Мартин Т. Инсайдерское руководство по STM32

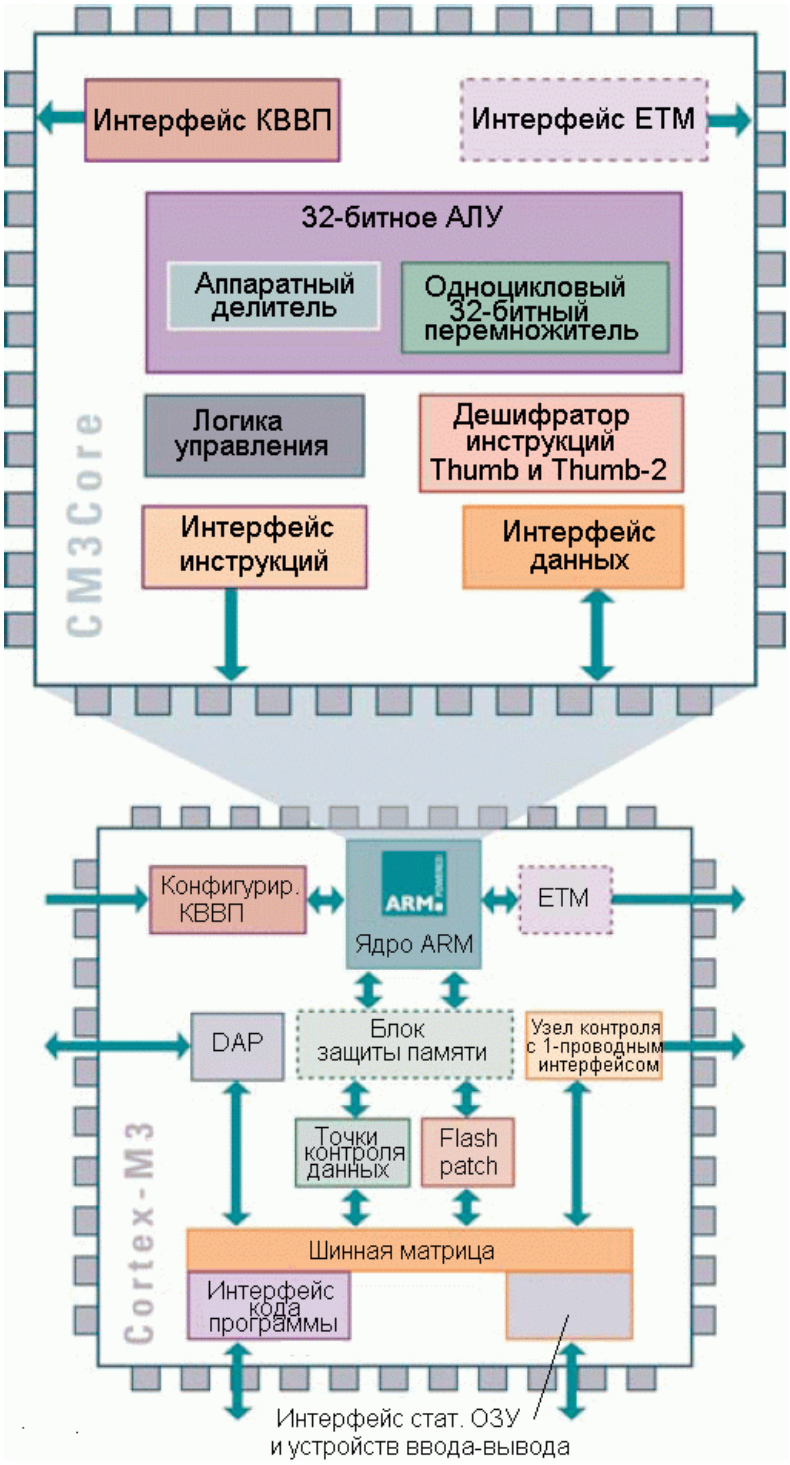
1. Введение

За прошедшие шесть-семь лет основные усилия разработчиков микроконтроллеров были потрачены на интегрирование ЦПУ ARM7 и ARM9 в микроконтроллеры общего назначения. И вот, в настоящее время различными производителями выпускается порядка 240 ARM-микроконтроллеров. К числу таких производителей относится и компания ST Microelectronics, семейство STM32 которой, стало первым ее семейством

микроконтроллеров, выполненных на основе нового ЦПУ ARM Cortex-M3. Эти микроконтроллеры стали новым эталоном по уровню рабочих характеристик и стоимости. Кроме того, они могут использоваться в применениях с малым энергопотреблением и жесткими требованиями к характеристикам управления в масштабе реального времени.

1.1. Знакомство с Cortex

Семейство ARM Cortex - новое поколение процессоров, которые выполнены по стандартной архитектуре и отвечают различным технологическим требованиям. В отличие от других ЦПУ ARM, семейство Cortex является завершенным процессорным ядром, которое объединяет стандартное ЦПУ и системную архитектуру. Семейство Cortex доступно в трех основных профилях: профиль A для высокопроизводительных применений, профиль R для реально-временных применений и профиль M для чувствительных к стоимости и микроконтроллерных применений. Микроконтроллеры STM32 выполнены на основе профиля Cortex-M3, которое специально разработано для применений, где необходимы развитые системные ресурсы и, при этом, малое энергопотребление. Они характеризуются настолько низкой стоимостью, что могут конкурировать с традиционными 8 и 16-битными микроконтроллерами. И хотя ЦПУ ARM7 и ARM9 были с успехом интегрированы в стандартные микроконтроллеры, в них все же прослеживается изначальная ориентированность на системы на кристалле (SoC). Это особенно заметно по способам обработки исключительных ситуаций и прерываний, т.к. у разных производителей микроконтроллеров и способы обработки реализованы различным образом. Cortex-M3 является стандартизованным микроконтроллерным ядром, которое помимо ЦПУ, содержит все остальные составляющие основу микроконтроллера элементы (в т.ч. система прерываний, системный таймер SysTick, отладочная система и карта памяти). 4 гигабайтное адресное пространство Cortex-M3 разделено на четко распределенные области кода программы, статического ОЗУ, устройств ввода-вывода и системных ресурсов. В отличие от ядра ARM7, Cortex-M3 выполнено по Гарвардской архитектуре и, поэтому, имеет несколько шин, позволяющие выполнять операции параллельно. Семейство Cortex имеет возможность оперировать с фрагментированными данными (unaligned data), что также отличает его от предшествующих архитектур ARM. Этим гарантируется максимальная эффективность использования внутреннего статического ОЗУ. Семейство Cortex также поддерживает возможности установки и сброса бит в пределах двух областей памяти размером 1 Мбайт по методу bit banding. Этот метод предоставляет эффективный доступ к регистрам и флагам УВВ, расположенных в области статического ОЗУ, и исключает необходимость интеграции полнофункционального битового процессора.



Основой STM32 является процессор Cortex-M3. Он представляет собой стандартизованный микроконтроллер, интегрирующий 32-битное ЦПУ, шинную структуру, блок вложенных прерываний, отладочную систему и предопределенную организацию памяти.

Еще одним ключевым компонентом ядра Cortex-M3 является контроллер векторизованных вложенных прерываний (NVIC). NVIC предоставляет стандартную структуру прерываний для всех Cortex-микроконтроллеров и способы их обработки.

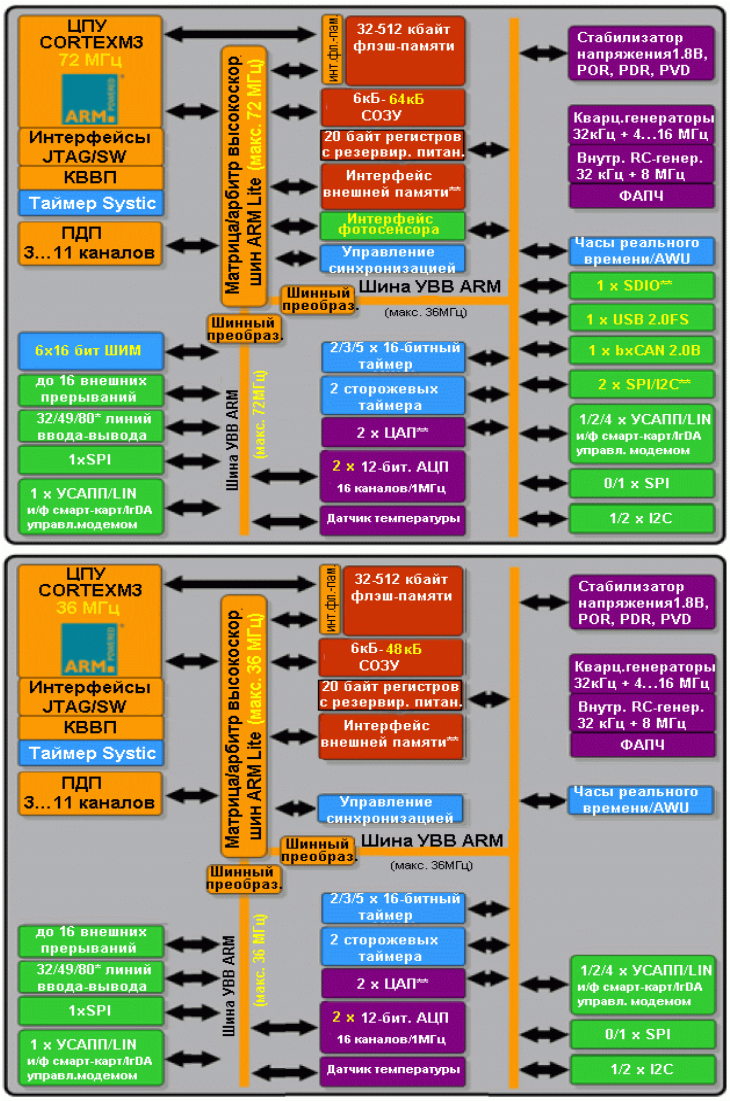
NVIC предписывает векторы прерываний для 240 источников, для каждого из

которых может быть установлен свой приоритет. При разработке NVIC особое внимание уделялось быстродействию обработки прерываний. С момента получения запроса на прерывание до выполнения первой команды процедуры обработки прерывания проходит всего лишь 12 циклов. Частично это достигнуто за счет автоматических операций со стеком, выполняемым специальным микрокодом внутри ЦПУ. Если же прерывания возникают практически одновременно, то NVIC использует способ упорядоченной обработки прерываний с задержкой перед вызовом очередной процедуры обработки прерывания всего лишь 6 циклов. В случае наложения прерываний, прерывание с более высоким приоритетом может вытеснить более низкоприоритетное прерывание, не расходуя при этом дополнительных циклов ЦПУ. Структура прерываний также тесно связана с поддерживаемыми ядром Cortex-M3 экономичными режимами работы. Предусмотрена возможность конфигурации ЦПУ на автоматический переход в экономичный режим работы по завершении обработки прерывания. После этого перехода ядро будет бездействовать вплоть до возникновения очередной исключительной ситуации.

Несмотря на то, что ядро Cortex-M3 разрабатывалось как недорогое ядро, оно остается 32-битным ЦПУ и, в связи с этим, поддерживает два режима работы: потоковый режим (Thread) и режим обработчика (Handler), для каждого из которых можно сконфигурировать свои собственные стеки. Благодаря этому, появляется возможность разработки более интеллектуального программного обеспечения и поддержки операционных систем реального времени (RTOS). В ядро Cortex также входит 24-битный автоматически перезагружаемый таймер, предназначенный для генерации периодических прерываний и используемый ядром RTOS. Если у ЦПУ ARM7 и ARM9 имеется два набора инструкций (32-битный ARM и 16-битный Thumb), то у семейства Cortex предусмотрена поддержка набора инструкций ARM Thumb-2. Он представляет собой смесь 16- и 32-битных инструкций, позволяющие добиться производительности 32-битного набора инструкций ARM и плотности кода, свойственной 16-битному набору инструкций Thumb. Thumb-2 - обширный набор инструкций, ориентированный на компиляторы языков C/ C++. Это означает, что программа для Cortex-микроконтроллера может быть полностью написано на Си.

1.2. Обзор семейства STM32

До появления STM32 компания ST уже имела в своем выпускаемом ассортименте 4 семейства микроконтроллеров на основе ядер ARM7 и ARM9, однако именно у микроконтроллеров STM32 было достигнуто существенное улучшение соотношения стоимости и рабочих характеристик. Микроконтроллеры STM32, цена которых за штуку при покупке больших количеств составляет чуть более одного Евро, бросают серьезный вызов существующим 8-битным микроконтроллерам. Микроконтроллеры STM32 изначально выпускались в 14 различных вариантах, разделенные на две группы: Performance Line, в которую вошли микроконтроллеры с тактовой частотой ЦПУ до 72 МГц, и Access Line (тактовая частота до 36 МГц). Обе группы микроконтроллеров совместимы по расположению выводов и программному обеспечению. Объем их встроенной Flash памяти достигал 128 кбайт, а статического ОЗУ - 20 кбайт. С момента первого появления микроконтроллеров STM32 их ассортимент был существенно расширен новыми представителями с повышенными размерами ОЗУ и Flash памяти, а также с более сложными УВВ.



