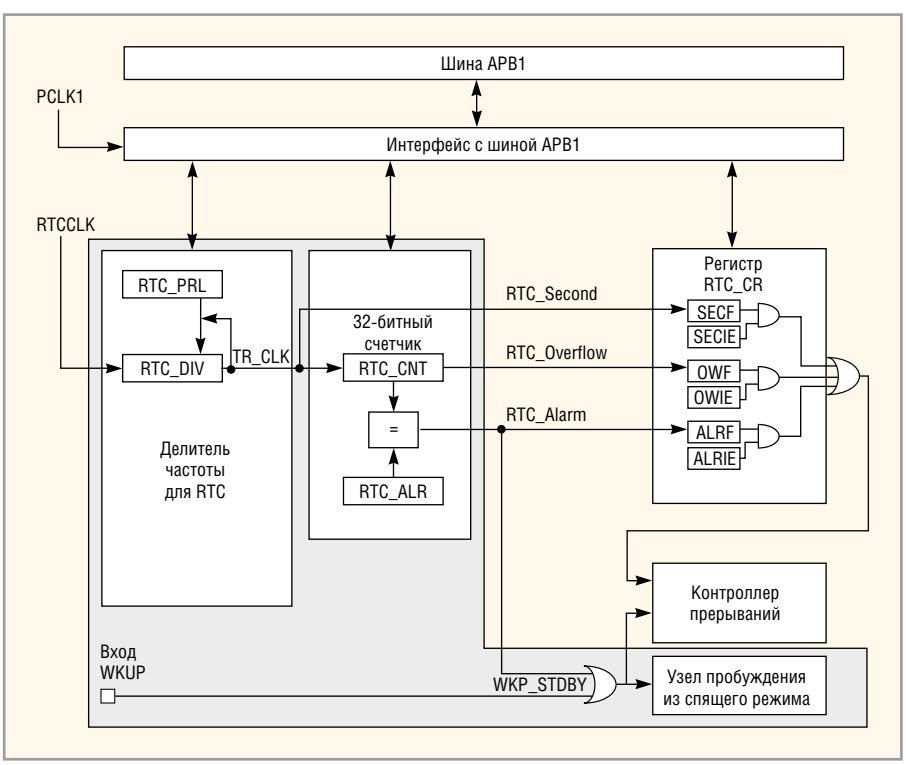
Приведено описание часов реального времени RTC 32-разрядных ARM-микроконтроллеров серии STM32 от компании STMicroelectronics. Рассмотрена архитектура, состав и назначение регистров конфигурирования RTC, а также примеры программ для работы с этим блоком.

Введение

Не все микроконтроллеры имеют в своём составе такой важный аппаратный блок, как часы реального времени (RTC, Real Time Clock). Данного электронного узла нет даже во многих широко распространённых микроконтроллерах семейства AVR. Таким образом, наличие часов реального времени является одним из достоинств микроконтроллеров STM32 компании STMicroelectronics [1].

Представленный блок непрерывно отсчитывает текущее время и позволяет в заданный момент формировать прерывания для процессора микроконтроллера, не отнимая при этом ресурсы у самого процессора на программный отсчёт времени.

Кроме того, блок RTC способен работать при обесточенном микроконтроллере от автономного источника питания. Благодаря тому что блок RTC потребляет ток величиной лишь в несколько микроампер, он может работать очень долгое время от малогабаритного источника с небольшой энергоёмкостью. Это позволяет создавать на базе микроконтроллеров STM32 устройства с энергонезависимыми часами, способными отслеживать время даже при пропадании питания и функционировать в соответствии с временны́м графиком. Например, одним из таких устройств могут быть простые электронные часы, которые автоматически восстанавливают верные показания времени после пропадания питания. Другим примером может служить устройство, которое раз в сутки опрашивает датчики, собирает информацию и передаёт её в центральный узел, а остальное время находится в спящем режиме, экономя энергию автономного источника и увеличивая тем самым продолжительность своей автономной работы.



Структурная схема часов реального времени

Архитектура RTC

Часы реального времени могут тактироваться от одного из трёх источников тактовой частоты. В качестве таких источников сигнала могут служить:

- внутренний высокочастотный сигнал HSE с предварительным делителем на 128;
- внутренний низкочастотный сигнал LSE;
- внешний сигнал LSI от низкочастотного кварцевого резонатора.

