life.augmented

AH4776

Примечание по применению

Поваренная книга таймеров общего назначения для микроконтроллеров STM32

Введение

Периферийный таймер является частью основного набора периферийных устройств, встроенных во все микроконтроллеры STM32. Количество периферийных устройств таймера и их соответствующие функции различаются от одного семейства микроконтроллеров STM32 к другому, но все они имеют некоторые общие функции и режимы работы.

Периферийный таймер STM32 был задуман как периферийное устройство Keystone для большого количества приложений: от приложений управления двигателем до приложений генерации периодических событий. Спецификации периферийного устройства таймера, доступные во всех справочных руководствах по STM32, очень широки из-за его универсальности.

Цель этих указаний по применению — предоставить простое и ясное описание основных функций и режимов работы периферийных устройств таймера общего назначения STM32. Этот документ дополняет спецификации периферийных устройств таймера STM32, доступные в их справочных руководствах.

Документ разделен на две основные части:

- В первом разделе в простой форме представлены основные функции таймеров STM32 и описаны некоторые специфические функции, которые обычно используются в приложениях на основе периферийных устройств с таймерами.
- Следующие разделы посвящены описанию конкретного варианта использования периферийного таймера STM32. В этих разделах содержится подробное описание основных функций таймера STM32, используемых для создания примера приложения. Они также описывают архитектуру используемого исходного кода.

Цель этих указаний по применению состоит в том, чтобы представить в общем и простом виде некоторые варианты использования периферийных устройств таймера STM32, и они не охватывают такие варианты использования, как приложения управления двигателем, из-за их сложности.

Список рисунков

1 Основные режимы работы универсальных таймеров STM32

1.1 Введение

Во все микроконтроллеры STM32 встроено по крайней мере одно периферийное устройство таймера, а в некоторые из них встроено более одного типа периферийного устройства таймера. Этот документ охватывает устройства общего назначения. Таймеры общего назначения можно отличить от других типов периферийных устройств таймеров STM32 по их имени.

В документации микроконтроллеров STM32 периферийное устройство таймера общего назначения всегда называется «таймер TIMx», где «x» может быть любым числом, и оно не отражает количество периферийных устройств таймера, встроенных в данный микроконтроллер. Например, в микроконтроллеры STM32F100 встроено периферийное устройство таймера с именем TIM17, но общее количество периферийных устройств таймера, встроенных в эти микроконтроллеры, меньше 17.

В общем, во всех семействах микроконтроллеров STM32 периферийные устройства таймера с одинаковыми именами также имеют одинаковый набор функций, но есть несколько исключений. Например, периферийное устройство таймера TIM1 является общим для серии STM32F1, серии STM32F2 и серии STM32F4, но для конкретного случая семейства микроконтроллеров STM32F30x периферийное устройство таймера TIM1 имеет немного более богатый набор функций, чем TIM1, присутствующий в других семействах.

Таймеры общего назначения, встроенные в микроконтроллеры STM32, имеют одинаковую магистральную структуру; они различаются только на уровне функций, встроенных в данное периферийное устройство таймера. Уровень интеграции функций для данного периферийного устройства таймера определяется в зависимости от области применения, для которой оно предназначено.

Периферийные устройства таймера можно разделить на:

- Таймеры расширенной конфигурации, такие как TIM1 и TIM8 среди прочих.
- Таймеры конфигурации общего назначения, такие как TIM2 и TIM3 среди прочих.
- Таймеры с упрощенной конфигурацией, такие как TIM9, TIM10, TIM12 и TIM16 среди прочих.
- Таймеры базовой конфигурации, такие как TIM6 и TIM7 среди прочих.

Замечание по применению *Обзор межсерийного таймера STM32*(AN4013) представляет подробный обзор периферийных устройств таймера STM32 в различных семействах микроконтроллеров STM32.

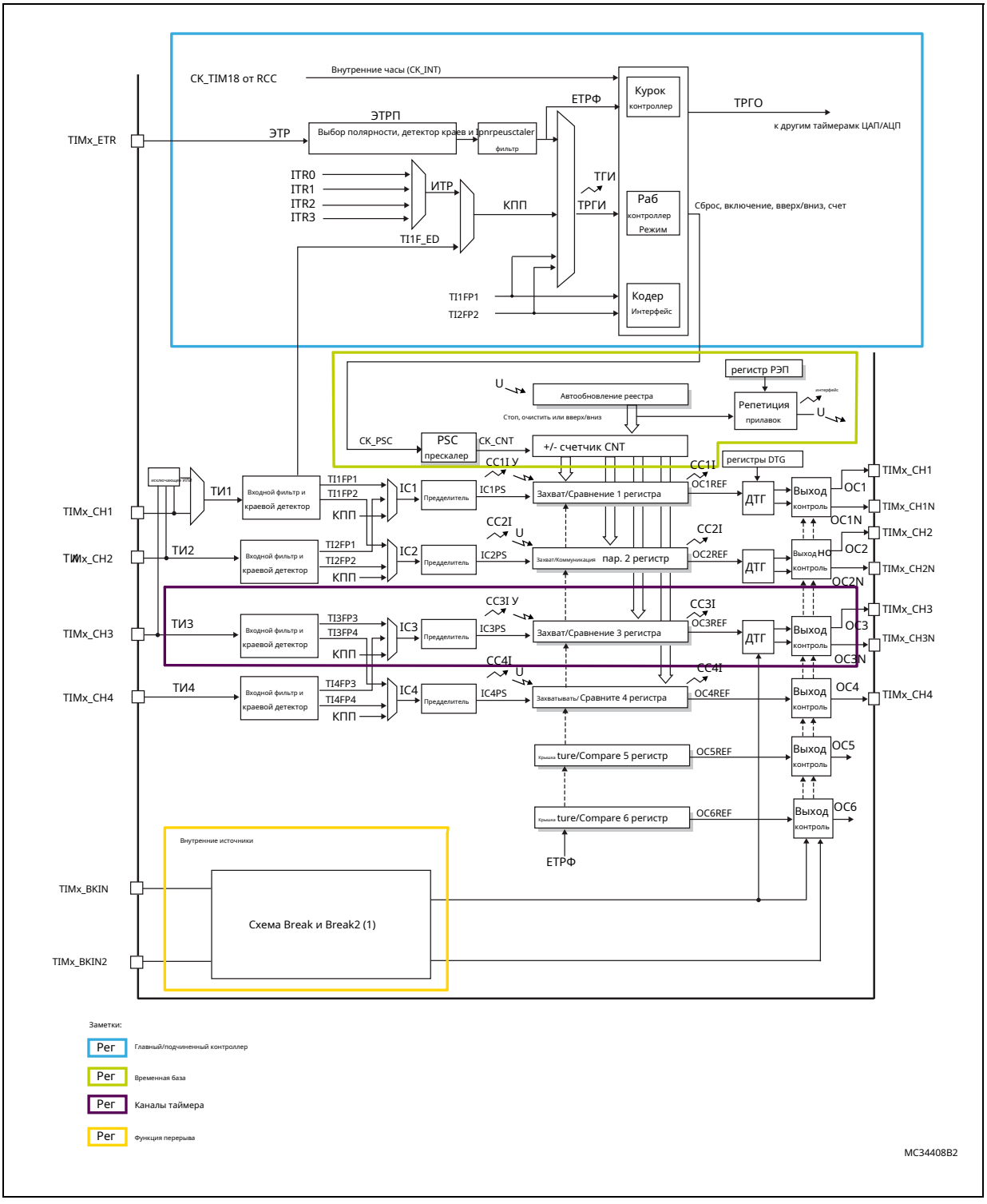
1,2 Разборка периферийного устройства таймера STM32

Все периферийные устройства таймера общего назначения STM32 имеют одинаковую магистральную структуру. В этом разделе рассматривается периферийное устройство таймера TIM1 с расширенной конфигурацией, которое является периферийным устройством таймера с наибольшим количеством функций.

*фигура 1*показывает блок-схему периферийного устройства таймера TIM1.
Периферийный таймер STM32 состоит из четырех блоков:

1. Главный/ведомый блок контроллера
2. Базовая единица времени
3. Блок таймерных каналов
4. Функциональный блок Break.

Рис. 1. Блок-схема таймер-периферия TIM1



1.2.1 Главный/подчиненный блок контроллера

Ведущий/подчиненный блок обеспечивает блок временной развертки тактовым сигналом счета (например, сигналом CK_PSC), а также сигналом управления направлением счета. Этот блок в основном обеспечивает сигналы управления для блока временной развертки.

Контроллер ведущий/ведомый выбирает правильную конфигурацию счета для единицы времени на основе конфигурации главного/ведомого таймера; он также определяет фактический статус подсчета.

Например, если таймер сконфигурирован в одном из режимов энкодера путем записи правильного значения в битовое поле управления SMS в регистре таймера TIMx_SMCR, то тактовый сигнал счета и сигнал управления направлением счета вычисляются на основе состояния входных сигналов TI1FP1 и TI2FP2.

Блок контроллера ведущий/ведомый выполняет синхронизацию между таймерами. Это устройство может быть сконфигурировано для вывода сигнала синхронизации (сигнал TRGO) рядом с определенным внутренним событием таймера. Его также можно настроить для управления счетчиком временной развертки в зависимости от внешних событий (например, внутренних событий других таймеров или внешних сигналов).

Можно настроить один подчиненный таймер для увеличения его счетчика на основе событий главного таймера, таких как событие обновления таймера. В этом примере событие ведущего таймера сигнализируется блоком контроллера ведущего/ведомого ведущего таймера. Этот блок управления использует выход главного таймера – сигнал TRGO. Выходной сигнал TRGO главного таймера соединен с входным сигналом TRGI подчиненного таймера. Блок контроллера ведущий/ведомый подчиненного таймера настроен на использование входного сигнала TRGI в качестве источника тактового сигнала для увеличения счетчика подчиненного таймера.

Не все периферийные устройства таймера STM32 имеют одинаковые возможности главного/ведомого контроллера. Периферийный таймер TIM1, взятый в качестве примера, реализует все возможности master/slave; в отличие от базовых таймеров TIM6 и TIM7, в которые встроен простейший контроллер ведущий/ведомый. Контроллер ведущий/ведомый в периферийных устройствах таймера TIM6 и TIM7 не имеет битового поля управления.

Для периферийных устройств таймера TIM6 и TIM7 счетчик времени всегда ведет обратный отсчет без возможности сброса его содержимого рядом с внешними событиями. Невозможно синхронизировать их ни с другим источником синхронизации, ни с внутренними часами периферийного таймера.

1.2.2 Базовая единица времени

Единицей базы времени является счетчик таймера в дополнение к каскаду предварительного делителя и счетчику повторений. Тактовый сигнал, подаваемый в блок временной развертки, сначала проходит через этап предварительного масштабирования, прежде чем попасть в счетчик временной развертки.

В зависимости от содержимого регистра предварительного делителя таймера TIMx_PSC частота сигнала счета может быть уменьшена до того, как он достигнет каскада счетчика. Выходной сигнал каскада предварительного масштабирования является сигналом тактового отсчета каскада таймера-счетчика.

Счетчик таймера управляется двумя регистрами таймера:

- Регистр таймера TIMx_CNT используется для чтения и записи содержимого счетчика таймера.
- Регистр таймера TIMx_ARR содержит значение перезагрузки счетчика таймера.
  - Если счетчик таймера ведет обратный отсчет и достигает содержимого регистра автоматической перезагрузки таймера (TIMx_ARR), то счетчик таймера сбрасывается и начинается новый цикл счета.
  - Если счетчик таймера ведет обратный отсчет и достигает нулевого значения, то значение счетчика таймера устанавливается равным содержимому регистра автоматической перезагрузки таймера (TIMx_ARR) и перезапускается новый цикл счета.

Каждый раз, когда перезапускается новый цикл счета, запускается «событие обновления» таймера до тех пор, пока содержимое счетчика повторений равно нулю. Если содержимое счетчика повторений не равно нулю, то «событие обновления» не инициируется, но перезапускается новый цикл подсчета, и содержимое счетчика повторений уменьшается на единицу. Рядом с каждым «событием обновления» содержимое счетчика повторений устанавливается равным значению, хранящемуся в регистре таймера TIMx_RCR.

Не все периферийные устройства таймера STM32 имеют встроенный счетчик повторений. Если счетчик повторений не встроен, то периферийное устройство таймера будет вести себя так, как будто счетчик повторений встроен, но его содержимое равно нулю.

1.2.3 Блок таймер-каналов

Каналы таймера являются рабочими элементами таймера; они являются средством, с помощью которого периферийное устройство таймера взаимодействует с внешней средой. Как правило, каналы таймера отображаются на выводы микроконтроллера STM32 за некоторыми исключениями, такими как каналы таймера 5 и 6 для периферийного устройства таймера TIM1 в семействе микроконтроллеров STM32F30x. Канал таймера, сопоставленный с выводом микроконтроллера STM32, может использоваться либо как вход, либо как выход.

Канал таймера настроен как выход

При настройке в качестве выхода канал таймера используется для генерации набора возможных сигналов. Пока канал сконфигурирован в режиме вывода, содержимое регистра канала TIMx_CCRy сравнивается с содержимым счетчика таймера.

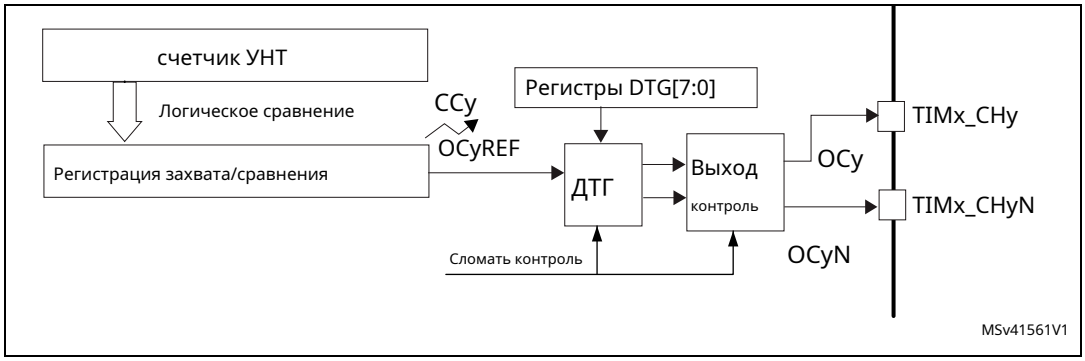
Основываясь на результатах этого непрерывного логического сравнения и на основе сконфигурированного выходного подрежима (например, режим PWM1 или неактивный режим), внутренний выход канала таймера OCyREF либо устанавливается, либо сбрасывается.

Затем внутренний выход канала таймера OCyREF подается на выходной каскад канала. Выходной каскад канала применяет набор операций кондиционирования сигнала OCyREF на основе набора сконфигурированных параметров (например, конфигурация полярности канала или генерация мертвого времени среди прочего).

Выходной сигнал выходного каскада канала отображается на выводы микроконтроллера в качестве альтернативной функции. Обратите внимание, что некоторые выходные каскады каналов таймера, как показано на рис.*фигура 2*, может выводить два дополнительных сигнала.

Управляющие битовые поля для выходного каскада такого канала предоставляют средства для конфигурирования каждого выходного сигнала отдельно (например, включенный/отключенный выходной сигнал или полярность).

Рисунок 2. Соответствующая блок-схема для канала таймера, сконфигурированного как выход



Канал таймера настроен как вход

Когда канал таймера сконфигурирован как вход, его можно использовать для временной отметки нарастающего и/или спадающего фронта внешних сигналов. Для обработки этой функции вход канала отображается на один из выводов микроконтроллера.

Некоторые входы канала таймера также отображаются на некоторые встроенные сигналы, например, на выход генератора для целей калибровки. Вход канала таймера TIy питает схему согласования входа канала, как показано на рис.*Рисунок 3*. Схема кондиционирования включает этап фильтрации и детектор фронта. Ступень фильтра отсеивает импульсы длительностью меньше заданной. Детектор фронта определяет, возник ли активный фронт на соответствующем входе таймера после фильтрации.

Конфигурация активного фронта устанавливается путем записи битовых полей управления полярностью канала в регистре таймера TIMx_CCER. Схема кондиционирования выдает два сигнала:

- TIyFPy: входной сигнал таймера TIy, который был отфильтрован и на котором обнаружен активный фронт в зависимости от полярности канала таймера «y».
- TIyFPz: всегда является входным сигналом таймера TIy, который был отфильтрован, но на котором обнаруживается активный фронт в зависимости от полярности канала таймера «z».

Сигнал TIyFPz перенаправляется на вход предварительного делителя канала «z», где сигнал TIzFPy перенаправляется на вход предварительного делителя канала «y», как показано на рис.*Рисунок 3*. Этот перекрестный обмен отфильтрованными входными сигналами очень полезен для маркировки времени как переднего, так и заднего фронта входного сигнала. Это очень полезно для реализации входных приложений PWM (широтно-импульсная модуляция).

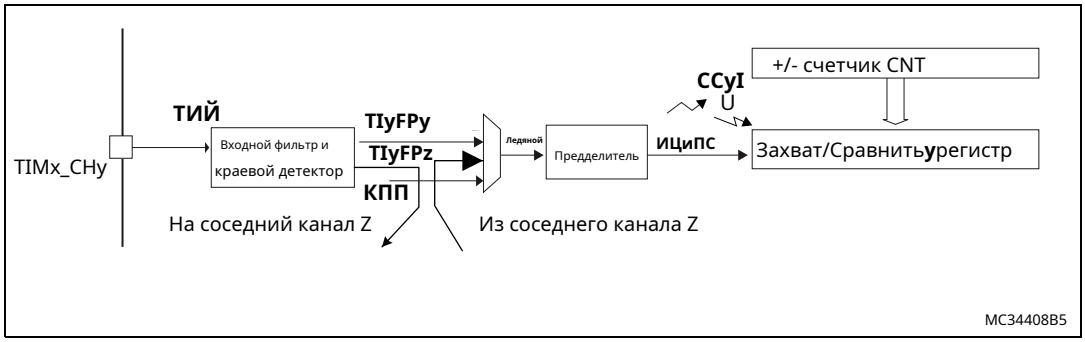
Каждый канал таймера может быть настроен на один из трех возможных входных режимов. Каждый входной режим соответствует одному из трех возможных входов мультиплексора прескалера, подключенного к прескалеру канала таймера. Битовое поле управления CCyS определяет, настроен ли канал таймера в режиме вывода (например, CCyS[1:0] = '00') или он настроен в одном из режимов ввода (например, CCyS[1:0] отличается от «00»).

Одни и те же регистры таймера TIMx_CCMRn («n» может быть любым числом, но обычно это либо 1, либо 2) используются для настройки каналов таймера либо как входных, либо как выходных. Некоторые управляющие битовые поля регистров таймера TIMx_CCMRn имеют различную интерпретацию в зависимости от конфигурации канала, режимов ввода или вывода.

Предварительный делитель канала таймера может быть сконфигурирован для уменьшения частоты активных фронтов, обнаруженных на входе TIy таймера. Обнаружение активного фронта на выходе предделителя канала запускает передачу содержимого счетчика таймера в регистр «y» канала таймера TIMx_CCRy.

Содержимое регистра «y» канала таймера TIMx_CCRy представляет собой метку времени последнего обнаруженного активного фронта на выходе предделителя канала «y». Это временная метка последнего обнаруженного активного фронта на входе таймера TIy, если предварительный делитель канала «y» сконфигурирован так, чтобы не уменьшать масштаб входного сигнала (например, когда коэффициент предварительного делителя = 1, другими словами, предварительный делитель канала обходится). .

Рисунок 3. Соответствующая блок-схема для канала таймера, сконфигурированного как вход



1.2.4 Функциональный блок Break

Функциональный блок Break встроен только в периферийные устройства таймера с дополнительными выходами. Другими словами, только периферийные устройства таймера, которые имеют хотя бы один канал с двумя дополнительными выходами, имеют встроенную функцию Break.

Функция Break воздействует на выходной каскад каналов таймера, сконфигурированных в режиме вывода. Как только на входе прерывания обнаруживается активный фронт, выходы каналов таймера, сконфигурированных в режиме вывода, либо выключаются, либо принудительно переводятся в предопределенное безопасное состояние.

Функция Break обычно используется для реализации функции безопасного отключения в инверторах электрической мощности при наличии аномалий.

Замечание по применению *Использование функций отключения ШИМ устройства STM32 для управления двигателем и цифрового преобразования мощности* (AN4277) содержит подробное описание функции Break для семейств микроконтроллеров STM32.

1,3 Периферийный таймер STM32, основные режимы работы

В этом разделе представлен набор фрагментов исходного кода для набора базовых конфигураций периферийного таймера STM32. Эти фрагменты были разработаны с использованием языка программирования C.

1.3.1 Конфигурация временной развертки таймера

```
# определить ANY_DELAY_RQUIRED 0x0FFF
/* Реализация цикла задержки с аппаратной точностью с использованием периферийного устройства таймера TIM6. Для выполнения этой функции можно использовать любой другой таймер STM32, но был выбран таймер TIM6, так как он имеет меньший уровень интеграции. Другие периферийные устройства таймера могут быть зарезервированы для более сложных задач */
/* Очистить флаг события обновления */
TIM6->SR = 0
/* Устанавливаем требуемую задержку */
/* Значение сброса предускорителя таймера равно 0. Если требуется более длительная задержка, регистр предупредителя может быть настроен на */
/*TIM6->PSC = 0 */ TIM6->ARR =
ANY_DELAY_RQUIRED /* Запуск
счетчика таймера */ TIM6->CR1 |=
TIM_CR1_CEN
```

```
/* Цикл, пока не будет установлен флаг события
обновления */ while (!(TIM6->SR & TIM_SR_UIF));
/* Требуемое время задержки истекло */ /*
Пользовательский код может быть выполнен */
```

1.3.2 Конфигурация канала таймера в режиме ввода

```
/* Переменная для хранения временной метки последнего обнаруженного активного
фронта */ uint32_t TimeStamp;
/* Значение сброса регистра ARR равно 0x0000FFFF для таймера TIM3. Так что для этого фрагмента
все должно быть в порядке */
/* Если вы хотите изменить его, раскомментируйте строку ниже */ /*
TIM3->ARR = ANY_VALUE_YOU_WANT */
/* Установите канал таймера 1 TIM3 в качестве входа */
/* Биты CC1S доступны для записи, только когда канал 1 выключен */ /*
После сброса все каналы таймера выключаются */ TIM3->CCMR1 |=
TIM_CCMR1_CC1S_0;
/* Включить канал TIM31 и сохранить конфигурацию по умолчанию (состояние после сброса)
для полярности канала */
TIM3->CCER |= TIM_CCER_CC1E; /*
Запуск счетчика таймера */ TIM3-
>CR1 |= TIM_CR1_CEN;
/* Сбросить флаг события Capture для канала 1 */ TIM3->SR =
~TIM_SR_CC1IF;
/* Цикл, пока не будет установлен флаг события захвата */
while (!(TIM3->SR & TIM_SR_CC1IF));
/* Обнаружено активное ребро, сохраняем отметку времени */
TimeStamp = TIM3->CCR1;
```

1.3.3 Конфигурация канала таймера в режиме вывода

```
/* Значение сброса регистра ARR равно 0x0000FFFF для таймера TIM3. Так что это должно быть
хорошо для этого фрагмента. Если вы хотите изменить его, раскомментируйте строку ниже */
/* TIM3->ARR = ANY_VALUE_YOU_WANT */
/* Канал 1 таймера TIM3 после сброса настраивается как выход */ /* Значение
сброса TIM3->CC1S равно 0 */
/* Чтобы выбрать режим вывода PWM2, установите битовое поле управления OC1M в '111'
*/ TIM3->CCMR1 |= TIM_CCMR1_OC1M_0 | TIM_CCMR1_OC1M_1 | TIM_CCMR1_OC1M_2; /*
Установите рабочий цикл на 50% */
TIM3->CCR1 = TIM3->ARR/2;
/* По умолчанию после сброса предварительная загрузка для канала 1 отключена */ /*
Чтобы изменить ее, раскомментируйте строку ниже */
/* TIM3->CCMR1 |= TIM_CCMR1_OC1PE;
/* Включить канал TIM31 и сохранить конфигурацию по умолчанию (состояние после сброса)
для полярности канала */
TIM3->CCER |= TIM_CCER_CC1E; /*
Запуск счетчика таймера */
```

TIM3->CR1 |= TIM_CR1_CEN

1,4 Дополнительные функции таймера STM32

В этом разделе представлено подробное описание некоторых общих функций таймера STM32, используемых в примерах приложений, подробно описанных далее в этом документе.

1.4.1 Стадия фильтрации

Входы таймера (такие как вход ETR или входы канала) имеют этап фильтрации, который можно активировать для фильтрации импульсов внешнего сигнала с длительностью меньше желаемого порога.

Максимальная длительность отфильтрованных импульсов зависит от двух параметров:

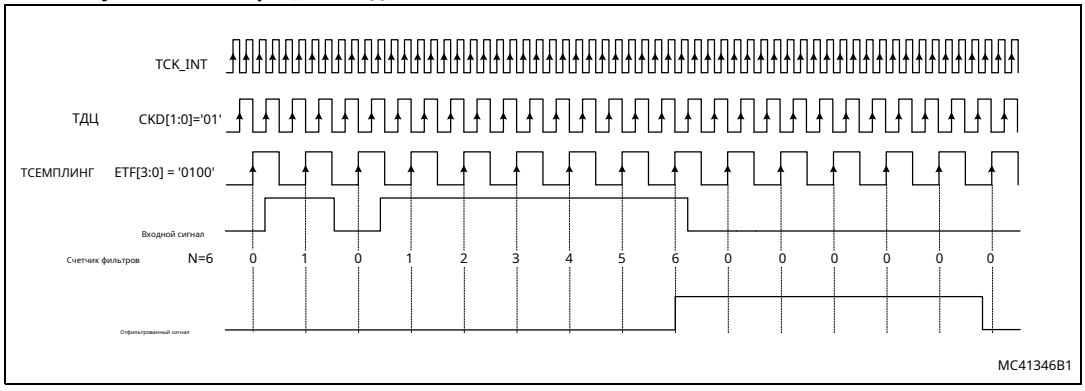
- Конфигурация этапа фильтрации в соответствии с определенным входом таймера. Например, этап фильтрации ввода ETR настраивается с помощью управляющего битового поля ETF[3:0] в регистре TIMx_SMCR. Конфигурация каскада входной фильтрации подразумевает выбор источника тактовой частоты дискретизации, установку тактовой частоты дискретизации и установку минимальной длительности действительного импульса в единицах тактов уже настроенного тактового генератора дискретизации.
- В случае, когда источник тактового сигнала FDTS используется в качестве источника тактового сигнала дискретизации, на минимальную допустимую длительность импульса этапа фильтрации влияет значение, записанное в битовое поле управления CKD[1:0] в регистре TIMx_CR1. Тактовый сигнал FDTS получается из тактового сигнала таймера, а битовое поле управления CKD[1:0] устанавливает соотношение между этими двумя тактовыми сигналами.

Один из двух источников тактовых импульсов может использоваться в качестве источника тактовых импульсов выборки: либо тактовый сигнал таймера FCK_INT, либо источник тактовых импульсов FDTS.

