Таймеры общего назначения. Переполнение

В МК **stm32f103c8** имеется 3 таймера общего назначения (`TIM1` , `TIM2` , `TIM3`) и один продвинутый (`TIM1`). Согласно схеме тактирования продвинутый таймер `TIM1` работает на шине APB2, частота которой равна частоте системной шины SYSCLK. Базовые таймеры `TIM2` , `TIM3` и `TIM4` работают от шины APB1, частоту которой нам пришлось поделить на два, т.е. на 32 МГц. Это нужно учитывать, при работе с данными таймерами.

Разрядность таймеров — 16 бит, они они могут считать от 0 до 65535. Рассчитаем максимальное время, которое можно ими отмерить:

$$\frac{65536}{32\ \mathrm{MHz}} \approx 0,002\ \mathrm{sec}$$

Не очень много, но в отличии от системного таймера SysTick на входе предусмотрен специальный блок, называемый предделителем. Он позволяет понизить частоту конкретного таймера, т.е. изменить его временное разрешение. Делается это через регистр `PSC` , в котором значимыми являются только первые 16 бит, т.е. он может хранить в себе всё тоже значение от 0 до 65535. Здесь стоит учитывать, что нумерация ведётся с нуля и следовательно от желаемого числа нужно отнять единицу. Для простоты поделим частоту APB1 на `32000 - 1` и посчитаем максимальное время.

$$\frac{65536 \cdot 32000}{32\ \mathrm{MHz}} = 65,536\ \mathrm{sec}$$

Уже намного лучше, чем тот же SysTick. Говоря о счёте, стоит заметить, что таймер может считать как снизу вверх, так и сверху вниз, а может и вовсе считать ёлочкой — сначала снизу вверх, а затем сверху вниз.

Сводная таблица имеется в документации.

Таймер	Разрядность	Предделитель	Счёт	DMA	Каналы	Компл. выход
`TIM1`	16 бит	1 ... 65536	Вверх, вниз, вверх-вниз	Да	4	Да
`TIM2` , `TIM2` , `TIM3`	16 бит	1 ... 65536	Вверх, вниз, вверх-вниз	Да	4	Нет

Таблица 4, datasheet

У таймеров общего назначения довольно много регистров, аж 18 штук. Рассматривать все их мы не будем, остановимся лишь на тех, что пригодятся нам для написания задержки, т.е. вызова прерывания по переполнению счётчика.

Основные настройки таймера находятся в регистрах `CR1` и `CR2` . Нам пригодится только один из них — `CR1` , бит `CEN` . Записав в него `0` таймер будет остановлен, записав туда `1` таймер будет включён. Трогать данный бит до полной настройки таймера не стоит.

Значение счётчика хранится в `CNT`. По умолчанию счётчик будет считать от 0 до значения записанного в регистре `ARR`. Чтобы включить прерывание по переполнению необходимо воспользоваться регистром `DIER`, который отвечает за прерывания и запросы к модулю DMA. Событие переполнения на английском называется update. Что бы его разрешить, необходимо записать `1` в бит `UIE`.

Последний регистр, который нас интересует, это регистр статуса `SR`. Так как обработчик прерывания всего один, а прерываний может быть несколько, по разным событиям, то как и в случае с модулем EXTI нужно вручную сбрасывать флаг произошедшего события.

```
// int main() {
RCC->APB1ENR |= RCC_APB1ENR_TIM4EN;

TIM4->PSC = (SystemCoreClock / 2) / 1000 - 1; // 63999
TIM4->ARR = 1000 - 1;                         // 1 секунда
TIM4->DIER |= TIM_DIER_UIE;                   // разрешаем прерывание по переполнению
NVIC_EnableIRQ(TIM4_IRQn);                    // глобально разрешаем прерывание

TIM4->CR1 |= TIM_CR1_CEN;                     // запускаем таймер
```

Обработчик прерывания нам в будущем не понадобиться, но для теста, давайте его напишем и помигаем светодиодом.

```
void TIM4_IRQHandler(void) {
    led_toggle();
    TIM4->SR &= ~TIM_SR_UIF; // сбрасываем прерывание
}
```

Настройка со стандартной библиотекой периферии.

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

TIM_TimeBaseInitTypeDef tim_base;

TIM_TimeBaseStructInit(&tim_base);
tim_base.TIM_Prescaler = (SystemCoreClock / 2) / 1000 - 1; // 31999
tim_base.TIM_Period = 1000 - 1;                            // 1 sec
TIM_TimeBaseInit(TIM4, &tim_base);
TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE);
TIM_Cmd(TIM4, ENABLE);

NVIC_InitTypeDef nvic;
nvic.NVIC_IRQChannel = TIM4_IRQn;
nvic.NVIC_IRQChannelPreemptionPriority = 0;
nvic.NVIC_IRQChannelSubPriority = 0;
nvic.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&nvic);

// TIM4_IRQHandler
TIM_ClearITPendingBit(TIM_IT_Update);
```

Код на можно найти на github: **CMSIS**.

Назад | Оглавление | Дальше