Лабораторная работа № 4

Контроллеры ARM: базовый таймер

Цель работы: изучение принципиального устройства и основ программирования таймеров на основе CMSIS

1. Таймеры микроконтроллеров STM32

Таймер - устройство, отмеряющее заданный интервал времени, по окончании которого оно генерирует импульсный сигнал (событие). Обычно таймер работает в **периодическом режиме**, т.е. не останавливается после выдачи сигнала, а начинает сразу же отмерять следующий интервал такой же длительности. В результате таймер формирует последовательность событий с регулярной частотой. **Сигнальный режим** подразумевает, что таймер отмеряет заданный интервал времени, а по его окончании выдает сигнал и останавливается (рис. 1).

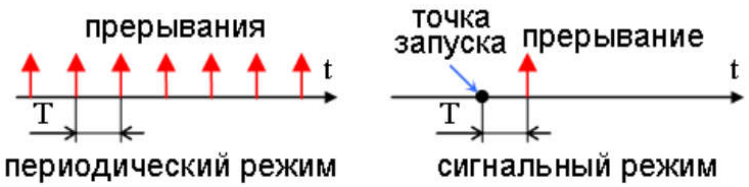


Рис. 1. Режимы работы таймера

Таймер может иметь аппаратную поддержку сигнального режима, либо реализовать его программно (достаточно отключить таймер в обработчике прерывания или выполнить требуемые действия внутри обработчика только при первом вызове).

Таймер строится на основе **счётчика**, который подсчитает импульсы входного тактирующего сигнала. Событие срабатывания таймера генерируется при переполнении счётчика. Это позволяет при фиксированной частоте тактового сигнала устанавливать требуемую частоту генерации сигналов от таймера. Программный доступ к счётчику таймера позволяет по текущему значению с высоким разрешением определять время, прошедшее с момента последнего переполнения. Таким образом, таймер даёт возможность микроконтроллеру и программе вести отсчёт времени, что определяет два основных направления применения:

1. Выполнение действий в заданные моменты времени.
2. Фиксация моментов времени возникновения событий.

К первому направлению можно отнести задачу управления внешними устройствами по заранее запланированному расписанию, в соответствии с которым программа посылает управляющие сигналы контролируемым устройствам. Интервал между двумя соседними моментами времени может варьироваться в широких пределах - от одного периода тактового сигнала до практически бесконечности (часы, сутки, годы). Способность таймера отмерять очень малые интервалы времени открывает возможность применения таймера для генерации сигналов: **цифровых** (для этого достаточно выполнять переключение состояния внешнего вывода в соответствии с законом изменения синтезируемого сигнала) или **аналого-**

вых (за счёт совместной работы таймера и DAC – цифро-аналогового преобразователя). Также таймер можно использовать для реализации многозадачности в операционной среде, когда переключение между выполняемыми задачами осуществляется на основе событий таймера. Не менее важным является второе направление использования таймера - фиксация момента наступления событий. Это может быть: ведение журнала событий, измерение параметров внешних сигналов (длительность импульса, период сигнала) и т.д. Сюда можно отнести решение задач управления, у которых реакция на внешний сигнал зависит от момента его поступления. Хорошей демонстрацией возможностей таймеров является построение частотомера на таймерах.

В микроконтроллерах STM32, всегда имеется, по крайней мере, один таймер - **системный таймер**, наличие которого гарантируется требованиями архитектуры. Стандартизация системного таймера на уровне архитектуры существенно упрощает его использование. Как минимум, это обеспечивает переносимость кода, использующего этот таймер между разными микроконтроллерами. Кроме системного таймера, микроконтроллеры STM32 имеют **сторожевой таймер**. Этот вид таймеров имеют достаточно узкую специализацию - он используется для автоматической перезагрузки в случае зависания программы. Также микроконтроллеры STM32 имеют множество универсальных таймеров, которые можно использовать на своё усмотрение. В отличие от системного, они не являются частью ядра, а относятся к периферийным устройствам микроконтроллера. STM32 включает следующие универсальные типы таймеров: **базовые**, **общего назначения** и **таймеры с расширенным управлением**. Базовые таймеры являются самыми простыми, в соответствии со своим названием имеют лишь набор базовых функций. Таймеры с расширенным управлением - самые сложные и имеют наибольшее количество реализованных функций.

> **Примечание.** Для таймеров в микроконтроллерах семейства STM32 используется следующая система именования: название начинается с префикса TIM, за которым следует номер таймера: TIM1, TIM2, и т.д. Важно, что номер таймера определяет его тип. И в разных микроконтроллерах таймеры с одинаковыми номерами обычно совместимы (одинаково устроены, имеют одинаковый набор функций и управляются одинаковым образом с помощью одинаковых наборов регистров). Такой подход позволяет легко переходить от одного микроконтроллера к другому - как в смысле изучения, так и при переносе кода. Используемая в микроконтроллерах STM система именования таймеров подразумевает то, что нумерация не обязательно должна быть последовательной. Она имеет пропуски при отсутствии какого-либо таймера в конкретной модели микроконтроллера. Поэтому, например, если микроконтроллер имеет таймер с максимальным номером 17 (TIM17), это ещё не означает, что всего имеется 17 универсальных таймеров TIM1..TIM17 - таймеры с какими-то номерами могут отсутствовать.

Микроконтроллеры STM32F072x содержат два базовых таймера TIM6 и TIM7. Структурная схема базового таймера TIM6 отражена на рисунке 2.