*Рисунок 4*показывает практический пример, когда стадия фильтрации активирована для входа ETR таймера. Ради этого показательного примера:

- Тактовая частота таймера FCK_INT = 1 МГц.
- ЧКД [1:0] = '01'. Это означает, что частота тактового сигнала FDTS в два раза меньше частоты тактового сигнала таймера: Fdts=Fck_int/2=500KHz
- ETF [3:0] = '0100'. Это означает, что тактовый сигнал FDTS выбирается в качестве тактового сигнала для фильтра с уменьшенной в два раза частотой. Это также означает, что действительный импульс на входе ETR должен иметь длительность не менее 6 тактовых циклов дискретизации.
- В этом примере любой импульсный сигнал на входе ETR таймера длительностью менее 6xTsampling=6x1/250kHz=24 мкс отклоняется.

Рисунок 4. Фильтрация входного сигнала (ETF [3:0]= 0100): FSAMPLING = FDTS/2, N=6



1.4.2 Функция предварительной загрузки регистров таймера

Функция предварительной загрузки в контексте периферийного устройства таймера STM32 относится к дублированию некоторых регистров таймера или некоторых битовых полей управления таймером. Поскольку содержимое некоторых регистров таймера и некоторых управляющих битовых полей напрямую влияет на формы сигналов, выдаваемые каналами таймера, обновление содержимого этих регистров и управляющих битовых полей должно быть точно синхронизировано с «событием обновления» таймера, инициируемым рядом с начало нового цикла счета. Такой тесной синхронизации было бы практически невозможно достичь, если бы эти регистры и управляющие битовые поля не были предварительно загружены.

Когда регистр таймера имеет возможность предварительной загрузки, существуют два физических экземпляра этого регистра таймера:

- Экземпляр активного регистра (также называемый экземпляром теневого регистра): его содержимое используется периферийной логикой таймера для генерации выходных сигналов канала таймера.
- Экземпляр регистра предварительной загрузки: это экземпляр регистра, к которому обращается программное обеспечение, когда функция предварительной загрузки соответствующего регистра включена.

Если функция предварительной загрузки отключена, как показано на *Рисунок 5*, следует выделить две основные характеристики:

- Экземпляр регистра предварительной загрузки выглядит несуществующим
- Любой доступ для записи к соответствующему регистру программным обеспечением выполняется в активном экземпляре регистра.

Если функция предварительной загрузки включена, как показано на *Рисунок 6*, следует выделить две характеристики:

- Любой доступ к соответствующему регистру выполняется в экземпляре регистра предварительной загрузки, в то время как содержимое активного экземпляра регистра не изменяется.
- Как только в периферийном устройстве таймера генерируется «событие обновления», содержимое экземпляра регистра предварительной загрузки немедленно передается в активный экземпляр регистра. Содержимое регистра предварительной загрузки передается в идеальной синхронизации с «событием обновления», другими словами, оно находится в идеальной синхронизации с новым циклом счета, соответствующим новому циклу в выводимом сигнале.

Рис. 5. Механизм предварительной загрузки регистра канала таймера — отключен

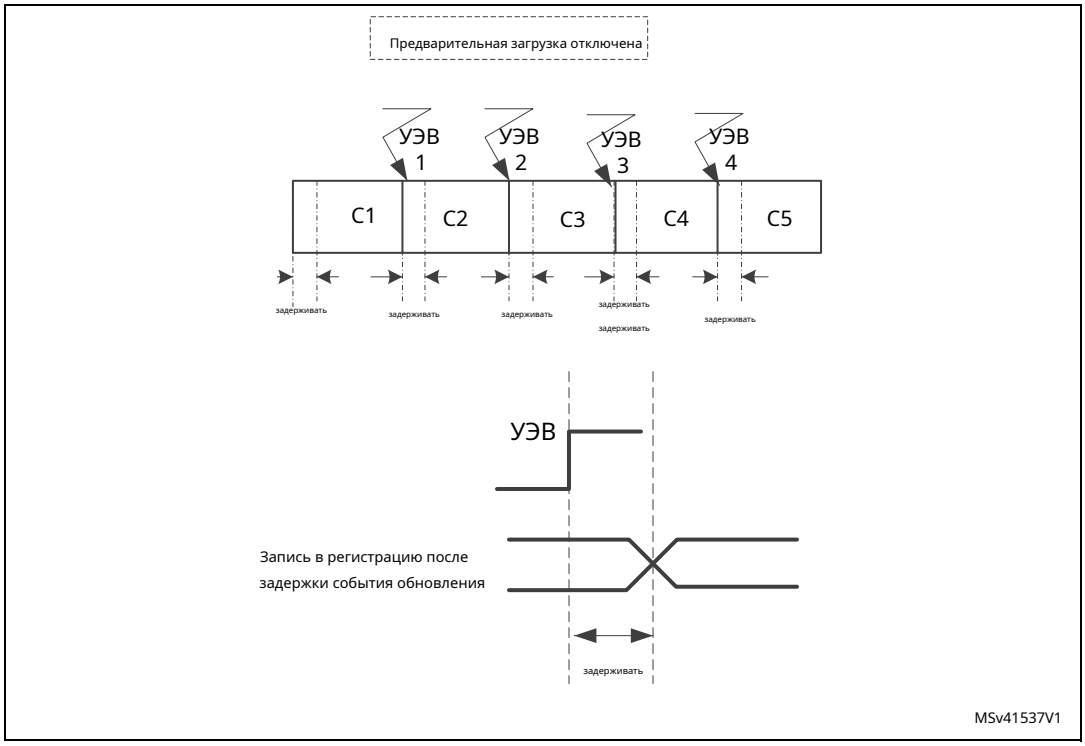
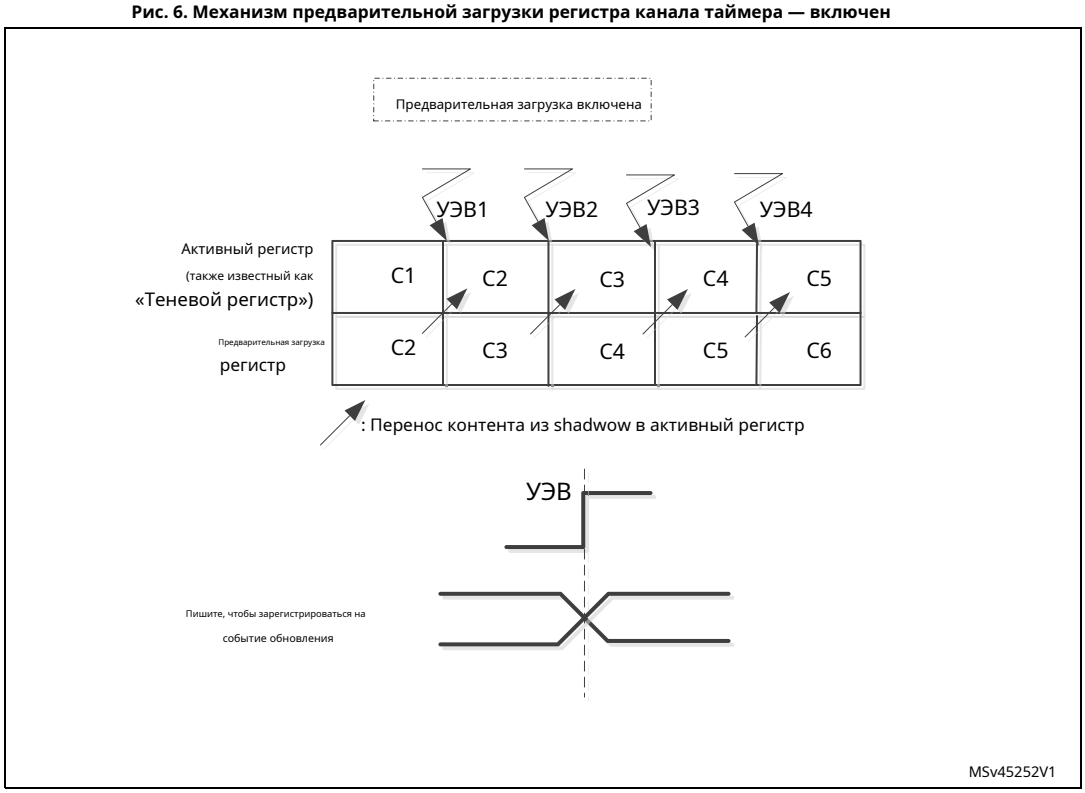


Рис. 6. Механизм предварительной загрузки регистра канала таймера — включен



Функция предварительной загрузки доступна для:

- Регистр таймера автоперезагрузки (TIMx_ARR)
- Регистр предделителя таймера (TIMx_PSC) (нельзя отключить)
- Регистры канала таймера (TIMx_CCRy)
- Битовые поля управления CCxE и CCxNE в регистре таймера TIMx_CCER
- Битовое поле управления OCxM в регистрах таймера TIMx_CCMRn.

Функция предварительной загрузки может представлять большой интерес при выводе ШИМ-сигнала на определенный канал таймера. Выходной уровень канала зависит от непрерывного сравнения между значением счетчика таймера и значением регистра канала таймера TIMx_CCRy, поэтому любое изменение в регистре канала таймера может привести к немедленному изменению выходного уровня канала таймера.

Как следствие, запись непосредственно в регистр канала в середине периода ШИМ может генерировать паразитные сигналы. Чтобы преодолеть эту проблему, функция предварительной загрузки должна быть включена для соответствующего регистра канала таймера. Когда функция предварительной загрузки включена, любой доступ для записи осуществляется в экземпляр регистра предварительной загрузки канала, в то время как активный экземпляр регистра канала остается неизменным. Возмущение текущего периода ШИМ отсутствует.

Как только таймер генерирует «событие обновления», содержимое экземпляра регистра предварительной загрузки передается в активный экземпляр регистра. Экземпляр активного регистра используется в течение следующего периода ШИМ для выполнения операции сравнения и определения нового коэффициента заполнения.

Подводя итог, можно сказать, что функция предварительной загрузки гарантирует, что при доступе к регистрам таймера и битовым полям управления не будут генерироваться паразитные сигналы, которые напрямую влияют на выходные данные каналов таймера; особенно в середине цикла вывода сигнала.

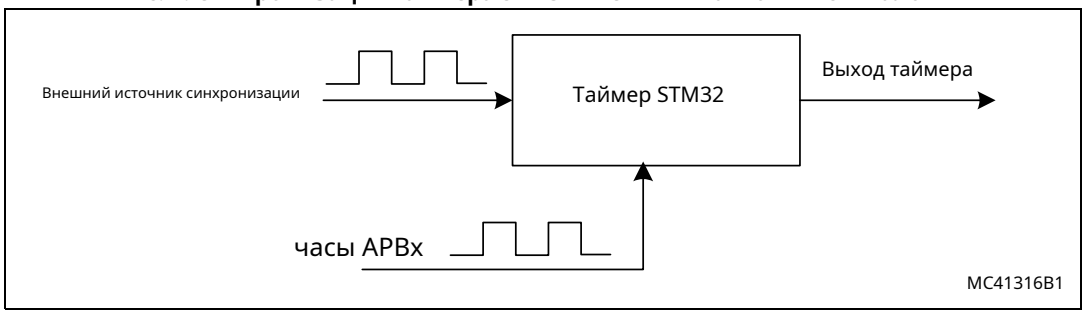
2 Синхронизация таймера с использованием внешнего источника синхронизации

2.1 Обзор

Периферийные устройства таймера STM32 могут синхронизироваться с помощью часов внешнего источника, но это не означает, что часы APB (расширенная периферийная шина) не требуются. Периферийный таймер STM32 синхронизирует внешний тактовый сигнал с собственным тактовым сигналом ядра (который является тактовым генератором APB). Полученный синхронизированный тактовый сигнал подается на предварительный делитель таймера, который, в свою очередь, подает сигнал на счетчик таймера.

Для периферийного устройства таймера STM32 требуется два источника тактовых импульсов, чтобы постоянно обновлять его временную базу (см.*Рисунок 7*). Период внешнего источника синхронизации — это единица времени, используемая для обновления базы времени периферийных устройств таймера.

Рис. 7. Синхронизация таймера STM32 внешним тактовым сигналом

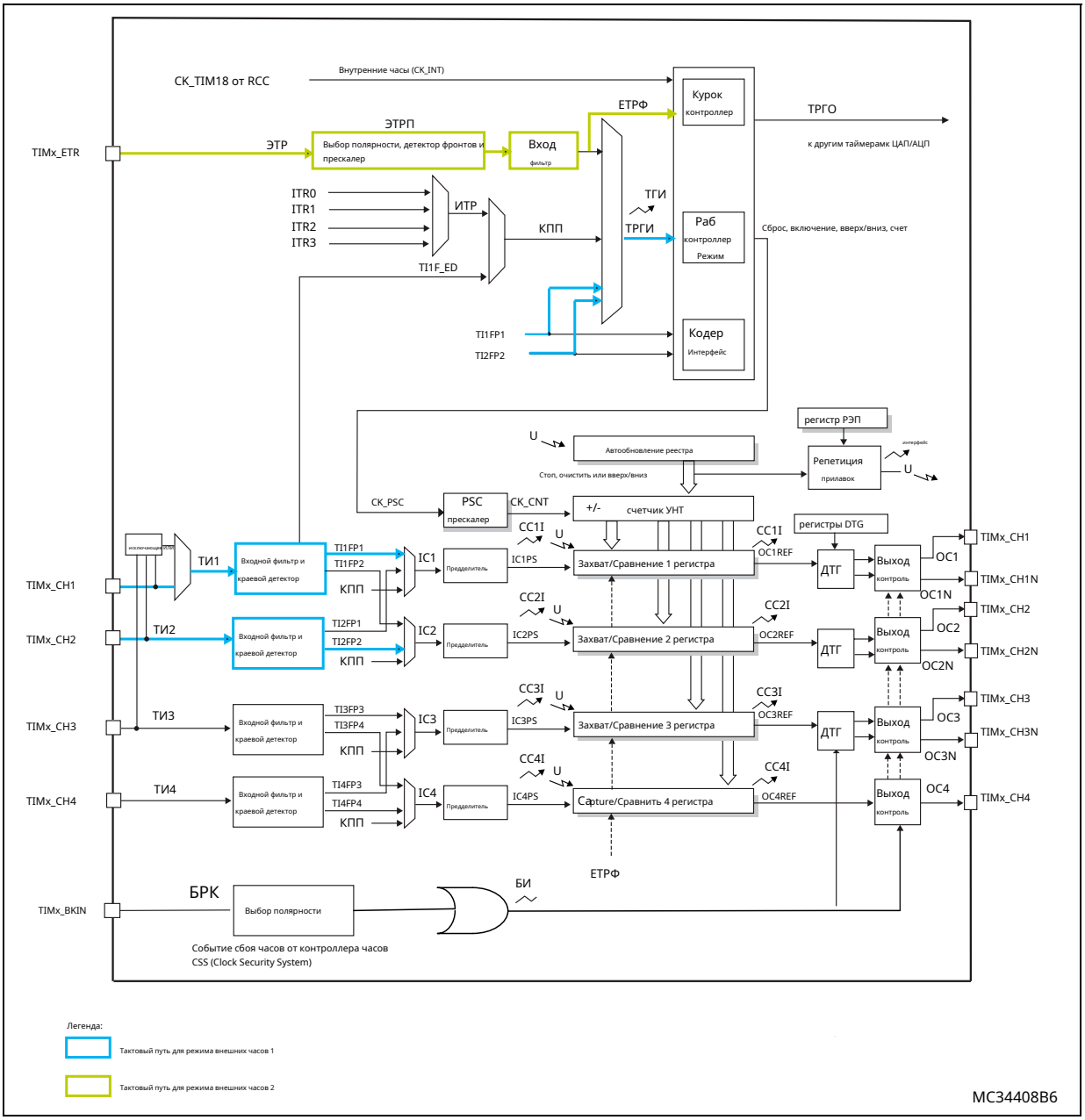


Есть два способа синхронизировать (или внешне синхронизировать) таймер STM32:

- Режим тактирования внешнего источника 1: путем подачи внешнего тактового сигнала на один из входов канала таймера, входы TIx.
- Режим тактирования внешнего источника 2: путем подачи внешнего тактового сигнала на вход ETR таймера (если он реализован и доступен).

The *Рисунок 8* ниже показан путь синхронизации для обоих режимов внешнего источника синхронизации.

Рисунок 8. Тактовый путь для режимов внешнего источника синхронизации



2.2 Блок синхронизации

Прежде чем вводить режимы тактирования периферийного таймера внешним источником тактового сигнала, важно сначала представить механизм синхронизации, реализованный периферийным устройством таймера STM32 при работе с внешними сигналами. Внешние сигналы — это сигналы, поступающие снаружи периферийного устройства таймера. Они могут быть синхронизированы с тактовым сигналом, отличным от того, который используется периферийным устройством таймера, или они могут быть полностью асинхронными.

Периферийный таймер должен иметь дело с внешними сигналами, которые могут быть полностью асинхронными, и обрабатывать их соответствующим образом. Периферийный таймер также

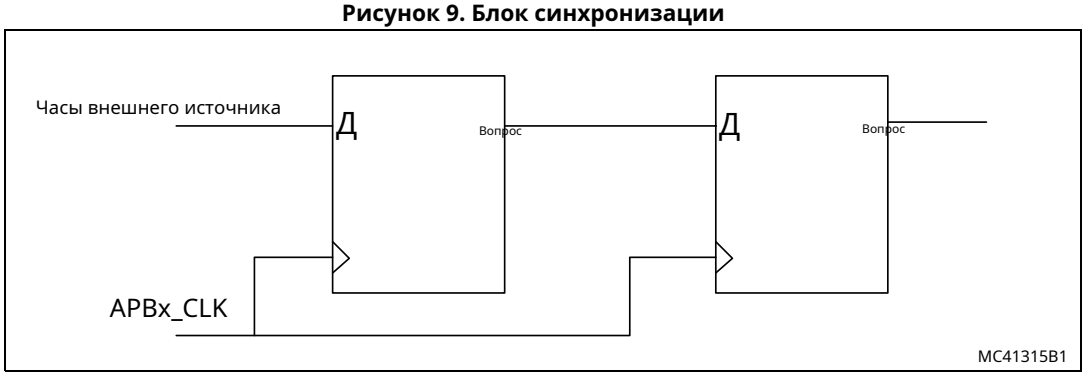
должен иметь возможность информировать программное обеспечение об отметке времени, когда сигнал на данном входе таймера изменил свое состояние (например, переключение сигнала).

Чтобы иметь дело с полностью асинхронным сигналом, периферийному устройству таймера необходимо сначала повторно синхронизировать его с собственным тактовым сигналом. Может потребоваться повторная синхронизация, например, с логическим тактовым сигналом ядра таймера, прежде чем подавать результирующий сигнал на различные подблоки периферийного устройства таймера. Это действие защищает основную логику таймера от возникновения проблем с метастабильностью.

Рисунок 9 показана синоптическая схема схемы синхронизации, используемой для синхронизации внешних сигналов, подаваемых на входы таймера. Схема синхронизации в основном состоит из двух каскадных D-триггеров, которые тактируются тактовым сигналом ядра таймера. Внешний сигнал подается на вход первого каскада D-триггера, где синхронизированный сигнал извлекается с выхода второго D-триггера. Эта схема синхронизации вносит задержку не менее двух тактов ядра таймера и не более трех тактов.

Таймер встраивает «сначала синхронизировать» на всех входах, кроме ETR, где сначала есть предварительный делитель, а затем повторная синхронизация.

Рисунок 9. Блок синхронизации



Как показано в приведенной ниже формуле, частота входного сигнала должна быть в три раза меньше частоты тактового сигнала ядра таймера.

$$Частота_{ТИМКЛК} \geq 3 \times Частота_{входной\ сигнал}$$

2.3 Режим внешнего источника синхронизации 1

Когда активирован режим 1 внешнего источника синхронизации, любой сигнал, который может быть направлен во внутренний сигнал таймера TRGI, также может тактировать счетчик таймера. *Рисунок 8* показывает возможный источник синхронизации для счетчика таймера, а именно:

- Входной сигнал ETRF: сигнал ETR после предварительного масштабирования, синхронизации и фильтрации.
- Сигналы синхронизации периферийных устройств между таймерами (входы ITR)
- Сигнал TI1FD, который является выходом канала таймера 1, но который чувствителен к обоим фронтам сигнала (каждый переход входа таймера 1 генерирует импульс)
- Входные сигналы TI1FP1 и TI2FP2, которые являются синхронизированными, отфильтрованными, а затем предварительно масштабированными входами таймера TI1 и TI2 соответственно.

В этом разделе рассматривается только режим внешних часов 1, когда часы внешнего источника подаются на таймер через один из его входов, другими словами, когда часы подаются на таймер через входы таймера ETR, TI1 или TI2.

Использование входа ETR для питания таймера является альтернативной конфигурацией для режима 2 внешнего источника тактового сигнала, поэтому в этом разделе он не рассматривается далее. Соответствующее справочное руководство содержит правильную конфигурацию для активации этой альтернативы тактирования.

Входы таймера, TI1 и TI2 в качестве источника синхронизации

Только входы TI1 и TI2 могут использоваться для подачи тактового сигнала на счетчик таймера, когда активирован внешний режим тактирования 1. Если таймер уже имеет четыре канала, входы TI3 и TI4 не могут синхронизировать таймер в этом режиме.

Тактовый сигнал, подаваемый на периферийные входы таймера TI1 или TI2, сначала обрабатывается перед тем, как попасть на счетчик таймера. Обработка входного сигнала состоит из каскада внешней синхронизации сигнала, затем каскада предварительного масштабирования и фильтрации.

Входной фильтр и прескалер настраиваются. Канал, связанный с целевым входом таймера, должен быть настроен как входной канал; затем те же поля управляющих битов, которые используются для настройки входного канала, используются для настройки входа для внешнего источника синхронизации.

Ниже приведена типичная последовательность настройки входов TI1 или TI2 в качестве входов для внешнего тактового сигнала:

1. Настройте канал, связанный с входом таймера, который подает внешний тактовый сигнал, как входной канал. Несмотря на то, что любое значение, записанное в битовое поле управления выбором захвата/сравнения (CCxS), за исключением значения 00, устанавливает канал таймера в режим ввода, правильным значением для настройки является CCxS = 01.

Примечание: *Битовое поле CCxS доступно для записи только тогда, когда связанный с ним канал таймера отключен (например, когда битовое поле включения/отключения управления для этого канала в регистре TIMx_CCER сброшено, CCxE = '0'.*

2. Настройте полярность активного фронта. Некоторые микроконтроллеры STM32 имеют встроенные таймеры с входами, чувствительными к спаду, нарастанию или обоим фронтам (например, микроконтроллеры STM32F2); другие семейства STM32 встраивают таймеры, входы которых чувствительны только к переднему или заднему фронту (например, микроконтроллеры STM32F1). В справочном руководстве для каждого микроконтроллера STM32 указано правильное значение для записи в битовые поля управления полярностью для настройки удобной полярности входа таймера.
   Например, чтобы сделать вход таймера на микроконтроллерах STM32F30x чувствительным к обоим фронтам, битовые поля управления полярностью для соответствующего канала должны быть настроены как CCxP = 1 и CCxNP = 1.
3. При необходимости настройте входной фильтр, связанный с соответствующим каналом, для подавления импульсов длительностью меньше заданного значения. Порог для отклонения или прохождения входного импульса настраивается через битовое поле ICxF[3:0] в регистре TIMx_CCMRx, связанном с используемым каналом таймера.
4. После правильной настройки входа таймера с требуемой настройкой перенаправьте обработанный входной сигнал на счетчик таймера. Этот шаг выполняется путем записи правильного значения в битовое поле управления выбором триггера TS[2:0] в регистре TIMx_SMCR. Например, если вход таймера TI1 используется в качестве входа внешнего тактового сигнала, то правильная конфигурация должна быть TS[2:0]=101.
5. Наконец, активируйте режим внешнего источника синхронизации 1. Это делается путем записи правильного значения в битовое поле выбора ведомого/ведущего (SMS) в регистре TIMx_SMCR. Обратите внимание, что это битовое поле имеет разную ширину в разных семействах микроконтроллеров STM32. Документ справочного руководства указывает правильное значение для записи в битовое поле SMS в

чтобы активировать различные режимы часов. Например, в серии STM32F1 битовое поле SMS имеет ширину 3 бита, а правильное значение для настройки — SMS[2:0] = 111, независимо от того, для микроконтроллеров STM32F30x битовое поле SMS имеет ширину 4 бита и правильное значение для настройки: SMS[3:0] = 0111.

6. Установите битовое поле управления CEN в регистре TIMx_CR1, чтобы включить счетчик таймера.

*Рисунок 10*показывает типичную последовательность подсчета счетчика таймера, где периферийное устройство таймера сконфигурировано в конфигурации прямого счета. Содержимое счетчика равно нулю во время установки битового поля управления CEN, а периферийное устройство таймера синхронизируется извне в режиме 1 через вход TI1.

В этом примере показано, что сконфигурированный активный фронт — это нарастающий фронт, где битовые поля управления фильтром и предварительным делителем сохраняются в своих значениях по умолчанию (как состояние после сброса).*Рисунок 10* также показывает влияние стадии ресинхронизации на входе таймера TI1.

Существует задержка между передним фронтом тактового сигнала на входе таймера TI1 и передним фронтом внутреннего сигнала счета, подаваемого на счетчик таймера CK_CNT. В этом примере содержимое регистра предварительного делителя таймера TIMx_PSC равно нулю, поэтому сигнал CK_CNT совпадает с сигналом CK_PSC (CK_CNT = CK_PSC).

Рис. 10. Приращение счетчика таймера (режим внешних часов 1)

Из-за стадии ресинхронизации требуется, чтобы частота внешнего тактового генератора была менее чем в два раза выше частоты таймера, как показано в формуле ниже:

$$TIMxклкчастота \geq 3 \times ITxчастота$$

2,4 Режим внешнего источника синхронизации 2

Когда активирован режим внешнего источника синхронизации, счетчик таймера обновляется при обнаружении активных фронтов тактового сигнала, подаваемого через вход таймера ETR. Для некоторых периферийных устройств таймера в определенных семействах STM32 вход ETR может не выводиться из микроконтроллера.

Для некоторых ETR вход не отображается на ввод-вывод микроконтроллера в качестве альтернативной функции. В некоторых других периферийных устройствах таймера вход таймера ETR мультиплексируется с входом таймера канала 1.

Основное преимущество использования внешнего источника тактового сигнала в режиме 2 по сравнению с режимом 1 заключается в том, что частота внешнего тактового сигнала может быть равна или даже превышать частоту внутреннего тактового сигнала ядра таймера (например, на тактовом сигнале шины APB).

Это не означает, что счетчик таймера может обновляться с частотой, превышающей частоту основных часов таймера (например, с увеличенной частотой, если таймер сконфигурирован в режиме прямого счета).

Вход таймера ETR является единственным входом таймера, который имеет предзагрузочный этап перед этапом ресинхронизации. Эта ступень предкаскалера является полностью асинхронной и может быть разделена до восьми раз меньше частоты асинхронного входного сигнала.

