**Лабораторная работа №3**

**Таймеры. Настройка тактирования микроконтроллера**

1. **Настройка тактирования**

Для настройки тактирования необходимо сгенерировать С-файл конфигурации при помощи специального excel файла – «STM32F4xx\_Clock\_Configuration\_V1.1.0.xls». Этот файл специально разработан производителями микроконтроллера для удобной и интерактивной настройки системы тактирования. Для начала работы необходимо запустить этот файл. Убедитесь в том, что в Exсel включены макросы, без них файл работать не будет. Вид открывшегося окна показан на рисунке 1.

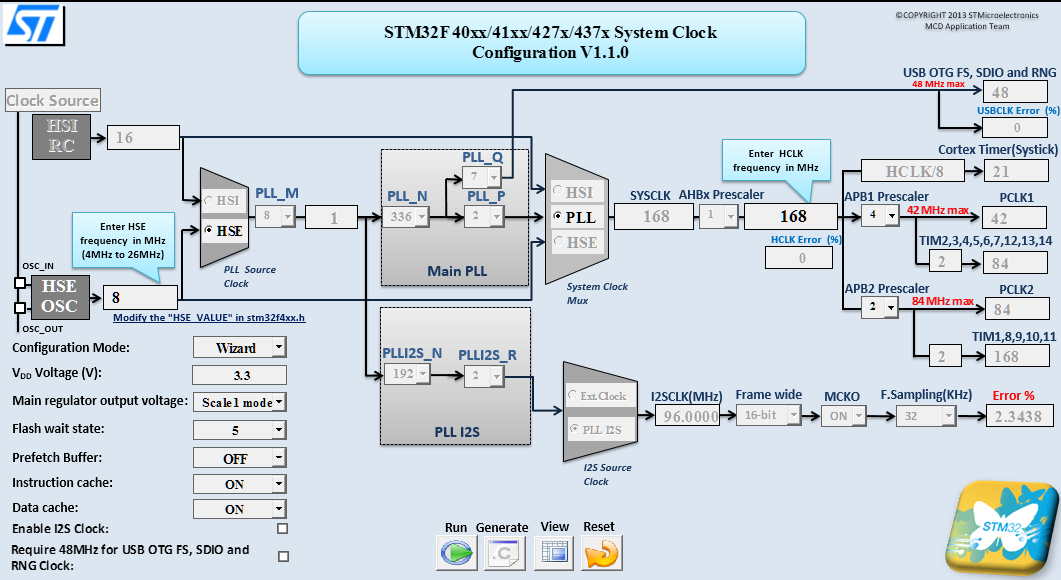


Рисунок 1 – Настройка тактирования микроконтроллера STM32F407VG

Микроконтроллер может тактироваться тремя основными источниками тактирования:

1. HSI RC – внутренний высокочастотный RC генератор, представляет собой RC-генератор с частотой 16 МГц. Находится внутри микроконтроллера и калибруется на заводе, гарантируется точность выходной частоты в 1% при температуре 25 °С. Недостатком является то, что частота может изменяться в зависимости от температуры, а также то, что частоту нельзя поменять. Преимуществом является то, что HSI находится внутри микроконтроллера, следовательно, нет необходимости усложнять схему устройства, а также то, что внутренний генератор имеет малое энергопотребление.
2. HSE – внешний высокочастотный генератор, подключается к выводам PH0-OSC\_IN и PH1-OSC\_OUT (Рисунок 2), может представлять собой как внешний тактовый генератор, так и обычный кварцевый или керамический резонатор. Недостатком является усложнение схемы устройства и повышенное энергопотребление. Преимущество же заключается в повышенной стабильности частоты тактирования и в возможности смены частоты тактирования путём смены резонатора.