На рисунке приведена структурная схема часов реального времени в микроконтроллере STM32.

Блок RTC состоит из интерфейса APB1, предварительного делителя тактовой частоты, набора программируемых счётчиков и регистров, а также логики управления. Интерфейс APB1 служит для связи регистров RTC с шиной APB1. Предварительный делитель тактовой частоты RTC формирует импульсы отсчётов времени TR_CLK для основного счётчика часов. Делитель включает в свой состав регистр RTC_PRL и счётчик RTC_DIV. Данный делитель может быть запрограммирован так, чтобы формировать импульсы с периодом в одну секунду. Например, если частота тактового сигнала взята от «часового» кварца, то для получения секундных импульсов необходимо разделить её на 32 768.

Основную роль часов в составе RTC выполняет 32-разрядный счётчик секунд RTC_CNT, который отсчитывает импульсы, поступающие от делителя частоты. Регистр RTC_ALR служит для формирования сигнала тревоги и позволяет хранить значение, которое регулярно сравнивается с текущим содержимым счётчика секунд. При

достижении счётчиком секунд значения регистра тревоги формируется сигнал RTC_Alarm. Этот сигнал пробуждает процессор и генерирует соответствующее прерывание.

С помощью регистра управления RTC_CR можно разрешить или запретить генерирование прерываний от сигнала секундных импульсов RTC_Second, сигнала переполнения RTC_Overflow и сигнала тревоги RTC_Alarm.

Серым фоном на рисунке выделена энергонезависимая область, которая питается от резервной батареи при отключении основного питания микроконтроллера.

Регистры RTC

Блок RTC включает в свой состав следующие десять регистров:

1. RTC_CRH – старший управляющий регистр;
2. RTC_CRL – младший управляющий регистр;
3. RTC_PRLH – старший регистр коэффициента делителя тактовой частоты;
4. RTC_PRLL – младший регистр коэффициента делителя тактовой частоты;
5. RTC_DIVH – старший счётный регистр делителя тактовой частоты;
6. RTC_DIVH – младший счётный регистр делителя тактовой частоты;
7. RTC_CNTH – старший счётный регистр часов;
8. RTC_CNTL – младший счётный регистр часов;
9. RTC_ALRH – старший сигнальный регистр;
10. RTC_ALRL – младший сигнальный регистр.

Формат этих регистров с названиями входящих в них разрядов представлены в таблице.

Рассмотрим поочерёдно структуру и назначение этих регистров.

Старший регистр **RTC_CRH** имеет структуру, представленную в таблице (см. строки Разряд – Обозначение – Обращение).

Разряд SECIE разрешает прерывание от секундных импульсов.

Разряд ALRIE разрешает прерывание от сигнального регистра.

Разряд OWIE разрешает прерывание по переполнению счётного регистра.

Здесь и далее способ обращения к разрядам регистров имеет следующие условные обозначения:

- rw – допускается чтение и запись разряда;
- r – допускается только чтение разряда;
- w – допускается только запись разряда;
- rc_w1 – допускается чтение разряда и его очистка путём записи 1;
- rc_w0 – допускается чтение разряда и его очистка путём записи 0.

Младший регистр **RTC_CRL** имеет свою структуру (см. табл.).

Три младших разряда являются флагами следующих событий:

- SECF – сформирован секундный импульс;
- ALRF – совпало значение счётного и сигнального регистра;
- OWF – счётный регистр переполнен.

Остальные разряды имеют следующее назначение:

- RSF – флаг синхронизации, устанавливается, когда обновляются регистры RTC_CNT и RTC_DIV;
- CNF – разрешает конфигурирование регистров RTC_CNT, RTC_ALR или RTC_PRL;
- RTOFF – указывает на окончание операции записи в регистр RTC_CNT, RTC_ALR или RTC_PRL.

Регистры **RTC_PRLH** и **RTC_PRLL** задают коэффициент деления входного делителя тактовой частоты часов. Их назначение приведено в соответствующих ячейках таблицы.