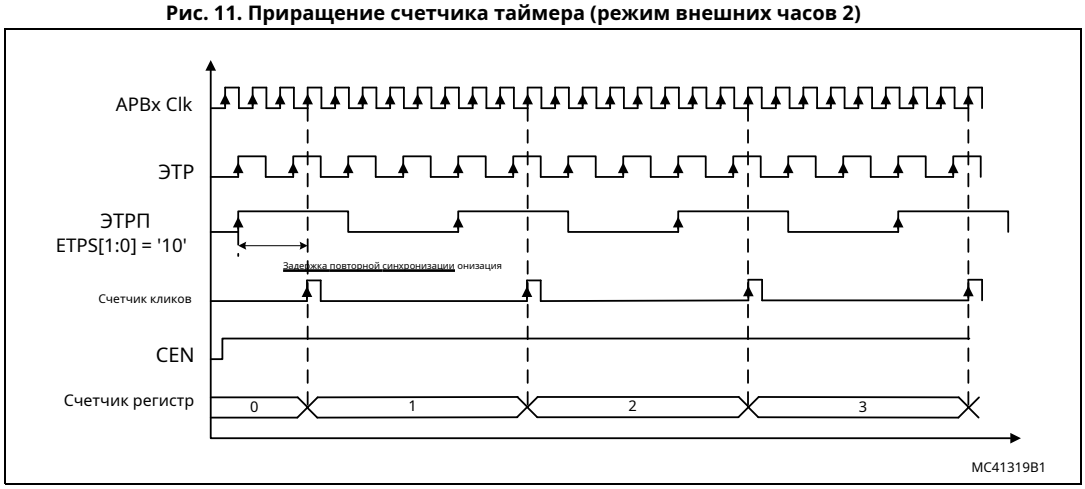
Выход каскада асинхронного предделителя, называемый сигналом ETRP, направляется на каскад ресинхронизации перед подачей на счетчик таймера. Выход схемы ресинхронизации называется сигналом ETRF. Сигнал ETRP имеет то же ограничение, что и любой другой асинхронный сигнал, подаваемый на таймер: его частота должна быть менее чем в три раза больше внутренней тактовой частоты ядра таймера.

Эта особенность режима 2 внешнего источника тактового сигнала представляет большой интерес во многих прикладных случаях. Например, может потребоваться подсчитать количество импульсов, генерируемых определенным датчиком, но выходная частота этого датчика выше, чем внутренние часы ядра таймера. В этом случае предварительный делитель должен быть сконфигурирован для уменьшения частоты входного сигнала в заданном отношении, что делает его совместимым с ограничением стадии ресинхронизации на частоту внешнего входного сигнала. При интерпретации числа подсчитанных импульсов следует учитывать, что частота сигнала делилась на заданное соотношение.

*Рисунок 11*показан пример, когда периферийное устройство таймера настроено на режим внешнего источника тактовой частоты 2 и где тактовый сигнал подается на таймер через его вход ETR. В этом примере частота внешнего тактового сигнала выше, чем частота внутреннего тактового сигнала ядра таймера (тактовый сигнал шины APB). Асинхронный предварительный делитель ETR был сконфигурирован для деления частоты входного сигнала на четыре путем установки битового поля управления предварительным делителем внешнего триггера в регистре TIMx_SMCR на ETPS = '10'.

Также,*Рисунок 11*показывает задержку между выходом предварительного делителя (сигнал ETRP) и тактовым сигналом счетчика CK_CNT, используемым для обновления счетчика таймера. Эта задержка вставляется этапом ресинхронизации.

Рис. 11. Приращение счетчика таймера (режим внешних часов 2)



Ниже приведена типичная и рекомендуемая последовательность настройки периферийного устройства таймера в режиме внешнего источника синхронизации 2:

1. Оцените максимальную частоту входного тактового сигнала и выберите коэффициент асинхронного предделителя. Если частота внешнего тактового сигнала меньше частоты внутреннего ядра таймера более чем в три раза, то использование предварительного делителя необязательно и не может быть использовано. В случае, если асинхронный прескалер не используется, битовое поле управления ETPS должно быть сброшено (по умолчанию битовое поле ETPS сброшено).
2. При необходимости активировать этап фильтрации для отбраковки импульсов тактового сигнала длительностью меньше определенного порога. Для получения дополнительной информации о том, как настроить функцию фильтрации на внешних входах периферийных устройств таймера STM32, см.*Раздел 1.4.1: Этап фильтрации на стр. 12*.
3. Настройте активный фронт внешнего тактового сигнала. Управляющий бит ETP устанавливает, для чего чувствителен фронт режима 2 внешнего источника тактовых импульсов. По умолчанию и после сброса управляющий бит ETP сбрасывается, что делает нарастающие фронты внешнего тактового сигнала активными (например, нарастающие фронты внешнего тактового сигнала запускают обновление счетчика таймера).
4. Включите режим внешнего источника тактовой частоты 2, установив управляющий бит включения внешней тактовой частоты, ECE = 1.
5. Обязательно в конце установить бит управления включением счетчика в регистре TIMx_CR1, чтобы включить счетчик таймера.

Примечание: *Вход ETR можно также использовать в качестве входа для внешнего тактового сигнала, если сконфигурирован режим 1 внешнего источника тактового сигнала. Можно одновременно активировать оба режима внешнего источника синхронизации 1 и 2 и заставить их одновременно использовать вход ETR. В этом случае приоритет отдается режиму 2 внешнего источника синхронизации, который используется для тактирования счетчика таймера.*

2,5 Режим 1 внешнего источника синхронизации по сравнению с режимом 2

Оба режима внешнего источника синхронизации, 1 и 2, похоже, имеют одинаковую функциональность, но их тщательное изучение показывает, что они различаются некоторыми специфическими характеристиками, которые делают каждый из них подходящим для различного диапазона прикладных случаев.

См. ниже основные различия между режимами 1 и 2:

- Можно использовать режим 1 внешнего тактового генератора для обновления счетчика таймера на обоих фронтах внешнего тактового сигнала. Это невозможно, когда используется режим 2 внешнего источника синхронизации.
- Используя режим внешнего источника синхронизации 2, можно синхронизировать периферийное устройство таймера с помощью внешнего источника синхронизации. В этом случае все еще возможно одновременно сконфигурировать периферийное устройство таймера в одном из совместимых режимов ведомого таймера. Например, если требуется повторно подсчитать количество импульсов, сгенерированных определенным датчиком за заданный период времени:
  - Сделайте одно периферийное устройство таймера, например TIMy, подсчитывающим количество импульсов, генерируемых датчиком, настроив его в режиме внешнего источника синхронизации 2.
  - Путем настройки второго периферийного устройства таймера, такого как TIMz, для многократной генерации сигнала запуска (TRGO) в каждый заданный период времени.
  - Первый таймер (TIMy) также должен быть сконфигурирован в режиме ведомого сброса и использовать выходной сигнал TRGO второго таймера (TIMz) в качестве триггера для сброса.

Один из способов заставить периферийное устройство таймера выдавать импульс на своем выходном сигнале TRGO через равные промежутки времени — это установить в регистре таймера TIMx_ARR определенное значение, чтобы генерировалось периодическое «событие обновления» таймера.

«Событие обновления» таймера может запускать импульс на сигнале TRGO таймера, если главный режим таймера установлен на «значение обновления» (например, если битовое поле MMS[2:0] установлено на «010» в регистре TIMx_CR2) .

Таймер можно сконфигурировать в режиме сброса ведомого, установив правильное значение в битовом поле управления выбором ведомого режима (например, SMS[2:0] = 100 в регистре TIMx_SMCR).

Выбор правильного триггера для ведомого режима сброса обеспечивается путем настройки правильного значения в битовом поле управления выбором триггера TS[2:0].

Значение для записи в управляющее битовое поле TS[2:0] зависит от того, к какому входу ITR первого таймера (TIMy) внутренне подключен выход TRGO второго таймера (TIMz). В справочном руководстве каждого микроконтроллера STM32 перечислены все внутренние соединения между периферийными устройствами таймера.

2,6 Применение: синхронизация таймера с использованием внешнего источника синхронизации на входе таймера ETR.

В этом приложении описывается один вариант использования внешнего источника синхронизации для синхронизации периферийного устройства таймера. Это приложение разработано для режима 2 внешнего источника тактовой частоты, но его можно легко изменить для использования режима 1 внешнего источника тактовой частоты.

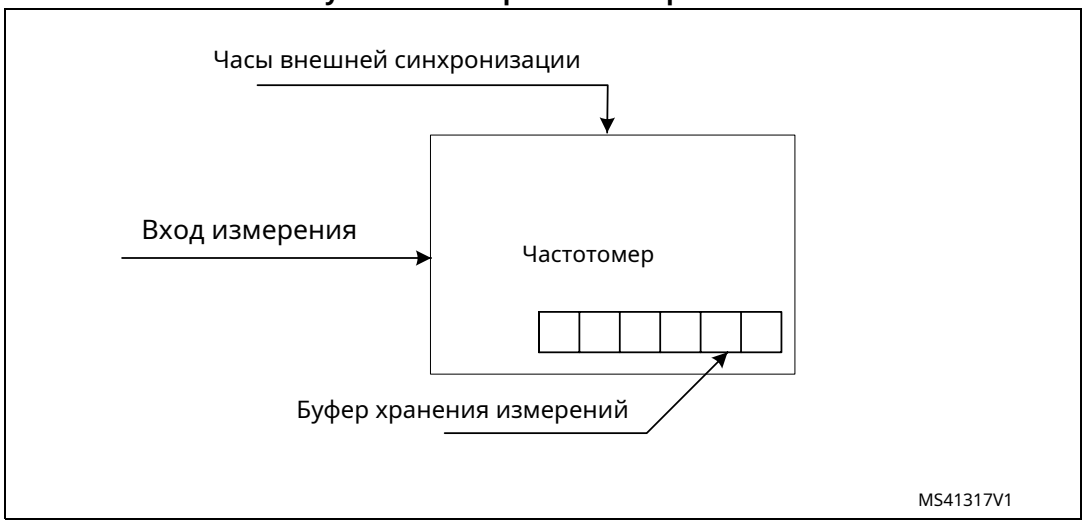
Это приложение является базовой реализацией счетчика частоты, например счетчика с уменьшенным диапазоном частот и прецизионного счетчика. Это приложение было разработано в следующих предположениях:

– Микроконтроллер использует свой внутренний высокоскоростной генератор (генератор HSI) в качестве источника тактового сигнала.

– Это приложение используется в среде, где доступен источник тактового сигнала с точностью до 10 МГц (например, в измерительном стенде, где несколько приборов обеспечивают калибровочный выходной сигнал с частотой 10 МГц).

– Эта демонстрация приложения разделена на две части. В первой части приложение частотомера тактируется только внутренним генератором HSI. Во второй части приложение счетчика частоты имеет доступ к относительно более точному источнику тактовой частоты 10 МГц, который подается на вход ETR таймера.

Рисунок 12. Обзор частотомера



Чтобы подчеркнуть влияние синхронизации периферийного устройства таймера с внешними часами на точность измерения, опорный тактовый сигнал вводится на вход измерения приложения, как показано на рис.*Рисунок 12*.

Затем следует провести серию измерений частоты входных сигналов в обоих случаях: есть ли внешний тактовый сигнал и нет ли. Разброс измерений обеспечивает хороший индикатор того, улучшило ли использование внешнего источника тактового сигнала производительность этого базового приложения частотомера или нет.

Для запуска этого приложения необходима следующая аппаратная настройка:

— Плата Nucleo STM32F302 (NUCLEO-F302R8)

– USB-кабель для подключения платы Nucleo к ПК для разработки (USB-кабель также обеспечивает питание платы).

– Один или несколько генераторов сигналов.

Для изменения исходного кода этого приложения и отладки различных конфигураций необходим набор программных инструментов:

– Цепочка инструментов разработки, поддерживающая отладчик ST-LINK.

– Последняя версия программного инструмента STM32CubeMX (версия 4.10.1 была последней версией на момент разработки этого приложения).

Таймер TIM2 был выбран потому, что он имеет самое высокое разрешение счетчика (32 бита) для получения большего количества возможных выборок PPM.

Канал 2 был выбран в качестве входного захвата, потому что вывод ETR отображается в канале 1 таймера TIM2.

Идея этого приложения состоит в том, чтобы вычислить период входящего сигнала, который представляет собой разницу между двумя последовательными значениями регистра CCR2, умноженную на период приращения счетчика TIM2, а затем вычесть частоту и PPM.

$$\text{частей на миллион} = \frac{\Delta\phi \times 10^6}{\phi} \quad \text{с} \quad \Delta\phi = \phi_м - \phi$$

– fm: измеренная частота

– f: номинальная частота

Конфигурация

Инициализация системных часов

Инициализация часов выполняется в основной процедуре с помощью функции SystemClock_Config() сразу после инициализации библиотеки HAL (HAL_Init).

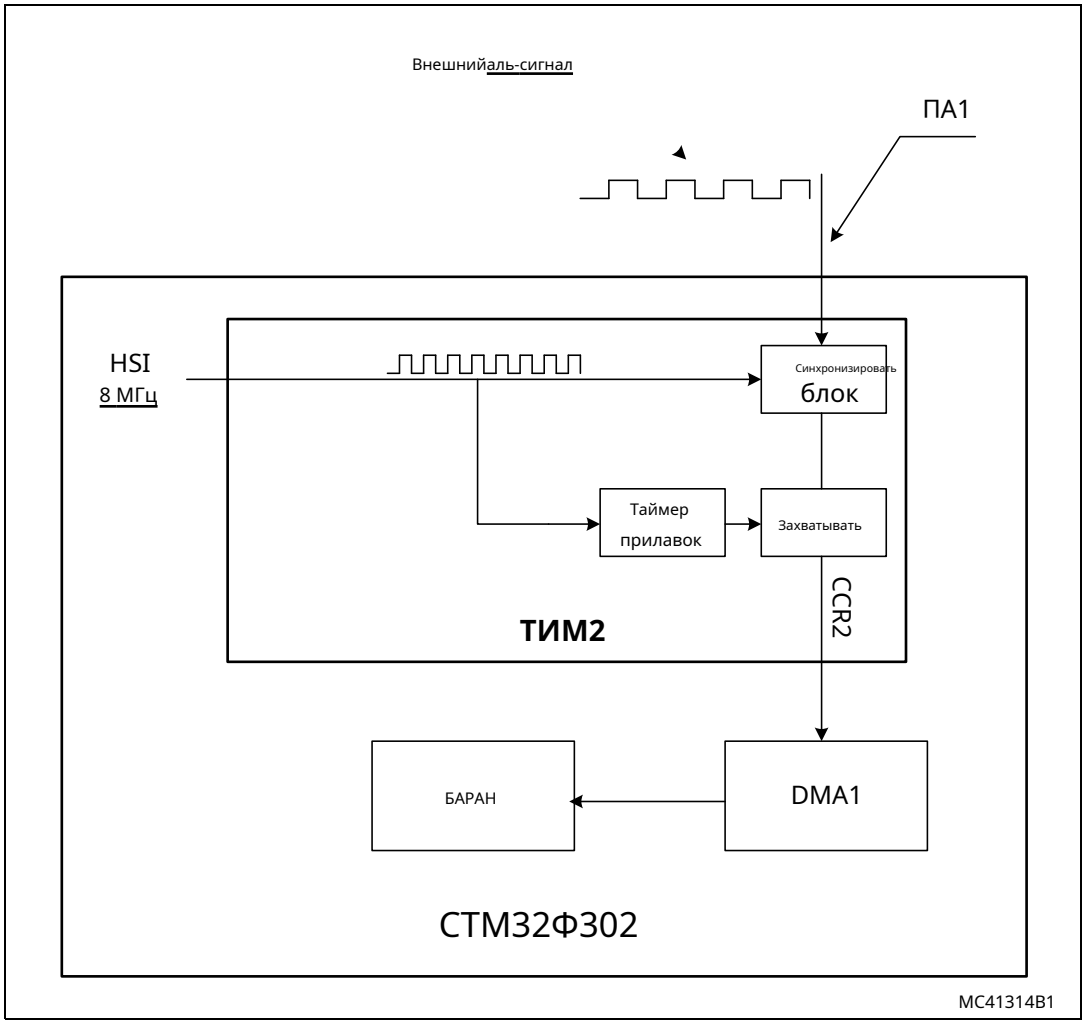
Конфигурация прямого доступа к памяти

DMA используется для копирования значений CCR2 в буфер для вычисления частоты входящего сигнала.

DMA настроен следующим образом:

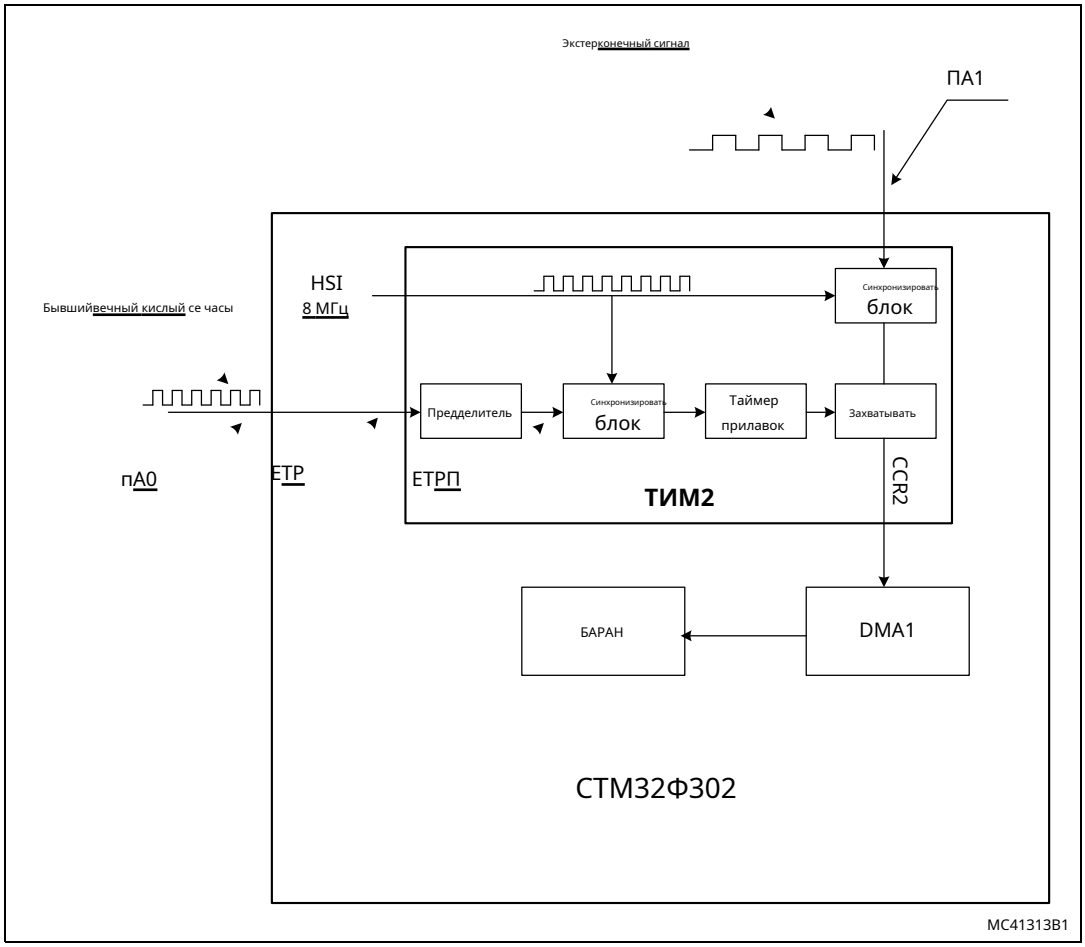
- DMA_Stream = DMA1_Stream6
- DMA_Channel = DMA_Channel_3
- Направление = периферия памяти
- Увеличение памяти = включить
- Выравнивание периферийных данных = слово выравнивания
- Выравнивание данных памяти = слово выравнивания
- Режим = нормальный
- Режим FIFO = включить.

Рис. 13. Архитектура измерителя частоты, тактируемая внутренним генератором HSI.



В этом примере таймер тактируется только внутренним HSI-генератором с частотой 8 МГц. Измеряемый сигнал сначала проходит через таймер с блоком тактовой синхронизации. Затем при каждом захвате таймера нарастающего фронта значение счетчика (доступное в регистре CCR2) затем сохраняет это значение в ОЗУ через DMA. Наконец, он применяет формулу для получения частоты входного сигнала.

Рис. 14. Архитектура частотомера, синхронизируемая внешним источником синхронизации в режиме 2



В этом примере таймер синхронизируется калиброванным внешним источником синхронизации от генератора сигналов с частотой 10 МГц. Затем он проходит через асинхронный пределитель для проверки условия синхронизации с тактовым таймером.

Измеренный сигнал сначала проходит через таймер с блоком синхронизации часов, а затем на каждом таймере переднего фронта захватывает значение счетчика, которое находится в регистре CCR2. Затем сохраняет это значение в ОЗУ через DMA. Наконец, он применяет формулу для получения частоты входного сигнала.

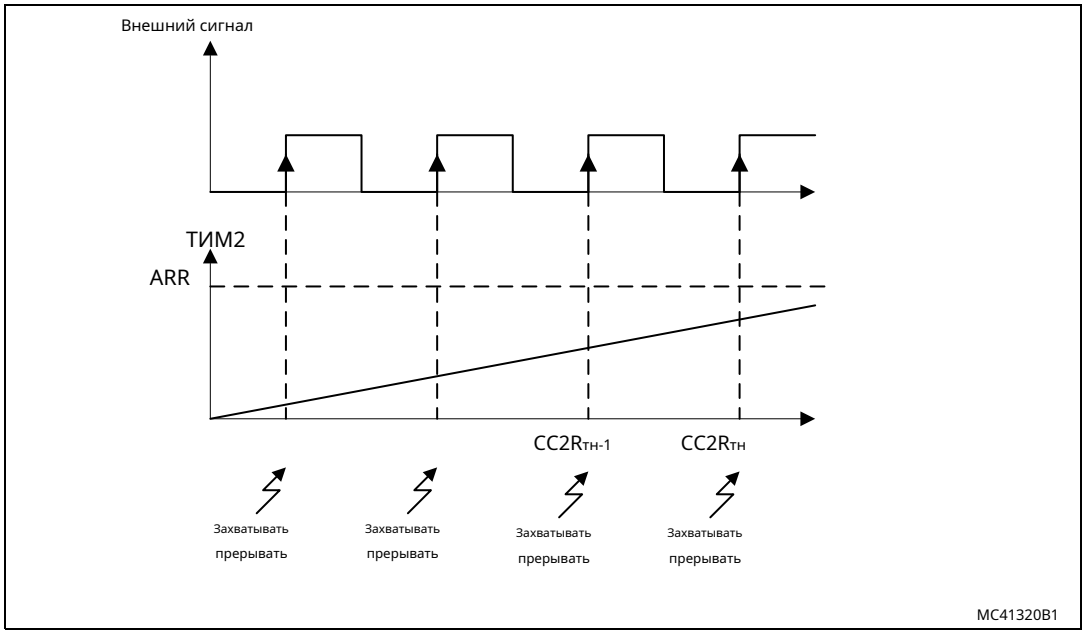
Входной захват

Для измерения периода внешнего сигнала используется таймер в режиме захвата входа. Максимальная частота, которую можно измерить с помощью 32-битного таймера, зависит от сигнала TIM2CLK.

Частота внешнего сигнала не измеряется напрямую, а вычисляется по количеству тактовых импульсов, подсчитанных с помощью таймера. Для этого должна быть доступна очень точная опорная частота.

После включения счетчика таймера, когда возникает первый нарастающий фронт опорного сигнала, значение счетчика таймера фиксируется и сохраняется в CC2R. Он также сохраняется в буфере с помощью DMA, чтобы не перезаписывать его последующим значением при следующем нарастающем фронте.

Рисунок 15. Временная диаграмма захвата ввода



Прошедшее время между двумя последовательными нарастающими фронтами CC2Rтн-CC2Rтн-1представляет собой весь период опорного сигнала.

Поскольку таймер тактируется системными часами (внутренний RC-генератор HSI), реальная частота, генерируемая внутренним RC-генератором, по сравнению с опорным сигналом определяется как:

*Измеренная частота = (CC2Rтн- CC2Rтн-1) * Опорная частота*

Ошибка (в Гц) вычисляется как абсолютное значение разницы между измеренной частотой и типичным значением. Следовательно, ошибка частоты выражается как:

Ошибка (Гц) =|Измеренная частота - типичное значение|

После расчета ошибки для каждого значения обрезки вычет ppm выражается как:

$$\text{частей на миллион} = \frac{\text{Ошибка} \times 10^6}{\text{типичное значение}}$$

В этом примере таймер тактируется на 8 МГц (TIM2CLK = 8 МГц), поэтому минимальная частота, которую можно измерить без переполнения счетчика таймера, составляет:

$$f = \frac{\text{ТИМ2КЛК}}{\text{ARR}} = \frac{8 \times 10^6}{\text{0xFFFFFFFF}} = 0{,}002\ \text{Гц}$$

Входной сигнал может быть измерен от генератора формы волны = 1 кГц. Режим захвата ввода настраивается следующим образом:

– Внешний сигнал подключается к контакту TIM2 CH2 (PA01)

– Нарастающий фронт используется как активный фронт

– TIM2 CCR2 используется для вычисления значения частоты.

Модуль захвата ввода используется для захвата значения счетчика после обнаружения перехода входным каналом 2. Для получения периода внешнего сигнала необходимы два последовательных захвата. Период рассчитывается путем вычитания этих двух значений.

Когда происходит захват, CC2IF (регистр TIM2_SR) устанавливается в 1. Если функция DMA включена, она генерирует запрос DMA. Если происходит захват, установлен флаг CC2IF, то устанавливается флаг избыточной выборки CC2OF.

Когда появляется нарастающий фронт, цифровой измеритель тока с таймером (TIM2_CNT) записывает в TIM2_CCR2. Дождитесь следующего нарастающего фронта, когда в записях TIM2_CNT появится другое значение счетчика. В соответствии с двумя значениями данных мы можем рассчитать цикл входных данных. Таймер переполнения не допускается.

Конфигурация внешнего источника синхронизации в режиме 2

Вторая часть этого примера состоит в том, чтобы сохранить эту конфигурацию и изменить источник тактового таймера на режим внешнего источника 2, который использует вывод ETR в качестве входного тактового сигнала таймера.