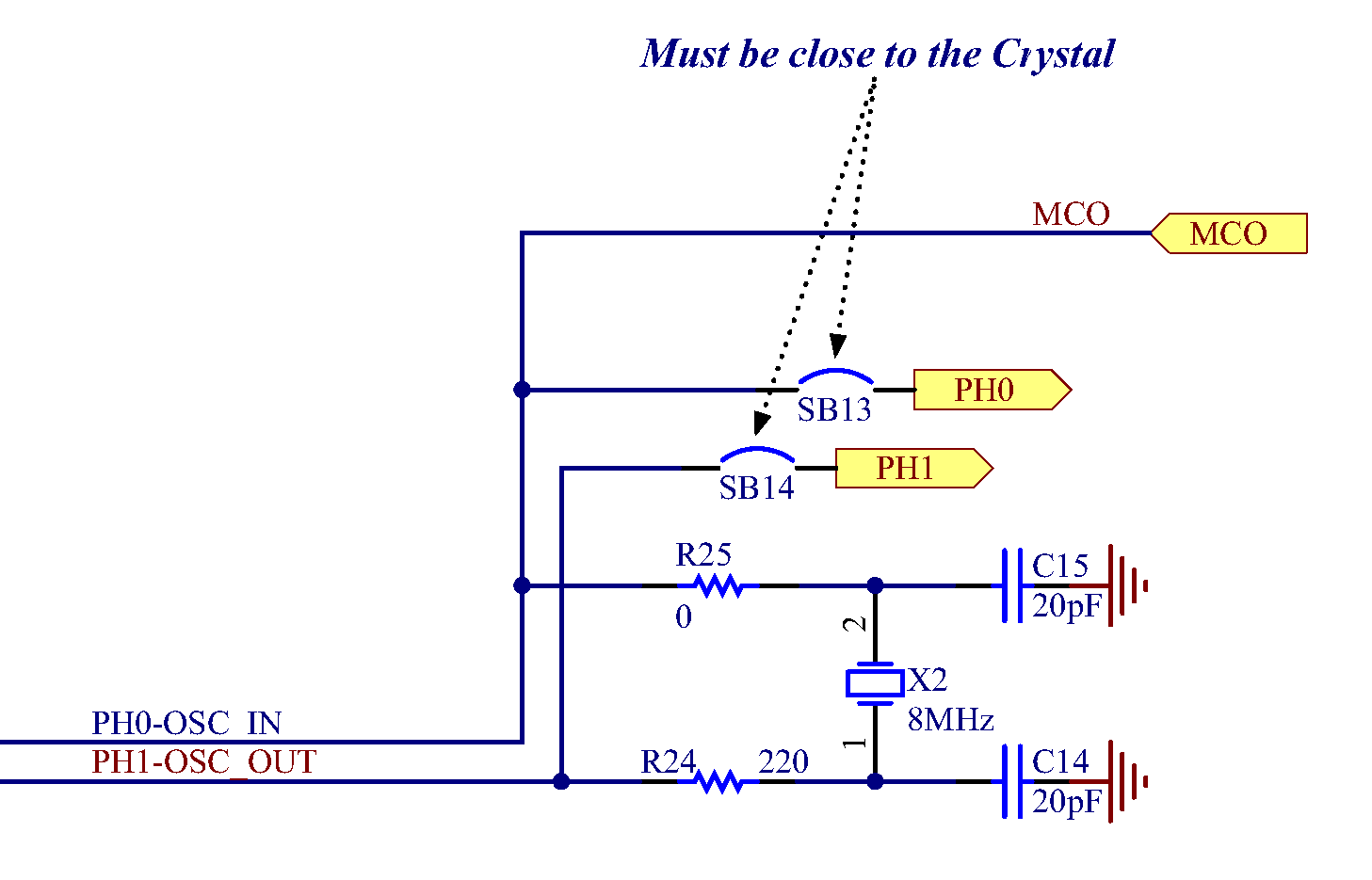


Рисунок 2 – Схема подключения внешнего кварцевого резонатора на 8 МГц к микроконтроллеру на отладочной плате STM32F4Discovery

1. PLL – система ФАПЧ (фазовая автоподстройка частоты), производит умножение опорного сигнала (частота входного сигнала должна лежать в пределах от 2 до 24 МГц) с заданным коэффициентом. На вход PLL обычно подаётся сигнал с HSI или HSE.

Выполнение пользовательской программы происходит с тактовой частотой микроконтроллера SYSCLK. Периферия микроконтроллера тактируется SYSCLK после предделителя частоты APBx Prescaler, после чего для шин APB1 и APB2 предделителями APB1 Prescaler и APB2 Prescaler соответственно. К шине APB1 относятся таймеры 2, 3, 4, 5, 6, 7, 12, 13, 14. К шине ABP2 относятся таймеры 1, 8, 9, 10, 11. Максимальная частота шины APB1 – 42 МГц. Максимальная частота шины APB2 – 84 МГц. Частота тактирования таймеров в два раза выше частоты шины.