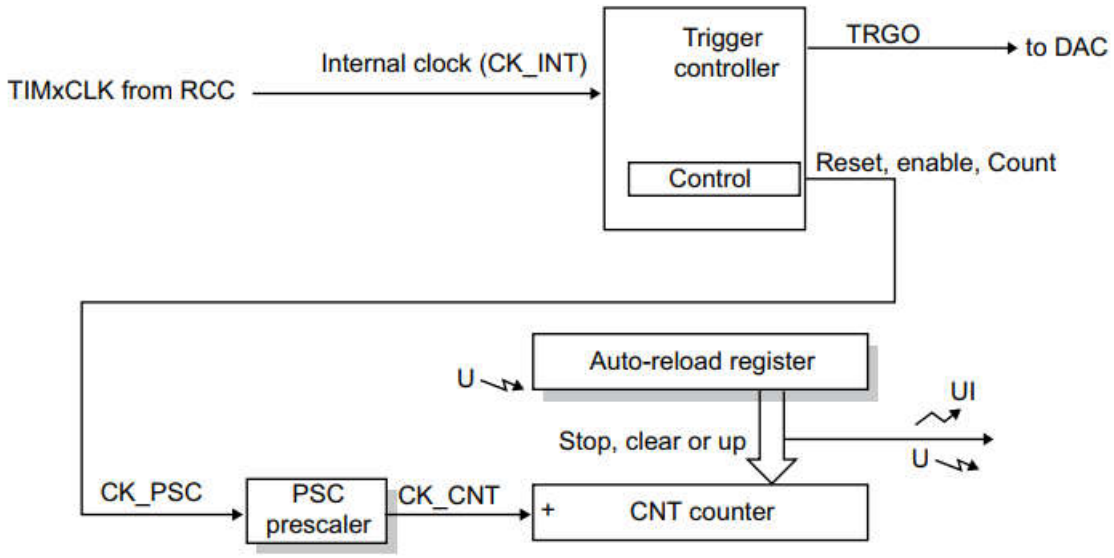


Рис. 2. Структурная схема базового таймера TIM6

Базовый таймер TIMx состоит из 16-битного счетчика – **Counter Register (TIMx_CNT)**, регистра автоперезагрузки – **Auto-Reload Register (TIMx_ARR)** и программируемого предварительного делителя тактового сигнала, включающего регистр предделителя – **Prescaler Register (TIMx_PSC)**. В общем случае, его можно использовать как базовый таймер для генерации временной базы. Специальным образом он применяется для управления цифро-аналоговым преобразователем (ЦАП) посредством триггерных выходов.

В микроконтроллерах STM32 счётчики универсальных таймеров являются 16-разрядными (за немногочисленными исключениями) и работают совместно с **регистром автоперезагрузки**. От последнего счетчик получает и сохраняет в собственном **скрытом регистре** значение, при достижении которого произойдет событие переполнения и сброс счетчика. В базовых таймерах счет выполняется только вверх. Переполнение формирует специальный сигнал - **событие обновления (UEV – update event)**, по которому выполняется ряд важных действий, а также возможна генерация прерывания. Тактовый сигнал на счётчик поступает через **предделитель** с управляемым коэффициентом деления, который можно изменять в пределах от 1 до 65536 (для 16-разрядного счётчика).

На рисунке 3 приведена диаграмма, поясняющая основные принципы работы базовых таймеров на примере изменения коэффициента деления тактовой частоты. Сигнал тактирования **CK_PSC** поступает через системную шину **APB** на предделитель, который первоначально имеет коэффициент равный 1. Коэффициент деления определяется значением регистра **TIMx_PSC** + 1. Поэтому коэффициенту 1 соответствует значение TIMx_PSC равное 0. С выхода предделителя тактирующий сигнал поступает на вход счетчика, где он обозначается **CK_CNT.** На диаграмме видно, что тактовый сигнал поступает на счетчик, когда разрешающий бит **CEN** управляющего регистра **TIMx_CR1** содержит 1. В противном случае, счетчик не работает и таймер бездействует.