Внешний сигнал должен удовлетворять условию синхронизации:

$$ТИМ2_{Clk} \geq 3 \times ф_{ЭТРП}$$

В этом примере CLK TIM2 = CLK APB1 = 8 МГц, тогда:

$$ф_{ЭТРП} \text{ знак равно} \times 8 \text{ МГц} знак\ равно 2{,}66\ МГц$$

Для этого выполните генератор формы сигнала с частотой 8 МГц и используйте асинхронный делитель для деления на четыре:

$$ф_{ЭТРП} = 2МГц \leq \frac{1}{3} \times 8МГц$$

Конфигурация TIM2

```
/* Конфигурация TIM2 */ /* Включение часов
TIM2 */ RCC->APB1ENR |=
RCC_APB1ENR_TIM2EN;
/* Установите предварительный делитель таймера на 8 МГц в качестве тактовой частоты */
Предделитель = (uint16_t)(SystemCoreClock/8000000)-1; TIM2->SMCR = СБРОС; /* Сброс регистра SMCR */
# ifdef USE_ETR
/* Настройка предварительного делителя ETR = 4 */
TIM2->SMCR |= TIM_ETRPRESCALER_DIV4 | /*
Настройте полярность = нарастающий фронт */
TIM_ETRPOLARITY_NONINVERTED |
/* Настройка источника часов ETR */
TIM_SMCR_ECE;
# else /* Внутренний источник часов */
/* Настройка источника внутренних часов */ TIM2-
>SMCR &= ~TIM_SMCR_SMS;
# endif /* USE_ETR */
TIM2->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /*
Выбор режима прямого счетчика */
```

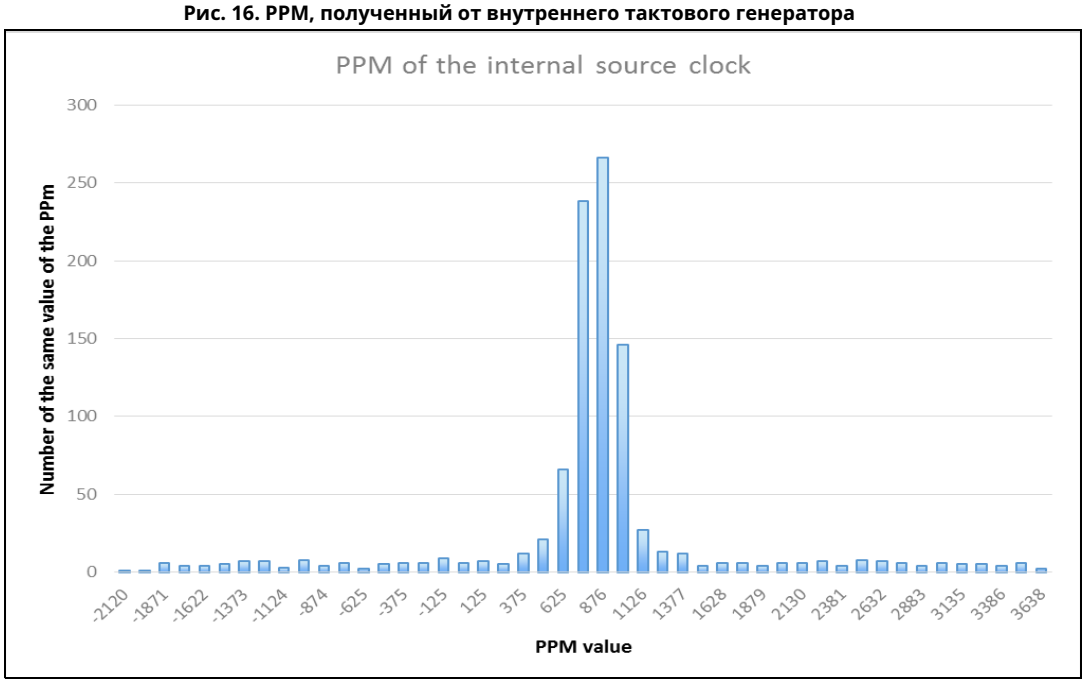
```
TIM2->CR1 |= TIM_COUNTERMODE_UP; TIM2-
>CR1 &= ~TIM_CR1_CKD; /* Устанавливаем
деление часов на 1 */ TIM2->CR1 |=
TIM_CLOCKDIVISION_DIV1; /* Установить
значение автоперезагрузки */ TIM2->ARR =
PERIOD;
/* Установить значение Prescaler */ TIM2->PSC =
(SystemCoreClock / 8000000)-1;
/* Создать событие обновления для немедленной перезагрузки значения прескалера */
TIM2->EGR = TIM_EGR_UG;
TIM2->CCMR1 &= ~TIM_CCMR1_CC2S; /*
Подключаем вход таймера к IC2 */ TIM2-
>CCMR1 |= TIM_CCMR1_CC2S_0;
```

Конфигурация захвата ввода

```
/* Включить DMA1 */
DMA1_Channel7->CCR |=        DMA_CCR_EN;
/* Включить запрос TIM Capture/Compare 2 DMA */
TIM2->DIER |= TIM_DMA_CC2; /* Включает TIM Capture Compare Channel 2 TIM2->CCER |=
TIM_CCER_CC2E;
/* Включить TIM2 */ TIM2->CR1 |
= TIM_CR1_CEN;
/* ждем завершения передачи */ while ((DMA1->ISR &
DMA_ISR_TCIF7) == RESET) {}
```

После взятия 1000 образцов и расчета соответствующего PPM получается приведенная ниже диаграмма.

Рис. 16. PPM, полученный от внутреннего тактового генератора

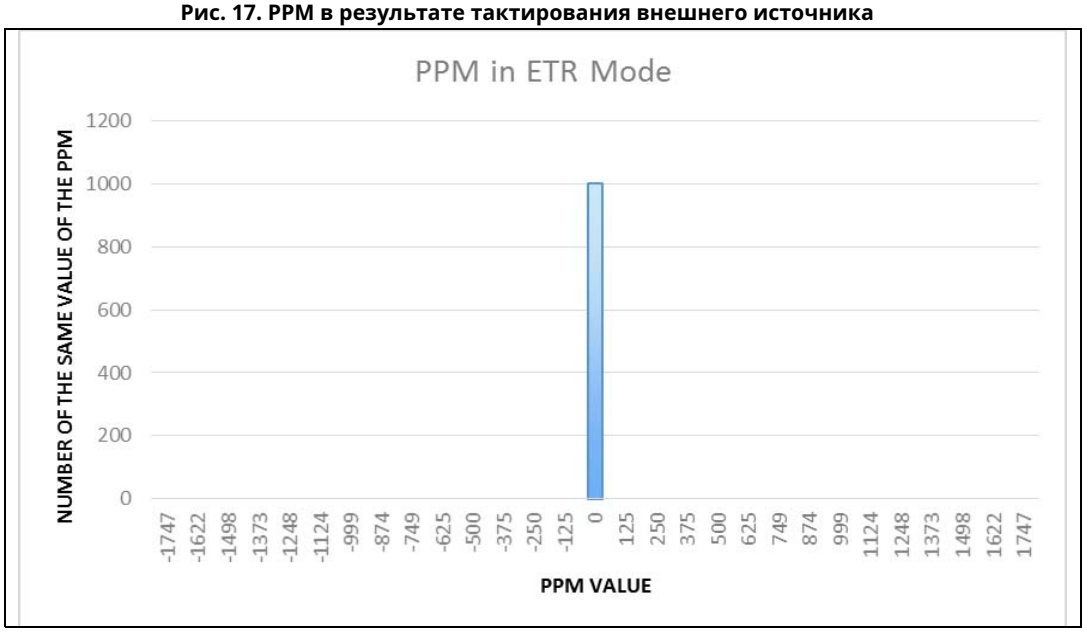


Распределение значений PPM далеко от нуля и имеет статическое и динамическое смещение. Статическое смещение возникает из-за нестабильности HSI (вариации производственного процесса), это систематические неисправности, и оно варьируется от одного микроконтроллера к другому. Динамическое смещение вызвано условиями окружающей среды, такими как температура.

Любая ошибка, возникающая из-за разницы между фактической частотой генератора временной развертки и номинальной частотой, напрямую преобразуется в ошибку измерения. Эта разница является кумулятивным эффектом всех отдельных ошибок генератора развертки.

Для второй части приложения, когда таймер синхронизируется с часами внешнего источника, приведенная ниже диаграмма получена после взятия 1000 выборок и расчета соответствующего PPM.

Рис. 17. PPM в результате тактирования внешнего источника



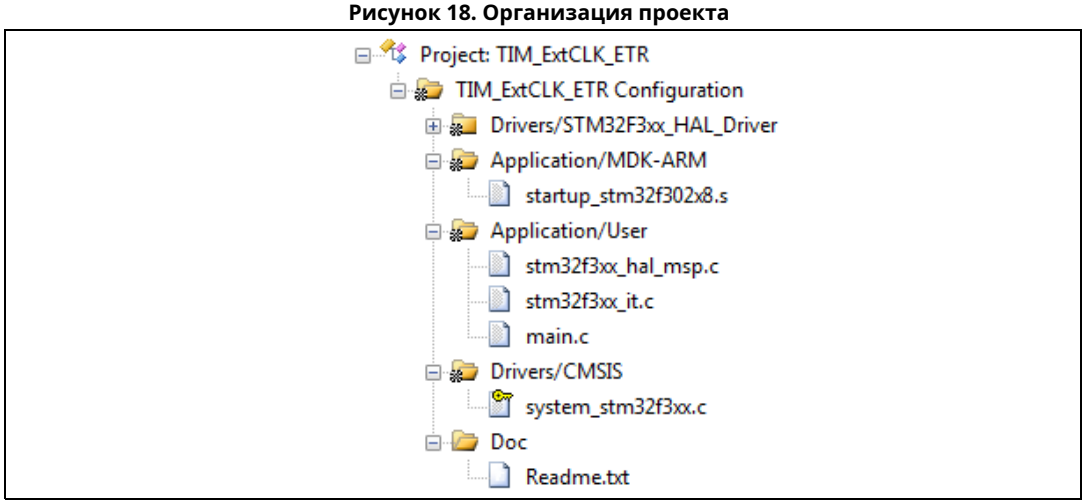
Все значения PPM равны нулю, поскольку все время между нарастающими фронтами измеренного сигнала одинаково. В этом случае обратите внимание на отсутствие ошибок в измерении, поскольку внешний источник тактового сигнала откалиброван должным образом.

2,7 Обзор прошивки

Прошивка была разработана на верстаке Keil µvision, IAR Embedded и SYSTEM WORKBENCH.

Разработанная прошивка поставляется в ZIP-файле и содержит все подкаталоги и файлы исходного кода .h и .c, составляющие ядро приложения.

Рисунок 18. Организация проекта



Прошивка содержит все исходные файлы задач приложения и связанные с ними файлы и состоит из следующих папок проекта:

- Библиотека STM32 MCU HAL
- Файл запуска
- Прикладной уровень.

3 Генерация N-импульсного сигнала в одноимпульсном режиме

3.1 Обзор

Одноимпульсный режим (OPM) периферийного таймера STM32 — это функция, которую можно использовать вместе с каналами таймера, сконфигурированными в режиме вывода. Это позволяет таймеру генерировать импульс программируемой длительности после программируемой задержки на каналах таймера, сконфигурированных в режимах сравнения выходов PWM1 или PWM2.

Этот режим активируется установкой бита OPM в регистре таймера TIMx_CR1. Каждый раз, когда устанавливается управляющий бит OPM и происходит событие обновления таймера, управляющий бит разрешения счетчика таймера сбрасывается, и счетчик фиксируется до своего значения в событии обновления. В конфигурации прямого счета регистр счетчика таймера фиксируется на значении 0. Для другой конфигурации счета, такой как конфигурация с выравниванием по центру и режим обратного счета, обратитесь к документу справочного руководства, чтобы определить значение, на котором счетчик заморожен.

Если событие обновления таймера каким-то образом маскируется, одноимпульсный операционный механизм не может сбросить управляющий бит CEN, и счетчик таймера продолжает работать. Выходы таймера продолжают выводить сконфигурированные сигналы.

Возможность замаскировать эффект события обновления таймера заключается в установке управляющего бита UDIS. Подробнее об этом управляющем бите см. в справочном руководстве. Другой способ замаскировать событие обновления таймера — сделать содержимое регистра счетчика повторений таймера отличным от нуля. Не все периферийные устройства таймера STM32 имеют встроенный счетчик повторений.

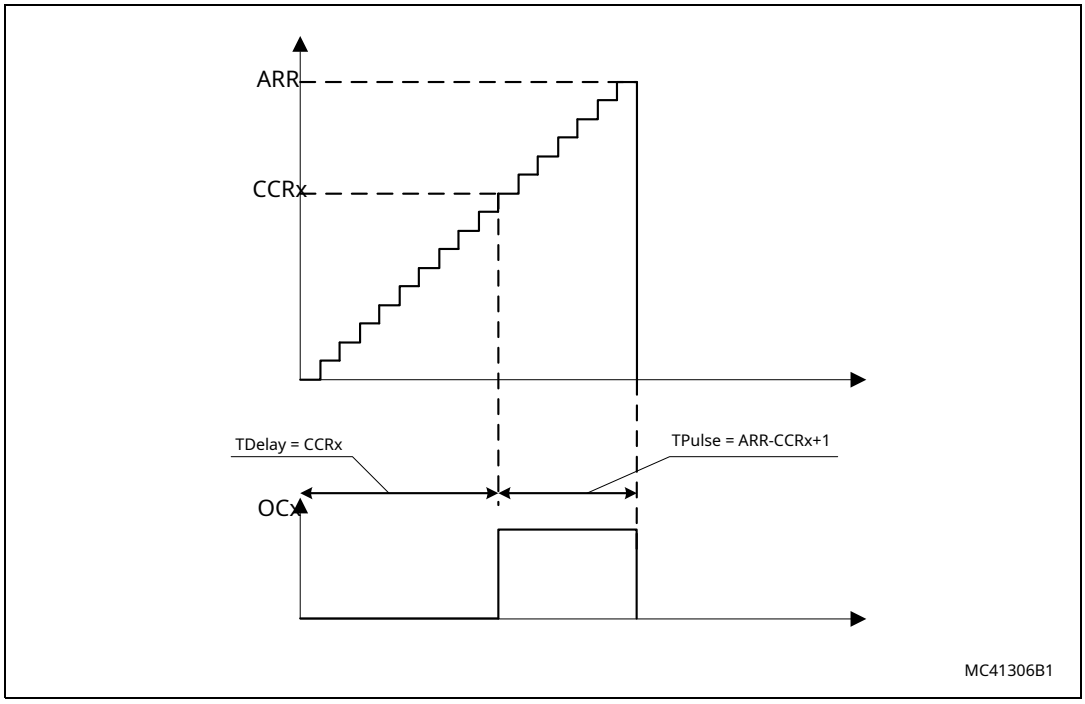
Учитывая предыдущее условие и принимая во внимание, что каждый раз, когда происходит событие обновления (с эффектом маскировки), содержимое счетчика повторений уменьшается, тогда это поведение можно использовать для генерации серии импульсов.

Например, чтобы сгенерировать серию из пяти плюсов, счетчик повторений должен быть установлен на четыре. Первые четыре обновленных события маскируются, поскольку содержимое счетчика повторений отличается от нуля. Эти четыре первых события обновления не сбрасывают управляющий бит CEN, но уменьшают содержимое счетчика повторений до нуля. Пятое событие обновления — это событие, которое сбрасывает элемент управления CEN (счетчик повторений уже равен нулю). Таким образом, счетчик таймера переполнится пять раз, прежде чем будет отключен, и будут сгенерированы пять необходимых импульсов.

При полярности выхода канала по умолчанию выходной режим PWM2 дает типичную форму импульса, например, задержку на заданное время, затем импульс заданной длительности. Для инверсной полярности либо полярность выхода канала должна быть инвертирована, либо должен использоваться выходной режим PWM1.

Первый сценарий с полярностью канала-выхода по умолчанию взят в качестве примера ниже в *Рисунок 19*. Выходной сигнал характеризуется двумя параметрами: длительностью импульса и длительностью задержки.

Рисунок 19. Пример одноимпульсного режима



$T_{Задерживать}$определяется значением, записанным в регистр захвата/сравнения таймера TIM_CCRx.

$T_{Пульс}$определяется разницей между значением, записанным в регистр автоперезагрузки таймера (TIMx_ARR), и значением $T_{Задерживать}$ценность. Значение TIMx_ARR – значение TIMx_CCRx + 1.

3.2 Применение: генерация N-импульсного сигнала в одноимпульсном режиме.

Целью этого приложения является генерация сигнала, состоящего из серии определенного количества импульсов на выходе одного канала таймера. Это достигается с помощью одноимпульсного режима и функций счетчика повторений периферийного таймера STM32.

Периферийный таймер TIM1 выбран для этого приложения, потому что он отвечает необходимым требованиям.

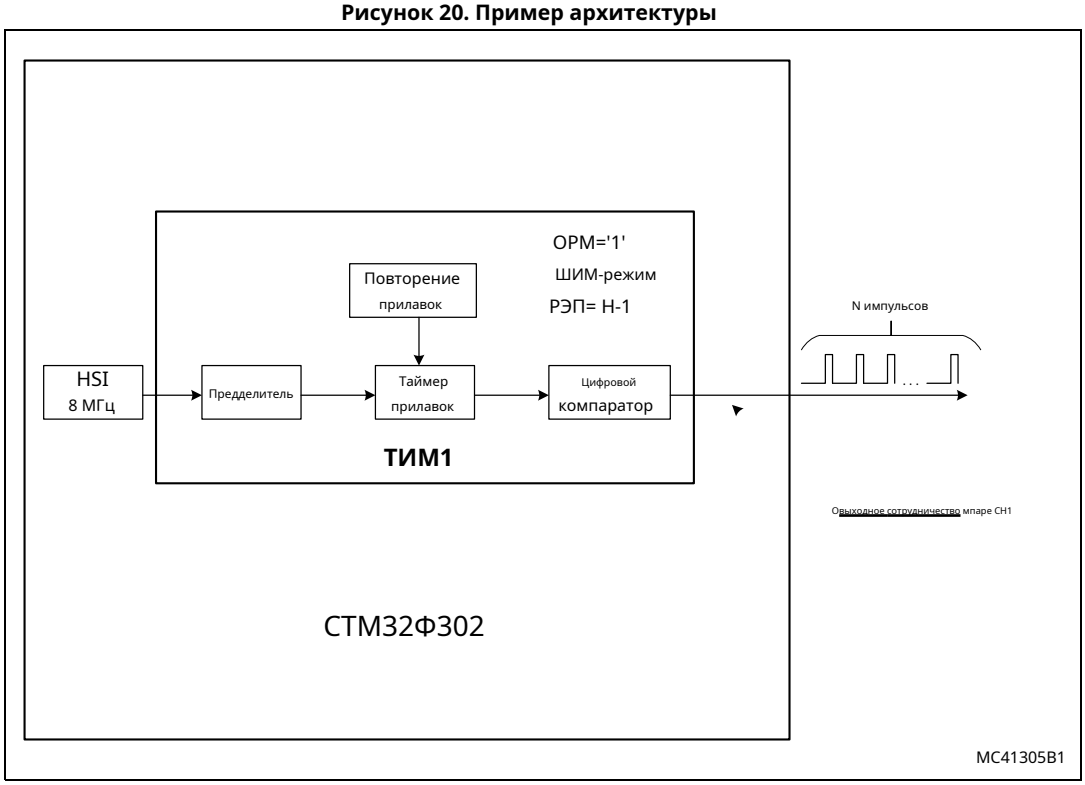
Для запуска этого приложения необходима следующая аппаратная настройка:

- Плата STM32F302 Nucleo (NUCLEO-F302R8).
- Кабель USB для подключения платы Nucleo к ПК для разработки (кабель USB также обеспечивает питание платы).
- Осциллограф для визуализации формы выходного сигнала.

Для изменения исходного кода этого приложения и отладки различных конфигураций необходим набор программных инструментов:

- Цепочка инструментов разработки, поддерживающая отладчик ST-LINK.
- Программный инструмент STM32CubeMX или более поздние версии (версия 4.10.1 была последней версией, когда разрабатывалось это приложение).

Рисунок 20. Пример архитектуры



В этом приложении таймер тактируется внутренним генератором HSI. Установите бит OPM в регистре таймера TIMx_CR1, чтобы активировать режим OPM. Для получения N-импульсов на выходах нужно настроить правильное значение (N-1) в регистре TIMx_RCR.

Инициализация системных часов

Инициализация часов выполняется в основной процедуре с помощью функции SystemClock_Config() сразу после инициализации библиотеки HAL (HAL_Init).

Код конфигурации выглядит следующим образом:

Генерация N-импульсного сигнала с использованием конфигурации одноимпульсного режима

RCC->APB2ENR |= RCC_APB2ENR_TIM1EN; /* Включение периферийных часов */

Prescaler = (uint16_t) (SystemCoreClock / 1000000) - 1;

/* Установите предварительный делитель таймера на 1 МГц в качестве тактовой частоты */

TIM1->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /* Выбор режима прямого счетчика */ TIM1->CR1 | = TIM_COUNTERMODE_UP;

TIM1->CR1 &= ~TIM_CR1_CKD

TIM1->CR1 |= TIM_CLOCKDIVISION_DIV1; /* Устанавливаем деление часов на 1*/ TIM1->ARR = PERIOD; /* Установить значение автоперезагрузки */

TIM1->CCR1 = ИМПУЛЬС; /* Установите значение импульса */ TIM1->PSC =

Prescaler; /* Установите значение предварительного делителя */

TIM1->RCR = ИМПУЛЬС_НОМЕР - 1; /* Установить значение счетчика повторений */

TIM1->EGR = TIM_EGR_UG; /* Создать событие обновления для немедленной перезагрузки прескалера и значения счетчика повторений */

TIM1->SMCR = СБРОС; /* Настройте источник внутренних часов */

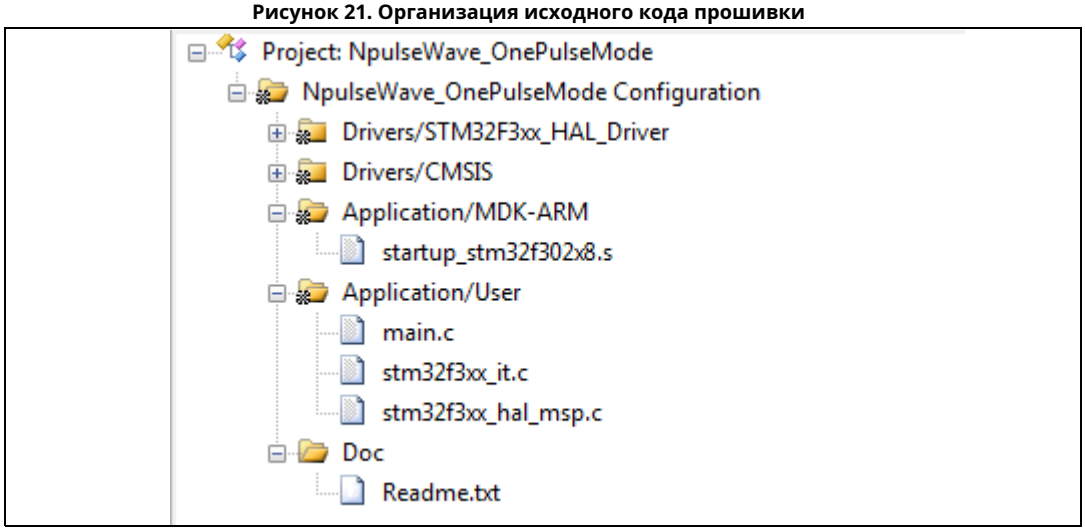
```
TIM1->CR1 |= TIM_CR1_OPM; /* Выбор режима OPM */ TIM1-
>CCMR1 &= (uint16_t)~TIM_CCMR1_OC1M; TIM1->CCMR1 &=
(uint16_t)~TIM_CCMR1_CC1S; TIM1->CCMR1 |= TIM_OCMODE_PWM2;

/* Выберите сравнение выхода канала 1 и режим */ TIM1->CCER &=
(uint16_t)~TIM_CCER_CC1P;
/* Установите для полярности сравнения выходных данных значение
High */ TIM1->CCER |= TIM_OCPOLARITY_HIGH;
TIM1->CCER = TIM_CCER_CC1E; /* Включить выходной канал сравнения 1 */ TIM1->BDTR |
= TIM_BDTR_MOE; /* Включить основной вывод TIM */ TIM1->CR1 |= TIM_CR1_CEN; /*
Включить периферийное устройство TIM */
```

3.3 Обзор прошивки

Прошивка была разработана на верстаке Keil µvision, IAR Embedded и SYSTEM WORKBENCH.

Разработанная прошивка поставляется в ZIP-файле и содержит все подкаталоги и файлы исходного кода «.h» и «.c», которые составляют ядро приложения.

Рисунок 21. Организация исходного кода прошивки



Прошивка содержит все исходные файлы задач приложения и связанные с ними файлы и состоит из следующих папок проекта:

- Библиотека STM32 MCU HAL
- Файл запуска
- Прикладной уровень.

4 Поцикловое регулирование с использованием входа прерывания

4.1 Обзор

Функция Break доступна в периферийных устройствах таймера расширенной конфигурации, таких как таймеры TIM1 и TIM8, а также в периферийных устройствах таймера облегченной конфигурации, таких как таймеры TIM15, TIM16 и TIM17. Функция Break в основном используется для защиты силового каскада, управляемого выходами таймера; он отключает или переводит их в предопределенное безопасное состояние, как только что-то пойдет не так в каскаде питания или в самом микроконтроллере.

Для обнаружения внешних запросов на прерывание, генерируемых силовым каскадом, функция прерывания связана с выделенным входом (например, BKIN или BKIN2), который отображается как альтернативная функция на несколько входов/выходов микроконтроллера. После включения функция останова отключает выходы ШИМ или переводит их в предопределенное безопасное состояние, как только обнаруживается событие останова, даже если часы отсутствуют. Например, возможно асинхронное отключение выходов.

Справочное руководство содержит дополнительную информацию о том, как настроить безопасные предопределенные состояния для выходов таймера.

Как только обнаружено допустимое событие прерывания, управляющий бит MOE в регистре TIMx_BDTR асинхронно сбрасывается. Он действует только на каналы таймера, сконфигурированные в режиме вывода. Чтобы возобновить нормальное рабочее состояние выходов канала таймера, бит управления MOE должен быть установлен программно или должен быть установлен бит управления AOE, чтобы выходы таймера возобновили свое нормальное рабочее состояние к началу нового ШИМ. цикл.

Некоторые семейства микроконтроллеров STM32 (например, семейство STM32F303) имеют встроенные периферийные устройства таймера с двумя внешними входами прерывания: BKIN и BKIN2. Для этих периферийных устройств таймера вход прерывания BRK2 имеет более низкий приоритет, чем вход прерывания BRK. Входы прерывания BKIN и BKIN2 могут отключать выходы канала таймера только при принудительном переводе их в состояние Hi-Z, и они не могут принудительно переводить их в предопределенное безопасное состояние.

4.2 Разрыв ввода по сравнению с использованием OCxRef-clear

Как упоминалось в предыдущем разделе, вход прерывания используется в основном для управления выходами канала таймера в условиях неисправности. Благодаря своей гибкой конструкции вход прерывания можно использовать и в других случаях, например, для поцикловой регулировки тока. Концептуально функция поциклового регулирования тока выполняется функцией OCxRef-clear периферийных таймеров STM32.

В некоторых ситуациях невозможно использовать функцию OCxRef-clear для управления функцией поциклового регулирования тока. Это связано с тем, что вход таймера ETR, используемый этой функцией, может совпадать с другой функцией таймера (например, внешняя синхронизация таймера использует вход таймера ETR для внешних часов). В такой ситуации может быть полезно использовать функцию Break для управления функцией поциклового регулирования тока.

С одной стороны, функция очистки OCxRef воздействует на выходы канала таймера (что также относится к функции Break); с другой стороны, функция OCxRef может быть активирована для каждого канала (например, активация функции очистки OCxRef осуществляется поканально посредством установки соответствующего управляющего бита OCxCE для регулируемых каналов). В этой второй ситуации функция паузы действует на все выходы канала таймера (например, нет возможности контролировать, на какой канал влияет событие паузы).