Семейство STM32 состоит из двух групп. Группа Performance Line работает на тактовых частотах до 72 МГц и оснащена полным набором УВВ, а группа Access Line работает на частотах до 36 МГц и интегрирует ограниченный набор УВВ

1.2.1. Многофункциональные УВВ

На первый взгляд набор встроенных УВВ, в т.ч. два АЦП, таймеры общего назначения, I2C, SPI, CAN, USB и часы реального времени (ЧРВ), кажется идентичным самым простым микроконтроллерам. Но если детально рассмотреть каждое из этих УВВ, то выяснится, что они гораздо более сложные. Например, 12-битный АЦП оборудован датчиком температуры и поддерживает несколько режимов преобразования, а МК с двумя АЦП могут использовать их совместно еще в девяти режимах преобразования. По аналогии с этим, каждый из четырех таймеров, оснащенных блоками захвата и сравнения, могут использоваться как отдельно, так и совместно, образую более сложные массивы таймеров. У расширенного таймера (advanced timer) добавлена поддержка управления электродвигателями. Для этого у него предусмотрено 6 комплементарных

ШИМ-выходов с программируемой паузой неперекрытия и вход экстренного останова, который переводит ШИМ-выходы в предварительно запрограммированное безопасное состояние. У интерфейса SPI предусмотрен аппаратный генератор CRC для 8 и 16 слов, что упрощает реализацию интерфейса карт Flash памяти SD и MMC.

В отличие от простых микроконтроллеров, у STM32 также предусмотрен 7-канальный блок прямого доступа к памяти (ПДП). Каждый канал может использоваться для передачи данных между регистрами любого из УВВ и запоминающими устройствами 8/16 или 32-битными словами. Каждое из УВВ может выполнять роль потокового контроллера ПДП, при необходимости отправляя данные или посылая запрос на их получение. Арбитр внутренней шины и матрица шин минимизируют потребность в арбитраже доступа к шинам со стороны ЦПУ и каналов ПДП. Это означает, что блок ПДП является универсальным, простым в применении и реально автоматизирует передачу потоков данных внутри микроконтроллера.

Микроконтроллеры STM32 отличаются казалось бы невозможным сочетанием характеристик малого энергопотребления и высокой производительности. Они способны работать от 2В-ого источника питания на тактовой частоте 72МГц и потреблять, при этом, ток, с учетом нахождения в активном состоянии всех встроенных ресурсов, всего лишь 36мА. Если же использовать поддерживаемые ядром Cortex экономичные режимы работы, то потребляемый ток можно снизить до ничтожных 2 мкА в режиме STANDBY. Для быстроты возобновления активной работы микроконтроллера используется внутренний RC-генератор на частоту 8 МГц. Его активность сохраняется на время запуска внешнего генератора. Благодаря быстроте перехода в экономичный режим работы и выхода из них результирующая средняя потребляемая мощность еще больше снижается.

1.2.2. Безопасность

Многие современные применения, помимо высокой производительности и многофункциональных УВВ, должны обладать инструментами, обеспечивающие безопасность. Данное требование было учтено у МК STM32. В них интегрирован ряд аппаратных блоков, отвечающих за безопасность работы микроконтроллера, в т.ч. маломощный супервизор питания, система защиты синхронизации и два отдельных сторожевых таймера. Первый сторожевой таймер относится к оконному типу (windowed watchdog). Его необходимо обновлять только в пределах отведенных временных рамок. Если это сделать слишком рано или слишком поздно, то он сгенерирует сигнал срабатывания. Другой сторожевой таймер полностью независим от первого. Он синхронизируется от отдельного внутреннего генератора, который не связан с главной системной синхронизацией. У МК также используется система защиты синхронизации, которая может выявлять перебои в работе основного внешнего генератора и безопасно переключаться на работу от внутреннего RC-генератора частоты 8 МГц.

1.2.3. Защищенность

Еще одним непростым требованием к современным разработкам является защита кода программы от хищения. В этом плане, Flash память МК STM32 оснащена программируемой блокировкой чтения через отладочный порт. После активизации этой блокировки будет также невозможно и записать что-либо во Flash память, что исключает возможность внесения изменений в таблицу векторов прерываний. В остальной части Flash памяти может быть активирована блокировка записи. У МК STM32 также имеются часы реального времени и небольшая область энергонезависимого статического ОЗУ, которые питаются от отдельного резервного батарейного источника. В этой области имеется вход реагирования на внешнее вмешательство. При изменении состояния на данном входе генерируется прерывание и обнуляется содержимое энергонезависимого статического ОЗУ.

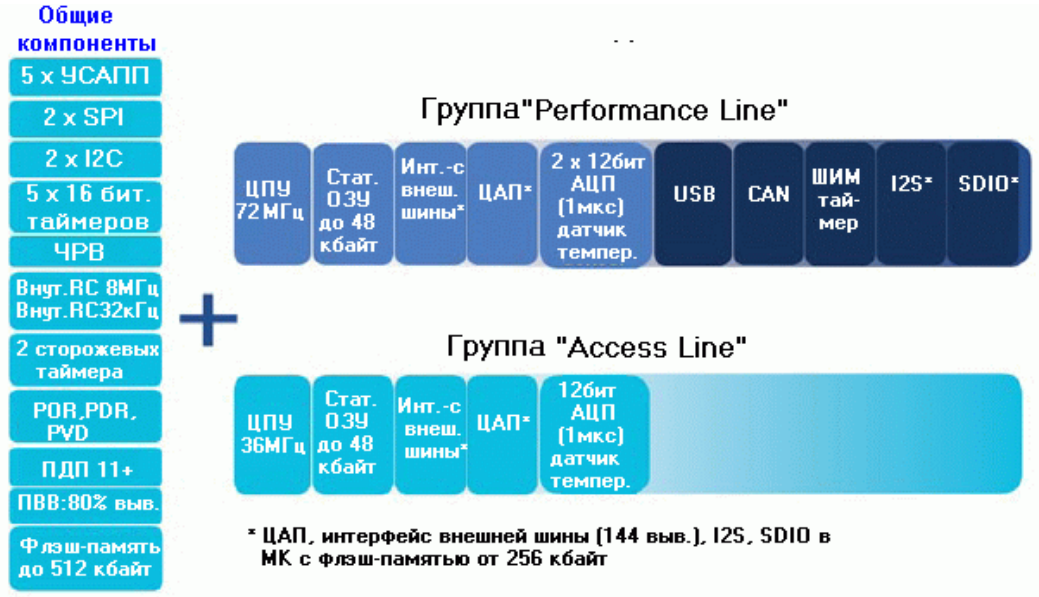
1.2.4. Разработка программ

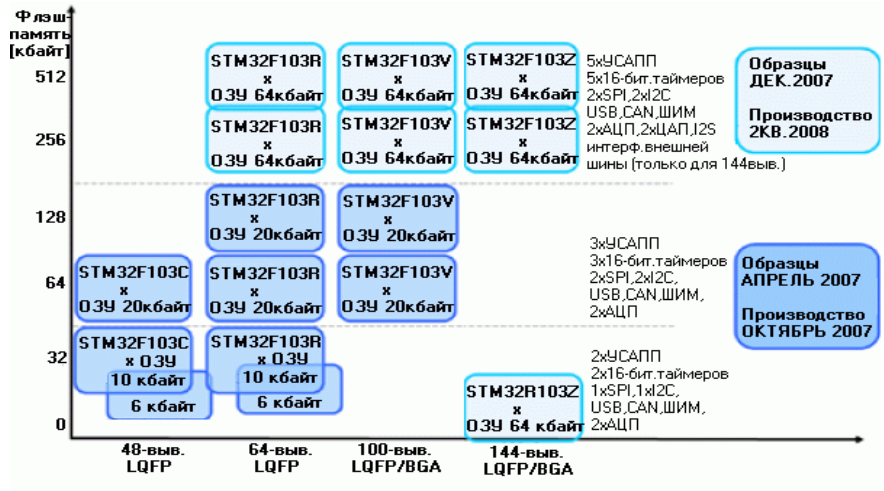
Для тех, кто уже работал с ARM-микроконтроллерами, приятной новостью может оказаться возможность поддержки имеющимися средствами для проектирования набора инструкций Thumb-2 и семейства Cortex. В худшем случае, чтобы появилась такая поддержка, потребуется обновление программного обеспечения. ST также предлагает библиотеку драйверов УВВ, библиотеку для разработки USB-приложений, а также библиотеку ANSI C и исходный код, который совместим с ранее выпущенными библиотеками для микроконтроллеров STR7 и STR9. На данный момент доступны порты данных библиотек для наиболее популярных компиляторов. Для семейства Cortex также предлагается множество коммерческих и с открытым кодом RTOS, а также связующего ПО (TCP/IP, файловые системы и т.п.).

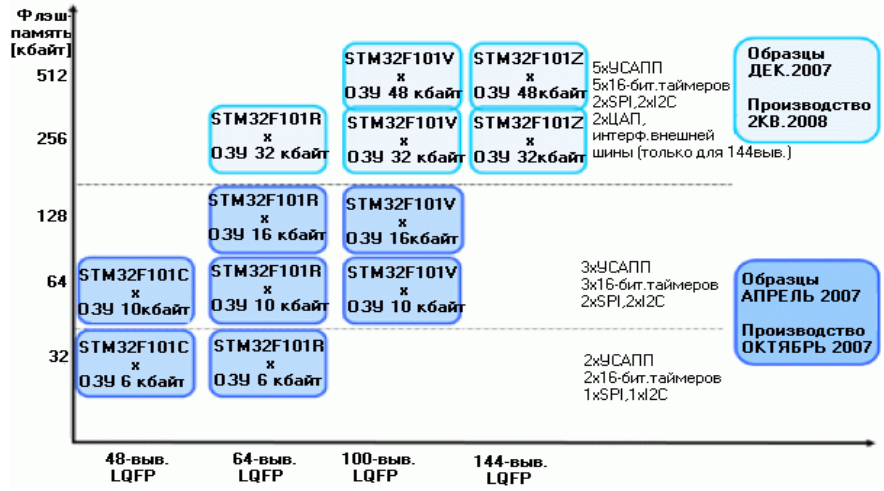
Микроконтроллеры Cortex-M3 также оснащены совершенно новой отладочной системой CoreSight. Доступ к системе CoreSight организован через специальный порт Debug Access Port (DAP порт), связь которым осуществляется, либо по стандартному интерфейсу JTAG или по последовательному 2-проводному интерфейсу. Система CoreSight, помимо управления исполнением программы в отладочном режиме, имеет возможность установки контрольных точек данных (data watchpoint) и инструментальной трассировки (instrumentation trace). Инструментальная трассировка имеет возможность отправлять выбранную прикладную информацию в отладочное средство. Эта функция может также предоставлять расширенную информацию о процессе отладки и использоваться в ходе тестирования программного обеспечения.

1.2.5. Группы Performance Line и Access Line

Семейство STM32 состоит из двух отдельных групп: Performance Line и Access Line. МК Performance Line оснащены полным набором УВВ и работают на тактовых частотах не более 72 МГц. В свою очередь, МК Access Line имеют ограниченный набор УВВ и имеют сниженное до 32 МГц быстродействие. Важно заметить, что среди микроконтроллеров разных групп имеются версии в одинаковых корпусах и с одинаковым расположением выводов. Благодаря этому, появляется возможность устанавливать на одну и ту же печатную плату совместимые микроконтроллеры из разных групп.


Общие компоненты
5 x УСАПП
2 x SPI
2 x I2C
5 x 16 бит. таймеров
ЧРВ
Внут.RC 8МГц
Внут.RC32кГц
2 сторожевых таймера
POR,PDR, PVD
ПДП 11+
ПВВ:80% выв.
Флэш-память до 512 кбайт
Группа"Performance Line"
ЦПУ 72 МГц
Стат. ОЗУ до 48 кбайт
Инт.-с внеш. шины*
ЦАП*
2 x 12бит АЦП (1мкс) датчик темпер.
USB
CAN
ШИМ тай-мер
I2S*
SDIO*
Группа "Access Line"
ЦПУ 36МГц
Стат. ОЗУ до 48 кбайт
Инт.-с внеш. шины*
ЦАП*
12бит АЦП (1мкс) датчик темпер.
* ЦАП, интерфейс внешней шины (144 выв.), I2S, SDIO в МК с флэш-памятью от 256 кбайт


Флэш-память [кбайт]
512
256
128
64
32
0
STM32F103R x ОЗУ 64кбайт
STM32F103V x ОЗУ 64кбайт
STM32F103Z x ОЗУ 64кбайт
5xУСАПП
5x16-бит.таймеров
2xSPI,2xI2C
USB,CAN,ШИМ
2xАЦП,2xЦАП,I2S
интерф.внешней шины (только для 144выв.)
Образцы ДЕК.2007
Производство 2КВ.2008
STM32F103R x ОЗУ 20кбайт
STM32F103V x ОЗУ 20кбайт
STM32F103C x ОЗУ 20кбайт
3xУСАПП
3x16-бит.таймеров
2xSPI,2xI2C,
USB,CAN,ШИМ,
2xАЦП
Образцы АПРЕЛЬ 2007
Производство ОКТЯБРЬ 2007
STM32F103C x ОЗУ 10 кбайт 6 кбайт
STM32F103R x ОЗУ 10 кбайт 6 кбайт
2xУСАПП
2x16-бит.таймеров
1xSPI,1xI2C,
USB,CAN,ШИМ,
2xАЦП
STM32R103Z x ОЗУ 64 кбайт
48-выв. LQFP
64-выв. LQFP
100-выв. LQFP/BGA
144-выв. LQFP/BGA