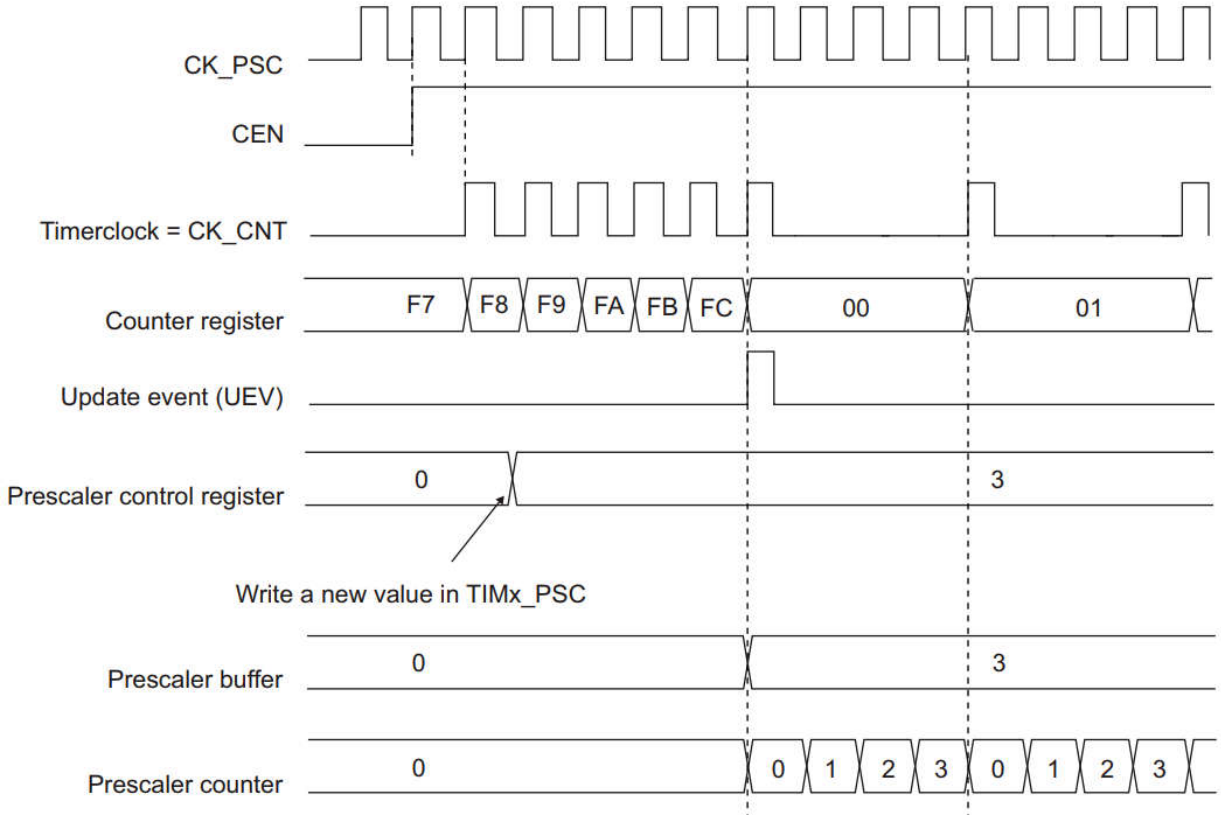


Рис. 3. Временная диаграмма счетчика с изменением значения в предделителе с 1 на 4

В примере регистр **TIMx_ARR** содержит значение 0xFDh. При возникновении **события обновления (UEV)** значение из TIMx_ARR копируется в скрытый (буферный) регистр счетчика. В процессе счета значение TIMx_CNT сравнивается со значение в скрытом регистре, а в момент возникновения равенства генерируется UEV, которое приводит сбросу TIMx_CNT (сигнал **Counter register** на диаграмме). На диаграмме видно, что новый коэффициент предделителя вступает в силу только в момент возникновения UEV. Сигнал **Prescaler counter** указывает, что предделитель для выполнения своей функции использует собственный счетчик, а сигнал **Prescaler buffer** говорит о наличии скрытого (буферного) регистра предделителя, в который помещается значение из TIMx_PSC при возникновении UEV. Кроме всего, следует отметить, что для синхронизации работы элементов таймера используется фронт тактирующего сигнала.

Регистры TIM6/TIM7

Регистр CR1 (программный доступ: **TIMx->CR1**) используется для управления работой таймера (рис. 4).

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

Рис. 4. Структура регистра CR1

Бит 3 **OPM** - одноимпульсный режим:

- 0: счетчик не останавливается при обновлении;
- 1: счетчик прекращает работу при следующем событии обновления (сброс бита CEN).

Бит 1 **UDIS** - запрет обновления. Этот бит устанавливается и сбрасывается программным обеспечением для включения/отключения генерации событий UEV.

- 0: UEV включен. В буферизованные регистры загружаются значения из регистров предварительной загрузки (ARR, PSC);
- 1: UEV отключен. Событие Update не генерируется, скрытые (буферные) регистры сохраняют свое значение.

Бит 0 **CEN -** включение счетчика:

- 0: счетчик отключен;
- 1: счетчик включен

CEN очищается автоматически в одноимпульсном режиме при возникновении события обновления.

Регистр DIER (программный доступ: **TIMx->DIER**) используется для разрешения прерывании и запросов DMA (запросов прямого доступа к памяти) (рис.5).

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Рис. 5. Структура регистра DIER

Бит 8 **UDE** - разрешение запроса DMA (при UEV):

- 0: запрос DMA отключен;
- 1: запрос DMA включен.

Бит 0 **UIE** - разрешение прерывания (при UEV):

- 0: прерывание отключено;
- 1: прерывание разрешено.

Регистр SR (программный доступ: **TIMx->SR**) – статусный регистр (рис. 6).

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Рис. 6. Структура регистра SR

Бит 0 **UIF** - флаг прерывания. Этот бит устанавливается аппаратно при событии обновления. Очищается программно.

- 0: UEV не произошло;

- 1: ожидание обработки прерывания. Этот бит устанавливается аппаратно при обновлении регистров таймера.

Регистр EGR (программный доступ: **TIMx->EGR**) используется для программной генерации UEV (рис. 7).

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Рис. 7. Структура регистра EGR

Бит 0 **UG** - генерация UEV. Этот бит может быть установлен программно. Сбрасывается он аппаратно автоматически.

- 0: никаких действий;
- 1: повторно инициализирует счетчик таймера и генерирует обновление регистров.

Регистр CNT (программный доступ: **TIMx->CNT**) используется для подсчета количества тактовых импульсов, поступивших на вход таймера (рис. 8).

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 8. Структура регистра CNT

Биты 15: 0 **CNT [15: 0]**: значение счетчика.

Регистр PSC (программный доступ: **TIMx->PSC**) задает коэффициент предделителя в таймере (рис. 9).

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рис. 9. Структура регистра PSC

Биты 15: 0 **PSC [15: 0]** - значение предделителя. Тактовая частота счетчика **CK_CNT = f(CK_PSC) / (PSC [15: 0] + 1)**. PSC содержит значение, которое загружается в буферный регистр предварительного делителя при каждом UEV.

Регистр ARR (программный доступ: **TIMx->ARR**) используется для значения счетчика таймера, при котором генерируется событие обновления UEV (рис. 10).

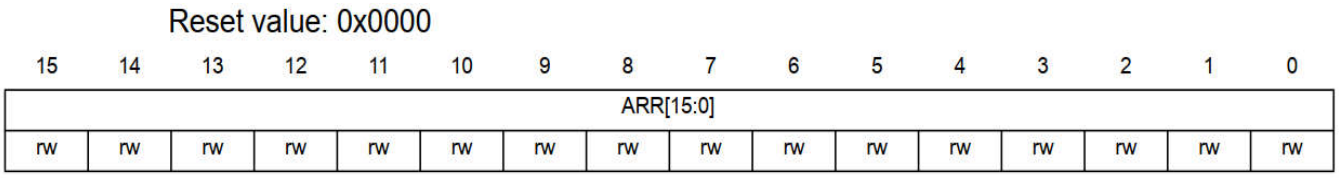


Рис. 10. Структура регистра ARR

ARR содержит значение, которое загружается в буферный регистр счетчика при возникновении события обновления UEV. Счетчик таймера заблокирован, пока значение автоперезагрузки равно нулю.