Концепция поциклового регулирования тока заключается в том, что как только величина регулируемого тока превышает заданный порог, сигнал ШИМ становится низким до начала следующего цикла ШИМ. Это поведение изначально поддерживается функцией OCxREf-clear и показано на *Рисунок 22*.

Для функции Break это поведение управляется битом управления AOE. Если бит управления AOE сохраняется в состоянии по умолчанию (состояние сброса), то, как только регулируемый ток пересекает предопределенный порог, выходы таймера становятся низкими. Таймер остается в этом состоянии до тех пор, пока бит MOE не будет установлен программным обеспечением, что не является типичным поведением логики поциклового регулирования.

Установка бита управления AOE перед активацией функции регулирования с помощью функции Break автоматически устанавливает бит управления MOE в начале каждого нового цикла ШИМ. Ранее описанное поведение функции Break по сравнению с конфигурацией битов управления AOE показано на *Рисунок 23* где построена форма сигнала на выходе одного канала таймера.

Рисунок 22. Время очистки TIMx OCxREF

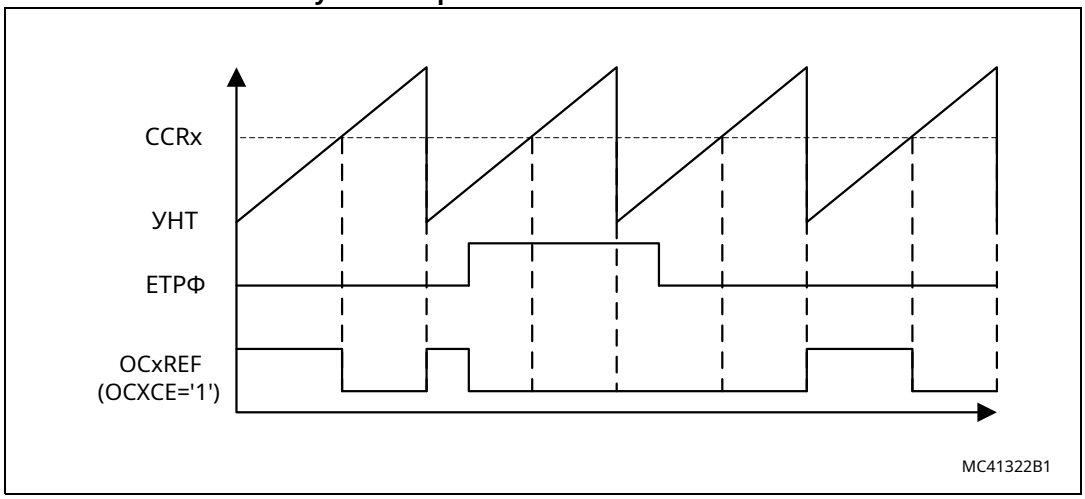


Рисунок 23. Функция времени перерыва

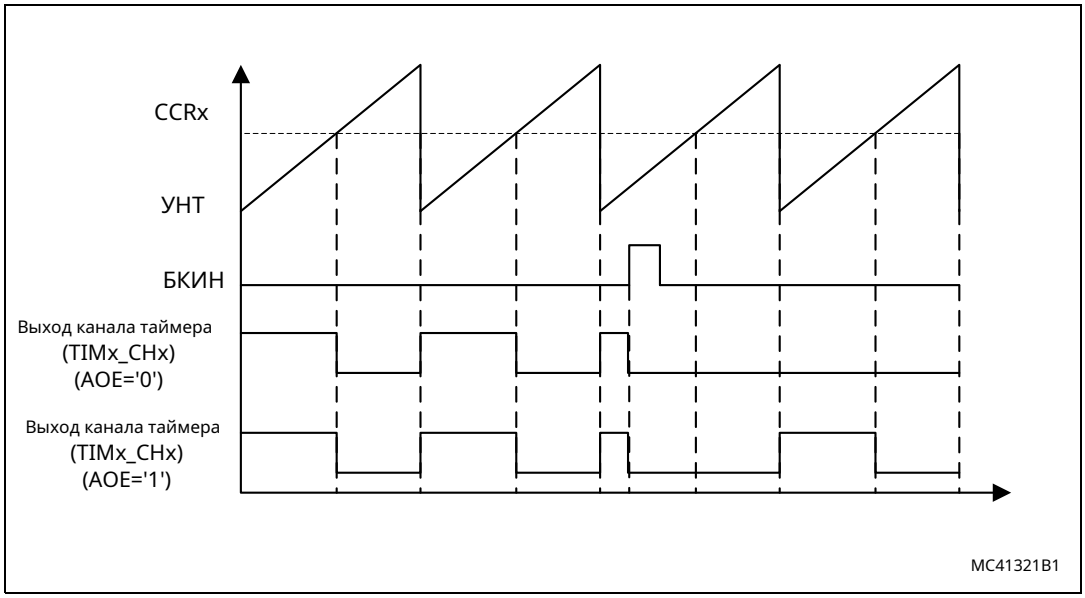


*Рисунок 23*иллюстрирует различное поведение выхода таймера между двумя конфигурациями бита управления AOE.

- Если AOE = '0', выход PWM отключен, даже если вход прерывания не активен.
- Если AOE = '1', выход ШИМ включается при следующем событии обновления, если вход прерывания не активен.

4.3 Применение: поцикловое регулирование с использованием функции паузы.

Поцикловое регулирование является одним из применяемых методов реализации защиты от перегрузки по току на уровне ступеней преобразования мощности. Эта функция контролирует ток, потребляемый силовым каскадом. Как только ток превысит заданный порог ($I_{Ссылка}$), сигнал ШИМ становится низким, чтобы уменьшить потребляемый ток ниже порогового значения в эффекте регулирования.

Сигнал ШИМ не может возобновить свой сконфигурированный рабочий цикл до начала нового цикла ШИМ. Если потребляемый ток все еще выше порогового значения, сигнал ШИМ остается в низком состоянии до следующего цикла ШИМ и так далее.

Сравнение тока, потребляемого силовым каскадом, с заданным пороговым током может быть реализовано с помощью электронной схемы на основе компаратора. Сам компаратор может быть встроен в микроконтроллер; (например, это касается микроконтроллеров STM32F302 и STM32F303). Выход этой схемы на основе компаратора должен подаваться на вход прерывания и, соответственно, на вход ETR в случае, когда используется функция очистки OCxRef.

Рисунок 24. Временные характеристики поциклового регулирования

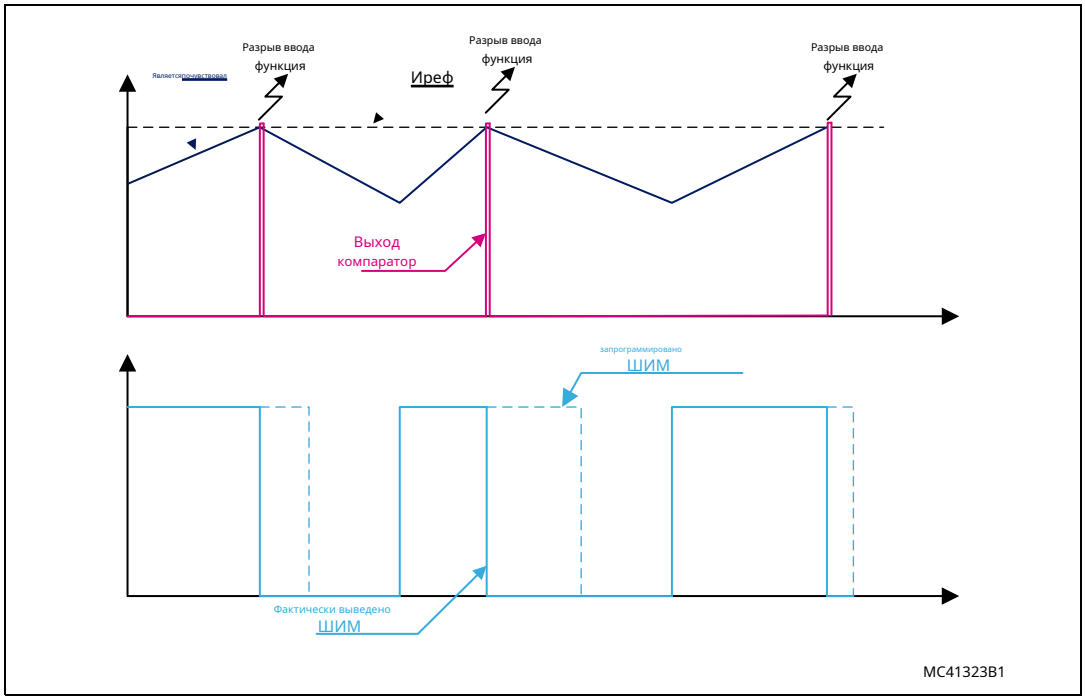
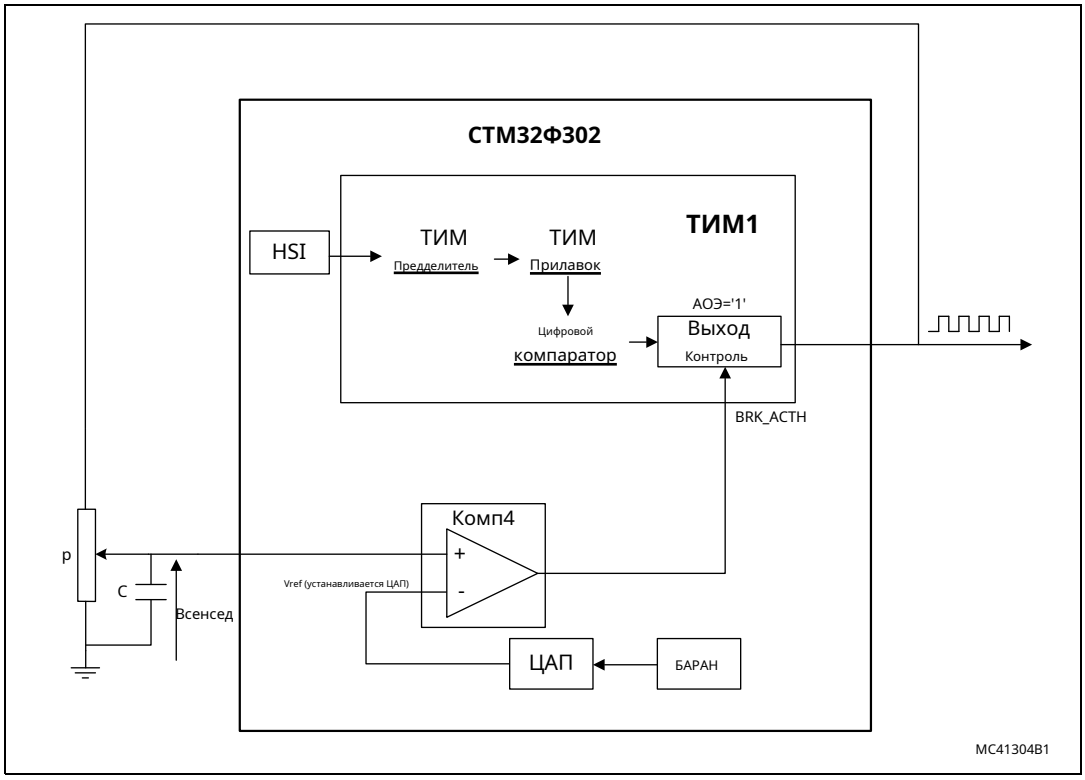


*Рисунок 24*показывает поведение функции прерывания, когда прерывание включено. Он показывает как фактически выведенное значение ШИМ (сплошная линия), так и запрограммированное значение ШИМ (пунктирная линия).

Функция прерывания переводит выходной канал в низкий уровень, подавая высокий уровень на вход BRK_ACTH. Функция прерывания остается на низком уровне до следующего «события обновления», поскольку установлен бит управления AOE.

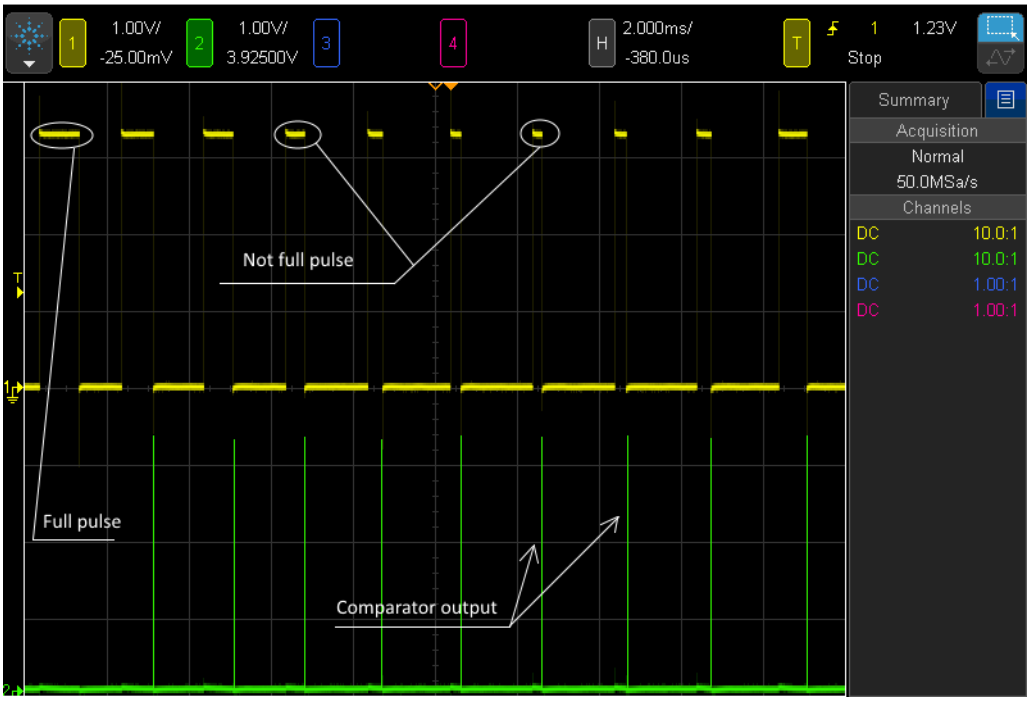
*Рисунок 25*показывает эмуляцию концепции поциклового регулирования с использованием функции паузы. Простая электронная схема, состоящая из пассивных электронных компонентов (резистора и конденсатора), использовалась для имитации сигналов обратной связи, выдаваемых реальной регулируемой системой инвертирования мощности.

Рисунок 25. Архитектура поциклового регулирования



Как показано в*Рисунок 25*, таймер используется в режиме ШИМ, Vref хранится в ОЗУ и сравнивается с входным напряжением после преобразования с помощью ЦАП и аналогового компаратора. Выход компаратора внутренне перенаправляется на сигнал BRK_ACTH, который управляет состоянием ШИМ.

Рис. 26. Скриншот полученной осциллограммы на осциллографе



Поцикловое регулирование является одним из применяемых методов реализации защиты от перегрузки по току на уровне ступеней преобразования мощности.*Рисунок 26*показывает изменение значения рабочего цикла в зависимости от времени из-за изменения входного напряжения относительно Vref.

На этом рисунке во время первого импульса ШИМ напряжение достигает значения «предел Vref» до того, как заканчивается входной импульс ШИМ. Выходной импульс ШИМ отключается в начале своего цикла. Затем напряжение спадает до момента подачи следующего импульса, который снова вызывает повышение напряжения. Во время второго импульса ШИМ ограничение тока не происходит, поскольку импульс заканчивается до того, как нарастающий ток достигает порога «ограничения тока».

Для запуска этого приложения необходима следующая аппаратная настройка:

- Плата STM32F302 Nucleo (NUCLEO-F302R8)
- Кабель USB для подключения платы nucleo к ПК для разработки (кабель USB также обеспечивает питание платы).
- Осциллограф для визуализации формы выходного сигнала.
- Потенциометр и конденсатор

Чтобы изменить исходный код этого приложения и отладить другую конфигурацию, необходим набор программных инструментов:

- Цепочка инструментов разработки, поддерживающая отладчик ST-LINK.
- Программный инструмент STM32CubeMX версии 4.10.1 или более поздних версий (версия 4.10.1 была последней версией на момент разработки этого приложения).

Инициализация системных часов

Инициализация часов выполняется в основной процедуре с помощью функции SystemClock_Config() сразу после инициализации библиотеки HAL (HAL_Init).

Конфигурация функции останова

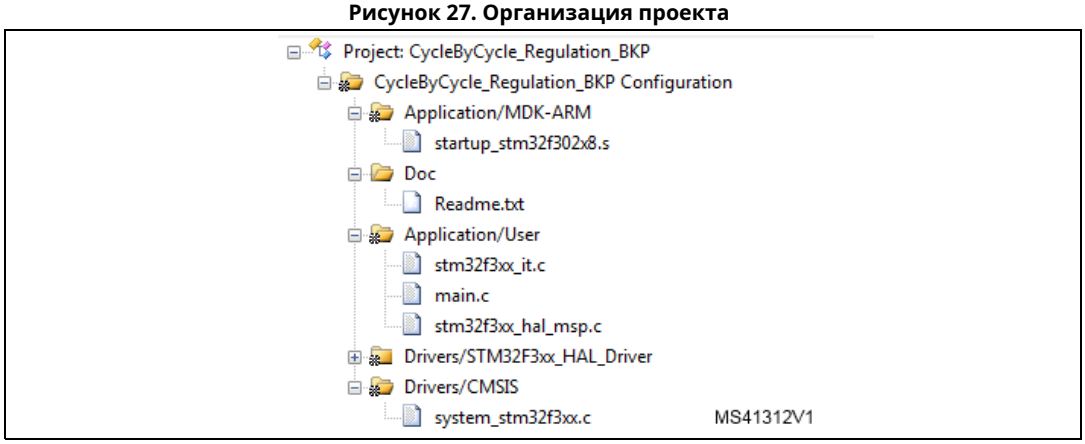
```
RCC->APB2ENR |= RCC_APB2ENR_TIM1EN; /* Включение часов TIM1 */
Предделитель = (uint16_t) (SystemCoreClock/500000) - 1; /* Установите предварительный
делитель таймера на 500 кГц в качестве тактовой частоты */
TIM1->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /* Выбор режима обратного счетчика*/ TIM1->CR1
|= TIM_COUNTERMODE_UP;
TIM1->CR1 &= ~TIM_CR1_CKD;
TIM1->CR1 |= TIM_CLOCKDIVISION_DIV1; /* Устанавливаем деление часов на 1*/ TIM1-
>ARR = PERIOD;                /* Установить значение автоперезагрузки */
TIM1->CCR1 = ИМПУЛЬС;         /* Установите значение регистра сравнения захвата */
TIM1->PSC = Предварительный делитель; /* Установите значение предварительного делителя */
TIM1->EGR = TIM_EGR_UG; /* Генерация события обновления для немедленной перезагрузки
прескалера и значения счетчика повторений */
TIM1->SMCR = СБРОС; /*Настройте источник внутренних часов */ TIM1->BDTR
= RESET; /* Очистить биты BDTR */
TIM1->BDTR |= DEAD_TIME; /* Установите значение Dead Time в 0 */ TIM1-
>BDTR |= TIM_LOCKLEVEL_OFF;          /* Отключение уровня блокировки*/ /*
TIM1->BDTR |= TIM_OSSI_ENABLE; TIM1-  Включение режима ожидания вывода */ /*
>BDTR |= TIM_OSSR_DISABLE;           Отключение режима запуска вывода */
TIM1->BDTR |= TIM_BREAK_ENABLE; /* Включить вход Break */ TIM1->BDTR |=
TIM_BREAKPOLARITY_HIGH; /* Установить полярность на высокий */ TIM1->BDTR |=
TIM_AUTOMATICOUTPUT_ENABLE; /* Включить автоматический вывод */
TIM1->CCMR1 &= ~TIM_CCMR1_OC1M; /* Выбор выхода канала 1. Сравнение и режим.
* /
TIM1->CCMR1 &= ~TIM_CCMR1_CC1S;
TIM1->CCMR1 |= TIM_OCMODE_PWM1;
TIM1->CCER &= ~TIM_CCER_CC1P; /* Установите для полярности сравнения выходных данных значение High
*/ TIM1->CCER |= TIM_OCPOLARITY_HIGH;
TIM1->CCER |= TIM_CCER_CC1E; /* Включить выходной канал сравнения 1 */ TIM1->CR1|
=TIM_CR1_CEN; /* Включить периферийное устройство TIM */
```

4.4 Обзор прошивки

Прошивка была разработана на Keil µvision, рабочей среде IAR Embedded и СИСТЕМНЫЙ ВЕРСТАК.

Разработанная прошивка поставляется в ZIP-файле и содержит все подкаталоги и файлы исходного кода .h и .c, составляющие ядро приложения.

Рисунок 27. Организация проекта



Прошивка содержит все исходные файлы задач приложения и связанные с ними файлы и состоит из следующих папок проекта:

- Библиотека STM32 MCU HAL
- Файл запуска
- Прикладной уровень.

5 Генерация сигналов произвольной формы с использованием функции таймера DMAburst

5.1 Обзор функции STM32 DMA-burst

Периферийный модуль с прямым доступом к памяти (DMA) используется для обеспечения высокоскоростной передачи данных как между периферийными устройствами и памятью, так и между памятью и памятью. Это действие экономит ресурсы ЦП, которые можно было бы использовать в других задачах.

Каждая передача DMA состоит из двух этапов:

- На первом этапе данные для передачи загружаются из исходного местоположения.
- На втором этапе полученные данные сохраняются в месте назначения.

Эта двухэтапная операция передачи данных связана с обновлением индексного регистра передачи периферийных устройств DMA; это регистр, используемый для отслеживания того, сколько данных осталось передать.

В семействах микроконтроллеров STM32 есть два варианта периферийных устройств DMA:

- Функция пакетной передачи DMA поддерживается только вариантом периферийного устройства DMA продуктов STM32F2, где периферийное устройство DMA может передавать настраиваемое количество элементов данных рядом с одним триггером передачи данных.
- В другом варианте, таком как продукты STM32F1, периферийный вариант DMA поддерживает только одиночные передачи; это означает, что только один элемент данных передается рядом с триггером передачи данных.

Вышеприведенный абзац не предназначен для подробного описания функции STM32 DMA-burst, поддерживаемой многими микроконтроллерами STM32. Информация, изложенная выше, направлена на то, чтобы смягчить любое неправильное понимание этой функции и любую возможную путаницу с функцией пакетной передачи таймера STM32, которая находится в центре внимания этой главы.

Дополнительные сведения о периферийном устройстве STM32 DMA см. в справочных руководствах по документации микроконтроллеров STM32 и примечаниях по применению. *Обзор межсерийного таймера STM32* (AH4013).

5.2 Функция таймера DMA-burst

Периферийные устройства таймера могут генерировать несколько последовательных запросов прямого доступа к памяти после одного события таймера. Основное использование этой функции заключается в обновлении содержимого нескольких регистров периферийного таймера каждый раз, когда запускается данное событие таймера. Это может быть сделано либо для динамической перенастройки режима работы таймера (переключение с одного режима вывода на другой, например, из режима ШИМ2 в режим принудительно-активного уровня), либо для изменения рабочих параметров одновременно на нескольких каналах (изменение рабочие циклы для более чем одного канала таймера одновременно).

Эту же функцию можно также использовать для передачи содержимого нескольких периферийных регистров таймера в буфер памяти.

Эта функция позволяет изменять на лету форму сигнала, выдаваемого выходом периферийного устройства таймера, путем настройки содержимого регистров периферийного устройства таймера. Например, он позволяет обновить содержимое регистра TIMx_ARR для настройки частоты выходного сигнала или обновить регистр TIMx_CCRx для настройки коэффициента заполнения сигнала.

Чтобы использовать функцию DMA-burst таймера, программист должен иметь дело со следующими периферийными регистрами таймера:

– Регистр адреса DMA (TIMx_DMAR): это регистр перенаправления доступа для чтения/записи.

– Регистр управления DMA (TIMx_DCR): это регистр управления машиной состояния пакетной передачи.

– Настройка регистра периферийного адреса контроллера прямого доступа к памяти.

Адресный регистр функции DMA-burst периферийного устройства таймера

Регистр периферийного адреса в периферийном DMA используется для настройки адреса регистра назначения передачи, когда DMA сконфигурирован в режиме «память-периферия». Он также используется для настройки адреса регистра источника передачи, когда DMA настроен в режиме периферийных устройств в память.

Периферийное устройство DMA должно обновлять содержимое ряда периферийных регистров таймера содержимым буфера памяти с предопределенными и/или предварительно вычисленными значениями, но это не означает, что адрес передачи, указываемый адресным регистром периферийного устройства DMA, должен быть сконфигурирован как постинкрементируется контроллером прямого доступа к памяти после каждой передачи данных. Регистр периферийного адреса контроллера DMA должен указывать на периферийный регистр таймера TIMx_DMAR.

Регистр таймера TIMx_DMAR является виртуальным регистром. Любой доступ к этому регистру перенаправляется логикой управления DMA-пакетом периферийного таймера на один из физических регистров периферийного таймера.

Доступ к регистру TIMx_DMAR может быть как для чтения, так и для записи. Перенаправление фактического доступа к регистру TIMx_DMAR на другой периферийный физический регистр таймера зависит от содержимого регистра TIMx_DCR в настройках интерфейса DMA-burst. Это также зависит от фактического состояния конечного автомата (FSM), управляющего интерфейсом пакета DMA таймера.

Настройка регистра адреса памяти контроллера DMA

Регистр адреса памяти в периферийном устройстве прямого доступа к памяти используется для настройки адреса места назначения передачи в памяти, когда устройство прямого доступа к памяти сконфигурировано в режиме «периферийное устройство-память». Он также используется для настройки адреса местоположения памяти источника передачи, когда DMA настроен в режиме памяти-периферии.

Периферийное устройство прямого доступа к памяти должно обновлять содержимое ряда регистров периферийного устройства таймера содержимым буфера памяти. Адрес передачи, указанный в регистре адреса памяти DMA, должен быть сконфигурирован так, чтобы он увеличивался контроллером DMA после каждой передачи данных.