Для примера настроим микроконтроллер на частоту тактирования 168 МГц, используя внешний осциллятор HSE с частотой 8 МГц. Для этого:

1. Установим частоту внешнего осциллятора 8 МГц (Рисунок 3);

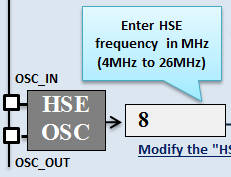


Рисунок 3 – Установка частоты внешнего осциллятора

1. Установим требуемую частоту HCLK 168 МГц (Рисунок 4);

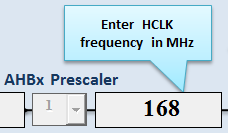


Рисунок 4 – Установка требуемой частоты микроконтроллера

1. Чтобы файл произвёл настройку, нажмите кнопку Run . В появившемся окне (Рисунок 5) необходимо выбрать источник тактирования для PLL, в данном случае HSE;

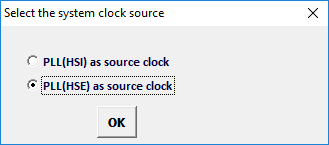


Рисунок 5 – Выбор источника тактирования PLL

1. Настроить предделители на шины APB1 и APB2 (Рисунок 6). В данном случае изменять нет необходимости.

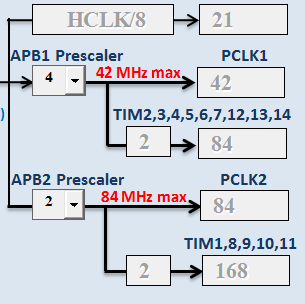


Рисунок 6 – Настройка предделителей на шины APB1 и APB2

1. Сгенерировать исходный файл, нажав кнопку Generate . Если настройка выполнена правильно появится окно «File successfully generated». Теперь в папке с Excel файлом должен появиться файл system\_stm32f4xx.c.
2. Добавить исходный файл в проект. Для этого в папке проекта (папка проекта в CooCox находится в папке установленной среды разработки workspace, т.е. {путь к CoIDE}\workspace\{название проекта}) в папке cmsis\_boot необходимо заменить файл system\_stm32f4xx.c нами сгенерированным файлом.
3. В исходном файле main.c необходимо подключить заголовочный файл system\_stm32f4xx.h:

**#include** "system\_stm32f4xx.h"

1. В функции main необходимо вызвать функцию SystemInit(), в которой и происходит установка в регистры микроконтроллера необходимых значений для переключения в заданный режим тактирования:

int main(void)

{

SystemInit();

while(1)

{

}

}

1. **Таймеры**

В микроконтроллере STM32F407VG всего имеется 14 таймеров. Из них:

– 6-й и 7-й базовые таймеры (Basic timers). Самый простой таймер, который умеет только генерировать прерывания в заданный промежуток времени, но при этом очень легко настраивается и управляется.

– 2, 3, 4, 5, 9, 10, 11, 12, 13, 14, 15 таймеры общего назначения (General-purpose timers). Более продвинутый таймер, позволяющий генерировать ШИМ, считывать состояние ног, обрабатывать данные от энкодера и т.д.

– 1-й и 8-й таймеры продвинутого управления (advanced-control timers). Самый продвинутый таймер, может использоваться как трехфазный ШИМ генератор.

В таблице 1 указаны параметры таймеров микроконтроллера.