Флэш-память [кбайт]
512
256
128
64
32
STM32F101V x ОЗУ 48 кбайт
STM32F101Z x ОЗУ 48кбайт
5xУСАПП
5x16-бит.таймеров
2xSPI,2xI2C
2xЦАП,
интерф.внешней шины (только для 144выв.)
Образцы ДЕК.2007
Производство 2КВ.2008
STM32F101R x ОЗУ 32 кбайт
STM32F101V x ОЗУ 32 кбайт
STM32F101Z x ОЗУ 32кбайт
STM32F101R x ОЗУ 16 кбайт
STM32F101V x ОЗУ 16кбайт
STM32F101C x ОЗУ 10кбайт
STM32F101R x ОЗУ 10 кбайт
STM32F101V x ОЗУ 10 кбайт
3xУСАПП
3x16-бит.таймеров
2xSPI,2xI2C
Образцы АПРЕЛЬ 2007
Производство ОКТЯБРЬ 2007
STM32F101C x ОЗУ 6 кбайт
STM32F101R x ОЗУ 6 кбайт
2xУСАПП
2x16-бит.таймеров
1xSPI,1xI2C
48-выв. LQFP
64-выв. LQFP
100-выв. LQFP
144-выв. LQFP

2. Обзор процессоров Cortex

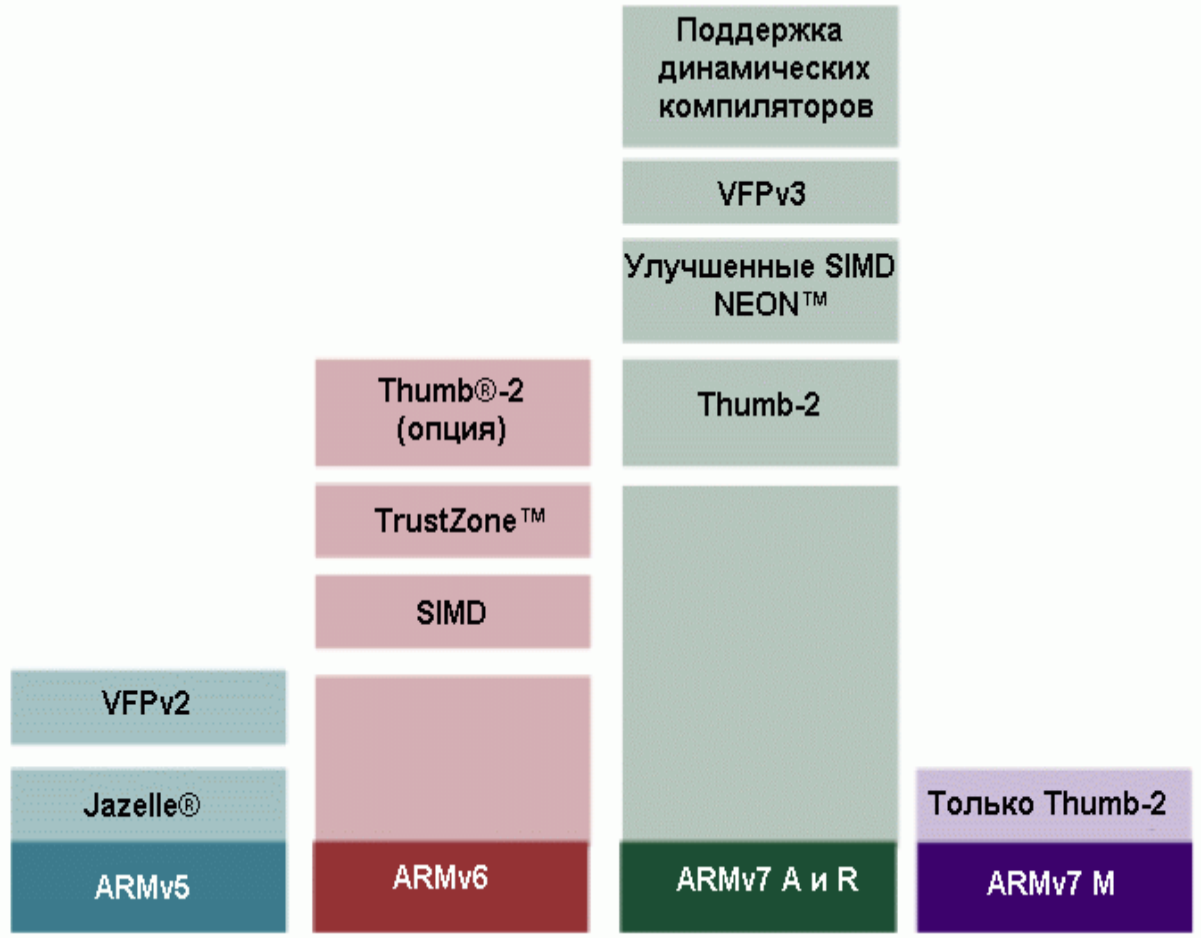
Как упоминалось во введении, процессор Cortex является встраиваемым ядром нового поколения компании ARM. Он выполнен с некоторыми отходами от идей, заложенных в основу предшествующих ЦПУ ARM, состоит из завершенного процессорного ядра (образован ЦПУ Cortex и окружающим его набором системных ресурсов) и является "сердцем" встраиваемых систем. В связи с большим разнообразием встраиваемых систем, процессор Cortex выпускается в различных прикладных профилях. Профиль обозначается после наименования Cortex. Существует три профиля:

- **Cortex-A** - прикладные процессоры для сложных операционных систем (ОС) и пользовательских приложений. Поддерживают наборы инструкций ARM, Thumb и Thumb-2.
- **Cortex-R** - профиль для операционных систем реального времени. Поддерживают наборы инструкций ARM, Thumb и Thumb-2.
- **Cortex-M** - микроконтроллерный профиль, оптимизированный под требования критичных к стоимости применений. Поддерживает только набор инструкций Thumb-2.

Число, завершающее наименование процессора Cortex, указывает на его уровень рабочих характеристик, причем 1 указывает на самый низкий уровень, а 8 - на самый высокий. На данный момент в микроконтроллерном профиле наивысший уровень характеристик - третий. Микроконтроллеры STM32 выполнены на основе процессора Cortex-M3.

2.1. Версии архитектур ARM

Иногда ARM ссылается на свои процессоры по наименованию версии архитектуры, что может сбить с толку. (Примеры записи версий архитектур: ARMV6, ARMV7 и т.д.). Версия архитектуры процессора Cortex M3 - ARMV7M.



Процессор Cortex-M3 выполнен по архитектуре ARMV7 и поддерживает выполнение инструкций Thumb-2

Таким образом, документация на Cortex-M3 состоит из справочного руководства по процессору Cortex-M3 и справочного руководства по архитектуре ARMV7M. Оба этих документа можно скачать с сайта ARM .

2.2. Процессор Cortex и ЦПУ Cortex

Чтобы подчеркнуть отличия между завершенным встраиваемым ядром Cortex и внутренним RISC ЦПУ, далее будут использоваться термины процессор Cortex и ЦПУ Cortex, соответственно. В следующем параграфе будут рассмотрены ключевые особенности ЦПУ Cortex, а затем системные ресурсы процессора Cortex.

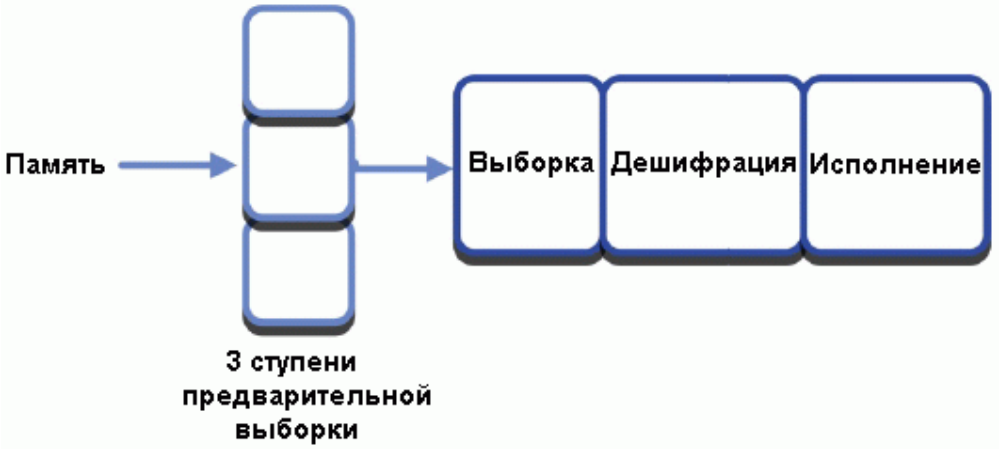
2.3. ЦПУ Cortex

Основой процессора Cortex является 32-битное RISC ЦПУ. Данное ЦПУ использует упрощенную модель программирования ARM7/9, но, при этом, более обширный набор инструкций с хорошей поддержкой целочисленной арифметики, улучшенными битовыми

операциями и более строгими реально-временными характеристиками.

2.3.1. Конвейер

ЦПУ Cortex способно выполнять большинство инструкций за один цикл. Также как и у ЦПУ ARM7 и ARM9, это достигается с помощью трехступенчатого конвейера.



ЦПУ Cortex-M3, также как и ARM7/ARM9, использует трехступенчатый конвейер. Однако Cortex-M3 также поддерживает предсказание переходов для минимизации количества перезагрузок конвейера

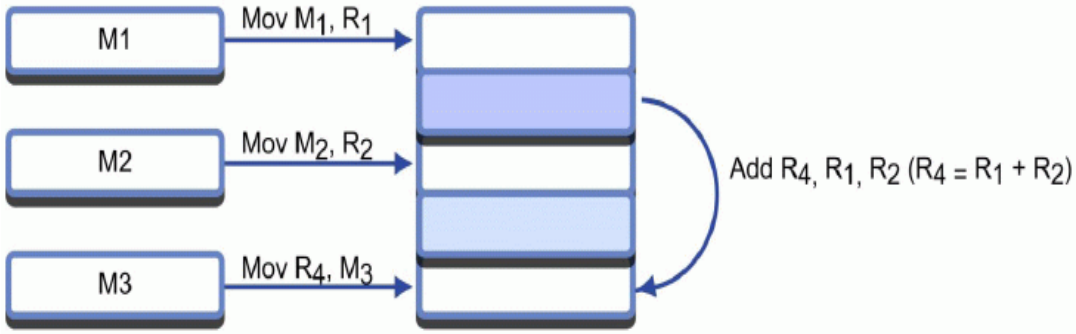
Во время выполнения одной инструкции, следующая - дешифрируется, а третья - считывается из памяти. Этот механизм отлично работает с линейным кодом, но, если требуется выполнить переход, то, прежде чем продолжить выполнение кода программы, потребуется очистка и перезагрузка конвейера. У ЦПУ ARM7 и ARM9 переходы существенно ограничивают производительность исполнения кода программы. Во избежание этого, трехступенчатый конвейер ЦПУ Cortex оснащен логикой предсказания переходов. Это означает, что при достижении инструкции условного перехода выполняется упреждающая выборка и, в результате, оба назначения инструкции условного перехода будут доступны для исполнения и снижение производительности не произойдет. Хуже обстоят дела с инструкциями косвенного перехода, т.к. в этом случае упреждающую выборку выполнить нельзя и перезагрузка конвейера может оказаться неизбежной.

Таким образом, конвейер - инструмент, от которого зависит результирующая производительность ЦПУ Cortex и который не требует каких-либо действий со стороны кода программы.

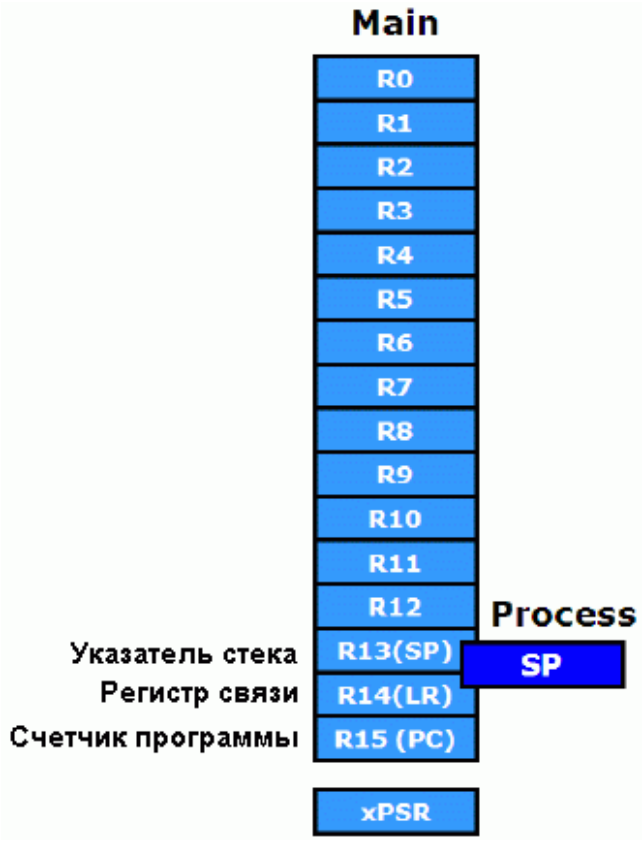
2.3.2. Модель программирования

ЦПУ Cortex является RISC-процессором, который выполнен по архитектуре чтения/записи. Для выполнения операций обработки данных вначале необходимо поместить операнды из памяти в центральный регистровый файл, затем выполнить требуемую

операцию над данными в регистрах и, наконец, перезаписать результат обратно в память.