Регистр управления DMA-burst периферийного устройства таймера

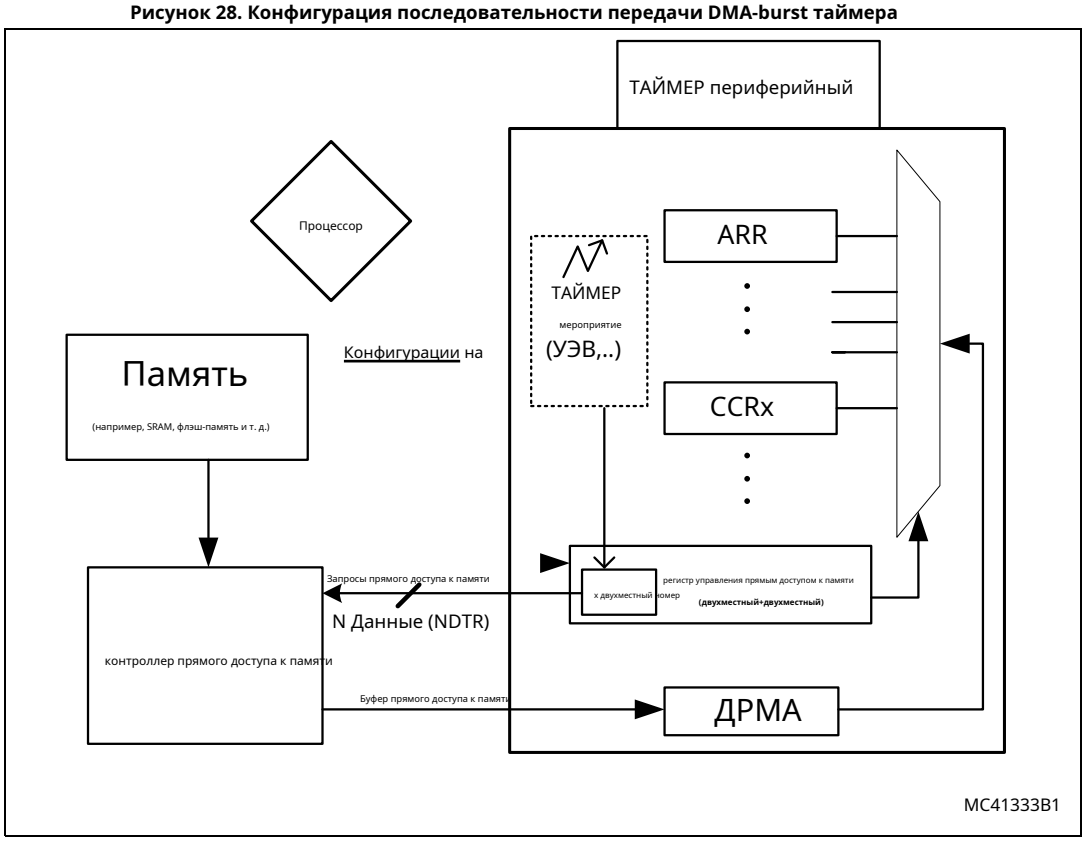
Регистр управления конечного автомата для функции пакетной передачи DMA таймера, TIMx_DCR, используется для настройки количества тактов во время передачи одного пакета. Он также используется для идентификации целевого регистра таймера или источника первой передачи данных во время пакетной передачи.

Управляющее битовое поле DBL [4:0] устанавливает количество тактов во время одной пакетной передачи, которое должно быть равно количеству регистров таймера, которые участвуют (либо для записи, либо для чтения) в пакетной передаче. Содержимое управляющего битового поля DBA [4:0] идентифицирует начальный регистр, участвующий в пакетной передаче между регистрами таймера.

Битовое поле управления DBA[4:0] может идентифицировать до 32 регистров таймера, поскольку оно имеет 5-битную ширину. Идентификационный номер регистров таймера получается путем деления относительного адреса регистра в карте регистров таймера на четыре. Например, регистр TIMx_CR1 имеет относительный адрес 0x00, тогда его идентификационный номер равен 0.

Чтобы пакетная передача начиналась с регистра TIMx_CR1, битовое поле управления DBA[4:0] должно быть установлено на 0. Чтобы пакетная передача начиналась с регистра TIMx_ARR, битовое поле DBA[4:0] должно быть установлено на 11. Это означает, что 44 / 4 = 11, где 44 — это относительный адрес регистра TIMx_ARR, 0x2C, после кодирования в десятичную систему счисления.

Рисунок 28. Конфигурация последовательности передачи DMA-burst таймера



Чтобы завершить одну последовательность передачи DMA-burst таймера, как показано *Рисунок 28*, DMA и периферийные устройства таймера должны взаимодействовать. Значения данных, используемые для обновления регистров таймера, должны храниться где-то в памяти микроконтроллера. Их можно сохранить в памяти статического ОЗУ, если шаблон необходимо обновить во время генерации сигнала.

Прикладное программное обеспечение должно настроить DMA так, чтобы он указывал на буфер данных как на источник передачи данных; он должен указывать на регистр таймера TIMx_DMAR как на место назначения передачи данных. Наконец, прикладное программное обеспечение должно настроить функцию DMA-burst таймера, записав правильные настройки в регистр таймера TIMx_DCR.

Вышеприведенные абзацы содержат подробное описание того, как настроить функцию пакетного DMA таймера. Как только будет выполнена правильная конфигурация, поток или канал DMA, используемый для передачи данных, должен быть включен, а затем включен счетчик таймера.

После включения счетчик таймера периодически обновляется, увеличиваясь или уменьшаясь. Через некоторое время, в зависимости от включенных запросов DMA таймера, периферийное устройство таймера может вызвать внутренний запрос DMA. Запрос DMA направляется внутренне к логике управления пакетом DMA таймера.

На основе значения, настроенного для длины пакета DMA (DBL[4:0], битовое поле в регистре таймера TIMx_DCR), запрос DMA отправляется на периферийное устройство DMA либо как есть, либо несколько раз умножается. [4:0] содержимое равно нулю, тогда запрос DMA отправляется как есть; в противном случае запрос DMA умножается на значение DBL + 1 коэффициент.

Если содержимое битового поля DBL[4:0] равно 2, то, как только внутренний запрос DMA таймера будет инициирован, логика управления пакетом DMA таймера отправит первый запрос DMA на периферийное устройство DMA. Периферийное устройство прямого доступа к памяти передает содержимое ячейки памяти; это место указывается источником передачи регистра DMA в регистр таймера TIMx_DMAR. Затем он увеличивает указатель источника и подтверждает запрос DMA таймера.

Как только получено первое подтверждение DMA, логика управления пакетом DMA таймера отправляет второй запрос DMA. Этот запрос DMA снова обрабатывается периферийным устройством DMA, и на таймер снова отправляется подтверждение.

После получения второго подтверждения DMA логика управления пакетом DMA таймера отправляет третий запрос DMA. Этот третий запрос снова обрабатывается периферийным устройством прямого доступа к памяти.

Как только таймер получает третье подтверждение от DMA, последовательность передачи, запущенная внутренним запросом DMA, считается успешно завершенной. Затем логика управления пакетом DMA таймера готова к новой последовательности передачи.

Для описанной выше последовательности передачи данных регистр назначения передачи, на который указывает периферийное устройство прямого доступа к памяти, остается одним и тем же на протяжении всей последовательности передачи. В этом случае регистр остается равным регистру таймера TIMx_DMAR.

Логика управления DMA-burst таймера каждый раз перенаправляет доступ для записи в регистр TIMx_DMAR таймера к правому физическому регистру таймера.

Для предыдущего примера, и если битовое поле базового адреса DMA DBA[4:0] в регистре TIMx_DCR установлено на 11, обратите внимание, что:

- Первый доступ к записи DMA в регистр таймера TIMx_DMAR перенаправляется в регистр таймера TIMx_ARR. Регистр таймера TIMx_ARR выбирается в качестве базового адреса для пакетной передачи.
- Второй доступ DMA к регистру таймера TIMx_DMAR перенаправляется на регистр таймера TIMx_RCR (при условии, что таймер, используемый в этом примере, включает регистр TIMx_RCR).
- Третий доступ DMA к регистру таймера TIMx_DMAR перенаправляется на регистр таймера TIMx_CCR1.
- В конце третьего доступа DMA к регистру TIMx_DMAR логика управления пакетом DMA таймера зацикливается на автомате перенаправления доступа для записи. Он снова указывает на сконфигурированный базовый регистр для пакетной передачи (в данном примере регистр таймера TIMx_ARR).

Для каждого нового запроса DMA, сгенерированного внутри периферийного таймера, описанная выше последовательность повторяется.

Чтобы использовать пакет DMA таймера для периодического считывания содержимого периферийных регистров таймера в определенный буфер памяти, приведенная выше последовательность также допустима. Должны быть изменены только направление передачи DMA, адрес источника и адрес назначения.

5.3 Пример применения: генерация сигналов произвольной формы с использованием функции DMA-burst таймера

Этот пример демонстрирует одно возможное использование периферийных устройств таймера STM32 для генерации сигнала произвольной формы без накладных расходов ЦП. Этот пример также направлен на то, чтобы демистифицировать функцию таймера DMA-burst, которая является краеугольным камнем для этого примера.

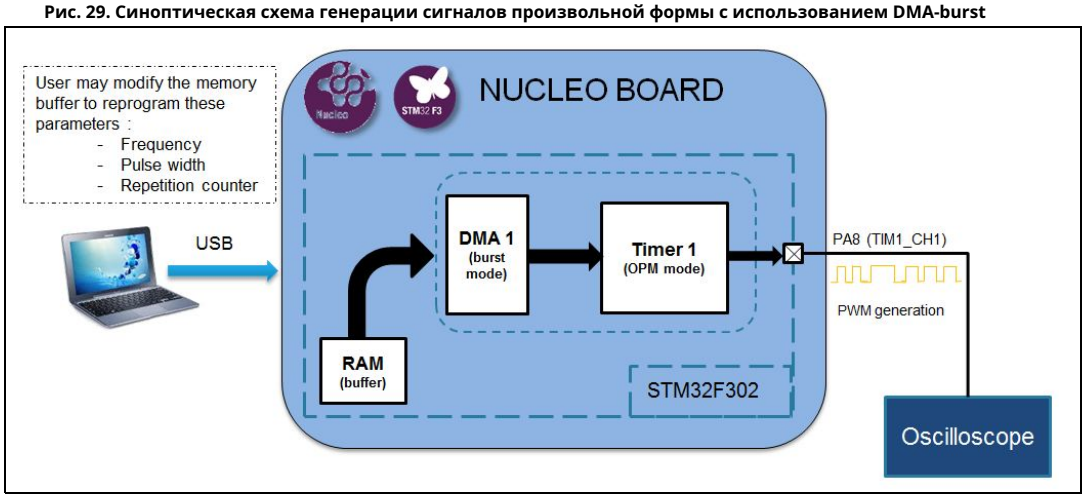
Обзор приложений

Это приложение разработано для платы Nucleo NUCLEO-F302R8, основанной на микроконтроллере STM32F302x8, в качестве основного контроллера платы. Тем не менее, это приложение можно легко портировать на любую аппаратную платформу STM32.

В плату Nucleo встроен собственный отладчик ST-Link/V2. Для подключения платы Nucleo к ПК требуется только USB-кабель как для загрузки двоичного образа приложения в микроконтроллер, так и для целей отладки приложения.

*Рисунок 29*иллюстрирует сводную диаграмму для приложения генератора сигналов произвольной формы, представленного в этом примере приложения. Сгенерированный сигнал выводится на PA.08 IO микроконтроллера STM32F302, который отображается на контакт 8 разъема CN9 платы Nucleo.

Рис. 29. Синоптическая схема генерации сигналов произвольной формы с использованием DMA-burst

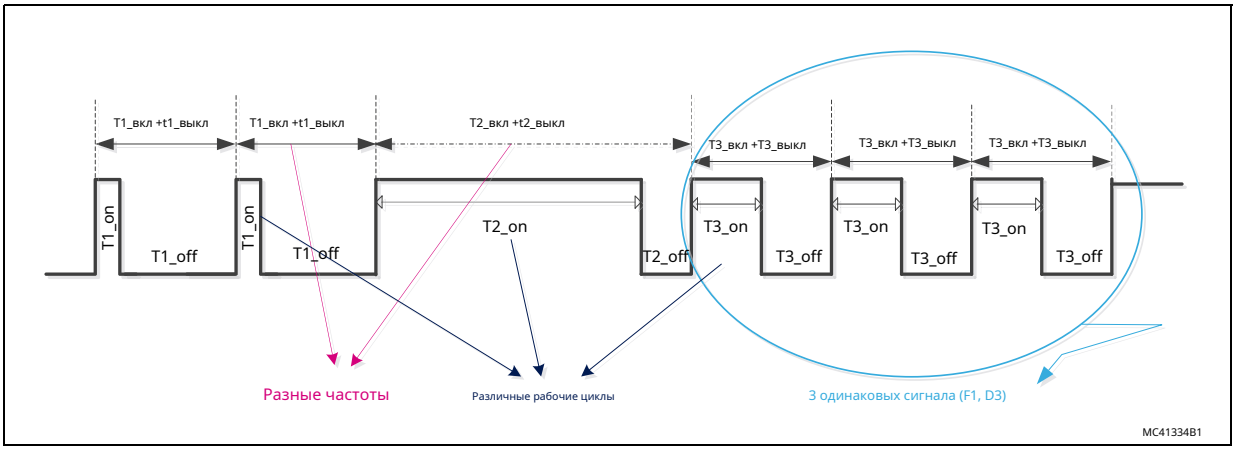


Генератор сигналов произвольной формы, описанный в этом примере, предназначен для вывода сигнала, подобного показанному на рис.*Рисунок 30*. Исходный код этого приложения можно легко изменить для вывода различных сигналов, но для демонстрационных целей тот, который показан на рис.*Рисунок 4 на странице 12*нацелен.

Целевая форма волны в*Рисунок 30*состоит из трех частей сигнала:

- Первая часть состоит из двух последовательных импульсов с определенным временем включения и выключения: t1_on и t1_off.
- Вторая часть состоит из одного импульса с разным временем включения и выключения: t2_on и t2_off.
- Третья часть состоит из трех импульсов с третьим набором параметров времени включения и выключения: t3_on и t3_off.

Рис. 30. Приложение генератора сигналов произвольной формы: целевая форма сигнала



Чтобы заставить периферийное устройство таймера STM32 выводить этот сигнал, можно использовать периферийное устройство таймера со счетчиком повторений. Периферийный таймер включает в себя регистр TIMx_RCR и имеет как минимум один канал таймера. В этом примере используется периферийное устройство таймера TIM1, поскольку оно имеет четыре канала таймера, а также счетчик повторений.

Канал таймера может генерировать ШИМ-сигнал с постоянной предопределенной частотой и параметрами рабочего цикла. Параметр частоты сигнала управляется содержимым регистра автоперезагрузки таймера TIMx_ARR. В этом регистре параметр рабочего цикла управляется содержимым регистра канала 1 таймера (TIM1_CCR1). Чтобы канал таймера выдавал запрошенную форму волны (см. *Рисунок 4*), содержимое этих двух регистров таймера в конце каждого цикла ШИМ должно обновляться.

Другими словами, требуется обновлять содержимое этих двух регистров синхронно с каждым «событием обновления» таймера. Интуитивно понятный способ сделать это — заставить прикладное программное обеспечение (модуль ЦП) обновлять содержимое этих двух регистров каждый раз, когда возникает «событие обновления» таймера. Интуитивно понятно, что это нагружает процессор и делает выходной сигнал недетерминированным.

В следующих абзацах объясняется, как использовать некоторые функции таймера в сочетании с периферийным устройством прямого доступа к памяти, чтобы снизить нагрузку на ЦП и генерировать сигнал детерминированной формы.

Чтобы обновить регистры таймера TIMx_ARR и TIMx_CCR1 одновременно рядом с одним «событием обновления» таймера, следует использовать функцию таймера DMA-burst в дополнение к одному каналу или потоку DMA. Как настроить и использовать функцию DMA-burst таймера, описано в предыдущих разделах этого документа.

Чтобы канал таймера не генерировал нежелательные сбои из-за обновления регистра канала таймера, используется функция предварительной загрузки периферийных устройств таймера STM32.

Чтобы канал таймера выдавал повторяющиеся импульсы с одинаковыми параметрами t_on и t_off и без того, чтобы таймер генерировал «событие обновления» на каждом цикле ШИМ, следует использовать счетчик повторений таймера.

Для вывода сигнала, показанного на *Рисунок 29* на канале 1 периферийного устройства таймера TIM1 содержимое следующих трех регистров должно быть обновлено следующим образом:

- **TIM1_ARR:** содержит сумму периодов t_on и t_off для продолжающегося импульса.
- **TM1_RCR:** содержит количество импульсов на часть текущего сигнала минус один (т.е. TIM1_RCR=Number_of_pulses_per_partion–1).
- **TIM1_CCR1:** содержит длительность периода t_on продолжающегося импульса.

Конфигурация регистров таймера, которая будет использоваться для восстановления сигнала, показанного на рис.*Рисунок 29*следующие:

Для первой части сигнала правильная конфигурация:

- TIM1_ARR = t1_on + t1_off, TIM1_RCR= 1, TIM1_CCR1= t1_on;

Для второй части сигнала правильная конфигурация:

- TIM1_ARR = t2_on + t2_off, TIM1_RCR= 0, TIM1_CCR1= t2_on;

Для третьей части сигнала правильная конфигурация:

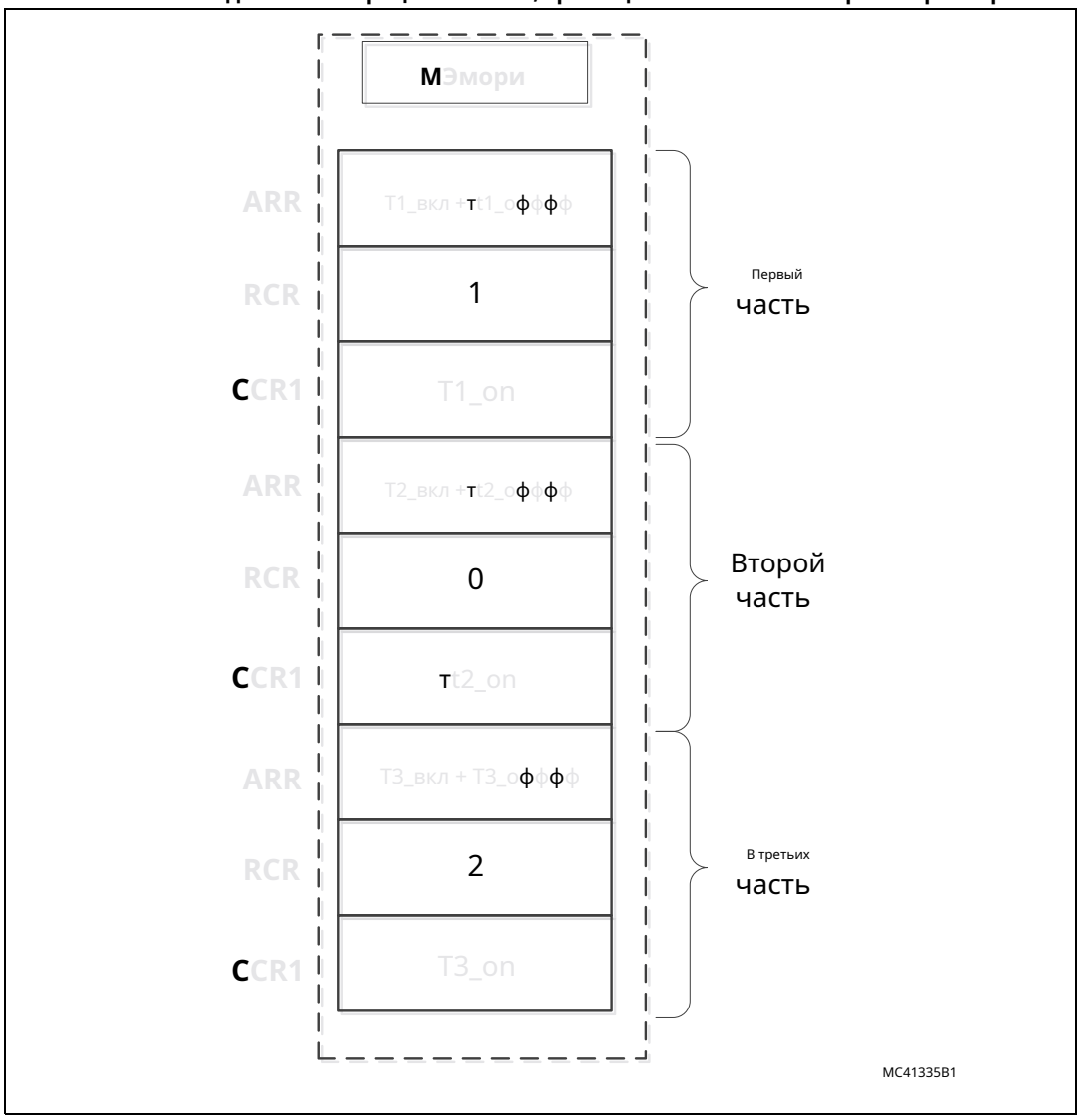
- TIM1_ARR = t3_on + t3_off, TIM1_RCR= 2, TIM1_CCR1= t3_on;

Содержимое буфера памяти:

Принимая во внимание приведенные выше значения для регистров таймера TIM1_ARR, TIM1_RCR и TIM1_CCR1 для каждой части целевого сигнала, см.*Рисунок 31*чтобы таблица значений определялась в памяти микроконтроллера.

Обратите внимание, что порядок значений данных в таблице такой же, как и для их соответствующих регистров в карте регистров периферийного таймера. Правильный порядок должен быть TIM1_ARR, TIM1_RCR, а затем регистр TIM1_CCR1.

Рис. 31. Шаблон данных генерации сигнала, хранящийся в памяти микроконтроллера



Исходный код на языке C для вышеописанной таблицы значений приведен ниже. *Постоянная*ключевое слово используется для указания того, что содержимое таблицы не изменяется на лету во время генерации сигнала.

Массив памяти размещается во Flash-памяти. Если требуется изменить параметры сигнала во время выполнения, ключевое слово*Постоянная*не следует использовать. Массив памяти распределяется в двух областях памяти: области флэш-памяти и области памяти SRAM.

```
uint32_t const aSRC_Buffer[9] = { t1_on+t1_off,1,t1_on,
t2_on+t2_off,0,t2_on, t3_on+t3_off,2,t3_on};
```

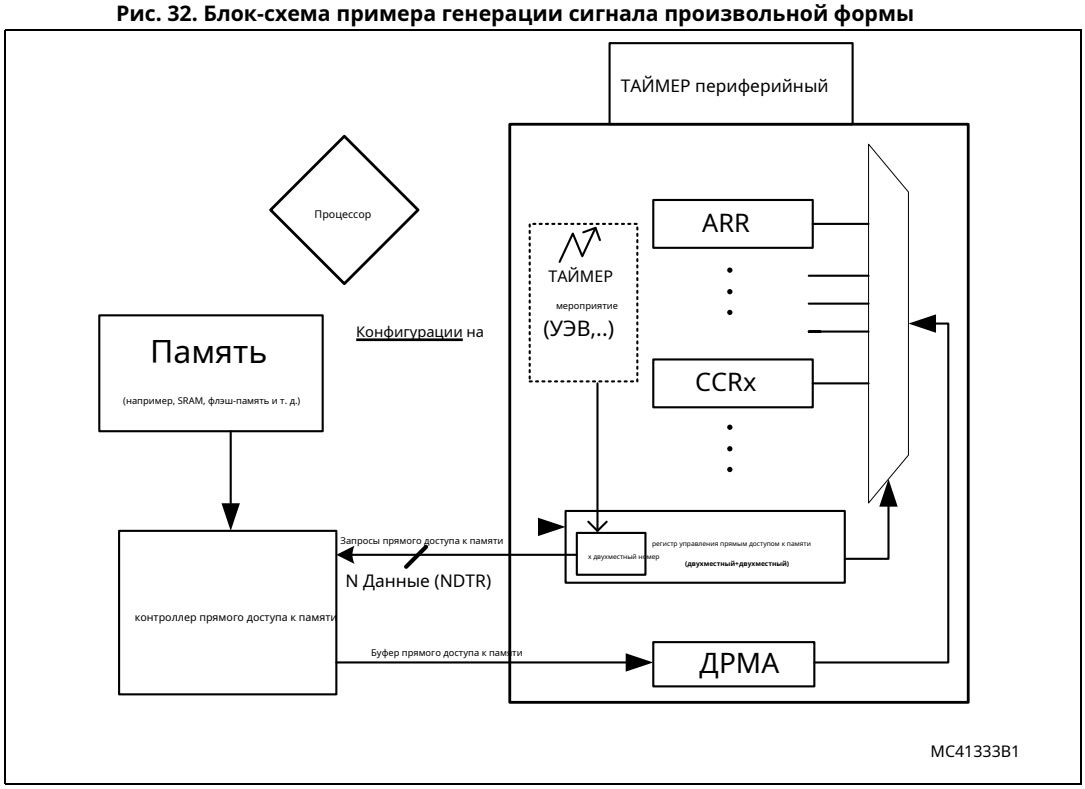
Конфигурация DMA-burst таймера

Для вывода сигнала нужной формы пакет DMA таймера TIM1 должен быть сконфигурирован, как показано ниже:

- Поскольку необходимо обновить три регистра таймера, длина пакетной передачи**3.**Битовое поле управления DBL[4:0] в TIM1_DCR должно быть установлено на 2. Три данных передаются каждым UEV, происходят три события обновления.
- Среди обновляемых регистров таймера регистр таймера TIM1_ARR является первым в карте регистров таймера TIM1, поэтому он определен как база для передачи. Битовое поле управления DBA[4:0] должно быть установлено так, чтобы оно указывало на регистр TIM1_ARR (DBA[4:0] = 11).

*Рисунок 32*иллюстрирует блок-схему для ранее описанных конфигураций.

Рис. 32. Блок-схема примера генерации сигнала произвольной формы



Конфигурация часов

В этом примере, чтобы получить часы счетчика TIM1 на частоте 32 МГц:

- Входные часы TIM1

Входные часы TIM1 (TIM1CLK) установлены на часы APB2 (PCLK2):

TIM1CLK = PCLK2 и PCLK2 = HCLK => TIM1CLK = HCLK = SystemCoreClock.