Частота импульсов, поступающих на вход счётного регистра, определяется по формуле: $F_{TR\_CLK} = F_{RTCCLK} / (PRL[19:0] + 1)$, где F_{RTCCLK} – частота импульсов на входе делителя. Поэтому при частоте входных импульсов, составляющей 32 768 Гц, для получения секундных импульсов в регистр PRL необходимо записать значение 32 767 = 0 × 7FFF, т.е. в регистр RTC_PRLH записать значение 0 × 7F, а в регистр RTC_PRLL – значение 0 × FF.

Регистры **RTC_DIVH** и **RTC_DIVL** представляют собой счётный входной делитель и доступны только для чтения. Назначение их разрядов при-

Формат регистров RTC

Сдвиг	Регистр	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0 × 00	RTC_CRH	Резерв																													OWIE	ALRIE	SECIE
	Исх.значение																														0	0	0
	Обращение																														rw	rw	rw
0 × 04	RTC_CRL	Резерв																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	Исх.значение																											1	0	0	0	0	0
	Обращение																											r	rw	rc_w0	rc_w0	rc_w0	rc_w0
0 × 08	RTC_PRLH	Резерв																													PRL[19:16]		
	Исх.значение																														0	0	0	0
	Обращение																														w	w	w	w
0 × 0C	RTC_PRLL	Резерв																PRL[15:0]																
	Исх.значение																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Обращение																	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
0 × 10	RTC_DIVH	Резерв																													DIV[19:16]		
	Исх.значение																														0	0	0	0
	Обращение																														r	r	r	r
0 × 14	RTC_DIVL	Резерв																DIV[15:0]																
	Исх.значение																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Обращение																	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
0 × 18	RTC_CNTH	Резерв																CNT[31:16]																
	Исх.значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Обращение																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0 × 1C	RTC_CNTL	Резерв																CNT[15:0]																
	Исх.значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Обращение																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
0 × 20	RTC_ALRH	Резерв																ALR[31:16]																
	Исх.значение																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Обращение																	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
0 × 24	RTC_ALRL	Резерв																ALR[15:0]																
	Исх.значение																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Обращение																	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Листинг 1

```
// Функция инициализации RTC
// Аргументы отсутствуют
// Результат: 1 - инициализация выполнена; 0 - часы уже были инициализированы
unsigned char init_rtc (void)
{
// разрешить тактирование модулей управления питанием и управлением резервной областью
RCC->APB1ENR |= RCC_APB1ENR_PWREN | RCC_APB1ENR_BKPEN;
// разрешить доступ к области резервных данных
PWR->CR |= PWR_CR_DBP;
// если часы выключены - инициализировать их
if ((RCC->BDCR & RCC_BDCR_RTCEN) != RCC_BDCR_RTCEN)
  {
  // выполнить сброс области резервных данных
  RCC->BDCR |=  RCC_BDCR_BDRST;
  RCC->BDCR &= ~RCC_BDCR_BDRST;
  // выбрать источником тактовых импульсов внешний кварц 32768 и подать тактирование
  RCC->BDCR |=  RCC_BDCR_RTCEN | RCC_BDCR_RTCSEL_LSE;
  RTC->CRL  |=  RTC_CRL_CNF;
  RTC->PRLL  = 0x7FFF; // записать в регистр деления 32768
  RTC->CRL  &=  ~RTC_CRL_CNF;
   // установить бит разрешения работы и дождаться установки бита готовности
  RCC->BDCR |= RCC_BDCR_LSEON;
  while ((RCC->BDCR & RCC_BDCR_LSEON) != RCC_BDCR_LSEON) {}
  RTC->CRL &= (uint16_t)~RTC_CRL_RSF;
  while((RTC->CRL & RTC_CRL_RSF) != RTC_CRL_RSF) {}
  return 1;
  }
return 0;
}
```

Листинг 2

```
// Функция чтения счётчика RTC
// Аргументы отсутствуют
// Результат: текущее значение счётного регистра
uint32_t RTC_GetCounter(void)
{
return (uint32_t)((RTC->CNTH << 16) | RTC->CNTL);
}
// Функция записи нового значения в счётчик RTC
// Аргументы: новое значение счётчика
// Результат: отсутствует
void RTC_SetCounter(uint32_t value)
{
RTC->CRL |= RTC_CRL_CNF; // включить режим конфигурирования
RTC->CNTH = value>>16; // записать новое значение счётного регистра
RTC->CNTL = value;
RTC->CRL &= ~RTC_CRL_CNF; // выйти из режима конфигурирования
}
```

ведено в соответствующих строках таблицы.