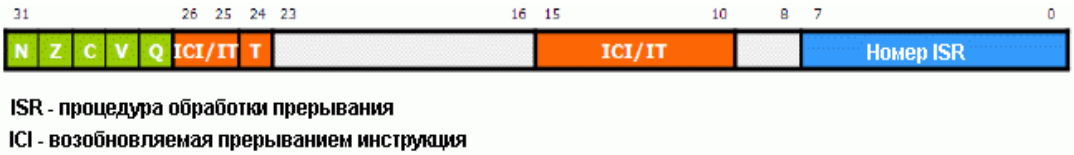
Cortex-M3 выполнен по архитектуре чтения/записи. Все данные, перед выполнением инструкции их обработки, необходимо поместить в центральный регистровый файл. Следовательно, вся активность программы фокусируется вокруг регистрового файла ЦПУ. Данный регистровый файл образуют шестнадцать 32-битных регистров. Регистры R0-R12 - обычные регистры, которые могут использоваться для хранения программных переменных. У регистров R13-R15 имеются особые функции в рамках ЦПУ Cortex. Регистр R13 выступает в роли указателя стека. Данный регистр является банковым, что делает возможной работу ЦПУ Cortex в двух режимах работы, в каждом из которых используется свое собственное пространство стека. Данная возможность обычно используется операционными системами реального времени (RTOS), которые могут выполнять свой "системный" код в защищенном режиме. У двух стеков ЦПУ Cortex имеются собственные наименования: основной стек и стек процесса. Следующий регистр R14 называется регистром связи. Он используется для хранения адреса возврата из подпрограммы. Благодаря нему, ЦПУ Cortex быстро переходит к подпрограмме и выходит из нее. Если же в программе используется несколько уровней вложений подпрограмм, то компилятор будет автоматически сохранять R14 в стек. Последний регистр R15 - счетчик программы; поскольку он является частью центрального регистрового файла, его чтение и обработка может выполняться аналогично любым другим регистрам.



У ЦПУ Cortex-M3 имеется регистровый файл, состоящий из 16 32-битных регистров. Также как и у предшествующих ЦПУ ARM7/9 регистр R13 выступает в роли указателя стека. R14 - регистр связи, R15 - счетчик программы. R13 является банковым регистром, что позволяет Cortex-M3 работать с двумя стеками: стеком процесса и основным стеком

2.3.2.1. XPSR

Помимо регистрового файла, имеется отдельный регистр, который называется регистром статуса программы. Он не входит в основной регистровый файл, а доступ к нему возможен с помощью двух специальных инструкций. В xPSR хранятся значения полей, влияющих на исполнение инструкций ЦПУ Cortex.



Регистр статуса программы содержит поля статуса, от которых зависит исполнение инструкций. Данный регистр разделен еще на три регистра: регистры статуса прикладной программы, исполнения программы и прерываний

Биты регистра xPSR разделены на три группы, к каждой из которых возможен доступ по собственному наименованию. Верхние пять бит (флаги кода условия) именуются регистром статуса прикладной программы. Первые четыре флага кода условия N, Z, C, V

(индикация отрицательного (N) или нулевого (Z) результата, переноса (C) и переполнения (V)) устанавливаются и сбрасываются по итогам выполнения инструкции обработки данных. Пятый бит Q используется при выполнении математических инструкций с насыщением алгоритмов цифровой обработки сигналов (ЦОС) для индикации достижения переменной своего максимального или минимального значения. Также как и 32-битные инструкции ARM, некоторые инструкции Thumb-2 выполняются только при условии совпадения кода условия инструкции и состояния флагов регистра статуса прикладной программы. Если коды условия инструкции не совпадают, то инструкция проходит по конвейеру как NOP (нет операции). Этим гарантируется равномерность прохождения инструкций по конвейеру и минимизируется число перезагрузок конвейера. У Cortex данный способ расширен регистром статуса исполнения программы, который связан с битами 26..8 регистра xPSR. Этот регистр состоит из трех полей: поле "If then" (IT), поле возобновляемой прерыванием инструкции и поле инструкции Thumb. Набор инструкций Thumb-2 реализует эффективный метод выполнения компактных блоков инструкций типа 'if then'. Если проверяемое условие истинно, записью значения в поле IT можно сигнализировать ЦПУ о необходимости выполнения до четырех следующих инструкций. Если же проверяемое условие - ложное, то данные инструкции пройдут по конвейеру как NOP. Ниже приводится пример кодирования типичной строки программы на Си:

```
If (r0 ==0)
CMP r0,#0 проверка r0 на ноль
ITTEE EQ если Да, выполняются следующие две инструкции
Then r0 = *r1 +2;
LDR r0,[r1] загрузка данных из памяти в r0
ADDr0,#2 сложение с 2
```

Несмотря на то, что большинство инструкций Thumb-2 выполняются за один цикл, несколько инструкций (например, инструкции чтения/записи) требуют для выполнения несколько циклов. Чтобы точно знать время отклика ЦПУ Cortex на прерывания, данные инструкции должны быть прерываемыми. В случае преждевременного прекращения исполнения инструкции в поле возобновляемых прерываниями инструкций запоминается номер следующего регистра, подлежащего обработке инструкцией многократного чтения или записи. Таким образом, сразу после завершения процедуры обработки прерывания, выполнение инструкции многократного чтения/записи может быть восстановлено. Последнее поле Thumb предусмотрено для совместимости с предшествующими ЦПУ ARM. Данное поле сигнализирует, что в данный момент ЦПУ выполняет инструкцию ARM или Thumb. У ЦПУ Cortex-M3 данный бит всегда равен единице. Наконец, в поле статуса прерывания хранится информация о любых приостановленных запросах прерывания.

2.3.3. Режимы работы ЦПУ

Несмотря на то, что процессор Cortex разрабатывался как быстродействующее, простое в использовании и с малым числом логических элементов микроконтроллерное ядро, в нем была учтена поддержка RTOS. Процессор Cortex поддерживает два режима

работы: режим Thread (или потоковый режим) и режим Handler (или режим обработчика). ЦПУ запускается в режиме Thread при непрерываемом, фоновом выполнении инструкций и переключается в режим Handler при обработке исключительных ситуаций. Кроме того, ЦПУ Cortex может выполнять код программы в привилегированном или непривилегированном режиме. В привилегированном режиме, ЦПУ имеет доступ ко всему набору инструкций, а в непривилегированном режиме некоторые инструкции отключаются (например, инструкции MRS и MSR, осуществляющие доступ к регистру xPSR и его битовым группам). В этом режиме также отключается доступ к большинству регистров управления системными ресурсами процессора Cortex. Также можно сконфигурировать использование стека. Основной стек (R13) может использоваться в обоих режимах Thread и Handler. Альтернативно, режим Handler можно настроить на использование стека процесса (банковый регистр R13).

Процессор Cortex-M3 может использоваться в обычном режиме ('flat'), а также поддерживает операционные системы реального времени. У него предусмотрены режимы Handler и Thread с возможностями выбора используемого стека (основной стек или стек процесса) и привилегированного доступа к регистрам управления системными ресурсами Cortex

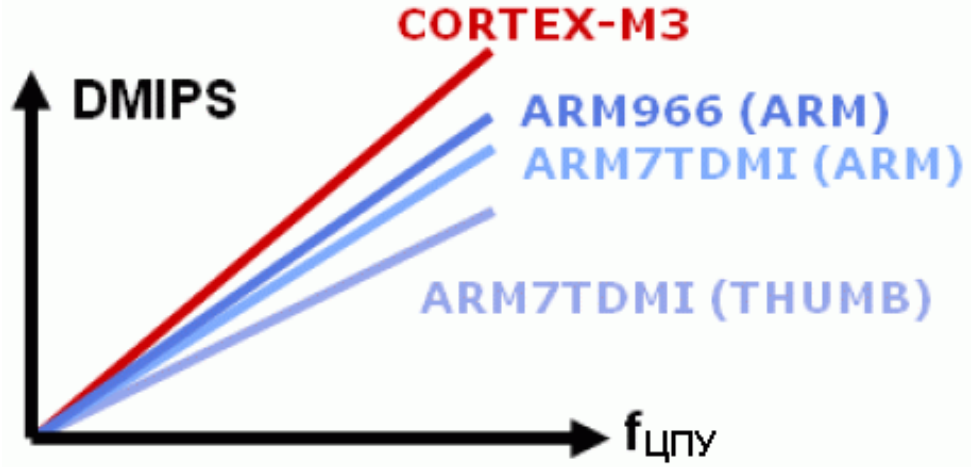
		Операции (после сброса - привилегированные)	Стек (после сброса - основной стек)
Режимы (Thread)	Handler - обработка исключительных ситуаций	Привилегированное исполнение Полное управление	Основной стек используется ОС и при обработке исключительных ситуаций
	Thread - исключительные ситуации не обрабатываются	Привилегированные/ непривилегированные	Основной стек или стек процесса

	- обычное выполнение кода		

Сразу после сброса процессор Cortex запускается в конфигурации 'flat'. В обоих режимах, Thread и Handler, инструкции выполняются в привилегированном режиме, т.о. какие-либо ограничения на доступ к процессорным ресурсам отсутствуют. В режимах Thread и Handler используется основной стек. Чтобы начать выполнение инструкций, достаточно указать процессору вектор сброса и стартовый адрес стека, после чего можно выполнять Си-код программы. Однако, если используется RTOS или выполняется разработка критичного к безопасности приложения, процессор может использоваться в более изощренной конфигурации: режим Handler (используется при обработке исключительных ситуаций и операционными системами реального времени) с привилегированными операциями и использованием основного стека и режим Thread при исполнении прикладного кода с непривилегированным доступом и использованием стека процесса. При таком подходе, весь код программы разделяется на системный и прикладной и, поэтому, ошибки в прикладном коде не вызывают сбоев RTOS.

2.3.4. Набор инструкций Thumb-2

ЦПУ ARM7 и ARM9 поддерживают два набора инструкций: 32-битный ARM и 16-битный Thumb. Благодаря этому, разработчик имеет возможность оптимизировать свою программу путем использования более оптимального набора инструкций для каждой конкретной процедуры: 32-битные инструкции, где более важно быстродействие, и 16-битные, где более важна плотность кода. ЦПУ Cortex поддерживает набор инструкций Thumb-2, который являются смесью 16- и 32-битных инструкций. Инструкции thumb-2 дают улучшение плотности кода на 26% по сравнению с 32-битными инструкциями ARM и производительности на 25% по сравнению с 16-битными инструкциями Thumb. В наборе инструкций Thumb-2 предусмотрено несколько улучшенных инструкций умножения, исполняющихся за один цикл, и аппаратный делитель, требующий 2..7 циклов.



По итогам тестирования процессор Cortex демонстрирует уровень производительности 1.2 DMIPS/МГц, что эквивалентно 1.2 циклам тактирования на инструкцию

Источник	Назначение	Циклы

16 бит x 16 бит	32 бит	1
32 бит x 16 бит	32 бит	1
32 бит x 32 бит	32 бит	1
32 бит x 32 бит	64 бит	3..7*

В наборе инструкций Thumb-2 также предусмотрены улучшенные инструкции переходов, в т.ч. с проверкой и сравнением; блоки условного выполнения типа if/ then и упорядочивание байт для обработки данных; а также инструкции извлечения байт или полуслов. Будучи RISC-процессором, ЦПУ Cortex обладает обширным набором инструкций, который специально разработан с учетом его использования Си-компилятором. Типичная программа для Cortex-M3 может быть полностью написана на ANSI Си с минимальным числом несовместимых с ANSI ключевых слов, за исключением таблицы векторов исключительных ситуаций, которую необходимо написать на Ассемблере.

2.3.5. Карта памяти

Процессор Cortex-M3 является стандартизованным микроконтроллерным ядром и, поэтому, его карта памяти четко расписана. Несмотря на использование нескольких внутренних шин, адресное пространство является линейным и имеет размер 4 Гбайт.
Первые 1 Гбайт памяти разделены равномерно между областью кода программы и областью статического ОЗУ. Пространство кода программы оптимизировано для работы с шиной I-Code. Аналогично, пространство статического ОЗУ доступно через шину D-code. Несмотря на то, что в области статического ОЗУ поддерживается загрузка и исполнение инструкций, их выборка осуществляется через системную шину, что требует дополнительного состояния ожидания.