- Предварительный делитель TIM1

Чтобы получить тактовую частоту счетчика TIM1 на частоте 32 МГц, предварительный делитель вычисляется следующим образом:

Предварительный делитель = (часы счетчика TIM1CLK / TIM1) - 1

Предварительный делитель = (SystemCoreClock/32 МГц) – 1

Расчет частот и рабочих циклов

В этом примере шаблон данных для сгенерированного сигнала определяется следующим образом:
uint32_t aSRC_Buffer[9] = {4000,1,800,10000,0,8500,4000,2,200};

И исходя из приведенных ниже расчетов:

– Расчет частоты TIM1 (F1,F2):

$$\text{TIM1Частота(F1)} = \frac{\text{ТИМ1}}{\text{ТИМARR}+1} = \frac{32\ \text{МГц}}{4000+1} = 8{,}0\ \text{кГц}$$

$$\text{TIM1Frequency(F2)} = \frac{\text{ТИМ1}}{\text{ТИМARR}} = \frac{32\ \text{МГц}}{10000+1} = 3{,}2\ \text{кГц}$$

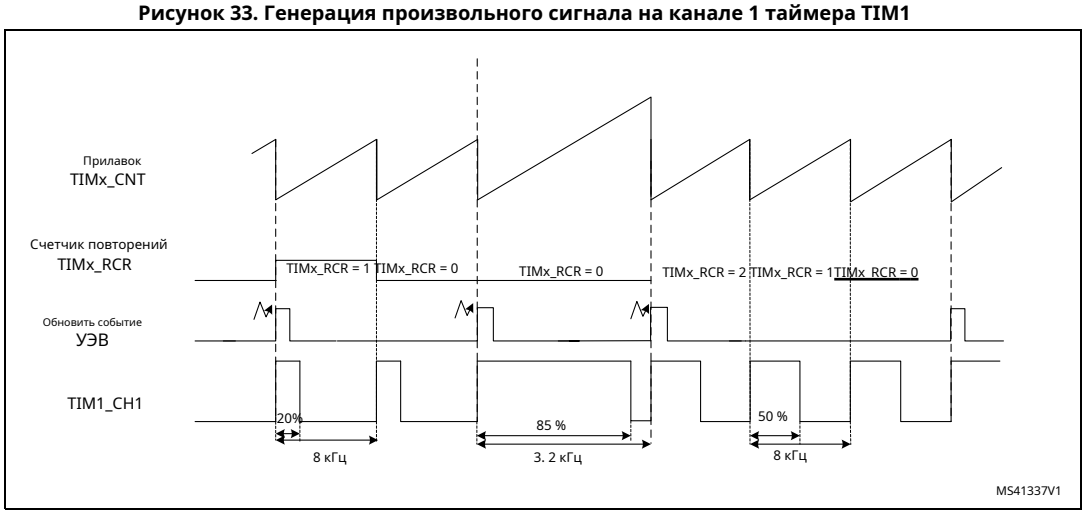
– Расчет рабочих циклов канала 1 TIM1 (D1, D2, D3):

$$\text{TIM1dutycycle(D1)} = \frac{\text{ТИМ1}}{\text{ТИМARR}} \times 100 = \frac{800}{4000} \times 100 = 20\%$$

$$\text{TIM1dutycycle(D2)} = \frac{\text{ТИМ1}}{\text{ТИМARR}} \times 100 = \frac{8500}{10000} \times 100 = 85\%$$

$$\text{TIM1dutycycle(D3)} = \frac{\text{ТИМ1}}{\text{ТИМARR}} \times 100 = \frac{2000\ \text{г.}}{4000} \times 100 = 50\%$$

Рисунок 33. Генерация произвольного сигнала на канале 1 таймера TIM1



Описание прошивки

Чтобы добиться генерации произвольного сигнала, описанного в предыдущем разделе, необходимо выполнить следующие шаги:

- **Системные часы**

  – PLL как источник системных часов: 64 МГц

  - HSI в качестве генератора (нет необходимости впаивать HSE в плату Nucleo)

  – AHB дел = 1 / APB1 дел = 2 / APB2 дел = 1

В этом примере используется DMA1 с TIMER1:

- **Конфигурация DMA2**

```
/* Включение часов DMA1 */ RCC->AHBENR
|= RCC_AHBENR_DMA1EN;
/* Настроить регистр CR DMA1 Channel5 */ /*
Сбросить управляющий регистр DMA1 Channel5 */
   DMA1_Channel5->CCR = 0;
/* Установите биты CHSEL в соответствии с каналом 5 DMA */
/* Установите биты DIR в соответствии с направлением памяти на периферию */ /*
Установите бит PINC в соответствии с отключенным приращением периферийных устройств
DMA */ /* Установите бит MINC в соответствии с включенным приращением памяти DMA */ /
* Установите биты PSIZE в соответствии с Peripheral DataSize = Word */ /* Установите биты
MSIZE в соответствии со словом Memory DataSize */
/* Установить бит CIRC в соответствии с круговым режимом */ /* Установить биты
PL в соответствии с очень высоким приоритетом */ /* Установить биты MBURST в
соответствии с одиночным пакетом памяти */ /* Установить биты PBURST в
соответствии с одиночным пакетом периферии */
   DMA1_Channel5->CCR |= DMA_MEMORY_TO_PERIPH |
                         DMA_PINC_DISABLE| DMA_MINC_ENABLE|
                         DMA_PDATAALIGN_WORD| DMA_MDATAALIGN_WORD |
                         DMA_CIRCULAR | DMA_PRIORITY_HIGH;
/* Запись в DMA1 Channel5 номер регистра данных */
   DMA1_Channel5->CNDTR = 9;
/* Запись в адресный регистр периферийного устройства DMA1 Channel5 */
   DMA1_Channel5->CPAR = (uint32_t)TIM1_DMAR_ADDRESS; /* Запись в
регистр адреса памяти DMA1 Channel5 */ /* Установка адреса в буфер
памяти «aSRC_Buffer» */
   DMA1_Channel5->CMAR = (uint32_t)aSRC_Buffer; /*
Включить канал 5 DMA1 */
   DMA1_Channel5->CCR |= (uint32_t)DMA_CCR_EN;
```

- **Конфигурация TIM1**

```
/* устанавливаем предварительный делитель таймера */
   Tim1Prescaler= (uint16_t) (SystemCoreClock/32000000) - 1; /* Настраиваем
период */
   TIM1->ARR = 0xFFFF;
/* Настройте предварительный делитель таймера */
   TIM1->PSC = Tim1Prescaler; /*
Настройка ширины импульса */
   TIM1->CCR1 = 0xFFF;
/* Выберите ClockDivision в 1 */ /* Сброс
битового поля clockDivision */
   TIM1->CR1 &= ~ TIM_CR1_CKD; /*
Выберите DIV1 в качестве деления часов*/
```

```
    TIM1->CR1 |= TIM_CLOCKDIVISION_DIV1;
/* Выбор прямого счета для счетчика TIM1 */ /* Сброс
битовых полей выбора режима*/
    TIM1->CR1 &= ~( TIM_CR1_DIR | TIM_CR1_CMS); /*
выбираем режим прямого счета */
    TIM1->CR1 |= TIM_COUNTERMODE_UP; /*
УСТАНОВИТЬ режим PWM1 */
/* Сброс битов режима сравнения вывода */
    TIM1->CCMR1 &= ~TIM_CCMR1_OC1M; TIM1-
    >CCMR1 &= ~TIM_CCMR1_CC1S; /* Выбираем
режим сравнения вывода 1*/
    TIM1->CCMR1 |= TIM_OCMODE_PWM1;
/* Включить предварительную загрузку сравнения 1 */
    TIM1->CCMR1 |= TIM_CCMR1_OC1PE; /* Включить
автоматическую перезагрузку Preload */
    TIM1->CR1 |= TIM_CR1_ARPE; /*
Включение обновления DMA TIM1 */
    TIM1->DIER |= TIM_DMA_UPDATE;
/* Настройка базового регистра DMA и длины пакета DMA */ /* Сброс битовых
полей DBA и DBL */
    TIM1->DCR &= ~TIM_DCR_DBA;
    TIM1->DCR &= ~TIM_DCR_DBL;
/* Выберите базовый регистр DMA и длину пакета DMA */
    TIM1->DCR = TIM_DMABase_ARR | TIM_DMABurstLength_3Трансферы;
/* Включить UEV, установив бит UG в значение «Загрузить данные буфера в регистры предварительной загрузки».
* /
    TIM1->EGR |= TIM_EGR_UG;
/* дождаться СБРОСА бита UG*/
    в то время как ((TIM1-> EGR & TIM_EGR_UG) == SET) {}
/* Включить UEV, установив бит UG для загрузки данных из предварительной загрузки в
активные регистры */
    TIM1->EGR |= TIM_EGR_UG; /* Включить
основной выход TIM1 */
    TIM1->BDTR |= TIM_BDTR_MOE; /*
Включить вывод CC1*/
    TIM1->CCER |= TIM_CCER_CC1E; /*
Включить счетчик TIM */
    TIM1->CR1 |= TIM_CR1_CEN;
```

- GPIO:
  - Контакт PA8: TIM1_ch1_output
  - Режим: двухтактный
  - Тянуть: подтягиваться
  - Скорость: высокая
  - Альтернативная функция: GPIO_AF6_TIM1

6 Генерация сигнала N-импульсов с использованием синхронизации таймера

Этот пример приложения разделен на две части, описывающие два похожих примера приложения. В обоих примерах используется синхронизация между таймерами для генерации N-импульсного сигнала, но с небольшой разницей:

- В первом примере приложения, описанном в части 1 этого раздела, на выходе и дополнительном выходе канала таймера TIM1 генерируется N-импульсный сигнал. В конце генерации N-импульсного сигнала канал таймера TIM1 Выходы 1 сохраняют свое последнее состояние, когда на выходе канала 1 низкий уровень, а на его дополнительном выходе высокий уровень.
- Во втором примере приложения, описанном в части 2 этого раздела, на выходе и дополнительном выходе канала 1 таймера TIM1 генерируется сигнал N-импульсов. В конце генерации сигнала N-импульсов канал 1 таймера TIM1 выводит оба должны быть низкими. Этого можно добиться с помощью функции коммутации, встроенной в таймеры STM32. Эта функция подробно описана во второй части этой главы.

6.1 Обзор синхронизации таймера

Некоторые таймеры STM32 внутренне связаны друг с другом для синхронизации или объединения таймеров. Периферийные устройства таймера STM32 с возможностью синхронизации между таймерами имеют только один выходной сигнал синхронизации с именем TRGO (аббревиатура от «Trigger Out»). У них также есть четыре входа синхронизации (названные ITRx, где x находится в диапазоне от 1 до 4), подключенные к другим выходам синхронизации таймеров STM32. В справочном руководстве для любого микроконтроллера STM32 указана матрица подключения между таймерами.

Обратите внимание, что только таймеры с контроллером ведущий/ведомый поддерживают синхронизацию между таймерами (за некоторыми исключениями). Например, периферийное устройство таймера TIM2 имеет блок контроллера ведущий/ведомый и может быть синхронизировано с другими периферийными устройствами таймера STM32.

TIM2 может запускать события внутри других таймеров через выходной сигнал синхронизации TRGO. В этом случае периферийный таймер TIM2 действует как главный таймер. Периферийный таймер TIM2 также может быть сконфигурирован для запуска другими выходными сигналами синхронизации таймера; в этом случае периферийный таймер TIM2 действует как подчиненный таймер. Периферийный таймер может действовать как ведомый таймер и как ведущий таймер одновременно.

Периферийный таймер TIM11 является примером периферийного устройства таймера, в котором нет контроллера ведущий/ведомый. Он не имеет синхронизации выходного сигнала TRGO, но может действовать как главный таймер для некоторых других периферийных устройств таймера. Это возможно благодаря использованию выхода канала таймера TIM11 в качестве выходного сигнала синхронизации. Выход канала таймера TIM11 подключен к входам синхронизации других периферийных устройств таймера.

Конфигурация мастера таймера

Когда периферийное устройство таймера сконфигурировано как главный таймер, его соответствующий выходной сигнал синхронизации TRGO может выдавать синхронизирующий импульс рядом с любым из событий таймера, перечисленных ниже. Обратите внимание, что представленный список не является исчерпывающим и перечислены только самые распространенные режимы. Список основных режимов может варьироваться от одного семейства микроконтроллеров к другому:

- Сброс: бит UG из регистра EGR используется как триггерный выход (TRGO).
- Включение: сигнал включения счетчика используется в качестве триггерного выхода (TRGO). Используется для одновременного запуска нескольких таймеров или для управления окном, в котором включен подчиненный таймер.
- Обновление: событие обновления выбирается в качестве триггерного выхода (TRGO). Например, главный таймер можно использовать в качестве предделителя для подчиненного таймера.
- Импульс сравнения: выход триггера отправляет положительный импульс, когда установлен флаг CC1IF (даже если он уже был высоким), как только происходит захват или сравнение.
- OC1Ref: сигнал OC1REF используется в качестве триггерного выхода (TRGO).
- OC2Ref: сигнал OC2REF используется в качестве триггерного выхода (TRGO).
- OC3Ref: сигнал OC3REF используется в качестве триггерного выхода (TRGO).
- OC4Ref: сигнал OC4REF используется в качестве триггерного выхода (TRGO).

Чтобы сконфигурировать событие таймера или внутренний сигнал для использования в качестве выхода синхронизации, правильное значение должно быть записано в битовое поле управления MMS (выбор главного режима) в регистре управления таймером TIMx_CR2 2.

Конфигурация подчиненного таймера

Каждое периферийное устройство таймера, в которое встроен блок контроллера ведущий/ведомый, имеет четыре входа синхронизации, уже подключенных к другим выходам синхронизации таймеров. Обратите внимание, что только один вход синхронизации может быть активен в каждый момент времени, и битовое поле управления TS (выбор триггера) используется для выбора того, какой вход синхронизации является активным.

Обнаружение активного фронта на входе синхронизации одного периферийного устройства таймера может вызвать одно из событий таймера внутри периферийного устройства, например, «событие обновления» и сброс счетчика или увеличение счетчика. Это зависит от сконфигурированного значения битового поля управления SMS (выбор ведомого режима) в регистре управления ведомым режимом таймера (TIMx_SMCR).

Чтобы выбрать, какой вход синхронизации использовать, ведомый таймер подключается к ведущему через входной триггер. Каждый ITRx внутренне подключен к другому таймеру, и это соединение специфично для каждого продукта STM32.

Функция коммутации таймера STM32

Функция коммутации используется в сочетании с функцией предварительной загрузки для изменения конфигурации канала таймера в идеальной синхронизации с внешними событиями таймера, которые поступают в него через один из его внутренних или внешних входов. Конфигурация, к которой это относится, может быть, например, режимом вывода канала, включенным/отключенным каналом или другим. Входы ITRx являются примерами внутренних входов, а ETR, TI1 или TI2 — примерами внешних входов.

Как указано в*Раздел 1: Основные режимы работы универсальных таймеров STM32*, битовые поля управления OCxM, CCxE и CCxNE имеют возможность предварительной загрузки. Когда функция предварительной загрузки этих битовых полей включена, любая запись в них не изменяет режим работы канала таймера. Причина в том, что операция записи выполняется для экземпляров предварительной загрузки этих управляющих битовых полей. Их активные экземпляры, эффективно управляющие режимом вывода канала таймера, остаются без изменений.

Как только внутри таймера генерируется коммутационное событие, содержимое экземпляров предварительной загрузки этих управляющих битовых полей передается в активные экземпляры, и, следовательно, режим вывода канала может быть изменен.

В зависимости от конфигурации битового поля управления CCUS в регистре управления таймером 2 (TIMx_CR2) событие коммутации может быть сгенерировано после обнаружения активного фронта входного сигнала триггера таймера (TRGI). В частности, он может генерироваться после обнаружения активного фронта на входе ITRx активной синхронизации таймера периферийного устройства таймера (входы ITRx являются частью источников сигнала TRGI).

Эта возможность может использоваться для управления одним таймером, когда событие коммутации на втором таймере должно быть запущено посредством синхронизации между таймерами. В качестве альтернативы событие коммутации может быть сгенерировано программным обеспечением посредством установки битового поля управления COMG в регистре таймера TIMx_EGR.

6.2 Пример приложения для генерации N-импульсного сигнала — часть 1

В этом примере показано, как использовать функцию соединения таймера STM32 для генерации определенного количества импульсов в каждый настраиваемый период.

Обзор приложений

Приложение разработано для платы Nucleo NUCLEO-F302R8, основанной на микроконтроллере STM32F302x8, в качестве основного контроллера платы. Тем не менее, это приложение можно легко портировать на любую аппаратную платформу STM32. В плату Nucleo встроен собственный отладчик ST-Link/V2. Для соединения платы Nucleo с ПК для загрузки двоичного образа приложения в микроконтроллер и для целей отладки приложения требуется только USB-кабель.

The *Рисунок 34*показывает синоптическую диаграмму для приложения периодической генерации N-импульсов, представленного в этом примере приложения. Сгенерированный сигнал выводится на PA.08 IO микроконтроллера STM32F302, который отображается на контакте 8 разъема CN9.

Основные описанные особенности:

- ТАЙМЕР 2 настроен как основной режим запуска для запуска ТАЙМЕРА 1.
- ТАЙМЕР 1 настроен как ведомый одноимпульсный (OPM) режим.

Рисунок 34. Блок-схема периодической генерации N-импульсов

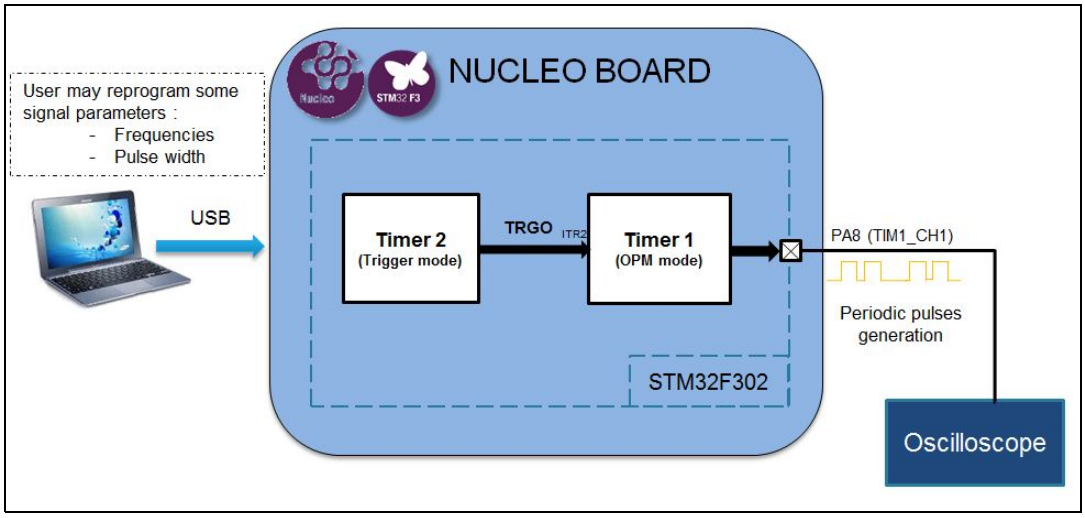
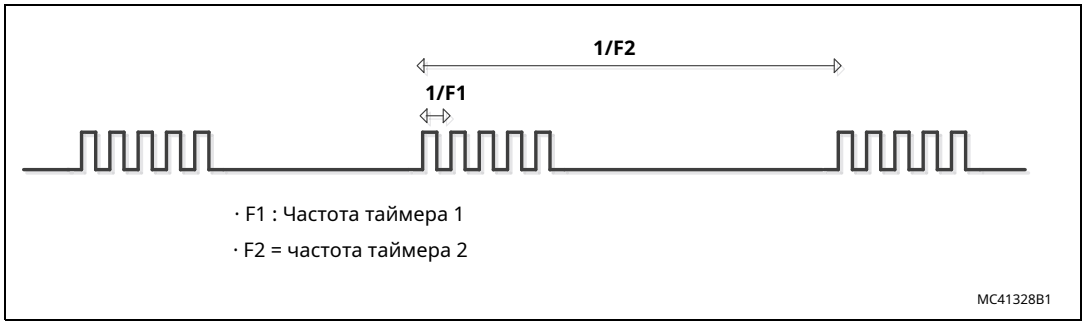


*Рисунок 35*показывает целевую форму волны, которую мы хотим сгенерировать в этом примере: периодическая генерация N=5 импульсов.

Рис. 35. Целевая форма выходного сигнала примера периодической генерации N-импульсов



Форма волны, описанная в этом примере, предназначена для периодического вывода шести импульсов в каждом цикле ШИМ, аналогично сигналу формы волны на*Рисунок 35*. Форма сигнала состоит из двух частот, заданных двумя запрограммированными периодами: T2 — период периферии таймера TIM2, а T1 — период периферии таймера TIM 1. Исходный код этого приложения может быть легко изменен для вывода различных сигналов, но для демонстрационных целей тот, который показан на*Рисунок 37* нацелен.

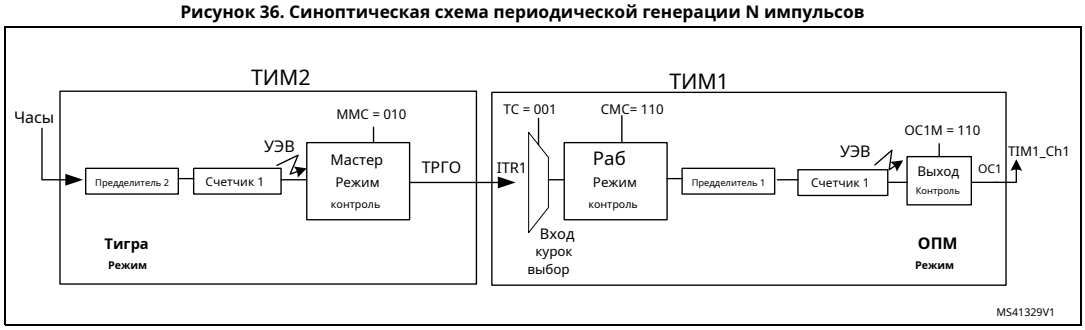
Функциональное описание

В отношении*Рисунок 35*, целевая форма волны представляет собой сигнал ШИМ, создаваемый периодическим Npulses для (шесть импульсов в этом примере). Импульсы производятся с определенным временем включения и выключения: t1_on и t1_off, t2_on и сумма равна T1. Каждый период T2 импульсы генерируются снова, а остаток цикла ШИМ поддерживает низкий уровень.

Чтобы сделать периферийный таймер STM32 для вывода этих периодических импульсов, это делается с использованием двух взаимосвязанных периферийных устройств таймера: периферийного устройства таймера TIM1 и периферийного устройства таймера TIM2. Периферийный таймер TIM1 генерирует сигнал PWM с рабочим циклом T1_on, запрограммированным в TIM1_CCR1, и периодом T1, запрограммированным в TIM1_ARR. Периферийный таймер TIM2 гарантирует, что период каждого цикла генерации равен T2, который запрограммирован в его регистре TIM2_ARR.

Как показано в*Рисунок 36*, периферийное устройство TIM2 и периферийное устройство таймера TIM3 последовательно соединяются как ведущий и ведомый.

Рисунок 36. Синоптическая схема периодической генерации N импульсов



Периферийный таймер TIM2 настроен как главный режим запуска для запуска периферийного таймера TIM1 ITR1 через его внутренний сигнал TRGO при каждом событии обновления. Время между двумя «событиями обновления» — это T2 целевого сигнала.

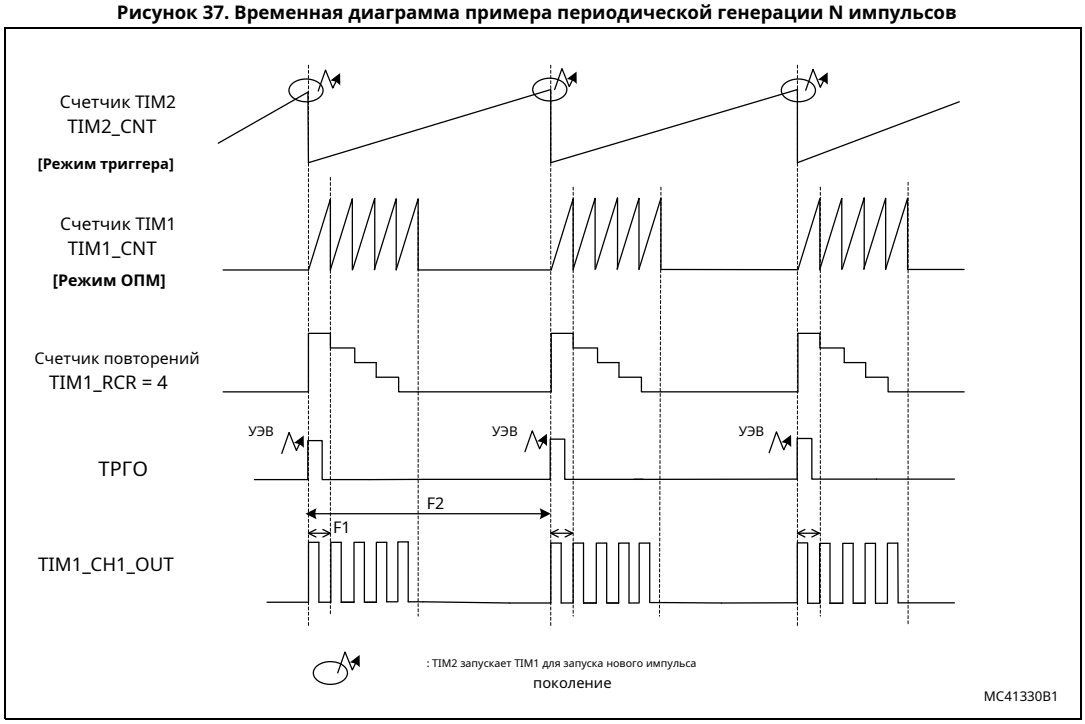
Периферийный таймер TIM1 настроен как ведомый режим одного импульса (OPM), чтобы генерировать один импульс при запуске периферийным устройством таймера TIM2.

Когда на триггерном входе (ITR1) возникает нарастающий фронт, счетчик начинает обратный счет только один раз, пока не совпадет с периодом настройки (TIM1_CCR1), равным 1/F1, а затем произойдет UEV.

Чтобы повторить сгенерированный импульс (прямой счет), мы используем**счетчик повторений** периферийного устройства таймера TIM1, которое должно быть настроено на количество импульсов на часть текущего сигнала минус один (TIM1_RCR = число_импульсов_на_часть – 1 = 6-1). UEV генерируется только после повторения обратного счета в течение времени, настроенного TIM1_RCR +1.

Временная диаграмма на *Рисунок 37* показывает, как синхронизированные таймеры с конфигурацией описателя позволяют получить желаемую форму сигнала.

Рисунок 37. Временная диаграмма примера периодической генерации N импульсов



Описание прошивки

- **Системные часы**

  – PLL как источник системных тактовых импульсов: 64 МГц

  — HSI в качестве генератора (нет необходимости впаивать HSE в платы Nucleo)

  – AHB дел = 1 / APB1 дел = 2 / APB2 дел = 1

- **ТАЙМЕР 1**

  Мы используем TIM1_channel1 в режиме сравнения вывода для генерации сигнала PWM1.

