## Принцип действия таймера

Таймер – это отдельный модуль на плате, который может использоваться в различных целях. На вход таймера поступает сигнал с генератора тактовой частоты платы. Таймер занимается только тем, что считает количество тактов и совершает различные действия, связанные с этим счётом (сравнение, сброс счётчика, генерация прерываний в зависимости от состояния счётчика и т.д.). Вообще, на плате несколько таймеров, но мы будем работать пока что только с одним – с первым (TIM1). Этому таймеру посвящён отдельный раздел в документации на контроллер (16. Advanced-control timers (TIM1/TIM8)). Для детального изучения принципов работы таймера рекомендуется обратиться к данной документации.

Количество посчитанных тактов сохраняется в регистре таймера – TIM1\_CNT (counter). В таймере есть регистр, в котором задаётся верхний предел счёта (TIM1\_ARR – auto-reload register), после преодоления которого счётчик (значение регистра TIM1\_CNT) сбрасывается и счёт начинается заново. На сброс счётчика уходит один такт.

График изменения счётчика приведён на рисунке №1

|  |
| --- |
|  |
| Рисунок №1 – Изменение счётчика во времени |

На рисунке задано значение TIM1\_ARR равное 32. Сброс таймера виден на рисунке не чётко, но смысл таков, что на такте 32 значение счётчика равно 32, а нулю оно будет равно только на такте 33. И дальше счёт продолжается как обычно. Событие обновления счётчика называется – update. Когда происходит это событие, таймер может генерировать прерывание, чем мы можем пользоваться.

Нужные нам регистры таймера рассмотрим в разделе, посвященном его настройке.

В данной работе нам нужно настроить таймер, так, чтобы он выдавал прерывания раз в секунду. По этому прерыванию будем менять состояние светодиода, настроенного в прошлом разделе.

## Настройка таймера

Первым делом настроим прерывания таймера (разумеется, предварительно включив его с помощью регистра RCC\_ APB2ENR). Чтобы разрешить прерывания от какого-либо модуля существует регистр NVIC\_ISER. Он описан в руководстве по программированию микроконтроллера. Нужно поместить единицу в бит, который отвечает за прерывание по обновлению нашего таймера. Для того, чтобы выяснить номер этого бита, необходимо обратиться к документации на микроконтроллер, где в разделе «Interrupts and events» есть таблица со всеми прерываниями. Находим там «TIM1\_UP», который находится на позиции 25. Это означает, что нам нужно выставить в бите с индексом 25 регистра NVIC\_ISER единицу, и тогда прерывания от таймера будут разрешены. Так как этот регистр большой, он разбит на три части по 32 бита. Очевидно, что 25 бит находится внутри нулевой части этого регистра (нумерация с нуля!), тогда код будет выглядеть:

NVIC->ISER[0] |= (1 << 25); //Разрешили прерывания таймера 1

Но на этом настройка прерываний не заканчивается. Необходимо указать в регистре самого таймера, какие прерывания он должен генерировать. Для настройки прерываний таймера 1 существует специальный регистр (TIM1\_DIER). Дело в том, что таймер может генерировать много различных прерываний, но нам нужно только прерывание по сигналу обновления таймера. В описании регистра можно найти, что за прерывания по обновлению таймера отвечает бит UIE (update interrupt enable), который находится в нулевой позиции. В него то нам и нужно записать единицу.

Осталось настроить частоту, с которой будет обновляться таймер. Для этого воспользуемся регистром авто-обновления счётчика (TIM1\_ARR - auto-reload register), который позволяет задать верхний предел счёта и регистром делителя (TIM1\_PSC - prescaler) который понижает частоту входящего сигнала тактовой частоты. Обратим внимание, что оба эти регистра состоят лишь из 16 бит, а значит в них можно поместить число не больше чем 216 – 1 = 65535. Так же заметим, что таймер делит частоту не на само значение регистра TIM1\_PSC, а автоматически прибавляет к нему единицу (так никогда не произойдёт деления на нуль, например). Отметим так же, что на обновление таймера уходит один такт, после чего и будет выставлено прерывание, а значит в регистре верхнего предела счёта нужно выставлять число на единицу меньше желаемого.

По умолчанию генератор тактовой частоты подаёт на таймер сигнал с частотой 8МГц. Чтобы получить сигнал прерываний по обновлению таймера с частотой 1 Гц надо понизить частоту в 8 000 000 раз. Сделать это можно различными способами. Один из вариантов: выставить верхний предел счёта равным 8000, а частоту исходного сигнала делить на 1000. Код будет выглядеть так:

TIM1->ARR = 7999; //Задали верхний предел счёта счётчика

TIM1->PSC = 999; //Задали делитель входной частоты

Если тактовая частота будет другой, то и числа подбирать нужно будет другие.

Осталось запустить таймер. Для управления таймером в нем есть специальные регистры CR (control register). За включение таймера отвечает бит CEN (count enable) внутри регистра CR. В нем надо выставить единицу. После этого таймер начнёт выполнять свою работу.

Прерывание таймера отображается в регистре SR (status register) таймера. За прерывание по обновлению отвечает бит UIF (update interrupt flag). Если в этом бите 1, значит прерывание действительно (таймер досчитал до нужного предела и сообщил об этом), иначе – нет.

Чтобы по прерыванию выполнять какую-то работу, можно, например, в вечном цикле проверять состояние бита UIF и, если он равен единице, делать необходимую работу и сбрасывать значение этого бита (обратим внимание, что доступ на запись этого бита у нас имеется). Однако, есть более правильное решение. Можно написать специальную функцию – обработчик прерывания. Эта функция должна быть объявлена в специальном файле startup\_stm32.s. В этом файле содержится таблица функций-обработчиков прерываний. Нужно сориентироваться в этой таблице. Для начала найти строчку:

«.word SysTick\_Handler»

Эта строчка соответствует прерыванию SysTick (смотри таблицу прерываний в разделе Interrupts and events в документации на контроллер). После этого прерывания начинаются прерывания, имеющие индексы. Помним, что наше прерывание имеет индекс 25. Отсчитываем в таблице нужное количество строк (помня, что нумерация в таблице прерываний идёт с нуля) и в нужной строке помещаем объявление нашей функции. Пусть она будет называться Tim1\_Handler, тогда строка в которой объявляется наша функция будет иметь вид:

«.word Tim1\_Handler»

Теперь можно написать реализацию нашей функции-обработчика прерываний. Сделать это можно в файле main.c, а лучше в специально созданном файле, который будет содержать все созданные нами функции (писать весь код внутри функции main – дурной тон).

Итак, функция-обработчик прерываний будет иметь примерно такой вид:

void Tim1\_Handler()

{

GPIOE->ODR ^= (1 << 15); //Зажигаем светодиод или гасим, в зависимости от того...

TIM1->SR &= ~1;

}

Здесь первая команда будет менять состояние светодиода, а вторая – сбрасывает статус прерывания.

Если вынести весь функционал по настройке таймера в отдельную функцию, получится что-то подобное:

void setupTimer()

{

NVIC->ISER[0] |= (1 << 25); // Разрешили прерывания по update

TIM1->DIER |= 1; //Разрешили прерывания по update внутри таймера

TIM1->ARR = 7999; //Задали верхний предел счёта счётчика

TIM1->PSC = 999; //Задали делитель

TIM1->CR1 |= 1; //Начали считать

}

Для того, чтобы светодиод загорался, необходимо, чтобы была проведена его настройка, которая описана в соответствующем разделе.