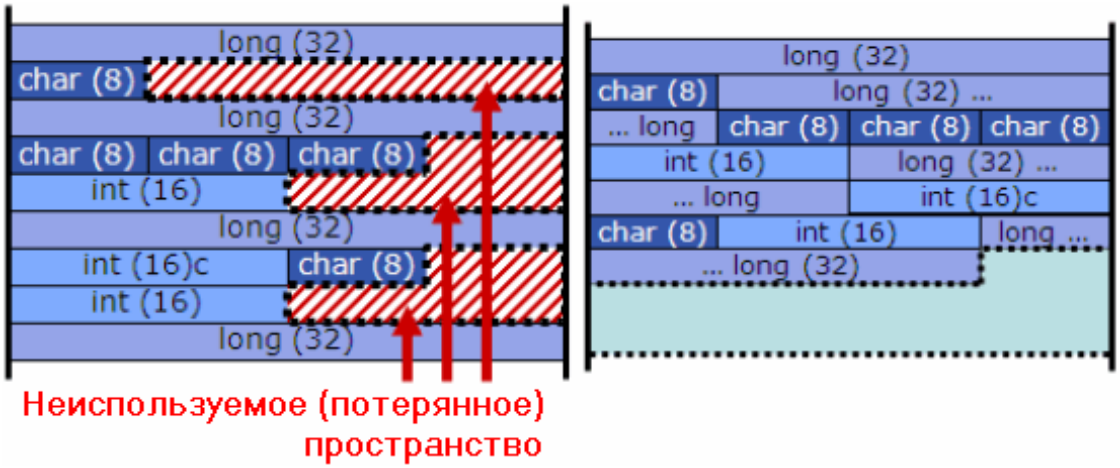
Таким образом, выполнение кода программы из статического ОЗУ будет более медленным, чем из встроенной Flash памяти, расположенной в области кода программы. Следующие 0.5 Гбайт памяти - область встроенных УВВ. В этой области находятся все предоставляемые пользователю производителем микроконтроллера УВВ. Первые 1 Мбайт в областях статического ОЗУ и УВВ являются битноадресуемыми. Для этого используется метод bit banding. Таким образом, все данные, хранящиеся в этих областях, могут обрабатываться как пословно, так и побитно. Следующие 2 Гбайт адресного пространства выделены для внешних статического ОЗУ и УВВ. Последние 0.5 Гбайт зарезервированы для системных ресурсов процессора Cortex и будущих расширений процессора Cortex. Все регистры процессора Cortex расположены по фиксированным адресам во всех Cortex-микроконтроллерах. Благодаря этому, облегчается портирование программ между различными МК STM32 и Cortex-микроконтроллерами других производителей.

Для процессора Cortex-M3 определена фиксированная карта памяти размером 4 Гбайт, в которой выделены конкретные области для хранения кода программы, статического ОЗУ, устройств ввода-вывода, внешней памяти и устройств, а также системных регистров

Cortex. Данная карта памяти одинакова для всех Cortex-микроконтроллеров

2.3.6. Доступ к фрагментированным данным

Набор инструкций ARM7 и ARM9 имеет возможность доступа к знаковым и беззнаковым переменным типа байт, полуслово и слово. Благодаря этому, ЦПУ естественным образом поддерживает целочисленные переменные без необходимости использования программных библиотек, как в случае 8- и 16-битных микроконтроллеров. Однако у предшествующих ЦПУ ARM есть один недостаток, который заключается в их способности оперировать только с нефрагментированными словами или полусловами. Вследствие этого линкер компилятора неэффективно размещает данные в статическом ОЗУ и некоторая его часть будет потеряна. (В зависимости от комбинации используемых переменных потери могут достигать 25%.)

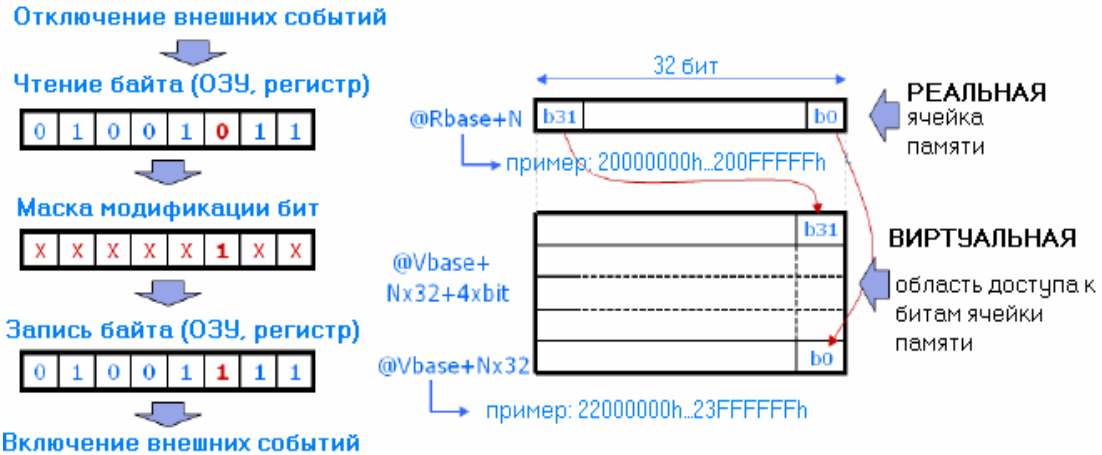


Cortex-M3 может осуществлять доступ к фрагментированным данным, что гарантирует эффективность использования статического ОЗУ

ЦПУ Cortex поддерживает режимы адресации для слов, полуслов, байт и может осуществлять фрагментированный доступ к данным. Благодаря этому, линкеру компилятора предоставляется полная свобода в очередности размещения данных в памяти. Кроме того, поддерживаемая ЦПУ Cortex возможность bit banding позволяет группировать флаги программы в переменные типа слово или полуслово, а не использовать для каждого флага отдельный байт.

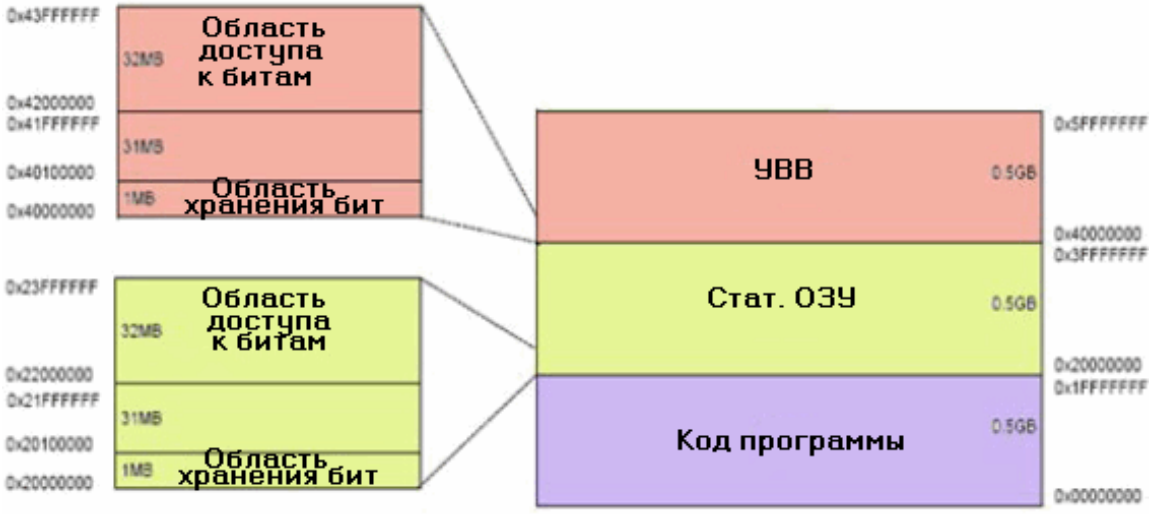
2.3.7. Метод "Bit Banding"

У предшествующих ЦПУ ARM7 и ARM9 битовые операции в статическом ОЗУ можно было выполнять только с помощью инструкций AND и OR. Для этого необходимо выполнить последовательность ЧТЕНИЕ - МОДИФИКАЦИЯ - ЗАПИСЬ. Однако использование этого метода приводит к чрезмерному расходованию количества циклов на выполнение установки и сброса отдельных бит и увеличению результирующего кода программы.



Метод bit banding позволяет выполнять битовые операции при сохранении логических вентилей ЦПУ Cortex-M3 на минимальном уровне

Преодолеть данное ограничение можно, если предусмотреть отдельные инструкции сброса и установки бит или интегрировать полнофункциональный процессор битовой обработки, однако это приведет к увеличению размеров и сложности ЦПУ. Вместо этого, способ, называемый bit banding, позволяет напрямую воздействовать на биты в памяти из областей УВВ и статического ОЗУ, не используя при этом каких-либо специальных инструкций. Битноадресуемые области карты памяти Cortex разделены на две части: область хранения бит (в нее входят до 1 Мбайт физической памяти или регистров УВВ) и область доступа к битам, которая занимает до 32 Мбайт карты памяти. Получить доступ к каждому отдельному биту из области хранения бит можно по соответствующему адресу слова из области доступа к битам. Таким образом, если выполнять запись по адресу в области доступа к битам на самом деле мы будем воздействовать на значение определенного бита в физической памяти.



Метод Bit Banding поддерживается в областях статического ОЗУ и УВВ (первые 1 Мбайт), охватывая все ресурсы STM32

В итоге, мы можем воздействовать на значение отдельных бит, не прибегая, при этом, к использованию специальных инструкций и сохраняя результирующие размеры ядра Cortex на минимально возможном уровне. Чтобы использовать этот метод на практике, необходимо вычислить адрес слова в области доступа к битам, который соответствует заданной ячейки памяти из области УВВ или статического ОЗУ. Выполняется это по следующей формуле:

Адрес в области доступа к битам **=** *Базовый адрес области доступа к битам* **+** *Смещение адреса слова доступа к биту*,

где *Смещение адреса слова доступа к биту = Смещение в байтах по отношению базовому адресу области хранения бит x 0x20 + номер бита x 4*

На самом деле все обстоит гораздо проще, чем может показаться на первый взгляд. Рассмотрим практический пример. Необходимо выполнить запись в выходной регистр порта ввода-вывода (ПВВ) для установки или сброса отдельных линий ввода-вывода.

Физический адрес выходного регистра порта B - 0x40010C0C. Предположим, что нужно устанавливать и сбрасывать бит 8 этого регистра. Воспользуемся приведенной выше формулой:

```
Адрес слова = 0x40010C0C
  Базовый адрес области хранения бит УВВ = 0x40000000
  Базовый адрес области доступа к битам УВВ = 0x42000000
  Смещение в байтах по отношению базовому адресу области хранения бит
= 0x40010C0C - 0x40000000 = 10C0C
  Смещение адреса слова доступа к биту = (0x10C0C x 0x20) +(8x4) =
0x2181A0
  Адрес в области доступа к битам = 0x42000000 + 0x2181A0 = 0x422181A0
```

Теперь мы можем создать указатель на этот адрес с помощью следующей Си-строки:

```
#define PortBbit8 (*((volatile unsigned long *) 0x422181A0 ))
```

После этого, этот указатель можно использовать для установки и сброса бит ПВВ:

```
PB8 = 1; //включаем светодиод
```

После компиляции будут сгенерированы следующие ассемблерные инструкции:

```
MOVS r0,#0x01
LDR r1,[pc,#104]
STR r0,[r1,#0x00]
```

Для отключения светодиода используем строку:

```
PB8 = 0; // отключаем светодиод
```

Ей соответствуют следующие ассемблерные инструкции:

```
MOVS r0,#0x00
LDR r1,[pc,#88]
STR r0,[r1,#0x00]
```

Таким образом, для установки или сброса бита необходимо выполнить три 16-битных инструкции. Микроконтроллер STM32, работающий на частоте 72 МГц, выполнит их за 80 нс. Альтернативно установку или сброс бита можно выполнить, если применить логическую операцию "ИЛИ" или "И" ко всему слову из области хранения бит УВВ или статического ОЗУ:

```
GPIOB->ODR |= 0x00000100; //Включение светодиода
      LDR r0,[pc,#68]
      ADDS r0,r0,#0x08
```

```
      LDR r0,[r0,#0x00]
      ORR r0,r0,#0x100
      LDR r1,[pc,#64]
      STR r0,[r1,#0xC0C]
GPIOB->ODR &=!0x00000100; //Отключение светодиода
      LDR r0,[pc,#40]
      ADDS r0,r0,#0x08
      LDR r0,[r0,#0x00]
      MOVS r0,#0x00
LDR r1,[pc,#40]
      STR r0,[r1,#0xC0C]
```

Но в таком случае, одна операция установки/сброса потребует выполнения смеси 16- и 32-битных инструкций, которые займут минимум 14 байт и на той же тактовой частоте будут выполняться как минимум 180 нс. Таким образом, в программе, где используется установка/сброс множества бит в регистрах УВВ, а также применяются семафоры и флаги в статическом ОЗУ, использование метода bit banding позволит существенно сэкономить, как размер кода программы, так и время его выполнения.

2.4. Процессор Cortex

2.4.1. Шины

Процессор Cortex-M3 выполнен по Гарвардской архитектуре, которая подразумевает использование раздельных шин данных и инструкций. Они называются шиной Dcode и Icode, соответственно. Обе эти шины могут осуществлять доступ к инструкциям и данным в диапазоне адресов 0x00000000 - 0x1FFFFFFF. Также имеется дополнительная системная шина, которая предоставляет доступ к области системного управления по адресам 0x20000000-0xDFFFFFFF и 0xE0100000-0xFFFFFFFF. У встроенной отладочной системы процессора Cortex имеется еще одна дополнительная шинная структура, которая называется локальной шиной УВВ (Private Peripheral Bus, PPB).