  – Предварительный делитель APB2 = 64 -1 (чтобы получить 1 МГц в качестве часов таймера)

  – Период (ARR) = 50 мкс -> частота F1 = 20 кГц

  – Импульс (CCR1) = период / 2 (50%)

  – RCR = 5-1 -> для повторения 5 импульсов

  – Выбран один импульсный режим

  – Выбран режим подчиненного триггера

  – Входной триггер: ITR1

  – Режим ШИМ: режим 1

  – Режим счета: прямой счет

  – Предварительная загрузка сравнения вывода: включена

```
/* установить предварительный делитель таймера на 1 МГц в качестве тактовой частоты */
   Tim1Prescaler= (uint16_t) (SystemCoreClock/1000000) - 1;
/* Инициализируйте период ШИМ, чтобы получить 20 кГц в качестве частоты от 1 МГц */
   Период = 1000000/20000;
/* настроить предварительный делитель таймера */
   TIM1->PSC = Tim1Prescaler; /*
настроить период */
   TIM1->ARR = Период-1;
/* настроить счетчик повторений */
   TIM1->RCR = ((uint32_t) 6) -1; /*
настройка ширины импульса */
   TIM1->CCR1 = период / 2;
/* Выберите для параметра Clock Division значение
1*/ /* Сброс битового поля Clock Division */
   TIM1->CR1 &= ~ TIM_CR1_CKD; /*
Выберите DIV1 в качестве деления часов*/
   TIM1->CR1 |= TIM_CLOCKDIVISION_DIV1; /* Выбор
прямого счета для счетчика TIM1 */ /* Сброс битовых
полей выбора режима*/
     TIM1->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /* выбор
режима прямого счета */
   TIM1->CR1 |= TIM_COUNTERMODE_UP; /*
УСТАНОВИТЬ режим PWM1 */
/* Сброс битов режима сравнения вывода */
   TIM1->CCMR1 &= ~TIM_CCMR1_OC1M;
```

```
   TIM1->CCMR1 &= ~TIM_CCMR1_CC1S; /*
Выбираем режим сравнения вывода 1*/
   TIM1->CCMR1 |= TIM_OCMODE_PWM1;
/*************** Конфигурация с одним импульсом ************/
   /* Выбор одного импульсного режима */ TIM1->CR1 |= TIM_CR1_OPM; /
   ********************************************************** *******/ /
********* Конфигурация ведомого режима: Режим триггера **********/ /*
Выберите сигнал TIM_TS_ITR1 в качестве входного триггера для TIM */

   TIM1->SMCR &= ~TIM_SMCR_TS;
   TIM1->SMCR |= TIM_TS_ITR1; /*
Выбор ведомого режима */
   TIM1->SMCR &= ~TIM_SMCR_SMS; TIM1->SMCR |= TIM_SLAVEMODE_TRIGGER; /
   ********************************************************** ******************/ /
* Включить предварительную загрузку сравнения 1 */

   TIM1->CCMR1 |= TIM_CCMR1_OC1PE; /*
Установите бит UG, чтобы включить UEV */
   TIM1->EGR |= TIM_EGR_UG; /* Включить
основной выход TIM1 */
   TIM1->BDTR |= TIM_BDTR_MOE;
/* Выбор активного низкого уровня в качестве уровня полярности
выхода */ /* Сброс уровня полярности выхода */
   TIM1->CCER &= ~TIM_CCER_CC1P; /*
Устанавливаем Низкий выход */
   TIM1->CCER |= TIM_OCPOLARITY_LOW; /*
Включить вывод CC1 на высоком уровне */
   TIM1->CCER |= TIM_CCER_CC1E; /*
Включить счетчик TIM */
   TIM1->CR1 |= TIM_CR1_CEN;
```

- **ТАЙМЕР 2**

Мы используем TIM2_channel1 в режиме сравнения вывода для генерации сигнала PWM1.

– Предварительный делитель APB1 = 64 -1 (чтобы получить 1 МГц в качестве часов таймера)

– Период (ARR) = 20 000 мкс -> частота F2 = 50 Гц

– Импульс (CCR1) = период/2 (50%)

– Выбран основной режим: обновление TRGO

– Режим счета: прямой счет

```
/* установить предварительный делитель таймера на 1 МГц в качестве тактовой частоты */
   Tim2Prescaler= (uint16_t) ((SystemCoreClock) / 1000000) - 1; /* Инициализируйте
период ШИМ, чтобы получить 50 Гц в качестве частоты от 1 МГц */
   Период = 1000000/50; /*
настроить период */
   TIM2->ARR = период-1;
/* настроить предварительный делитель таймера */
```

```
  TIM2->PSC = Tim2Prescaler; /* Выберите для
параметра Clock Divison значение 1*/ /* Сброс
битового поля Clock Division */
  TIM2->CR1 &= ~ TIM_CR1_CKD; /*
Выберите DIV1 в качестве деления часов*/
  TIM2->CR1 |= TIM_CLOCKDIVISION_DIV1;
/* Выбор прямого счета для счетчика TIM1 */ /* Сброс
битовых полей выбора режима */
  TIM2->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /* выбор
режима прямого счета */
  TIM2->CR1 |= TIM_COUNTERMODE_UP;
/****** Конфигурация ведущего режима: Запуск режима обновления *******/ /*
Запуск обновления TIM2 в подчиненный TIM1 */
  TIM1->CR2 &= ~ TIM_CR2_MMS; TIM2->CR2 |= TIM_TRGO_UPDATE; /
  ********************************************************** ***********/
/* Включить счетчик TIM */

  TIM2->CR1 |= TIM_CR1_CEN;
```

- GPIO
  - Контакт PA8: TIM1_ch1_output
  - Режим: двухтактный
  - Тянуть: подтягиваться
  - Скорость: высокая
  - Альтернативная функция: GPIO_AF6_TIM1

6.3 Пример приложения для генерации N-импульсного сигнала — часть 2

Обзор приложений

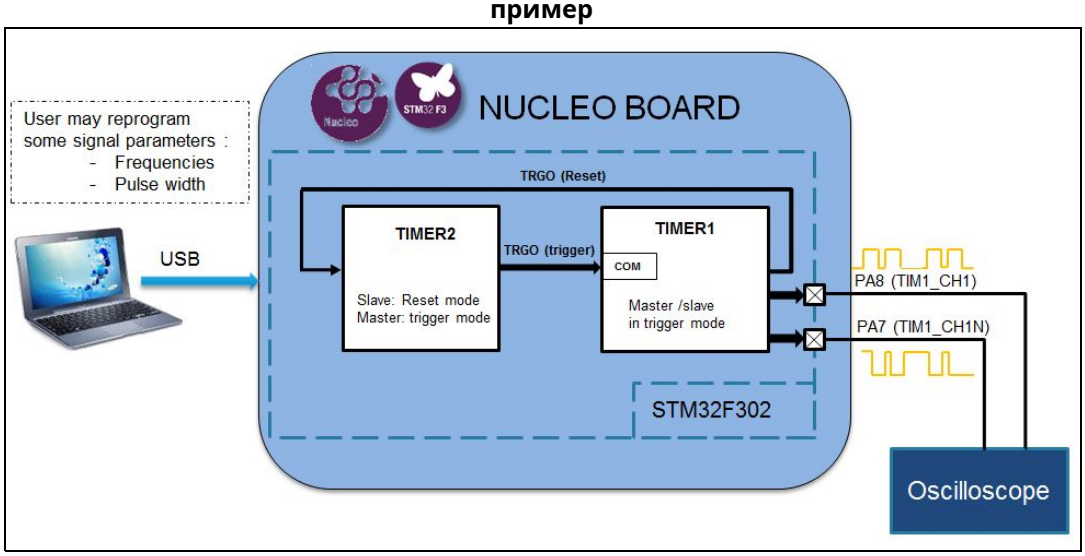
Этот пример приложения разработан на основе платы Nucleo NUCLEO-F302R8, основанной на микроконтроллере STM32F302x8, в качестве основного контроллера платы. Тем не менее, это приложение можно легко портировать на любую аппаратную платформу STM32. В плату Nucleo встроен собственный отладчик ST-Link/V2. Для соединения платы Nucleo с ПК для загрузки двоичного образа приложения в микроконтроллер и для целей отладки приложения требуется только USB-кабель.

*Рисунок 38*показывает синоптическую диаграмму для приложения периодической генерации N-импульсов, представленного в этом примере приложения. Оба дополнительных генерируемых сигнала выводятся на PA.08 IO и PA.07 IO микроконтроллера STM32F302, которые отображаются соответственно на контакт 8 разъема CN9 и на контакт 26 разъема CN10.

Основные описанные особенности:

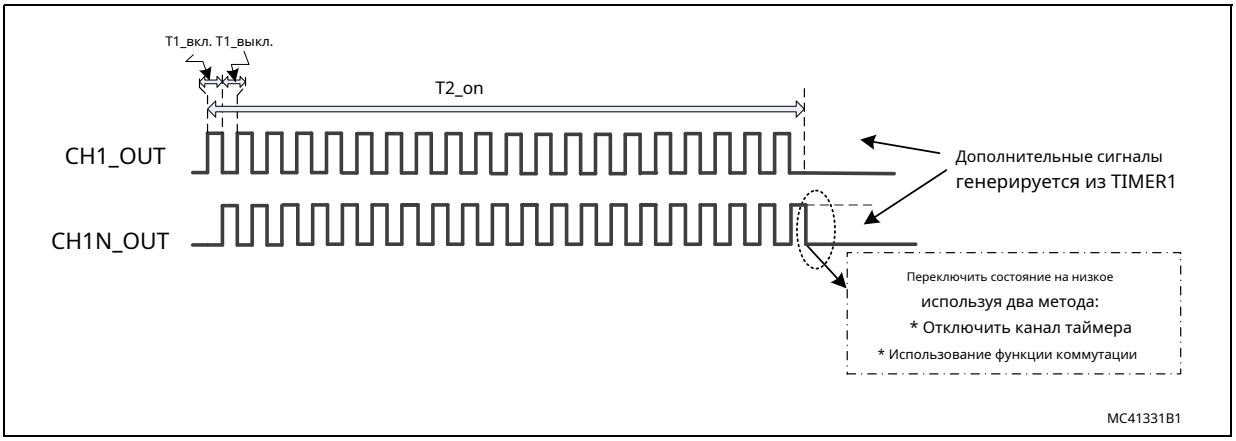
- ТАЙМЕР 2 настроен как режим ведомого сброса и как режим ведущего триггера.
- ТАЙМЕР 1 настроен как ведомый и ведущий в режиме триггера.

Рис. 38. Синоптическая схема генерации сигналов двух комплементарных импульсов N пример



Генератор двух комплементарных сигналов N-импульсов, описанный в этом примере, предназначен для вывода двух комплементарных сигналов, каждый из которых состоит из N-импульсов и которые имеют одинаковое конечное состояние. Результат должен быть похож на сигналы, показанные на *Рисунок 39*. Исходный код этого приложения может быть легко изменен для вывода различных сигналов, но в демонстрационных целях целью является тот, который показан на *Рисунок 39*.

Рис. 39. Пример генерации комплиментарных сигналов на выходе из двух N-импульсов



Функциональное описание

Согласно формам волны, показанным на *Рисунок 39*, целевые сигналы представляют собой два дополнительных ШИМ-сигнала, созданных только N-импульсами для 20 импульсов в нашем случае. Импульсы производятся с определенным временем включения и выключения, t1_on и t1_off. T2_on – активное время формирования кадра импульсов. Оба дополнительных сигнала должны заканчиваться одним и тем же конечным состоянием, в нашем примере это низкий уровень, как показано на *Рисунок 39*.

Чтобы заставить периферийный таймер STM32 выводить эти ШИМ, необходимо использовать два взаимосвязанных периферийных таймера. Один из них работает как дополнительный генератор ШИМ, а другой контролирует окончание генерации импульсов.

Первый таймер управляет временем начала и окончания генерации импульсов, отсчитывая T2_on и запуская таймер генератора, чтобы отключить выход ШИМ. Таким образом, оба периферийных счетчика таймера начинают отсчет одновременно, синхронизируя время запуска.

Интуитивно понятный способ сделать это — настроить таймер генератора как главный режим триггера, а другой — как режим сброса подчиненного, а также сбросить периферийный счетчик первого таймера, когда таймер генератора включен. С другой стороны, сгенерированный таймер также конфигурируется как режим подчиненного триггера, а первое периферийное устройство таймера — как режим обновления главного триггера для генерации «события обновления», когда счетчик времени T2_on переполняется и останавливает генерацию ШИМ.

Когда отсчет T2_on заканчивается и генерируется «событие обновления» для запуска таймера генератора, есть два способа остановить генерацию ШИМ в процедуре прерывания: либо отключить оба выхода захвата/сравнения, либо принудительно перевести выходы в неактивный режим с помощью кнопки функция коммутации.

Управление обновлением коммутации в периферийном устройстве таймера TIM1 должно быть включено и настроено для запуска его входа TRGI путем установки битового поля CCUS в периферийном устройстве TIM1_CR2. Источник прерывания коммутации таймера включается установкой бита COMIE в регистре TIM1_DIER. Конфигурация для следующего шага события COM должна быть запрограммирована заранее, эта конфигурация переводится в неактивный уровень путем установки битового поля OC1M [2:0] регистра TIM1_CCMR1 на 4.

Когда TIM2 запускает вход TRGI TIM1, происходит коммутационное событие (COM) путем обнаружения нарастающего фронта. Затем битовое поле предварительной загрузки OC1M передается в теневые биты в событии коммутации COM. Наконец, на выходах TIM1 одновременно устанавливается низкий уровень.

Оба таймера должны быть конфигурируемы как режим ведущий/ведомый, а для таймера генератора также необходимы дополнительные выходы. В этом примере периферийный таймер TIM1 является хорошим кандидатом на роль генератора таймеров, поскольку он имеет четыре канала таймера, которые могут выводить как дополнительные сигналы, так и событие коммутации. Используются канал 1 и канал 1N таймера TIM1. Мы используем TIM2 для другого таймера.

Чтобы канал таймера не генерировал нежелательные сбои из-за обновления регистра канала таймера, используется функция предварительной загрузки сравнения захвата периферийных устройств таймера STM32.

Чтобы избежать напрасных циклов времени перед периферийным устройством таймера TIM1, «событие обновления» генерируется после включения его регистра канала, битовое поле генерации обновления (UG) в регистре TIM1_EGR должно быть установлено. Это делает синхронизацию между счетчиками периферийных устройств обоих таймеров более точной.

To output the waveform shown on *Рисунок 39* на дополнительных каналах периферийного устройства таймера TIM1, как периферийное устройство таймера TIM1, так и периферийное устройство таймера TIM2 настроены, как описано ниже:

Периферийный таймер TIM1:основной режим сброса триггера /подчиненный триггерный режим

ТАЙМЕР 1 настроен как основной режим сброса триггера для запуска периферийного устройства таймера TIM2 и сброса его счетчика. Это делается для того, чтобы синхронизировать время начала отсчета обоих счетчиков периферийных таймеров. ТАЙМЕР 1 настроен как подчиненный режим запуска, который будет запускаться событием обновления, когда TIM2 заканчивает отсчет периода T2_on, чтобы снизить уровень сигнала до низкого уровня.

- **TIM1_ARR:**содержит период продолжающегося импульса; сумма периодов t1_on и t2_off
- **TIM1_CCR1**: содержит продолжительность периода импульса t1_on
- **Периферийный таймер TIM1**:мастер-триггер-обновление /режим сброса ведомого триггера

Таймер 2 настроен как основной режим запуска для запуска TIMER1 «событием обновления», когда заканчивается период T2_on, через сигнал TRGO. Таймер 2 настроен как режим сброса ведомого триггера, который сбрасывается TIM1, когда он включен.

- **TIM2_ARR:**содержит период T2_on.

6.3.1 Конфигурация часов

Этот пример направлен на получение тактовой частоты счетчиков TIM1 и TIM2 на частоте 1 МГц.

- **Входные часы TIM1**

Входные часы TIM1 (TIM1CLK) установлены на часы APB2 (PCLK2):

TIM1CLK = PCLK2 и PCLK2 = HCLK => TIM1CLK = HCLK = SystemCoreClock = 64 МГц

- **Входные часы TIM2**

Входные часы TIM2 (TIM2CLK) установлены на часы APB2 (PCLK2):

TIM2CLK = PCLK2 и PCLK2 = HCLK => TIM2CLK = HCLK = SystemCoreClock = 64 МГц

- **Предделитель TIM1**

Чтобы получить тактовую частоту счетчика TIM1 на частоте 1 МГц, предварительный делитель вычисляется следующим образом:

Предварительный делитель = (часы счетчика TIM1CLK / TIM1) - 1

Предварительный делитель = (SystemCoreClock/1 МГц) – 1

- **Предделитель TIM2**

Чтобы получить тактовую частоту счетчика TIM2 на частоте 1 МГц, предварительный делитель вычисляется следующим образом:

Предделитель = (часы счетчика TIM2CLK / TIM2) - 1

Предварительный делитель = (SystemCoreClock/1 МГц) – 1

В этом примере цель состоит в том, чтобы периферийное устройство таймера TIM1 выдавало 20 импульсов, каждый импульс с периодом (t1_on +t1_of) 500 мкс и рабочим циклом (t1_on) 50%. Следовательно, периферийное устройство таймера должно быть запрограммировано следующим образом:

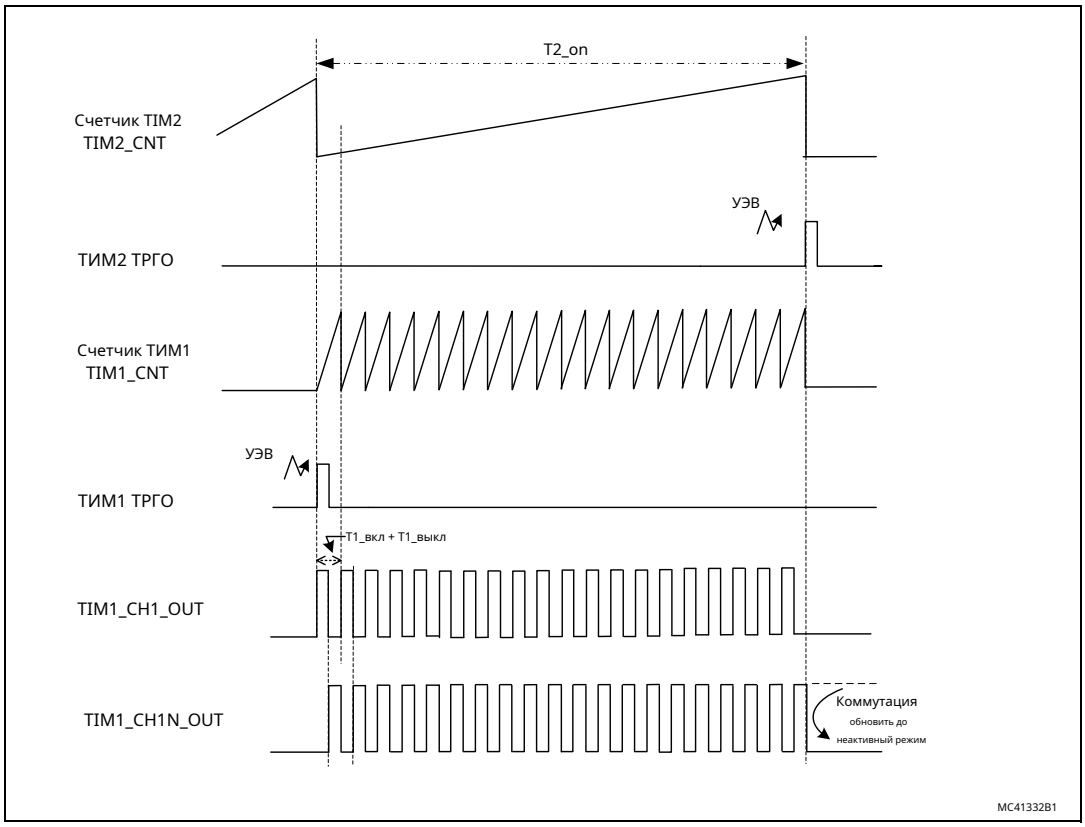
- **ТИМ1 период**
  – TIM1_ARR = T1_on+T1_off = 500 мкс
- **Рабочий цикл TIM1**
  – TIM1_CCR1 = 250 мкс

Расчет периода, который мы программируем на периферийном устройстве таймера TIM2, который позволяет точно получить 20 импульсов, показан ниже:

- **период TIM2**
  - TIM2_ARR=T2_ON = 500 x 20 = 10000 мкс

Временная диаграмма на *Рисунок 40* показывает, как синхронизированный ТАЙМЕР с описанной конфигурацией позволяет пользователю получить желаемую форму волны.

Рис. 40. Временная диаграмма генерации комплиментарных N-импульсов с аналогичное конечное состояние



Описание прошивки

- Системные часы
  - PLL как источник системных часов: 64 МГц
  - HSI в качестве генератора (нет необходимости впаивать HSE в платы Nucleo)
  - AHB дел = 1 / APB1 дел = 2 / APB2 дел = 1
- **ТАЙМЕР 1**

TIM1_channel1 используется в режиме сравнения выходов для генерации сигнала PWM1.

- Предварительный делитель APB2 = 64 -1 (чтобы получить 1 МГц в качестве часов таймера)
- Период (ARR) = 50
- 0 мкс -> частота F1 = 2 кГц
- Импульс (CCR1) = период / 2 (50%)
- Режим ШИМ: режим 1
- Предварительная загрузка сравнения вывода: включена
- Установить событие генерации обновления
- Включить функцию коммутации
- Режим счета: прямой счет
- Выбран режим сброса триггера Matser
- Входной триггер: ITR1
- Выбран подчиненный режим триггера

```
/* установите предварительный делитель Timer на 1 МГц в качестве тактовой частоты, SystemCoreClock =
64 МГц */
   Tim1Prescaler= (uint16_t) (SystemCoreClock/1000000) - 1; /* Инициализируем
период ШИМ, чтобы получить 2 кГц как частоту 10000*/ Period = 1000000 / 2000;

/* настроить предварительный делитель таймера */
   TIM1->PSC = Tim1Prescaler; /*
настроить период */
   TIM1->ARR = Период-1;
/* настройка ширины импульса */
   TIM1->CCR1 = период / 2;
/* Выберите для параметра Clock Divison значение
1*/ /* Сброс битового поля Clock Division */
   TIM1->CR1 &= ~ TIM_CR1_CKD; /*
Выберите DIV1 в качестве деления часов*/
   TIM1->CR1 |= TIM_CLOCKDIVISION_DIV1; /* Выбор
прямого счета для счетчика TIM1 */ /* Сброс битовых
полей выбора режима */
   TIM1->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /* выбор
режима прямого счета */
   TIM1->CR1 |= TIM_COUNTERMODE_UP; /*
УСТАНОВИТЬ режим PWM1 */
/* Сброс битов режима сравнения вывода */
   TIM1->CCMR1 &= ~TIM_CCMR1_OC1M; TIM1-
   >CCMR1 &= ~TIM_CCMR1_CC1S; /* Выбираем
режим сравнения вывода 1*/
   TIM1->CCMR1 |= TIM_OCMODE_PWM1;
/* Выберите активный высокий уровень в качестве уровня выходной
полярности */ /* Сбросьте уровень выходной полярности */
   TIM1->CCER &= ~TIM_CCER_CC1P;
/* Установите высокую полярность сравнения вывода */
   TIM1->CCER |= TIM_OCPOLARITY_HIGH; /*
Включить вывод CC1 на высоком уровне*/
   TIM1->CCER |= TIM_CCER_CC1E;
/* Выберите активный высокий уровень в качестве выходного уровня дополнительной
полярности */ /* Сброс состояния выхода N */
   TIM1->CCER &= ~TIM_CCER_CC1NP;
/* Установите полярность выхода N на высокий уровень */
   TIM1->CCER |= TIM_OCNPOLARITY_HIGH; /*
Включить вывод CC1 на высоком уровне*/
   TIM1->CCER |= TIM_CCER_CC1NE;
/********************* Конфигурация обновления COM Control ******************/ /* Установить
захват Сравнить предварительную загрузку */
  TIM1->CR2 |= TIM_CR2_CCPC;
/* Установите бит CCUS, чтобы выбрать обновление элемента управления COM для запуска ввода TRGI*/
```

```
   TIM1->CR2 |= TIM_CR2_CCUS;
/* Включить источники прерывания коммутации */
   TIM1->DIER |= TIM_IT_COM;
/********** Конфигурация ведущего режима: режим сброса триггера **********/ /*
настройка выхода триггера TIM1 Обновление для запуска TIM2 */
   TIM1->CR2 &= ~ TIM_CR2_MMS; TIM1-
   >CR2 |= TIM_TRGO_RESET;
/************ Конфигурация ведомого режима: Режим триггера ************/ /*
Выберите сигнал TIM_TS_ITR1 в качестве входного триггера для TIM */
   TIM1->SMCR &= ~TIM_SMCR_TS;
   TIM1->SMCR |= TIM_TS_ITR1; /*
Выбор ведомого режима */
   TIM1->SMCR &= ~TIM_SMCR_SMS; TIM1->SMCR |= TIM_SLAVEMODE_TRIGGER; /
   ********************************************************
*****************/ /* Установите бит UG, чтобы включить UEV */

   TIM1->EGR |= TIM_EGR_UG; /* Включить
основной выход TIM1 */
   TIM1->BDTR |= TIM_BDTR_MOE; /*
Включить счетчик TIM */
   TIM1->CR1 |= TIM_CR1_CEN;
```

- **ТАЙМЕР 2**

TIM2_channel1 используется в режиме сравнения выходов для генерации сигнала PWM1.

– Предварительный делитель APB1 = 64 -1 (чтобы получить 1 МГц в качестве часов таймера)

– Период (ARR) = 10 000 мкс

– Режим счета: прямой счет

– Выбран режим обновления основного триггера: обновление TRGO

– Входной триггер: ITR0

– Выбран режим сброса ведомого триггера

```
/* установить предварительный делитель таймера на 1 МГц в качестве тактовой частоты счетчика; SystemCoreClock = 64
МГц */
   Tim2Prescaler= (uint16_t) ((SystemCoreClock) / 1000000) - 1; /* Инициализируйте период
ШИМ, чтобы получить 100 Гц в качестве частоты от 1 МГц */
   Период = 1000000/100; /*
настроить период */
   TIM2->ARR = период-1;
/* настроить предварительный делитель таймера */
   TIM2->PSC = Tim2Prescaler;
/* Выберите для параметра Clock Divison значение 1
*/ /* Сброс битового поля Clock Division */
   TIM2->CR1 &= ~ TIM_CR1_CKD; /*
Выберите DIV1 в качестве деления часов*/
   TIM2->CR1 |= TIM_CLOCKDIVISION_DIV1; /* Выберите
счетчик прямого счета для TIM1 */
```

```
/* Сбросить битовые поля выбора режима */
   TIM2->CR1 &= ~(TIM_CR1_DIR | TIM_CR1_CMS); /* выбор
режима прямого счета */
   TIM2->CR1 |= TIM_COUNTERMODE_UP;
/********** Конфигурация ведущего режима: запуск обновления **********/ /*
Запуск обновления TIM2 в ведомом TIM1 */
   TIM2->CR2 &= ~TIM_CR2_MMS; TIM2-
   >CR2 |= TIM_TRGO_UPDATE;
/************ Конфигурация ведомого режима: Режим триггера ************/ /*
Выберите сигнал TIM_TS_ITR0 в качестве входного триггера для TIM */
   TIM2->SMCR &= ~TIM_SMCR_TS;
   TIM2->SMCR |= TIM_TS_ITR0;
/* Выбор ведомого режима: Режим сброса триггера */
   TIM2->SMCR &= ~TIM_SMCR_SMS; TIM2->SMCR |= TIM_SLAVEMODE_RESET; /
   ************************************************************
***************/ /* Включить счетчик TIM1 */

   TIM2->CR1 |= TIM_CR1_CEN;
```

- **GPIO**
  - Контакт PA8: выход TIM1_CH1
  - Контакт PA7: выход TIM1_CH1N
  - Режим: двухтактный
  - Тянуть: подтягиваться
  - Скорость: высокая
  - Альтернативная функция: GPIO_AF6_TIM1
- **Управление обновлением коммутации**

```
/* Сбросить биты OC1M в регистре CCMR1 */ TIM1->CCMR1
&= TIM_CCMR1_OC2M;
/* установить биты OC1M в регистре CCMRx в неактивный режим*/ TIM1->CCMR1 |=
TIM_OCMODE_FORCED_INACTIVE;
```

7 Лист регистраций изменений

Таблица 1. История изменений документа

Свидание	Редакция	Изменения
23 июня 2016 г.	1	Первый выпуск.
03 мая 2017 г.	2	Обновлено *Раздел 1.4.2: Функция предварительной загрузки регистров таймера* а также *Рис. 6: Механизм предварительной загрузки для регистра канала таймера — включен*.
11 июля 2019 г.	3	Обновлен объем документа для всех устройств STM32 и удалена Таблица 1: Применимые продукты на титульной странице.