2. Задание

1. Создайте папку «LabHard(<фамилия на латинице>-<номер группы>)-4». Например, студент Иванов из группы 1191б должен создать папку: LabHard(Ivanov-1191)-4.
2. Внутри папки «LabHard…» создайте папки: «Task4_1», «Task4_2» и «Task4_3». В них далее следует размещать проекты среды Keil µVision, соответственно предназначенные для решения: обучающей, самостоятельной и индивидуальных частей задания.

2.1. Обучающая часть

Задача «TimerBlink»: разработать программные средства для микроконтроллера лабораторного комплекса STM_01, которые реализуют мигание светодиодом на основе базового таймера TIM6 и управление частотой мигания с помощью переключателей SW3 и SW4.

1. В Keil µVision создайте новый проект и сохраните его в папке Task4_1. Название проекта задайте по следующему шаблону: «<инициалы ФИО на латинице>4_1». Например, проект Сидорова Ивана Петровича должен иметь название: **sip4_1**.
2. Добавьте в проект файл **main.h** и файл программного кода **main.c**.
3. В заголовочный файл добавьте код:

```
#include "stm32f0xx.h"              //Подключение биб-ки с моделью stm32f0xx
void InitPortB(void);               //Декларация функции инициализации порта B
void InitTimer6(void);              // Декларация ф-ции инициализации таймера
void TIM6_DAC_IRQHandler(void);     /*Декларация функции обработки
                                    прерывания от таймера TIM6*/
```

4. В начале файла **main.c** подключите заголовочный файл **main.h**. Перед описанием функции main() добавьте глобальные переменные, которые будут использоваться для настройки частоты мигания светодиода и контроля его состояния:

```
static _Bool statePB0=0;   /*Текущее состояние на выводе PB.0: 0 – светодиод
                             погашен; 1 - светодиод горит*/
static uint32_t half_period=125;   //Полупериод мигания светодиода
```

Модификатор **static** в Keil µVision используется для описания глобальных переменных.

5. Код функции main() должен начинаться запретом всех прерываний. Для этого используйте команду:

```
__disable_irq();                    //Глобальное запрещение прерываний
```

Затем следует вызвать функции инициализации порта B и таймера TIM6:

```
InitPortB();                        //Инициализация порта B
InitTimer6();                       //Инициализация таймера TIM6
```

В следующих строчках кода нужно разрешить прерывание от базового таймера TIM6 в системе. Сигнал прерывания от TIM6 подается на 17 линию NVIC (см. лабораторную работу №3, рис. 2). Для активации обработки сигнала на этой линии в NVIC задействуем команду:

```
NVIC->ISER[0] |= 0x20000;           /*Разрешение в NVIC прерывания
                                    от модуля TIM6 (17 линия)*/
```

Далее можно разрешить работы всех прерываний и включить таймер TIM6:

```
__enable_irq();                     //Глобальное разрешение прерываний
TIM6->CR1 |=0x1;                    //Разрешение работы таймера
```

Завершаться код функции main() должен бесконечным пустым циклом:

```
while (1);                          //Пустой цикл основной программы
```

В противном случае завершение работы main() приведет к остановке всех программных модулей, связанных с ней. В том числе, обработчиков прерываний.

6. Инициализация порта B проводится стандартным образом, описанном в лабораторной работе №1. Команды инициализации порта B вынесены в отдельную функцию InitPortB():

```
/*Включение тактирования порта B: RCC_AHBENR_GPIOBEN=0x00040000*/
RCC->AHBENR|=RCC_AHBENR_GPIOBEN;

/*Переключение линий 0 и 8 порта B в режим "Output":
GPIO_MODER_MODER0_0=0x1; GPIO_MODER_MODER8_0=0x10000*/
GPIOB->MODER|=GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;

//Переключение линий 12(SW4) и 13(SW3) порта B в режим "Input"
GPIOB->MODER&=~(GPIO_MODER_MODER12 | GPIO_MODER_MODER13);

/*Разрешение работы светодиодов на стенде STM_01 с помощью установки
логической "1" на выводе PB.8*/
GPIOB->BSRR=0x100;
```

7. Инициализация базового таймера в функции InitTIM6() начинается с включения тактирования модуля TIM6 и установки коэффициента предделителя, позволяющего подать на вход счетчика таймера сигнал с частотой 1 кГц. Чтобы определить значение, размещаемое в регистре TIM6_PSC, нужно частоту сигнала на шине APB (8 МГц) разделить на частоту сигнала на входе счетчика (1 кГц) и вычесть единицу. В итоге получим, что в регистр

TIM6_PSC нужно записать значение 7999, а коэффициент предделителя составит 8000.

```
RCC->APB1ENR=RCC_APB1ENR_TIM6EN; /*Включение тактирования
                                   модуля TIM6*/
TIM6->PSC=7999;   //Установка коэффициента предделителя равного 8000
```

В регистр TIM6_ARR помещаем значение переменной half_period. Это позволит установить длительность интервала обновления состояния светодиода равную 125 / 1 кГц = 125 / 1000 = 0.125 секунды:

```
TIM6->ARR=half_period;
```

А период мигания светодиода составит 2*half_period=0.25 секунды. Последней строкой кода функции InitTimer6() разрешаем модулю таймера генерировать прерывание при возникновении события обновления UEV:

```
TIM6->DIER|=0x1;
```

8. В качестве функции-обработчика прерывания таймера TIM6 используется стандартное название, определенное в библиотеке CMSIS (см. лабораторную работу №3, рис. 8) – TIM6_DAC_IRQHandler(). Обработчик прерывания таймера TIM6 включает код, который выполняет расчет длительности временного интервала, отсчитываемого таймером, на основе положения микропереключателей SW3 и SW4, инвертирует переменную состояния светодиода L1 (см. лабораторную работу №1, рис. 15), выключает светодиод L1, устанавливает новое состояние L1 в соответствие с переменной состояния:

```
TIM6->ARR = (uint32_t)half_period<<(((GPIOB->IDR)&0x3000)>>12);
statePB0 = !statePB0;
GPIOB->BRR = 0x1;
GPIOB->BSRR = statePB0;
```

В окончание кода обработчика TIM6_DAC_IRQHandler() необходимо сбросить флаг прерывания командой:

```
TIM6->SR=0;
```

Полный код проекта «TimerBlink» приведен на рисунке 11.