2.4.2. Матрица шин

Системная шина и шина данных подключаются к внешнему микроконтроллеру через набор высокоскоростных шин, называемых матрицей шин. Она образует несколько параллельных соединений между шинами Cortex и другими внешними шинными мастерами, как например, каналы ПДП к встроенным ресурсам, статическое ОЗУ и УВВ. Если два шинных мастера (например, ЦПУ Cortex и канал ПДП) предпринимают попытку доступа к одному и тому же УВВ, то вступит в действие внутренний арбитр, который разрешит конфликт, предоставив доступ к шине тому, кто имеет наивысший приоритет . Однако, благодаря тесной взаимосвязи блоков ПДП с ЦПУ Cortex, необходимость арбитража во многих случаях исключается. В этом мы убедимся при рассмотрении особенностей блока ПДП.

2.4.3. Системный таймер

В ядро Cortex входит 24-битный вычитающий счетчик с функциями автоматической перезагрузки и генерации прерывания. Он называется таймером SysTick и предназначен для использования в качестве стандартного таймера во всех Cortex-микроконтроллерах. Таймер SysTick может использоваться для формирования шкалы времени в RTOS или для генерации периодических прерываний для обработки запланированных задач. С помощью регистра управления и статуса таймера SysTick, который расположен в области системных ресурсов процессора Cortex-M3, пользователь может выбрать источник синхронизации таймера. Если установить бит CLKSOURCE, то таймер SysTick будет работать на тактовой частоте ЦПУ. Если же его сбросить, то таймер будет работать на частоте, равной 1/8 тактовой частоты ЦПУ.

Таймер SysTick - 24-битный автоматически-перезагружаемый таймер, являющийся частью процессора Cortex-M3. Он предназначен для формирования шкалы времени в операционных системах реального времени

У таймера SysTick имеется три регистра. Для задания периодичности счета необходимо инициализировать регистр текущего значения и регистр перезагружаемого значения. В регистре управления и статуса имеются биты ENABLE, позволяющий активизировать работу таймера, и TICKINT, управляющий активностью линии прерывания таймера. Далее мы будем рассматривать структуру прерываний Cortex и использование таймера SysTick
для генерации первой исключительной ситуации в микроконтроллере STM32.

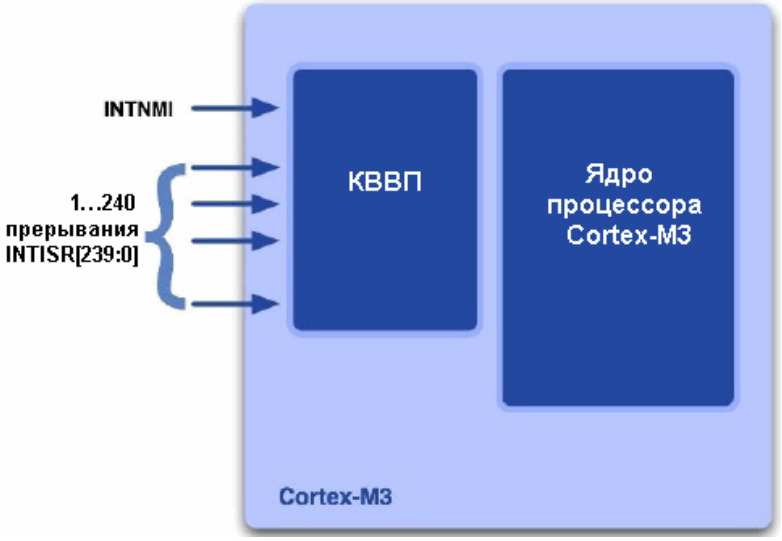
2.4.4. Обработка прерываний

Одними из главных усовершенствований ядра Cortex по сравнению с предшествующими ЦПУ ARM являются структура прерываний и механизм обработки исключительных ситуаций. ЦПУ ARM7 и ARM9 использовали две линии прерывания: быстродействующая линия прерывания и линия прерывания общего назначения. Данные линии поддерживали все источники прерываний в рамках микроконтроллера конкретного производителя. Однако, несмотря на использование казалось бы одинаковых подходов, конкретная

реализация могла отличаться между разными производителями МК. Структуре прерываний ARM7 и ARM9 свойственны еще две проблемы. Во-первых, она недетерминистическая, т.к. время, которое требуется на завершение текущей инструкции при возникновении прерывания непостоянно. Конечно же, во многих приложениях это не создает проблем, но в системах реального времени могут возникнуть большие трудности. Во-вторых, поддержка вложенных прерываний в архитектуре ARM7 и ARM9 реализована неэффективно и требует написания дополнительных кодов программы в виде ассемблерных макросов или RTOS. Таким образом, ключевой задачей, которая стояла перед разработчиками ядра Cortex, являлась преодоление всех этих ограничений и разработка стандартной структуры прерываний, отличающейся предельным быстродействием и детерминистичностью.

2.4.5. Контроллер вложенных векторизованных прерываний

Контроллер вложенных векторизованных прерываний (NVIC) является стандартным блоком ядра Cortex. Это означает, что у любого Cortex-микроконтроллера будет присутствовать одна и та же структура прерываний, независимо от его производителя. Таким образом, прикладной код и операционные системы можно легко портировать с одного МК на любой другой и, при этом, программисту не потребуется изучение нового набора регистров. При разработке NVIC также учитывалось, что задержка реагирования на прерывание должна быть очень малой. Данная задача решена двояким образом: собственно возможностями NVIC, а также набором инструкций Thumb-2, многоцикловые инструкции которого, как например, инструкции многократного чтения/записи, являются прерываемыми. Задержка реагирования на прерывание то же является детерминистичной, что важно для систем реального времени. Как следует из наименования NVIC, им поддерживаются вложенные прерывания и, в частности, у МК STM32 используется 16 уровней приоритетов. Структура прерываний NVIC разработана с учетом программирования полностью на Си и исключает потребность в написании каких-либо ассемблерных макросов или специальных, несовместимых с ANSI, директив.



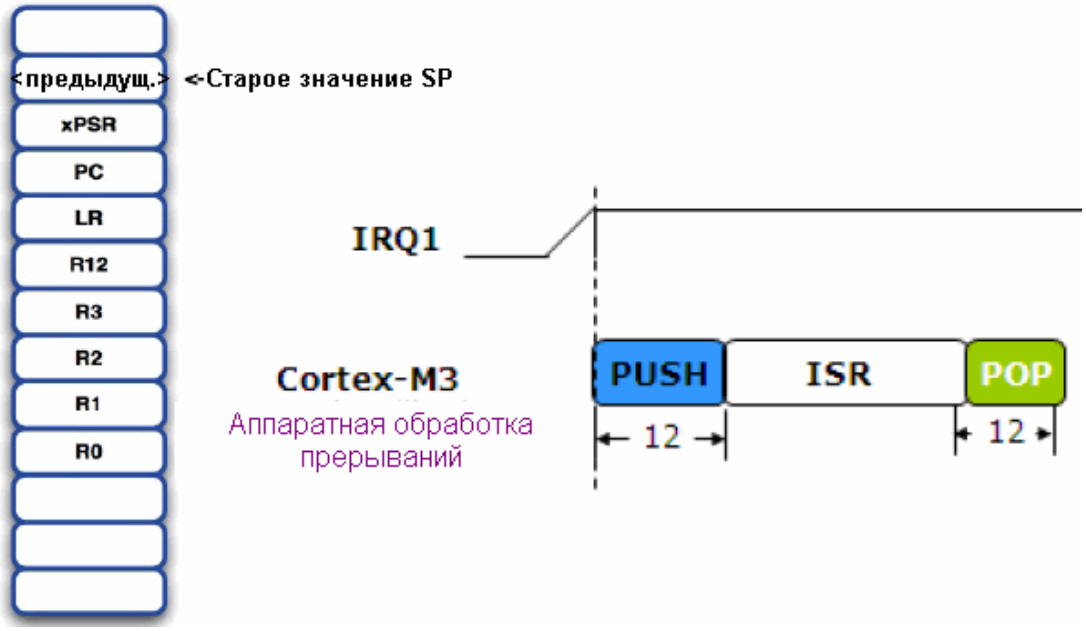
В процессор STM32 входит контроллер вложенных векторизованных прерываний, поддерживающий до 240 внешних УВВ

Несмотря на то, что NVIC является стандартным блоком ядра Cortex, в целях

минимизации количества логических вентилей, разработчик МК может сконфигурировать количество линий прерываний, идущих к NVIC. Контроллер поддерживает одно немаскируемое прерывание и еще до 240 внешних линий прерывания, которые можно подключить к пользовательским УВВ. В ядре Cortex поддерживается еще 15 дополнительных источников прерываний, использующихся для обработки внутренних исключительных ситуаций ядра Cortex. Максимальное число маскируемых линий прерывания NVIC микроконтроллеров STM32 равно 43.

2.4.5.1. Работа NVIC при входе в исключительные ситуации и выходе из них

Если прерывание инициируется УВВ, то NVIC подключит ЦПУ Cortex к обработке прерывания. После перехода ЦПУ Cortex в режим прерывания, он помещает набор регистров в стек. Эта операция выполняется с помощью специального микрокода, что упрощает прикладной код. В процессе записи данных в стек на шине инструкций осуществляется выборка начального адреса процедуры обработки прерывания. Благодаря этому, с момента возникновения прерывания до выполнения первой инструкции его обработки проходит всего лишь 12 циклов.



NVIC реагирует на обработку прерывания с задержкой всего лишь 12 циклов. В них входит выполнение микрокода, который автоматически помещает набор регистров в стек

К числу помещаемых в стек данных относятся регистр статуса программы, счетчик программы и регистр связи. Благодаря этому, запоминается состояние, в котором находилось ЦПУ Cortex CPU. Кроме того, также сохраняются регистры R0 - R3. Эти регистры широко используются в инструкциях для передачи параметров, поэтому, помещение в стек делает возможным их использование в процедуре обработке прерывания. Замыкает список помещаемых в стек регистров - R12. Он выступает в роли рабочего регистра внутри подпрограммы. Например, если в компиляторе активизировать проверку стека, то будет генерироваться дополнительный код, который при потребности в регистре ЦПУ будет использовать R12. По завершении обработки прерывания

все действия выполнятся в обратном порядке: с помощью микрокода извлекается содержимое стека и, параллельно с этим, осуществляется выборка адреса возврата, таким образом, для возобновления выполнения фоновой программы потребуется 12 циклов.

2.4.5.2. Улучшенные режимы обработки прерывания

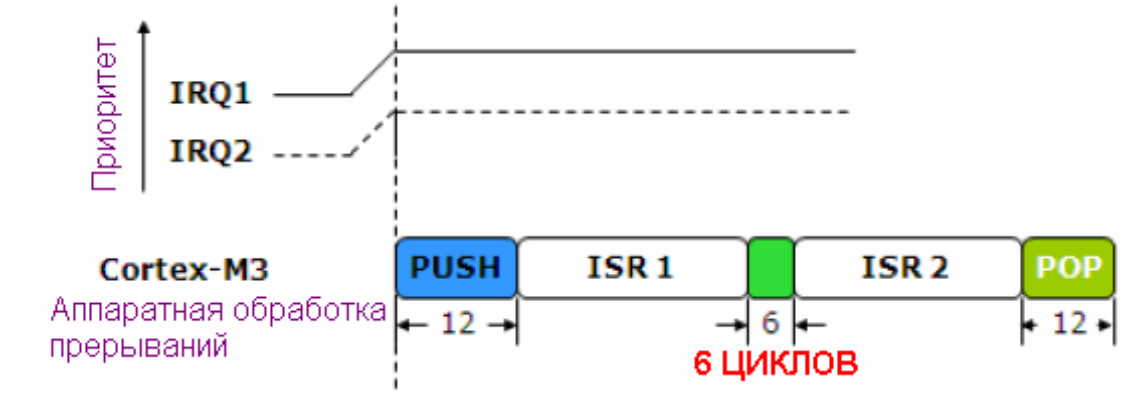
Помимо высокого быстродействия обработки одного прерывания, NVIC также характеризуется эффективной обработкой нескольких прерываний, что важно для высокобыстродействующих систем реального времени. Для этого у NVIC поддерживается несколько оригинальных методов, позволяющие обрабатывать несколько запросов прерываний с минимальными задержками между прерываниями и с гарантированием приоритетности обработки прерываний.

2.4.5.2.1. Приостановка прерываний

NVIC имеет возможность приостановки находящегося на обработке прерывания, если возникает прерывание с более высоким приоритетом. В этом случае, обработка активного прерывания прекращается, в течение последующих 12 циклов выполняется сохранение в стек нового набора данных и запускается обработка высокоприоритетного прерывания. По завершении его обработки, содержимое стека автоматически извлекается и обработка низкоприоритетного прерывания возобновляется.