Таблица – Таймеры в микроконтроллере STM32F407VG

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тип таймера | Таймер | Разрешение счётчика | Направление счёта | Значение пред-делителя | Генерация запроса DMA | Каналы захвата/сравнения | Комплементарные выходы | Max  частота шины  (МГц) | Max  частота таймера  (МГц) |
| Advanced-Control | TIM1, TIM8 | 16-bit | Up, Down,  Up/Down | Любое число от 1 до 65536 | Да | 4 | Да | 84 | 168 |
| General-Purpose | TIM2, TIM5 | 32-bit | Up, Down,  Up/Down | Любое число от 1 до 65536 | Да | 4 | Нет | 42 | 84 |
| TIM3, TIM4 | 16-bit | Up, Down,  Up/Down | Любое число от 1 до 65536 | Да | 4 | Нет | 42 | 84 |
| TIM9 | 16-bit | Up | Любое число от 1 до 65536 | Нет | 2 | Нет | 84 | 168 |
| TIM10, TIM11 | 16-bit | Up | Любое число от 1 до 65536 | Нет | 1 | Нет | 84 | 168 |
| TIM12 | 16-bit | Up | Любое число от 1 до 65536 | Нет | 2 | Нет | 42 | 84 |
| TIM13, TIM14 | 16-bit | Up | Любое число от 1 до 65536 | Нет | 1 | Нет | 42 | 84 |
| Basic | TIM6, TIM7 | 16-bit | Up | Любое число от 1 до 65536 | Да | 0 | Нет | 42 | 84 |

Таймеры общего назначения имеют 4 независимых канала, которые могут использоваться для:

* + Захвата сигнала (input capture)
  + Сравнения (output compare)
  + Генерации ШИМ (выровненного по границе или по центру)
  + Генерации одиночного импульса

Так же таймеры имеют следующие возможности:

* + Схемы синхронизации для управления таймерами при помощи внешних сигналов и для соединения нескольких таймеров друг с другом;
  + Комплементарные выходы с программируемым dead-time (Рисунок 7);

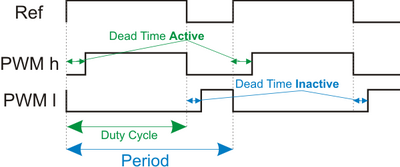


Рисунок 7 – Пример комприментарных выходов ШИМ с программируемым dead-time

* + Счётчик повторений;
  + Вход BRK для сброса выходов таймера или выставления их в известное состояние;
  + Поддерживают инкрементальные (квадратурные) энкодеры и датчики Холла;

Возможна генерация прерывания (список обработчиков прерываний от таймера приведён в приложении 1) или запроса DMA (прямой доступ к памяти) по следующим событиям:

* + Обновление: переполнение счётчикa
  + Захват сигнала
  + Сравнение
  + Событие-триггер: старт, остановка, инициализация счётчика или его обновление внутренним или внешним триггером.

Все эти возможности реализуют таймеры продвинутого управления TIM1 и TIM8, структурная схема которых приведена на рисунке 8.

