Лекция 3

Таймеры

Одна из основных задач таймеров в микроконтроллерах это отсчитывать точные интервалы времени. Но, помимо этого таймеры могут использоваться для измерения частоты, периодов, генерации ШИМа и переменных сигналов различной формы. Всего

TIM9-TIM11	Самые простые 16 битные таймеры.
TIM2-TIM5	Таймеры общего назначания. (TIM2 и TIM5 32 битные) (TIM3 и TIM4 16 битные)
TIM1	Расширенный 16 битный таймер
SYSTEM TIMER	Также существуют системный таймер **SysTick** таймер и Watchdog таймер.

В данном курсе мы рассмотрим только 32 битные таймеры общего назначения. Самостоятельно можно будет ознакомиться с TIM1.

Системный таймер

Самый просто таймер, встроенный в ядро ARMv7, на котором построено ядро CortexM4 и наш микроконтроллер stm32F411, т.е. его поддерживают все микроконтроллеры на этом ядре.

- 24 - битный таймер считающий вниз от заданного значения до нуля.

Для его настройки используется всего 3 регистра. В документации на ядро ARMv7 (кому интересно, небольшая брошурка (на 2720 страниц) про это ядро находится здесь: https://documentation-service.arm.com/static/5f8daeb7f86e16515cdb8c4e?token=)

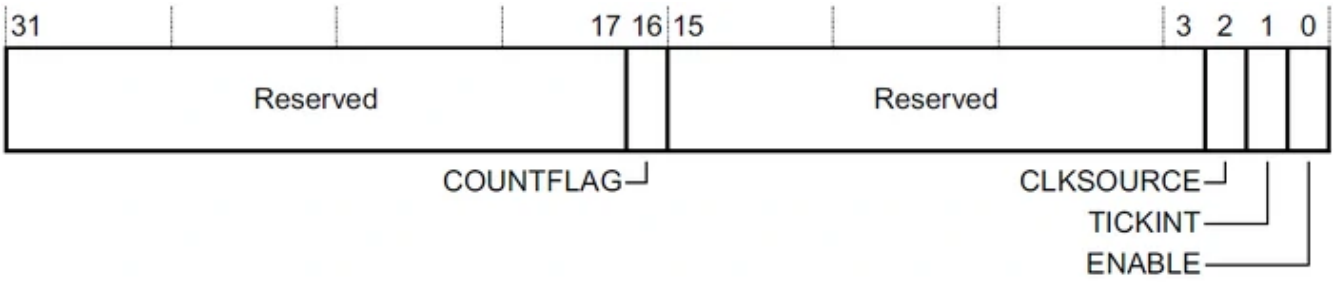
Table B8-1 Timer registers summary for the Generic Timer

	PL1 physical timer[a]	PL2 physical timer[b]	Virtual timer
CompareValue register	CNTP_CVAL	CNTHP_CVAL	CNTV_CVAL
TimerValue register	CNTP_TVAL	CNTHP_TVAL	CNTV_TVAL
Control register	CNTP_CTL	CNTHP_CTL	CNTV_CTL

В обертке над регистрами, я назвал их немного по другому, но суть от этого не меняется. Итак:

SysTick Control and Status Register (SYST_CSR)

- 32 битный регистр для управления системным таймером



Биты	Доступ	Обозначение	Описание
31:17			Зарезервировано
16	Read/Rewrite	COUNTFLAG	Показывает, дошёл ли счётчик до нуля с момента последнего считывания этого регистра: **NoOverflow** (0) — счётчик ещё не достигал нуля **Overflow** (1) — счётчик достигал нуля Этот бит сбрасывается считыванием данного регистра, а также записью значения в регистр SYST_CVR. Его установка производится, когда значение счётчика переходит из 1 в 0
15:3			Зарезервировано
2	Read/Write	CLKSOURCE	Выбирает источник синхронизации для таймера: **ExternalClock**(0) — используется внешний источник **CpuClock** (1) — для синхронизации используется частота процессора
1	Read/Write	TICKINT	Определяет, будет ли генерироваться запрос на прерывание при достижении счетчиком 0: **DisableInterrupt** (0) — запрос прерывания не выдаётся **EnableInterrupt** (1) — запрос прерывания выдаётся Достижением нуля считается только декремент значения счётчика, приводящий к появлению в нём нуля, но не его сброс в результате явной записи в регистр SYST_CVR

Биты	Доступ	Обозначение	Описание
0	Read/Write	ENABLE	Определяет, разрешена ли работа таймера (уменьшение счётчика): **Disable**(0) — счётчик выключен **Enable** (1) — счётчик включён

Регистр перезагружаемого значения LOAD

- 32 битный регистр, из которого используются только первые 24 бита. В этом регистре хранится значение, которое будет записано в системный таймер как только его счетчик достигнет 0.

