Современные 32-разрядные ARM-микроконтроллеры серии STM32: базовые таймеры

Олег Вальпа (г. Миасс, Челябинская обл.)

В статье приведено описание таймеров 32-разрядных ARM-микроконтроллеров серии STM32 от компании STMicroelectronics. Рассмотрена архитектура и состав регистров базовых таймеров, а также приведены практические примеры программ.

Введение

Для любого микроконтроллера таймер является одним из важнейших узлов, который позволяет очень точно отсчитывать интервалы времени, считать импульсы, поступающие на входы, генерировать внутренние прерывания, формировать сигналы с широтно-импульсной модуляцией (ШИМ) и поддерживать процессы прямого доступа к памяти (ПДП).

Микроконтроллер STM32 [1] имеет в своём составе несколько типов таймеров, отличающихся друг от друга по функциональному назначению.

Первый тип таймеров является самым простым и представляет собой базовые таймеры (Basic Timers). К данному типу принадлежат таймеры TIM6 и TIM7. Эти таймеры очень просто настраиваются и управляются при помощи минимума регистров. Они способны отсчитывать интервалы времени и генерировать прерывания при достижении таймером заданного значения.

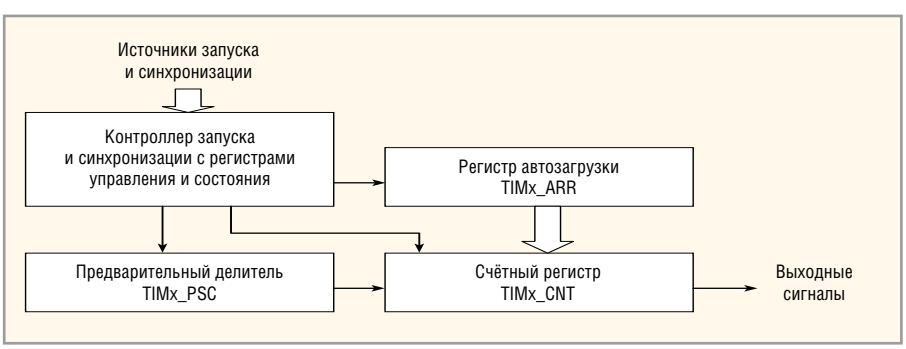
Второй тип представляет собой таймеры общего назначения (General-Purpose Timers). К нему относятся таймеры с TIM2 по TIM5 и таймеры с TIM12 по TIM17. Они могут генерировать ШИМ, считать импульсы, поступающие на определённые выводы микроконтроллера, обрабатывать сигналы от энкодера и т.п.

Третий тип определяет таймеры с развитым управлением (Advanced-Control Timer). К этому типу относится таймер TIM1, который способен выполнять все перечисленные выше операции. Кроме того, на основе данного таймера можно построить устройство, способное управлять трёхфазным электроприводом.

Устройство базового таймера

Рассмотрим устройство и работу базового таймера, структурная схема которого представлена на рисунке.

Базовый таймер построен на основе 16-битных регистров. Его основой является счётный регистр TIMx_CNT. (Здесь и далее символ «x» заменяет номер 6 или 7 для базовых таймеров TIM6 и TIM7 соответственно.) Предварительный делитель TIMx_PSC позволяет регулировать частоту тактовых импульсов для счётного регистра, а регистр автозагрузки TIMx_ARR даёт возможность задавать диапазон отсчёта таймера. Контроллер запуска и синхронизации вместе с регистрами управления и состояния служат для организации режима работы таймера и позволяют контролировать его функционирование.



Структурная схема базового таймера

Благодаря своей организации счётчик таймера может считать в прямом и в обратном направлении, а также до середины заданного диапазона в прямом, а затем в обратном направлении.

На вход базового таймера может подаваться сигнал от нескольких источников, в том числе тактовый сигнал синхронизации от шины APB1, внешний сигнал или выходной сигнал других таймеров, подаваемый на выводы захвата и сравнения.

Таймеры TIM6 и TIM7 тактируются от шины APB1. Если использовать кварцевый резонатор с частотой 8 МГц и заводские настройки тактирования по умолчанию, то тактовая частота с шины синхронизации APB1 составит 24 МГц.

Регистры базового таймера

В таблице приведена карта регистров для базовых таймеров TIM6 и TIM7.

Базовые таймеры включают в свой состав следующие 8 регистров:

- TIMx_CNT – Counter (счётный регистр);
- TIMx_PSC – Prescaler (предварительный делитель);
- TIMx_ARR – Auto Reload Register (регистр автоматической загрузки);
- TIMx_CR1 – Control Register 1 (регистр управления 1);
- TIMx_CR2 – Control Register 2 (регистр управления 2);
- TIMx_DIER – DMA Interrupt Enable Register (регистр разрешения ПДП и прерываний);
- TIMx_SR – Status Register (статусный регистр);
- TIMx_EGR – Event Generation Register (регистр генерации событий).

Регистры TIMx_CNT, TIMx_PSC и TIMx_ARR используют 16 информационных разрядов и позволяют записывать значения от 0 до 65535.