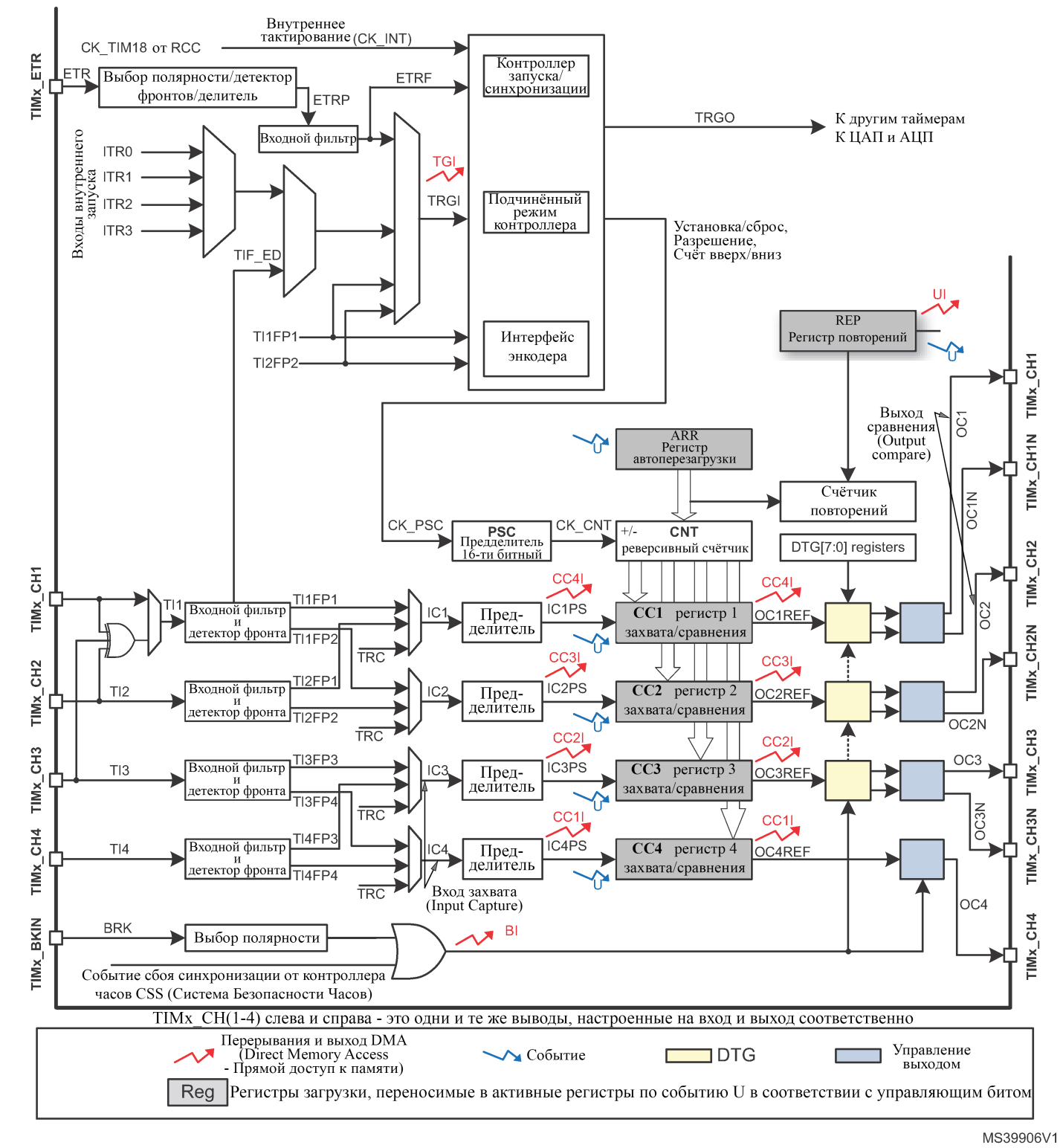


Рисунок 8 – Структурная схема таймеров продвинутого управления микроконтроллера STM32F407

Сердцем таймера является регистр-счётчик CNT, который, в зависимости от настройки, инкрементируется (увеличивает своё состояние на 1-цу), либо декрементируется (уменьшает своё состояние на 1-цу) в момент изменения тактового сигнала CK\_CNT с нуля на единицу (Рисунок 9). Когда значение регистра-счётчика CNT становится равным значению регистра автоперезагрузки ARR, происходит перезагрузка, и CNT устанавливается в 0. Если же значения регистра-счётчика декрементируется, то перезагрузка происходит, когда значение CNT достигает нуля, и СNT устанавливается в значение, которое хранится в ARR.