Регистр текущего значения VAL

- 32 битный регистр, из которого используются только первые 24 бита. В этом регистре хранится текущее значение счетчика.

Алгоритм работы с системным таймером

С помощью системного таймера можно задать практически любое значение задержки для этого необходимо:

- Записать в регистре **LOAD** значение задержки. Так как счетчик системного таймера уменьшается на 1 с каждым тактом процессора, то нетрудно посчитать, что если системная частота микроконтроллера подключена к внутренней источнику тактирования HSI, то для создания задержки в 1мс, необходимо записать в этот регистр значение (16'000'000/1000 - 1). Счетчик начнет уменьшаться с этого значения на 1 с каждым тактом процессора, и как только он дойдет до 0 установится флаг прерывания в регистре **ICSR** в поле **PENDSTSET** (26 бит) 1.
- Записать в текущее значение счетчика в регистр **VAL** - 0
- Подключить системный таймер к частоте процессора в регистре **CTRL**
- Включить системный таймер в регистре **CTRL**
- Дождаться готовности флага **PENDSTSET** в регистре **ICSR**

Задание

1. Подключить микроконтроллер к внешнему источнику тактирования HSE.
2. Написать программу морганиями всеми 4 светодиодами на плате с периодом в 0.5 секунды
3. По нажатию кнопки увеличивать период моргания на 0.1 секунды.

Таймеры TIM2 и TIM5, основные особенности

- Таймеры 32 битные (то есть могут считать до 2^32), умеют работать:
  - с инкрементальными энкодерами и датчиками Холла,
  - несколько таймеров можно синхронизировать между собой.
- Таймеры могут использоваться для:
  - Захвата сигнала (Защелкивать значение, когда на выводе порта например 0 сменился на 1)
  - Сравнения (Считать до значения в регистре сравнения и установить/сбросить/переключить вывод порта)
  - Генерации ШИМ (Генерировать прямоугольный сигнал с различной скважностью на вывод порта)
  - Генерации одиночного импульса

Таймеры TIM2 и TIM5

- Таймеры могут генеририровать следующие события:
  - Переполнение
  - Захват сигнала
  - Сравнение
  - Событие-триггер

Таймеры TIM2 и TIM5 начальная запуск

- Таймеры тактируются от шины APB1.

Поэтому для каждый отчсет таймера по умолчанию происходит на частоте шины, т.е. если шина **APB1** работает на частоте 1 Мгц, то один отсчет таймера произойдет через 1 мкс. Таким образом можно организовать измерение времени с разрешением в 1 мкс. Чтобы таймер заработал, его нужно подключить к системе тактирования, т.е. к шине **APB1**.

- Подключение к системе тактирование выполняется через регистр **APB1ENR** модуля **RCC**.
- Входную частоту таймера можно поделить, записав делитель частоты в решистр **PSC**.
- Включение таймера производиться с помощью бита **CEN** в регистре **CR1** модуля таймера (TIM2 или TIM5)

Таймеры TIM2 и TIM5 переполнение

Как только таймер начал считать, его счетчик будет увеличиваться с каждым тактом подающейся на таймер частоты. Т.е. если входная частота таймера 1 МГц, то через секунду таймер достчитает до 1 000 000.

- Значение счетчика таймера можно прочитать из регистра **CNT**.
  - Поскольку таймеры **TIM2** и **TIM5** 32 битных, то переполнение наступит когда в регистре **CNT** будет значение **0xFFFFFFFF**, нетрудно посчитать, что при частоте работе таймера 1 МГц он переполнится через примерно 71.5 минуты.
  - При переполнении таймера, он сгенерирует событие (запрос на прерывание).
- Проверить случилось ли переполнение можно, считав бит **UIF** в регистре **CR**.

Таймеры TIM2 и TIM5 режим счета до значения

Допустим, нам нужно раз в 71.5 минуты моргнуть светодиодом. Мы можем запустить таймер и и постоянно проверять значение бита **UIF**, как только оно установится в 1, моргнуть светодиодом.

- Используя переполнение невозможно задать таймером произвольный интервал времени.
- Задать производльный интервал можно, используя регистр автоперезагрузки **ARR**. В этот регистр записывается число, до которого будет идти счет. При достижении этого значения, содержимое счетчика **CNT** обнуляется и формируются прерывание или запрос DMA (если они разрешены).