9. Откомпилируйте программу и постройте исполняемый hex-файл.
10. Загрузите hex-файл на микроконтроллер и протестируйте работу программы.

```c
//-----------------------------------------------------------------------------------------------------------------------------------------------------------
//**Проект: "TimerBlink".
//Разработчик проекта: Долматов Алексей Викторович - ИЦЭ
//Цель: изложить основы программирования в среде Keil uVision базовых таймеров микроконтроллера STM32F072RBT на стенде STM_01
//Решаемые проектом задачи:
//    1. Продемонстрировать назначение основных регистров для программирования базовых таймеров на основе CMSIS
//    2. Выполнить настройку обработчика прерывания базового таймера
//    3. Разработать программу управления частотой мигания светодиода на основе базового таймера
//-----------------------------------------------------------------------------------------------------------------------------------------------------------

#include "main.h"

static _Bool statePB0=0;          //Текущее состояние на выводе PB.0: 0 - светодиод погашен; 1 - светодиод горит
static uint32_t half_period=125;  //Полупериод мигания светодиода

//Основная программа микроконтроллера
int main()
{
  __disable_irq();              //Глобальное запрещение прерываний
  InitPortB();                  //Инициализация порта B
  InitTimer6();                 //Инициализация таймера TIM6
  NVIC->ISER[0] |= 0x20000;     //Разрешение в NVIC прерывания от модуля TIM6 (17 линия)
  __enable_irq();               //Глобальное разрешение прерываний
  TIM6->CR1 |=0x1;              //Разрешение работы таймера
  while (1);                    //Пустой цикл основной программы
}

//Функция инициализации порта B с подключенным светодиодом VD0 и микропереключателями SW3 и SW4
void InitPortB(void)
{
  RCC->AHBENR|=RCC_AHBENR_GPIOBEN;                          //Включение тактирования порта B: RCC_AHBENR_GPIOBEN=0x00040000
  GPIOB->MODER|=GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;  //Переключение линий 0 и 8 порта B в режим "Output": GPIO_MODER_MODER0_0=0x1; GPIO_MODER_MODER8_0=0x10000
  GPIOB->MODER&=~(GPIO_MODER_MODER12 | GPIO_MODER_MODER13); //Переключение линий 12(SW4) и 13(SW3) порта B в режим "Input"
  GPIOB->BSRR=0x100;                                        //Разрешение работы светодиодов на стенде STM_01 с помощью установки логической "1" на выводе PB.8
}

//Функция инициализации базового таймера TIN6
void InitTimer6(void)
{
  RCC->APB1ENR=RCC_APB1ENR_TIM6EN;    //Включение тактирования модуля таймера TIM6
  TIM6->PSC=7999;                     //Установка коэффициента предделителя равного 8000: после предделителя частота сигнала = Частота на шине APB / 8000 = 8 МГц / 8000 = 1кГц
  TIM6->ARR=half_period;              //Длительность периода обновления таймера = half_period / 1 кГц = 125 / 1000 = 0.125 секунды при инициализации таймера
  TIM6->DIER|=0x1;                    //Разрешить аппаратную установку сигнала прерывания при событии обновления таймера (UEV)

}

//Функция-обработчик прерывания таймера TIM6
void TIM6_DAC_IRQHandler(void)
{

  TIM6->ARR = (uint32_t)half_period << (((GPIOB->IDR)&0x3000)>>12);     //Установка длительности периода обновления таймера
                                                                        //в соответствии с SW3 и SW4: 00 - 4 Гц; 01 - 2 Гц; 10 - 1 Гц; 11 - 0.5 Гц
  statePB0 = !statePB0;                                                 //Инвертировать состояние светиодиода
  GPIOB->BRR = 0x1;                                                     //Отключить светодиод     на выводе PB.0
  GPIOB->BSRR = statePB0;                                               //Установить светодиод на выводе PB.0 в соответствии с состоянием, хранимым в переменной statePB0
  TIM6->SR=0;                                                           //Сбросить флаг прерывания таймера TIM6
}
//-----------------------------------------------------------------------------------------------------------------------------------------------------------
//Руководство пользователя:
//    1. Для запуска загруженной программы удалите перемычку "boot" на стенде и нажмите кнопку "reset"
//    2. В управлении частотой мигания светодиода используйте микропереключатели SW3(старший разряд) и SW4(младший разряд);
//    3. Частота мигания светодиода VD0 на линии PB.0: 00 - 4 Гц; 01 - 2 Гц; 10 - 1 Гц; 11 - 0.5 Гц
//    4. Интервал между переключениями светодиода задается с помощью базового таймера TIM6
//-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

Рис. 11. Проект «TimerBlink»

2.2. Самостоятельная работа

Цель Task4_2: управление яркостью светодиода с помощью ШИМ.

На стенде STM_01 рассмотрим подключенную к выводам PB.0 и PB.8 внешнюю цепь со светодиодом L1 (см. лабораторную работу №1, рис. 15). Если на эти выходы порта B подать постоянное напряжение $+U_{пит}$, то через светодиод будет протекать электрический ток, что вызовет его свечение с максимальной мощностью. Когда выключена одна из указанных линий порта светодиод не горит (мощность его излучения равна нулю). Переключение цифровой линии PB.0 на малой частоте (0 – 20 Гц) вызывает мигание светодиода, наблюдаемое глазом человека. Однако мигание с частотой 1 кГц глаз различить уже не сможет, и человеку будет казаться, что светодиод горит постоянно. Это происходит потому, что глаз усредняет световой поток за время около 50 мс. И хотя в данный промежуток времени укладывается 20 периодов мигания светодиода, глаз будет воспринимать среднюю яркость как постоянную величину. Если интервал свечения равен половине периода мигания, то за 50 мс на мигающий светодиод будет подано в 2 раз меньше электрической энергии, чем на постоянно горящий. Поэтому средняя яркость мигающего светодиода будет ниже, чем у постоянного горящего. Отношение интервала времени с высоким уровнем бинарного сигнала к периоду его колебания называют **коэффициентом заполнения** (рис. 12).

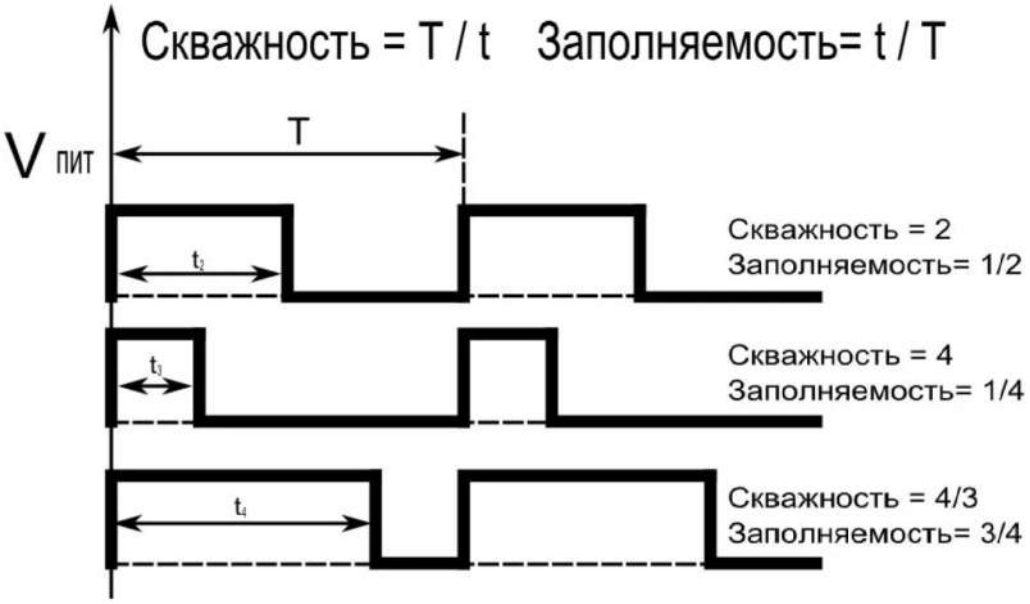


Рис. 12. Пояснение скважности и коэффициента заполнения сигнала

Когда светодиод включен постоянно, коэффициент заполнения сигнала равен 100%. Если интервал свечения равен половине периода мигания светодиода, то коэффициент заполнения равен 50%. В целом, регулировку яркости свечения светодиодов на стенде STM_01 можно выполнять с помощью контроля коэффициента заполнения сигнала от 0 до 100% на выходах цифровых портов PB0 – PB7. Такое управление мощностью нагрузки называют **широтно-импульсной модуляцией (ШИМ)**.

1. В проекте Task4_2 настройте предделитель таймера TIM6 так, чтобы на вход счетчика поступал сигнал с частотой f = 100 кГц (период сигнала T=10 мкс).
2. Запрограммируйте считывание кода состояния микропереключателей SW2, SW3, SW4 только в момент перехода сигнала PB.0 с низкого уровня на высокий.
3. Выключенным переключателям (SW2 – старший разряд кода, SW3, SW4 – младший разряд кода) должен соответствовать бинарный код 000, а включенным – 111. Интервал горения светодиода для кода 000 должен равняться 2040 мкс, а для кода 111 – 10 мкс. Шаг изменения интервала горения светодиода считать постоянным.
4. Сконфигурируйте выход PB.0 на максимальную частоту синхронизации и используйте атомарный метод управления этой линией порта.
5. Период ШИМ сигнала светодиода должен равняться 2050 мкс. Запрограммируйте обработчик прерывания таймера TIM6 так, чтобы таймер мог последовательно отсчитывать интервалы свечения и гашения светодиода, управляемого через линию PB.0.
6. Откомпилируйте и загрузите программу на стенд STM_01.
7. Протестируйте программу и понаблюдайте как изменяется яркость светодиода при переключении SW2-SW4.

2.3. Индивидуальное задание

Цель Task4_3: в соответствие с вариантом индивидуального задания на стенде STM_01 запрограммировать зависимость яркости светодиода от времени, используя два таймера: TIM6 и TIM7.

1. В файле «Назначение вариантов (группа).pdf» определить номер Вашего вариант для выполнения индивидуального задания.
2. В **Приложении 1** находится описание индивидуальных характеристик управляющего светодиодом сигнала.
3. Исходя из положения переключателей SW1 и SW2 с помощью TIM6 задайте частоту ШИМ, опираясь на ее максимальное и минимальное значение из индивидуального варианта задания (см. **Приложение 2**). Шаг изменения частоты ШИМ считать постоянным.