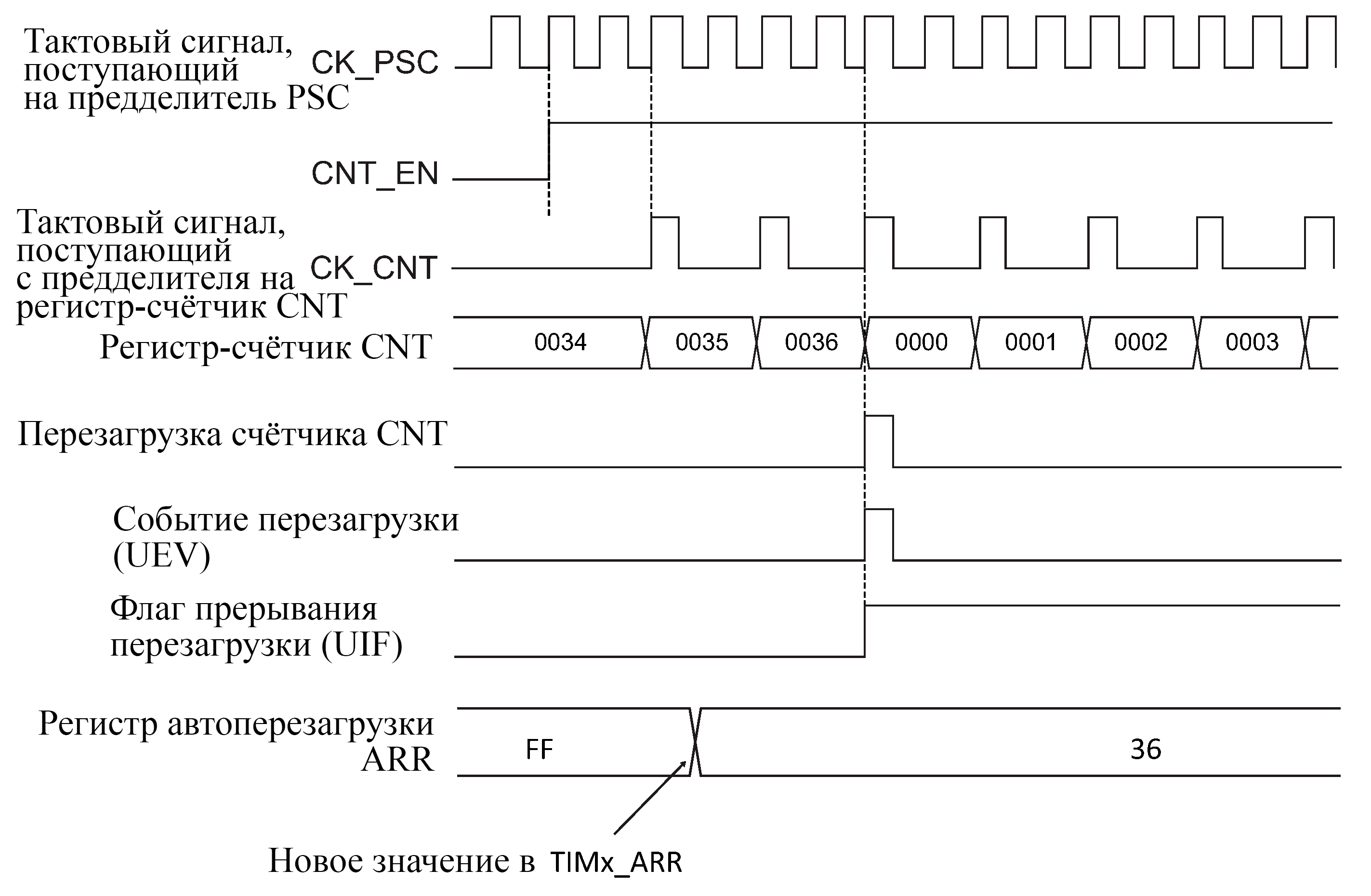


Рисунок 9 – Пример перезагрузки регистра-счётчика CNT, значение предделителя PRS = 1 (деление входной частоты происходит на велечину PRS + 1)

**Режим захвата сигнала.** В этом режиме с выбранного канала захватываются импульсы, на каждый из них текущее значение счётчика таймера кладётся в регистр TIM\_CCRx, где x — номер канала. Таким образом, период следования импульсов равен разнице между текущим значением TIM\_CCRx и предыдущим. Ну а для того, чтобы получить период в каких-то внятных единицах измерения, нужно правильно настроить предделитель.

При этом можно настроить генерацию прерывания и запроса DMA на приход очередного импульса, и если в это время предыдущее значение TIM\_CCRx не было считано, будет сгенерировано так называемое прерывание over-capture, т.е. сигнал о том, что предыдущее значение потерялось.

Захватывать можно фронты, спады или и то, и другое вместе. Есть настройка так называемого фильтра — числа выборок, после которого переход уровня будет считаться состоявшимся (полезно для устранения дребезга). Значение фильтра может принимать значения от 0 (фильтр выключен) до 15 (0xF). Также настраивается делитель входной частоты — 2, 4 или 8: будет ловиться каждый 2й, 4й или 8й импульс соответственно.

**Режим сравнения**. Тут просто подключаем какой-нибудь канал таймера к соответствующему выводу, и как только таймер досчитает до определенного значения (оно в TIM\_CCR) состояние вывода изменится в зависимости от настройки режима (либо выставится в единицу, либо в ноль, либо изменится на противоположное).

**Режим генерации ШИМ**. Генерируется ШИМ (Широтно-импульсная модуляция, по английски PWM – pulse-width modulation) – это способ управления подачей мощности к нагрузке. Управление заключается в изменении длительности импульса при постоянной частоте следования импульсов. Можно провести аналогию с механикой. Если при помощи двигателя вращать тяжелый маховик, то поскольку двигатель может быть либо включен, либо выключен, то и маховик будет либо раскручиваться и продолжать вращаться, либо станет останавливаться из-за трения, когда двигатель выключен.