Например: мы хотим раз в 1 секунду моргать светодиодом. Частота работы таймера 1 Мгц. Чтобы таймер генерировал запрос на прерывание каждыые 1 секунду, нужно записать число 1 000 000 в регистр **ARR** и число 0 в регистр **CNT** и после этого запустить таймер. Как только таймер досчитает до 1 000 000 он выставит флаг **UIF**.

Таймеры TIM2 и TIM5 регистры для режима счета

TIMx::CNT

Счетный 16/32 разрядный регистр таймера суммирующий, с приходом каждого тактового импульса инкрементирует свое содержимое. На вычитание работать не может.

TIMx::PSC

16 разрядный регистр - делитель частоты для таймера. Коэффициент деления задается в 16-разрядном регистре, этот коэффициент можно задать в пределах от 1 до 65536.

TIMx::ARR

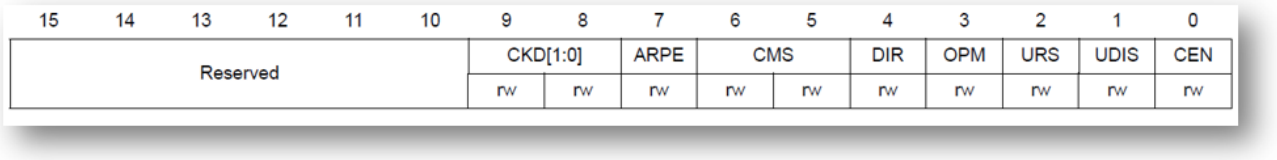
16/32 разрядный регистр автоперезагрузки. В этот регистр записывается число, до которого будет идти счет. При достижении этого значения, содержимое счетчика TIMx_CNT обнуляется и формируются прерывание или запрос DMA (если они разрешены).

TIMx::SR

Регистр статуса. Можно узнать о всех возможных запросах на прерывания от таймера

Таймеры TIM2 и TIM5. Управляющий регистр (CR1)

Основные настройки таймера производятся через регистр CR1. Нам понадобятся всего несколько бит.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 1. Регистр CR1

Bit 2: URS Источник генерации прерываний

- **0**: Любые из следующих событий будут генерировать прерывание или запрос DMA, если они включены:
  - Переполнение счетчика или установлен UG бит
- **1**: Только после переполнения счетчика может сгенерировать прерывание или запрос DMA

Bit 1: UDIS Отключить событие по изменению (Update Event)

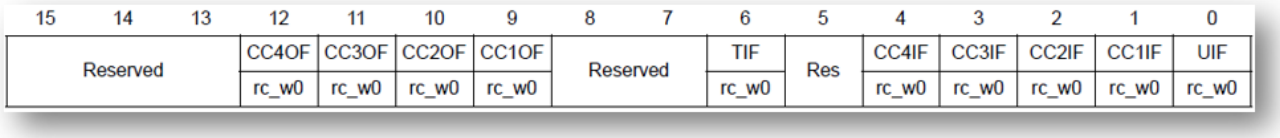
- **0**: UEV включен. Событие по изменению(UEV) генерируются следующими событиями:
  - Переполнение счетчика или установлен UG бит
- **1**: UEV отключен.

Bit 0 CEN Включить счетчик

- **0**: Counter выключен
- **1**: Counter включен

Таймеры TIM2 и TIM5. Регистр статуса (SR)

Регистр SR хранит статусы запросов на прерывания



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Figure 2. Регистр SR

Bit0: UIF Флаг прерывания по событию обновления. Бит устанавливается аппаратно, скидываться должен программно

- **0**: Флаг прерывания сбршен
- **1**: Флаг прерывания установлен

Работа с таймером в качестве счетчика

Для организации задержки

- Подать тактирование на модуль таймера
- Установить делитель частоты для таймера в регистре **PSC**
- Установить источник генерации прерываний по событию переполнение с помощью бита **URS** в регистре **CR1**
- Установить значение до которого счетчик будет считать в регистре перезагрузке **ARR**
- Скинуть флаг генерации прерывания **UIF** по событию в регистре **SR**
- Установить начальное значение счетчика в 0 в регистре **CNT**
- Запустить счетчик с помощью бита **EN** в регистре **CR1**
- Проверять пока не будет установлен флаг генерации прерывания по событию **UIF** в регистре **SR**
- Как только флаг установлен остановить счетчик, сбросить бит **EN** в регистре **CR1**, Сбросить флаг генерации прерывания **UIF** по событию в регистре **SR**

Задание 2. Простое

1. Написать программу морганиями всеми 4 светодиодами на плате с периодом в 0.5 секунды
2. По нажатию кнопки увеличивать период моргания на 0.1 секунды.