Счётный регистр состоит из двух 16-разрядных регистров **RTC_CNTH** и **RTC_CNTL**.

Аналогичную структуру имеют сигнальные регистры **RTC_ALRH** и **RTC_ALRL**, информация о которых также представлена в таблице. Отличается лишь способ обращения к ним.

Более подробное описание назначения регистров RTC можно найти в источнике [2].

Программирование

Инициализация и использование

Для инициализации блока RTC необходимо выполнить следующие действия:

- разрешить тактирование и доступ к резервной области данных;
- выбрать источник тактовых импульсов;
- настроить входной делитель.

Функция, выполняющая инициализацию RTC, может иметь код, приведённый в листинге 1.

В данной программе инициализации часто производится обращение к регистру BDCR. Этот регистр управляет сбросом и тактированием области резервных данных BKP. Регистры RTC расположены именно в этой области. В начале программы выполняется проверка работы блока RTC, и если он уже работает, то инициализация не выполняется.

Сброс регистров области резервных данных производится для того, чтобы иметь возможность изменять разряды RTCSEL, задающие источник тактового сигнала. В соответствии с описанием блока часов реального времени, после выбора источника синхронизации для RTC, он может быть изменён только через сброс BKP. Для сброса BKP можно использовать разряд BDRST регистра BDCR.

В данной программе в качестве источника тактового сигнала выбран генератор с внешним кварцем. Если использовать часовой кварц, имеющий частоту 32 768 Гц, то для получения секундных импульсов задаётся коэффициент деления входного делителя, который вычисляется по формуле: PRL[19:0] = = (F_{RTCCLK} / F_{TR_CLK}) – 1.

Для изменения регистра PRL разряд CNF регистра RTC_CRL переводится в единичное состояние, а по окончании модификации сбрасывается.

После такой инициализации блок RTC будет работать даже после выключения основного питания, используя питание блока BKP от резервной батареи.

Для работы со счётным регистром RTC применяются функции чтения и записи, приведённые в листинге 2.

Аналогичным образом можно работать с сигнальным регистром RTC. Конечно, при этом функции должны иметь другое название, например: RTC_GetAlarm и RTC_SetAlarm, и в этих функциях вместо регистров CNTH и CNTL необходимо использовать регистры ALRH и ALRL.

Функции преобразования времени

Поскольку в прикладных программах часто возникает необходимость использования показаний времени в часах, минутах и секундах, а в RTC счётный регистр отсчитывает только секунды, появляется потребность в дополнительных функциях преобразования времени.

Рассмотрим две такие функции. Одна функция будет преобразовывать текущее время в формат, пригодный для записи в счётный регистр, а вторая будет выполнять обратное преобразование.

Для удобства работы с форматом времени удобно использовать специальную структуру. В листинге 3 приведена такая структура, а также функции преобразования времени.

Листинг 3

```
// Структура для хранения параметров времени
typedef struct
{
unsigned char hour; // часы
unsigned char min; // минуты
unsigned char sec; // секунды
} RTC_Time;
// Функция преобразования часов, минут и секунд в секунды для счётчика RTC
// Вход: указатель на структуру, хранящую время для преобразования
// Выход: время в формате секунд
uint32_t TimeToRtc(RTC_Time *time )
{
 uint32_t result;
 result = (uint32_t)time->hour * 3600; // преобразовать часы
 result += (uint32_t)time->min * 60; // преобразовать минуты
 result += time->sec; // преобразовать секунды
 return result;
}
// Функция преобразования секунд счётчика RTC в часы, минуты и секунды
// Вход: значение счётчика и указатель на структуру времени
// Выход: данные структуры времени
void RtcToTime( uint32_t cnt, RTC_Time *time )
{
time->sec = cnt % 60; // сохранить секунды
cnt /= 60;            // вычислить минуты
time->min = cnt % 60; // сохранить минуты
cnt /= 60;            // вычислить часы
time->hour = cnt % 24; // сохранить часы
}
```