Но если двигатель включать на несколько секунд в минуту, то вращение маховика будет поддерживаться, благодаря инерции, на некоторой скорости. И чем дольше продолжительность включения двигателя, тем до более высокой скорости раскрутится маховик. Так и с ШИМ, на выход приходит сигнал включений и выключений (0 и 1), и в результате достигается среднее значение. Проинтегрировав напряжение импульсов по времени, получим площадь под импульсами, и эффект на рабочем органе будет тождественен работе при среднем значении напряжения.

Отношение полной длительности периода импульса ко времени включения (положительной части импульса) называется скважностью импульса. Так, если время включения составляет 10 мкс, а период длится 100 мкс, то при частоте в 10 кГц, скважность будет равна 10, и пишут, что S = 10. Величина обратная скважности называется коэффициентом заполнения импульса, по-английски Duty cycle, или сокращенно DC (Рисунок 7).

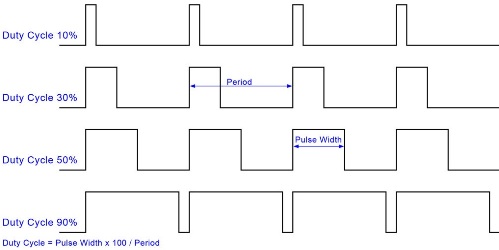


Рисунок 7 – ШИМ с различным коэффициентом заполнения импульсов

Так, для приведенного примера DC = 0.1, поскольку 10/100 = 0,1. При широтно-импульсной модуляции, регулируя скважность импульса, то есть варьируя DC, добиваются требуемого среднего значения на выходе электронного или другого электротехнического устройства, например двигателя.

Для инициализации таймера первым делом необходимо включить тактирование этого таймера:

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE);

Как вы, наверное, заметили, название функции немного отличается от того, которую мы использовали при инициализации GPIO. Отличается наименованием шины: вместо AHB (Advanced High-performance Bus или Расширенная Высокопроизводительная Шина) используется APB (Advanced Peripheral Bus, или Расширенная Шина Периферии).

Следующим шагом будет объявление структуры инициализации:

TIM\_TimeBaseInitTypeDef TIM\_InitStruct;

Далее необходимо проинициализировать параметры структуры значениями по умолчанию. Для этого пишем:

TIM\_TimeBaseStructInit(&TIM\_InitStruct);

**Приложение 1**

Список обработчиков прерываний таймеров микроконтроллера STM32F407

|  |  |
| --- | --- |
| TIM1\_BRK\_TIM9\_IRQHandler() | Обработчик прерываний перерыва TIM1 и всех прерываний TIM9 |
| TIM1\_UP\_TIM10\_IRQHandler() | Обработчик прерываний перезагрузки TIM1 и всех прерываний TIM10 |
| TIM1\_TRG\_COM\_TIM11\_IRQHandler() | Обработчик прерываний триггера и подключения TIM1 и всех прерываний TIM11 |
| TIM1\_CC\_IRQHandler() | Обработчик прерываний захвата/сравнения TIM1 |
| TIM2\_IRQHandler() | Обработчик всех прерываний TIM2 |
| TIM3\_IRQHandler() | Обработчик всех прерываний TIM3 |
| TIM4\_IRQHandler() | Обработчик всех прерываний TIM4 |
| TIM5\_IRQHandler() | Обработчик всех прерываний TIM5 |
| TIM6\_DAC\_IRQHandler() | Обработчик всех прерываний TIM6 и прерываний ошибки отставания ЦАП1 и ЦАП2 |
| TIM7\_IRQHandler() | Обработчик всех прерываний TIM7 |
| TIM8\_BRK\_TIM12\_IRQHandler() | Обработчик прерываний перерыва TIM8 и всех прерываний TIM12 |
| TIM8\_UP\_TIM13\_IRQHandler() | Обработчик прерываний перезагрузки TIM8 и всех прерываний TIM13 |
| TIM8\_TRG\_COM\_TIM14\_IRQHandler() | Обработчик прерываний триггера и подключения TIM8 и всех прерываний TIM14 |
| TIM8\_CC\_IRQHandler() | Обработчик прерываний захвата/сравнения TIM8 |