4. В индивидуальном варианте задания (см. **Приложение 2**) даны: форма зависимость яркости светодиода от времени, минимальная и максимальная величина коэффициента заполнения управляющего сигнала и количество его фаз за один период колебания. Используя эти параметры для одного периода колебания яркости светодиода рассчитать массив значений коэффициента заполнения управляющего сигнала. Временные промежутки между соседними фазами сигнала считать одинаковыми.

5. Исходя из положения переключателей SW3 и SW4 с помощью TIM7 задайте частоту мигания светодиода, опираясь на ее максимальное и минимальное значение из индивидуального варианта задания (см. **Приложение 2**). Шаг изменения частоты мигания светодиода считать постоянным.
6. Произведите тестирование и отладку программы.

3. Подготовка отчета, представление и оценка работы

Комментарии в программном коде в качестве «Отчета о выполнении задания»

1. Комментарии в программном коде каждого проекта выступают в роли отчета и оцениваются в процессе защиты работы.
2. Структура комментариев, описывающих программный код должна быть следующей:
    - В начале файла «main.c» должны присутствовать сведения о разработчике (ФИО - группа), цели выполнения работы и задачах, которые решались в ходе создания программы;
    - Перед реализацией каждой функции следует указать ее назначение и привести список входных/выходных параметров с описанием;
    - Внутри тела функции следует выделить отдельные блоки, реализующие единую комплексную операцию обработки данных, и вынести в их заголовок сведения о назначении блока;
    - Каждая элементарная операция в строке кода должна быть снабжена комментарием;
    - В конце файла «main.c» нужно привести комментарии, содержащие краткое руководство использования программы на стенде STM_01.
3. В заголовочных файлах (с расширением «h»), созданных разработчиком проекта, следует прокомментировать все строки кода. Однако можно опустить заголовочный и финальный комментарии со сведениями о проекте и руководстве пользователя.

Представление и защита работы

4. Для каждой части **Задания** нужно построить отдельный проект Keil µVision, протестировать и отладить его работу на стенде STM_01, снабдить программный код комментариями в соответствие с п. 3.1- 3.3.
5. Представление работы начинается с демонстрации преподавателю работоспособности программного кода проекта на стенде STM_01. Каждый проект (часть задания лабораторной работы) может быть представлена отдельно. В ходе представления преподаватель оценивает функциональность программы и качество написания программного кода. Студент должен быть готов ответить на вопросы преподавателя по использованным в проекте алгоритмам и их программной реализации.

6. После получения оценки за все проекты лабораторной работы следует создать единый архив (со всеми проектами), и загрузить его на сайт Eluniver.
7. По комментариям к проекту преподаватель оценивает качество отчета по каждому из выполненных заданий.
8. После представления всех заданий лабораторной работы студент получает право ее защиты, которая заключается в ответе на 2 контрольных вопроса из обозначенного в **разделе 4** списка. Каждый контрольный вопрос оценивается отдельно.

Структура оценки лабораторной работы

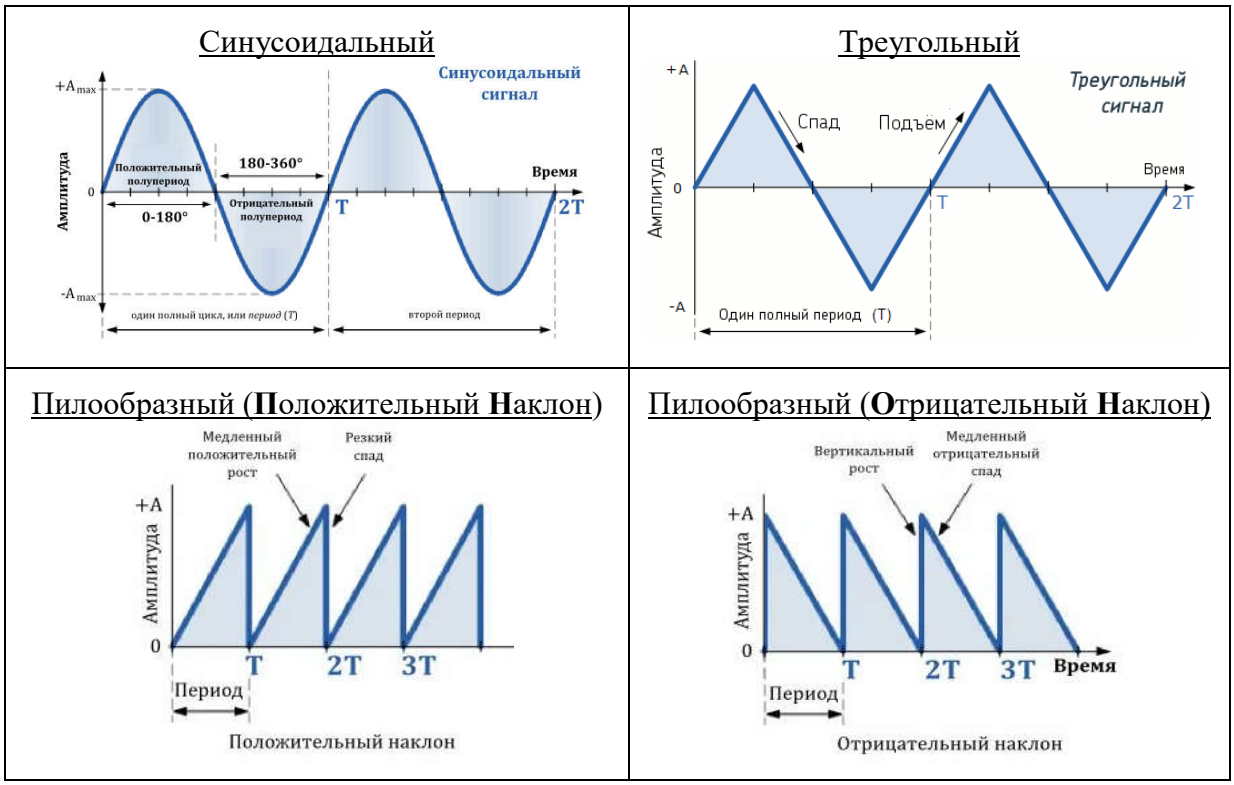
№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	10
2.	Проект «Самостоятельная работа»	15
3.	Проект «Индивидуальное задание»	25
4.	Отчет «Обучающая часть»	5
5.	Отчет «Самостоятельная работа»	5
6.	Отчет «Индивидуальное задание»	10
7.	Контрольный вопрос 1	15
8.	Контрольный вопрос 2	15
	Итого:	**100**

4. Контрольные вопросы

1. На чем основана способность таймера измерять временные промежутки?
2. От чего зависит временное разрешение таймера?
3. Какие существуют типы таймеров по принципу действия? Назовите их отличительные черты?
4. Из каких элементов построен таймер? Назовите функции каждого элемента.
5. Какие основные направления выделяют при использовании таймеров?
6. Как называется сигнал, генерируемый таймером, и когда он формируется?