Частота тактовых импульсов для счётного регистра TIMx_CNT, прошедших через делитель TIMx_PSC, рассчитывается по формуле:

$$Fcnt = Fin/(PSC + 1),$$

где Fcnt – частота импульсов счётного регистра таймера; Fin – тактовая

частота; PSC – содержимое регистра TIMx_PSC таймера, определяющее коэффициент деления.

Если записать в регистр TIMx_PSC значение 23999, то счётный регистр TIMx_CNT при тактовой частоте 24 МГц будет изменять своё значение 1000 раз в секунду.

Регистр автоматической загрузки хранит значение для загрузки счётного регистра TIMx_CNT. Обновление содержимого регистра TIMx_CNT производится после его переполнения или обнуления, в зависимости от заданного для него направления счёта.

Регистр управления TIMx_CR1 имеет несколько управляющих разрядов.

Разряд ARPE разрешает и запрещает буферирование записи в регистр автоматической загрузки TIMx_ARR. Если этот бит равен нулю, то при записи нового значения в TIMx_ARR оно будет загружено в него сразу. Если бит ARPE равен единице, то загрузка в регистр произойдёт после события достижения счётным регистром предельного значения.

Разряд OPM включает режим «одного импульса». Если он установлен, после переполнения счётного регистра счёт останавливается и происходит сброс разряда CEN.

Разряд UDIS разрешает и запрещает генерирование события от таймера. Если он обнулён, то событие будет генерироваться при наступлении условия генерирования события, то есть при переполнении таймера или при программной установке в регистре TIMx_EGR разряда UG.

Листинг 1

```
#define FAPB1 24000000 // Тактовая частота шины APB1
// Функция задержки в миллисекундах и микросекундах
void delay(unsigned char t, unsigned int n)
{
// Загрузить регистр предварительного делителя PSC
If(t = = 0) TIM7->PSC = FAPB1/1000000-1; // для отсчёта микросекунд
If(t = = 1) TIM7->PSC = FAPB1/1000-1; // для отсчёта миллисекунд
TIM7->ARR = n; // Загрузить число отсчётов в регистр автозагрузки ARR
TIM7->EGR |= TIM_EGR_UG; // Сгенерировать событие обновления
// для записи данных в регистры PSC и ARR
TIM7->CR1 |= TIM_CR1_CEN|TIM_CR1_OPM; // Пуск таймера
// путём записи бита разрешения счёта CEN
// и бита режима одного прохода OPM в регистр управления CR1
while (TIM7->CR1&TIM_CR1_CEN != 0); // Ожидание окончания счёта
}
```

Разряд CEN включает и отключает таймер. Если обнулить этот разряд, то будет остановлен счёт, а при его установке счёт будет продолжен. Входной делитель при этом начнёт счёт с нуля.

Регистр управления TIMx_CR2 имеет три управляющих разряда MMS2... MMS0, которые определяют режим мастера для таймера.

В регистре TIMx_DIER используется два разряда. Разряд UDE разрешает и запрещает выдавать запрос DMA (ПДП) при возникновении события. Разряд UIE разрешает и запрещает прерывание от таймера.

В регистре TIMx_SR задействован только один разряд UIF в качестве флага прерывания. Он устанавливается аппаратно, при возникновении события от таймера. Сбрасывать его нужно программно.

Регистр TIMx_EGR содержит разряд UG, который позволяет программно генерировать событие «переполнение счётного регистра». При установке этого разряда, происходит генерация события и сброс счётного регистра и предварительного делителя. Обнуляется этот разряд аппаратно. Благодаря этому разряду можно программно генерировать событие от таймера, и тем самым принудительно вызывать функцию обработчика прерывания таймера.