2.4.5.2.2. Непрерывная обработка прерываний с исключением внутренних операций над стеком

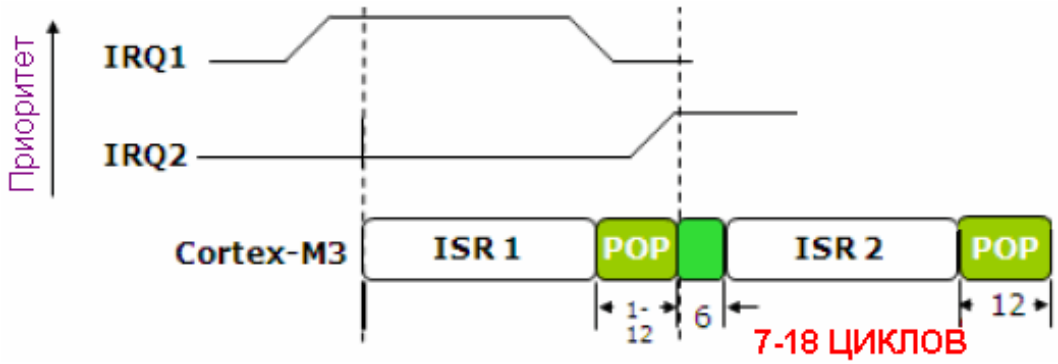
Если на обработке находится высокоприоритетное прерывание и, при этом, возникает низкоприоритетное, NVIC Cortex использует метод непрерывной обработки с исключением внутренних операций над стеком, который гарантирует минимальность задержки при переходе к обработке следующего прерывания.



Несколько прерываний обрабатываются непрерывно с исключением внутренних операций над стеком, поэтому задержка после завершения обработки одного прерывания и перед началом обработки следующего минимальна

Если возникает два прерывания, первым со стандартной задержкой в 12 циклов

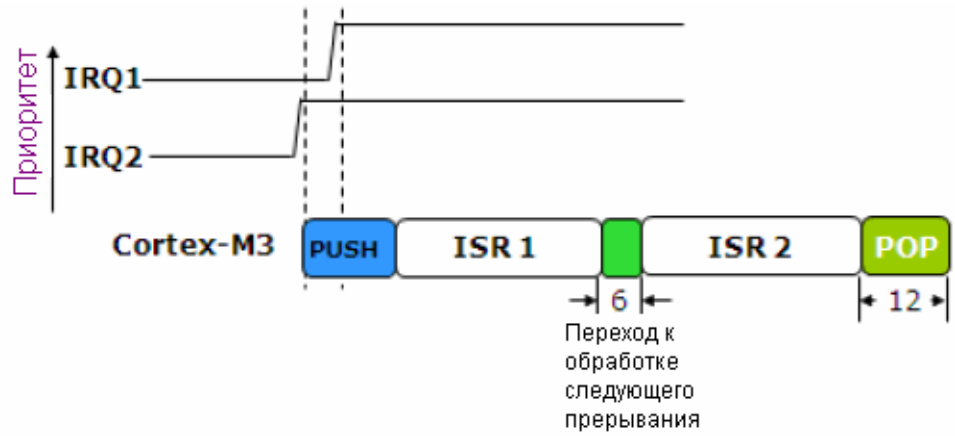
обслуживается прерывание с более высоким приоритетом. Однако, по окончании его обработки, ЦПУ Cortex не возвращается к выполнению фоновой программы и содержимое стека не извлекается. Вместо этого, осуществляется выборка адреса процедуры обработки следующего прерывания с учетом приоритета. Таким образом, задержка перехода к обработке следующего прерывания составит всего лишь 6 циклов. По завершении обработки последнего прерывания извлекается содержимое стека и выполняется выборка адреса возврата. Таким образом, через 12 циклов возобновляется выполнение фоновой программы. Если же во время выхода из активного прерывания возникает еще одно низкоприоритетное прерывание, то операция извлечения из стека прекращается и указатель стека вернется к своему исходному значению, а выполнение обработки нового прерывания начнется через 6 дополнительных циклов. В итоге, задержка вызова процедуры обработки нового прерывания будет в пределах 7-18 циклов.



Переход к обработке низкоприоритетного прерывания, которое возникает при выходе из текущего прерывания, выполняется с задержкой 7-18 циклов

2.4.5.2.3. Обработка опоздавшего высокоприоритетного прерывания

В системах реального времени часто возникают ситуации, когда во время перехода к обработке низкоприоритетного прерывания возникает еще одно прерывание с более высоким приоритетом. Если такая ситуация возникнет еще на фазе загрузки стека, то по его завершении NVIC инициирует переход к обработке высокоприоритетного прерывания. Загрузка стека будет продолжаться еще минимум 6 циклов с момента возникновения высокоприоритетного прерывания, после чего будет выполнена выборка нового адреса процедуры обработки прерывания.



Высокоприоритетное прерывание будет обработано первым, даже если оно возникнет уже на фазе перехода к обработке низкоприоритетного прерывания, при этом, дополнительные операции над стеком будут исключены

Отложенное низкоприоритетное прерывание будет обработано сразу по завершении обработки высокоприоритетного прерывания с задержкой 6 циклов.

2.4.5.3. Конфигурация и использование NVIC

Чтобы включить в работу NVIC необходимо выполнить три действия. Вначале сконфигурировать таблицу векторов используемых прерываний. Затем настроить регистры NVIC с целью активизации и установки уровней приоритета прерываний NVIC. И, наконец, настроить УВВ и разрешить поддержку ими прерываний.

2.4.5.3.1. Таблица векторов исключительных ситуаций

Таблица векторов Cortex начинается с нижней части адресного пространства. Однако таблица векторов начинается не с нулевого адреса, а с адреса 0x00000004, т.к. первые четыре байта используются для хранения начального адреса указателя стека.

Таблица векторов исключительных ситуаций содержит адреса, которые загружаются в счетчик программы, когда ЦПУ переходит в исключительную ситуацию

Номер	Тип исключительной ситуации	Приоритет	Тип приоритета	Описание
1	Reset	-3 (высший)	фиксированнный	Сброс
2	NMI	-2	фиксированнный	Немаскируемое прерывание
3	Hard Fault	-1	фиксированнный	Обработчик аварийный состояний по умолчанию, если другой не реализован

4	MemManageFault	0	устанавливаемый	Сбой в блоке защите памяти или доступ по несуществующему адресу
5	BusFault	1	устанавливаемый	Ошибки в интерфейсе AHB
6	UsageFault	2	устанавливаемый	Исключительные ситуации, вызванные программными ошибками
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	устанавливаемый	Вызов системных служб
12	DebugMonitor	4	устанавливаемый	Точки прерывания, контрольные точки, внешняя отладка
13	Reserved	N.A.	N.A.	
14	PendSV	5	устанавливаемый	Отправляемый запрос системному устройству
15	SYSTICK	6	устанавливаемый	Срабатывание системного таймера
16	Прерывание 0	7	устанавливаемый	Внешнее прерывание 0
.....	.....	.....	устанавливаемый	.....
256	Прерывание 240	247	устанавливаемый	Внешнее прерывание 240

Каждый из векторов прерываний занимает 4 байта и указывает на начальный адрес каждой конкретной процедуры обработки прерывания. Первые 15 векторов - адреса обработки исключительных ситуаций, возникающих в ядре Cortex. К ним относятся вектор сброса, немаскируемое прерывание, управление авариями и ошибками, исключительные ситуации отладочной системы и прерывание таймера SysTick. Набором инструкций Thumb-2 также поддерживается инструкция, выполнение которой приводит к генерации исключительной ситуации. Начиная с 16 вектора, следуют адреса обработки прерываний пользовательских УВВ. Их назначение зависит от каждого конкретного производителя. В программе таблица векторов обычно приводится в отдельном файле и содержит адреса процедур обработки прерываний:

```
        AREA RESET, DATA, READONLY
         EXPORT __Vectors
__Vectors     DCD __initial_sp        ; Верхняя граница стека
         DCD Reset_Handler       ; Обработчик сброса
         DCD NMI_Handler         ; Обработчик немаскируемого прерывания
```

```
        DCD HardFault_Handler    ; Обработчик аварий типа HardFault
        DCD MemManage_Handler    ; Обработчик аварий блока защиты памяти
        DCD BusFault_Handler     ; Обработчик аварий типа BusFault
        DCD UsageFault_Handler   ; Обработчик аварий типа UsageFault
        DCD 0                    ; Резерв
        DCD 0                    ; Резерв
        DCD 0                    ; Резерв
        DCD 0                    ; Резерв
        DCD SVC_Handler          ; Обработчик программно-сгенерированного
прерывания
        DCD DebugMon_Handler     ; Обработчик прерывания встроенной отладочной
системы
        DCD 0                    ; Резерв
        DCD PendSV_Handler       ; Обработчик PendSV
        DCD SysTick_Handler      ; Обработчик прерывания таймера SysTick
```

Например, если используется прерывание таймера SysTick, то объявление на Си процедуры обработки прерывания выполняется следующим образом:

```
void SysTick_Handler (void) {  }
```

Теперь, когда сконфигурирована таблица векторов и объявлена процедура обработки прерываний, мы можем настроить NVIC на обработку прерывания таймера SysTick. Обычно, для этого выполняют две операции: задается приоритет прерывания, а затем разрешается источник прерывания. Регистры NVIC расположены в области системных ресурсов.

Регистры NVIC находятся в области системных ресурсов Cortex-M3 и доступ к ним возможен при работе ЦПУ только в привилегированном режиме

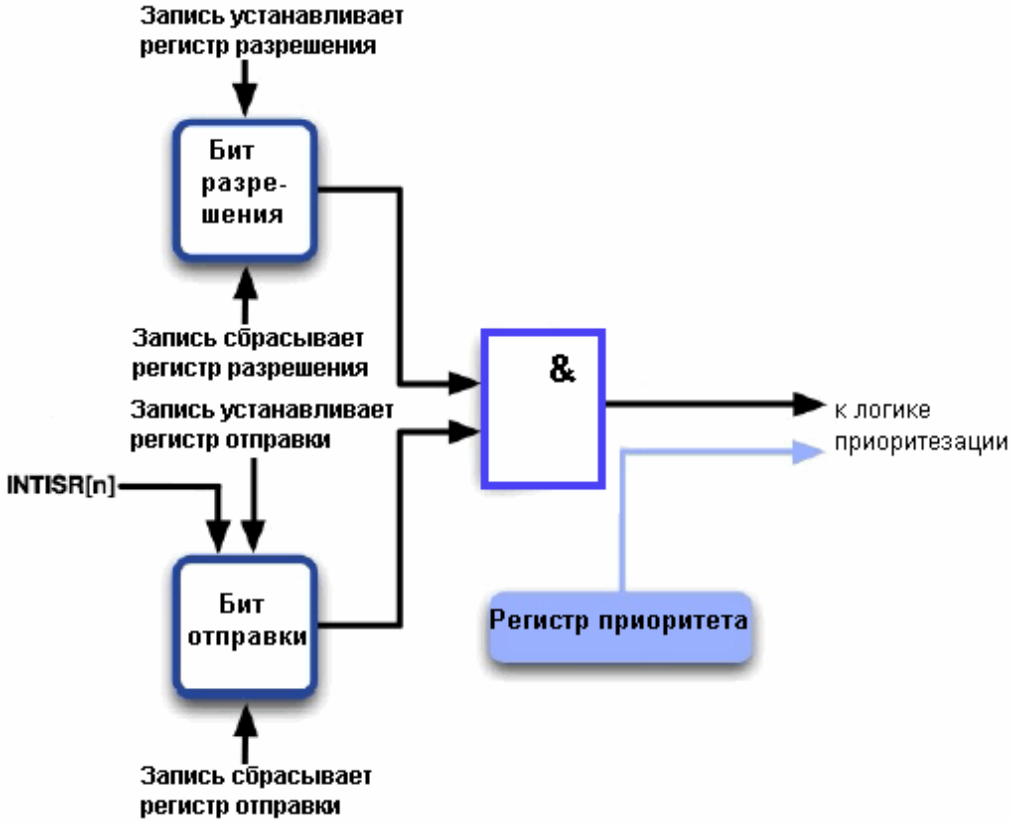
Настройка внутренних исключительных ситуаций процессора Cortex выполняется с помощью регистров системного управления и системных приоритетов, а пользовательских УВВ - с помощью регистров IRQ. Прерывание SysTick является

внутренней исключительной ситуацией процессора Cortex и, поэтому, управляется через системные регистры. Некоторые внутренние исключительные ситуации постоянно разрешены. К ним относятся прерывание по сбросу, немаскированное прерывание, а также и прерывание таймера tSysTick, поэтому, никаких действий с NVIC по разрешению этого прерывания делать не нужно. Для настройки прерывания SysTick нам необходимо активизировать сам таймер и его прерывание с помощью соответствующего регистра управления:

```
SysTickCurrent = 0x9000;                //Начальное значение счетчика SysTick
SysTickReload = 0x9000;                 //Перезагружаемое значение
SysTickControl = 0x07;                  //Запуск счета и разрешение прерывания
```

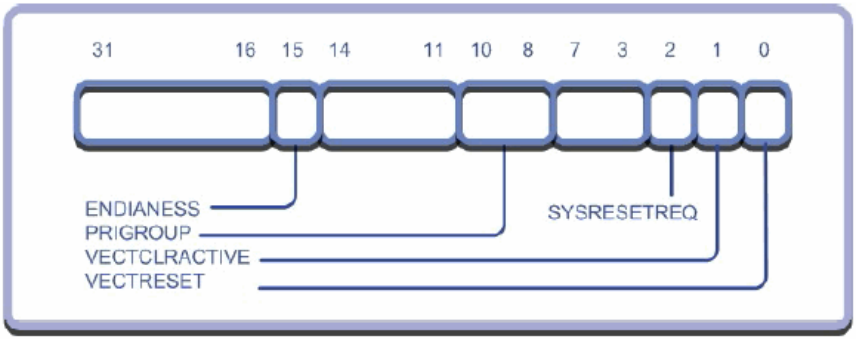
Приоритет каждой внутренней исключительной ситуации Cortex можно задать в системных регистрах приоритета. У исключительных ситуаций Reset, NMI и hard fault он фиксированный. Этим гарантируется, что ядро всегда будет переходить к обработке известной исключительной ситуации. У всех остальных исключительных ситуаций имеется восьмибитное поле, которое расположено в трех системных регистрах приоритета. МК STM32 используют только 16 уровней приоритета, поэтому, у них активно только 4 бита этого поля. Однако важно запомнить, что приоритет устанавливается четырьмя старшими битами.