7. Какова форма сигнала таймера?
8. Когда при возникновении сигнала таймера в микроконтроллере генерируется прерывание?
9. Какова классификация таймеров в микроконтроллерах STM32 по функциональным возможностям?
10. Чем отличается универсальный таймер от системного?
11. К какому классу относятся таймеры общего назначения?
12. Какой вид таймеров имеет минимальные функциональные возможности? Перечислите функциональные возможности этих таймеров.
13. Определяет ли номер таймера в микроконтроллере его тип?
14. Какие функциональные блоки содержит базовый таймер микроконтроллера? Поясните назначение каждого функционального блока.
15. Какой максимальный коэффициент деления тактовой частоты можно задать в таймерах STM32? Поясните.
16. Чем определяется частота тактирования счетчика таймера? В каких пределах ее можно изменять?
17. Каково назначение регистра предделителя и буферируется ли его значение?
18. Каким образом можно таймер TIM6 перевести в сигнальный режим работы?
19. Что происходит внутри таймера при возникновении события обновления?
20. Каково назначение в таймере регистра EGR?
21. Как называется регистр, который определяет коэффициент предделителя в таймере? Укажите его значения для установки коэффициента деления 540.
22. Какой регистр таймера хранит значение счетчика, при котором генерируется событие обновления?
23. Как с помощью регистров таймера можно определить временной отрезок до следующего события обновления?
24. Каково назначение статусного регистра таймера?
25. К чему приведет выполнение программной инструкции: ***TIM6->DIER=0x1000;***
26. Какой осуществляется регулировка мощности нагрузки в методе ШИМ?
27. Как связаны между собой коэффициент заполнения и скважность сигнала?
28. Почему яркость горения светодиода, воспринимаемая глазом человека, не зависит от периода сигнала, когда частота ШИМ больше 100-300 Гц?
29. Какой коэффициент заполнения имеет сигнал, если интервал свечения составляет 30 мкс, а интервал гашения – 270 мкс?
30. Как с помощью таймеров можно синтезировать на выходе микроконтроллера сигнал заданной скважности?

Приложение 1. Характеристики управляющего сигнала

1. $f_{\min}$ - минимальная частота ШИМ;
2. $f_{\max}$ - максимальная частота ШИМ;
3. $K_{\min}$ - минимальный коэффициент заполнения управляющего сигнала;
4. $K_{\max}$ - максимальный коэффициент заполнения управляющего сигнала;
5. *TD* (time dependence) - форма зависимость коэффициента заполнения управляющего сигнала от времени:



Примечание. В сигнале треугольной формы считать, что «Подъем» и «Спад» происходят с одинаковой абсолютной скоростью.

6. N - количество фаз в одном периоде управляющего сигнала;
7. $F_{\min}$ - минимальная частота колебания яркости светодиода;
8. $F_{\max}$ - максимальная частота колебания яркости светодиода.

Приложение 2. Варианты индивидуальных заданий

Вариант	f_{min}, кГц	f_{max}, кГц	K_{min}, %	K_{max}, %	*TD*	*N*	F_{min}, Гц	F_{max}, Гц
1.	4	7	1	95	синусоидальный	250	0.5	3.5
2.	2	5	0.1	75	треугольный	400	0.25	1.75
3.	1	4	5	99	пилообразный (ПН)	100	2	8
4.	3	9	10	80	пилообразный (ОН)	333	1	7
5.	1,5	3	1	90	треугольный	220	0.5	2
6.	1	7	0.5	50	пилообразный (ПН)	150	0.2	1.4
7.	2	8	2	85	синусоидальный	340	0.4	2.2
8.	2	5	0.1	75	пилообразный (ОН)	75	1	4
9.	1	4	5	99	пилообразный (ПН)	160	0.6	1.5
10.	3	9	10	80	треугольный	120	0.1	2.2
11.	1,5	3	1	90	пилообразный (ОН)	98	0.25	1.75
12.	1	7	0.5	50	синусоидальный	200	2	8
13.	4	7	1	95	пилообразный (ПН)	100	1	7
14.	2	5	0.1	75	треугольный	240	0.4	2.2
15.	1	4	5	99	синусоидальный	310	1	4
16.	2	5	0.1	75	пилообразный (ОН)	170	0.6	1.5
17.	1	4	5	99	треугольный	86	0.5	3.5
18.	3	9	10	80	пилообразный (ПН)	94	0.25	1.75
19.	3	9	10	80	синусоидальный	186	2	8
20.	1,5	3	1	90	пилообразный (ОН)	115	0.2	1.4
21.	4	7	1	95	синусоидальный	360	0.4	2.2
22.	2	5	0.1	75	пилообразный (ПН)	290	1	4
23.	1	4	5	99	пилообразный (ОН)	174	0.6	1.5
24.	3	9	10	80	треугольный	236	0.5	3.5
25.	1,5	3	1	90	пилообразный (ОН)	80	0.25	1.75
26.	1,5	3	1	90	синусоидальный	110	0.1	2.2
27.	1	7	0.5	50	треугольный	260	0.25	1.75
28.	4	7	1	95	пилообразный (ПН)	390	2	8
29.	2	5	0.1	75	синусоидальный	234	1	4
30.	1	4	5	99	пилообразный (ОН)	136	0.6	1.5