Листинг 4

```
/* Программа для работы с RTC */
RTC_Time Time; // структура для работы с временем
uint32_t tmp; // вспомогательная переменная
// Загрузить в структуру время 12:30:45
Time.hour= 12;
Time.min = 30;
Time.sec = 45;
tmp = TimeToRtc(&Time); // преобразовать время в формат секунд RTC
RTC_SetCounter(tmp); // записать время в счётный регистр RTC
// Место для других процедур
// Обратное преобразование
tmp = RTC_GetCounter(); // прочитать секунды из RTC
RtcToTime(tmp,&Time); // преобразовать секунды в формат времени
```

Листинг 5

```
// Функция для обработки прерывания, генерируемого RTC
void RTC_IRQHandler(void)
{
PWR->CR |= PWR_CR_DBP; // разрешить доступ к области резервных данных
 // Если причина прерывания - переполнение входного делителя, т.е.
очередная секунда
 if(RTC->CRL & RTC_CRL_SECF)
   {
   RTC->CRL &= ~RTC_CRL_SECF; // сбросить флаг
   // здесь можно выполнить действия, связанные с данным событием,
   // например, отобразить новое время на дисплее часов
   }
 // Если причина прерывания - совпадение счётного и сигнального
регистра
  if(RTC->CRL & RTC_CRL_ALRF)
  {
     RTC->CRL &= ~RTC_CRL_ALRF; // сбросить флаг
   // здесь можно выполнить действия, связанные с данным событием,
   // например, включить сигнал будильника часов
  }
   // Если причина прерывания - переполнение счётного регистра
  if(RTC->CRL & RTC_CRL_OWF)
  {
     RTC->CRL &= ~RTC_CRL_OWF; // сбросить флаг
   // здесь можно выполнить действия, связанные с данным событием,
   // например, скорректировать время
  }
PWR->CR &= ~PWR_CR_DBP; // запретить доступ к области резервных данных
}
```

Листинг 6

```
PWR->CR |= PWR_CR_DBP; // разрешить доступ к области резервных данных
RTC->CRL |=  RTC_CRL_CNF; // разрешить конфигурирование регистров RTC
RTC->CRH  =  RTC_CRH_SECIE; // разрешить прерывание от секундных
импульсов
RTC->CRH  =  RTC_CRH_ALRIE; // разрешить прерывание при совпадении
счётного и сигнального регистра
RTC->CRH  =  RTC_CRH_OWIE; // разрешить прерывание при переполнении
счётного регистра
RTC->CRL &= ~RTC_CRL_CNF; // выйти из режима конфигурирования
PWR->CR &= ~PWR_CR_DBP; // запретить доступ к области резервных
данных
```

Рассмотрим пример программы для работы с RTC, использующий эти функции. В программе будет осуществляться запись в RTC времени 12 час, 30 минут и 45 секунд, а затем чтение этого времени. Пример программы приведён в листинге 4.

Аналогично можно преобразовывать время для сигнального регистра RTC.

Прерывания RTC

Существует три события, которые могут генерировать прерывание от RTC:

- переполнение входного делителя, т.е. секундные импульсы;
- переполнение счётного регистра;
- совпадение сигнального и счётного регистров.

Чтобы использовать прерывания от RTC, необходимо создать функцию для обработки прерывания. Такая функция может иметь вид, приведённый в листинге 5.

Данная функция позволяет обработать любое из событий, вызвавших прерывание от RTC, с помощью проверки флагов соответствующих событий. При обнаружении причины прерывания сбрасывается флаг события и выполняются необходимые действия для его обработки. Сброс флага нужен, чтобы событие постоянно не генерировало прерывание.

В программе, которая будет использовать функцию обработки прерываний от RTC, требуется активировать события, вызывающие эти прерывания. Это делается с помощью команд, приведённых в листинге 6.

После чего следует разрешить прерывание от RTC с помощью команды:

```
NVIC_EnableIRQ (RTC_IRQn); //
разрешить прерывание от модуля
RTC
```

Заключение

В качестве практического закрепления материала читателям предлагается самим написать программу, которая выполнит инициализацию блока RTC и будет ежесекундно формировать отправку показаний часов через последовательный порт USART микроконтроллера.

Литература

1. www.st.com.
2. www.st.com/web/en/resource/technical/document/reference_manual/CD00246267.pdf.