Карта регистров для базовых таймеров TIM6 и TIM7

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
0x00	TIMx_CR1	Резерв																								ARPE	Резерв			OPM	URS	UDIS	CEN
	Исх.значение																									0				0	0	0	0
0x04	TIMx_CR2	Резерв																									MMS[2:0]			Резерв			
	Исх.значение																										0	0	0				
0x08	Резерв																																
0x0C	TIMx_DIER	Резерв																							UDE	Резерв							UIE
	Исх.значение																								0								0
0x10	TIMx_SR	Резерв																															UIF
	Исх.значение																																0
0x14	TIMx_EGR	Резерв																															UG
	Исх.значение																																0
0x18	Резерв																																
0x1C	Резерв																																
0x20	Резерв																																
0x24	TIMx_CNT	Резерв																CNT[15:0]															
	Исх.значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Резерв																PSC[15:0]															
	Исх.значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Резерв																ARR[15:0]															
	Исх.значение																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Листинг 2

```
// Подключение библиотек
#include <stm32f10x.h>
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_tim.h>
#include <misc.h>
// Назначение выводов для светодиодных индикаторов
enum { LED1 = GPIO_Pin_8, LED2 = GPIO_Pin_9 };
// Функция инициализации портов управления светодиодными индикаторами
void init_leds()
{
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
 GPIO_InitTypeDef gpio;
 GPIO_StructInit(&gpio);
 gpio.GPIO_Mode = GPIO_Mode_Out_PP;
 gpio.GPIO_Pin = LED1 | LED2;
 GPIO_Init(GPIOC, &gpio);
}
// Функция инициализации таймера TIM6
void init_timer_TIM6()
{
// Включить тактирование таймера
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE);
TIM_TimeBaseInitTypeDef base_timer;
TIM_TimeBaseStructInit(&base_timer);
// Задать делитель равным 23999
base_timer.TIM_Prescaler = 24000 - 1;
// Задать период равным 500 мс
base_timer.TIM_Period = 500;
TIM_TimeBaseInit(TIM6, &base_timer);
// Разрешить прерывание по переполнению счётчика таймера
TIM_ITConfig(TIM6, TIM_IT_Update, ENABLE);
// Включить таймер
TIM_Cmd(TIM6, ENABLE);
// Разрешить обработку прерывания по переполнению счётчика таймера
NVIC_EnableIRQ(TIM6_DAC_IRQn);
}
// Функция обработки прерывания таймера
void TIM6_DAC_IRQHandler()
{
// Если произошло прерывание по переполнению счётчика таймера TIM6
if (TIM_GetITStatus(TIM6, TIM_IT_Update) != RESET)
 {
 // Обнулить бит обрабатываемого прерывания
 TIM_ClearITPendingBit(TIM6, TIM_IT_Update);
 // Инвертировать состояние светодиодных индикаторов
 GPIO_Write(GPIOC, GPIO_ReadOutputData(GPIOC) ^ (LED1 | LED2));
 }
}
// Главный модуль программы
int main()
{
 init_leds();
 GPIO_SetBits(GPIOC, LED1);
 GPIO_ResetBits(GPIOC, LED2);
 init_timer_TIM6();
 while (1)
  {
  // Место для других команд
  }
}
```

Рассмотрим назначение регистров управления и состояния таймера на конкретных примерах программ.

Примеры программ

Для запуска таймера необходимо выполнить несколько операций, таких как подача тактирования на таймер и инициализация его регистров. Рассмотрим эти операции на основе примеров программ для работы с таймерами.

Довольно часто в процессе программирования возникает задача реализации временных задержек. Для решения данной задачи необходима функция формирования задержки. Пример такой функции на основе базового таймера TIM7 для STM32 приведён в листинге 1.

Эта функция может формировать задержки в микросекундах или миллисекундах в зависимости от параметра «t». Длительность задержки задаётся параметром «n».

В данной программе задействован режим одного прохода таймера TIM7, при котором счётный регистр CNT выполняет счёт до значения переполнения, записанного в регистре ARR. Когда эти значения сравняются, таймер остановится. Факт остановки таймера ожидается в цикле while, путём проверки бита CEN статусного регистра CR1.

Включение тактирования таймеров производится однократно в главном модуле программы при их инициализации. Базовые таймеры подключены к шине APB1, поэтому подача тактовых импульсов выглядит следующим образом:

```
RCC->APB1ENR |= RCC_APB1ENR_
TIM6EN; // Включить тактирование
на TIM6
RCC->APB1ENR |= RCC_APB1ENR_
TIM7EN; // Включить тактирование
на TIM7
```

Описанный выше программный способ формирования задержки имеет существенный недостаток, связанный с тем, что процессор вынужден заниматься опросом флага на протяжении всего времени задержки и поэтому не имеет возможности в это время выполнять другие задачи. Устранить такой недостаток можно с помощью использования режима прерываний от таймера.

Функции обработки прерывания для базовых таймеров обычно выглядят следующим образом:

```
void TIM7_IRQHandler()
{
TIM7->SR &= ~TIM_SR_UIF; //
Обнулить флаг
// Выполнить операции
}
void TIM6_DAC_IRQHandler()
{
// Если событие от TIM6
if(TIM6->SR & TIM_SR_UIF)
 {
 TIM6->SR &= ~TIM_SR_UIF; //
Обнулить флаг
 // Выполнить операции
 }
}
```

Рассмотрим пример программы для организации задержки на базовом таймере TIM6, которая использует прерывания от таймера. Для контроля выполнения программы задействуем один из выводов микроконтроллера для управления светодиодными индикаторами, которые должны будут переключаться с периодичностью, определяемой программной задержкой, организованной на таймере TIM6.

Пример такой программы приведён в листинге 2.

В данной программе функция задержки вызывается один раз, после чего процессор может выполнять другие операции, а таймер будет регулярно формировать прерывания с заданным интервалом задержки.

Аналогичную программу можно написать и для таймера TIM7. Отличие такой программы будет состоять в именах регистров и названии обработчика прерывания. Обработчик прерывания таймера TIM6 имеет одну особенность, связанную с тем, что вектор обработки прерывания этого таймера объединён с прерыванием от цифро-аналогового преобразователя (ЦАП). Поэтому в функции обработчика прерывания выполняется проверка источника прерывания. Подробнее ознакомиться с таймерами микроконтроллера STM32 можно на сайте St.com [2].

Для таймера существует множество других задач, описанных выше, которые он может успешно решить. Поэтому его применение в программе значительно облегчает нагрузку на процессор и делает программу эффективнее.

Литература

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