Каждое пользовательское УВВ управляется через блоки регистров IRQ. У каждого такого УВВ имеется бит разрешения прерывания. Все эти биты находятся в пределах двух 32-битных регистров установки разрешения прерываний. Для отключения источника прерывания предусмотрены отдельные регистры отмены разрешения прерываний. У NVIC также имеются регистры отправленных и активных прерываний, которые позволяют отследить состояние источника прерывания.



У каждого источника прерывания имеется бит разрешения, как в NVIC, так и в УВВ. У МК STM32 используется 16 уровней приоритетов

Всего предусмотрено 16 регистров приоритета. Каждый из них разделен на четыре 8-битных поля для задания приоритета. Каждое поле связано с конкретным вектором прерывания. У МК STM32 используется только половина такого поля, т.к. реализовано только 16 уровней приоритета. Однако необходимо помнить, что активные биты приоритета находятся в старшей тетраде поля. По умолчанию поле приоритета определяет 16 уровней приоритета, причем уровень 0 - наивысший приоритет, а 15 - наинизший. Поле приоритета также можно представить в виде групп и подгрупп приоритета. Это не добавляет дополнительных уровней приоритета, просто облегчает управление ими при необходимости задания в поле PRIGROUP регистра прикладных прерываний и управления сбросом большого числа прерываний.



Поле PRIGROUP разделяет уровни приоритетов на группы и подгруппы. Это необходимо для повышения программной абстракции при работе с большим числом прерываний

PRIGROUP (3 бита)	Положение запятой в двоичном числе (группа.подгруппа)		Группа приоритета		Подгруппа приоритета	
			Кол-во бит	Кол-во уровней	Кол-во бит	Кол-во уровней
011	4.0	гггг	4	16	0	0
100	3.1	гггп	3	8	1	2
101	2.2	ггпп	2	4	2	4
110	1.3	гппп	1	2	3	8
111	0.4	пппп	0	0	4	16

Трехбитное поле PRIGROUP управляет разделением 4-битных полей приоритета на группы и подгруппы. Например, запись в PRIGROUP числа 3 приведет к созданию двух групп с 4 уровнями приоритетов в каждой. После этого, вы можете в программе выполнить определения высокоприоритетной и низкоприоритетной групп прерываний. В рамках каждой группы можно задавать подуровни, в т.ч. низкий, средний, высокий и очень высокий. Ранее уже говорилось, что это позволяет более абстрактно смотреть на структуру прерываний и помогает программисту управлять большим числом прерываний. Конфигурация прерываний УВВ очень похожа на конфигурацию внутренних исключительных ситуаций процессора Cortex. Если взять в качестве примера прерывание АЦП, то вначале необходимо установить вектор прерывания и создать процедуру обработки прерываний:

```
DCD          ADC_IRQHandler ;
void ADC_Handler void { }
```

Затем необходимо инициализировать АЦП и разрешить прерывание в регистрах УВВ и NVIC:

```
ADC1->CR2    = ADC_CR2;//Включение АЦП в режиме непрерывных преобразований
ADC1->SQR1   = sequence1; //Выбор номеров каналов в очереди преобразования
ADC1->SQR2   = sequence2; //и выбор каналов для преобразования
ADC1->SQR3   = sequence3;
ADC1->CR2   |= ADC_CR2;   //Перезапись бита включения
ADC1->CR1    = ADC_CR1;   //Запуск группы каналов, разрешение прерывания АЦП
GPIOB->CRH   = 0x33333333;//Настройка светодиодных выводов на выход
NVIC->Enable[0] = 0x00040000; //Разрешение прерывания АЦП
NVIC->Enable[1] = 0x00000000;
```

2.5. Режимы работы, влияющие на энергопотребление

Возможности управления энергопотреблением МК STM32 будут рассмотрены позже. Здесь же, мы остановимся на режимах работы ядра Cortex, влияющих на энергопотребление. ЦПУ Cortex поддерживает режим SLEEP, который переводит ядро Cortex в экономичный режим работы и приостанавливает выполнение инструкций. В этом состоянии частично продолжает работу NVIC, что позволяет возобновить работу ядра Cortex при генерации прерываний встроенными в МК STM32 УВВ.

2.5.1. Переход в экономичный режим работы

Для перевода ядра Cortex в режим SLEEP необходимо выполнить инструкцию WFI (ожидание прерывания) или WFE (ожидание события). После выполнения инструкции WFI, ядро Cortex приостановит выполнение программы и обработку отправленных прерываний. Существует два сценария завершения выполнения процедуры обработки прерывания. По первому сценарию ЦПУ выходит из процедуры обработки прерывания и возвращается к дальнейшему выполнению фоновой программы. Однако, если установить бит SLEEPON EXIT в регистре системного управления, ядро Cortex после выхода из процедуры обработки прерываний автоматически перейдет в режим SLEEP. Такая возможность позволяет создавать управляемые прерываниями маломощные применения, в которых микроконтроллер после возобновления работы исполняет определенный код программы, а затем снова, не выполняя каких-либо инструкций для управления энергопотреблением, переходит в режим SLEEP.

Прерывание WFE позволяет возобновить работу ядра Cortex с того же места, с которого оно было переведено в режим SLEEP. Это прерывание не приводит к переходу к процедуре обработки прерывания. Возобновляющее работу событие - это обычная линия прерывания УВВ, но которая на активизирована как прерывание в NVIC. Благодаря этому, УВВ может сигнализировать ядру Cortex о необходимости возобновления работы и продолжения обработки, не прибегая к использованию процедуры обработки прерывания. Инструкции WFI и WFE не поддерживаются языком Си, однако компиляторы для набора инструкций Thumb-2 поддерживают встроенные макросы, которые можно использовать в программе, как стандартные Си-команды:

```
__WFI
__WFE
```

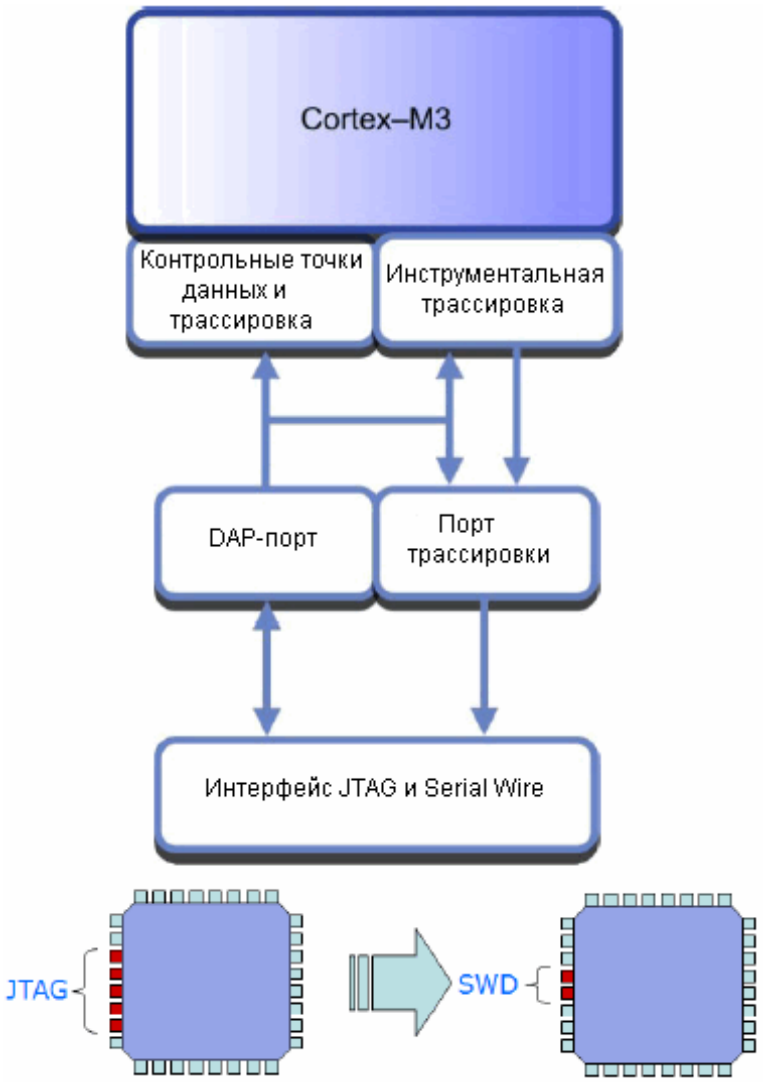
Помимо экономичных режимов работы SLEEPNOW и SLEEPONEXIT, ядро Cortex может генерировать сигнал SLEEPDEEP для остальной части микроконтроллерной системы.

Регистр системного управления конфигурирует режимы SLEEP процессора Cortex. STM32 поддерживают дополнительные экономичные режимы, которые используют генерируемый процессором Cortex сигнал DeepSleep

Благодаря этому появляется возможность реализации таких дополнительных функций, как приостановка ФАПЧ и УВВ, с помощью которых МК STM32 может переходить в режимы работы с наименьшим энергопотреблением.

2.5.2. Отладочная система CoreSight

У всех ЦПУ ARM имеется собственная встроенная отладочная система. ЦПУ ARM7 и ARM9, как минимум, имеют порт JTAG, который является стандартным интерфейсом для подключения к ЦПУ, а также загрузки кода программы во внутреннее ОЗУ или Flash память. Порт JTAG также поддерживает базовые функции управления исполнением программы (пошаговое выполнение, установка контрольных точек и др.) и делает возможным просмотр содержимого ячеек памяти. С помощью специального блока, который называется встроенной трассировочной макроячейкой (ETM), ЦПУ ARM7 и ARM9 также предоставляют возможности трассировки в масштабе реального времени. Несмотря на то, что данная отладочная система работает отлично, она имеет некоторые ограничения. Когда ЦПУ ARM остановлено, связанная с портом JTAG отладочная система, способна предоставлять только отладочную информацию и не имеет возможностей обновления в реальном времени. Кроме того, количество аппаратных контрольных точек ограничено двумя. Из-за этого в наборы инструкций ARM7 и ARM9 включены инструкции контрольных точек, которые отладочное средство может вставлять в код программы (обычно они называются программными контрольными точками). Наконец, чтобы появилась поддержка реально-временной трассировки, производитель должен в ущерб стоимости микроконтроллера интегрировать в него ETM. В итоге, в микроконтроллерах такой поддержки чаще всего не оказывается. В новом ядре Cortex применена абсолютно иная отладочная система, которая получила название CoreSight.



Отладочная система CoreSight использует интерфейс JTAG или Serial Wire. Она отвечает за выполнение функций управления исполнением программы и трассировки. Ее дополнительное преимущество заключается в возможности оставаться в работе, даже когда МК STM32 находится в экономичном режиме. Это большой шаг вперед относительно стандартной JTAG отладки

Отладочная система CoreSight имеет DAP-порт, который отвечает за подключение к микроконтроллеру посредством JTAG инструментальных средств. Отладочные инструментальные средства могут использовать для подключения стандартный 5-выводной интерфейс JTAG или 2-проводной интерфейс Serial Wire. Помимо возможностей JTAG отладки, отладочная система CoreSight содержит блоки трассировки Data Watch и ETM. Для тестирования программы предусмотрены блоки инструментальной трассировки и корректировки Flash памяти (FLASH patch). У МК STM32 используется сокращенная версия отладочной системы CoreSight, которая отличается отсутствием в ее структуре блока ETM. По сути, используемая в МК STM32 структура отладочной системы CoreSight, является версией стандартной JTAG отладочной системы, но с улучшенными реально-временными характеристиками. Отладочная система CoreSight

микроконтроллеров STM32 поддерживает 8 аппаратных контрольных точек, которые можно задавать и отменять во время исполнения ЦПУ Cortex кода программы, не оказывая на это никакого влияния. Отладочная система CoreSight может оставаться активной, даже после перехода ядра Cortex в экономичный режим работы. Такая возможность несете в себе множество преимуществ для отладки маломощных устройств. Кроме того, таймеры STM32 могут останавливаться одновременно с выполненной системой CoreSight остановкой ЦПУ. Благодаря этому, появляется возможность пошагового выполнения программы и поддержания таймеров в синхронизме с выполнением инструкций ЦПУ Cortex. Отладочная система CoreSight микроконтроллеров STM32 позволяет существенно улучшить возможности отладки в масштабе реального времени по сравнению с предшествующими ЦПУ ARM7 и ARM9 и, при этом, предоставляет возможность использовать столь же недорогую аппаратную часть.

3. Схема включения

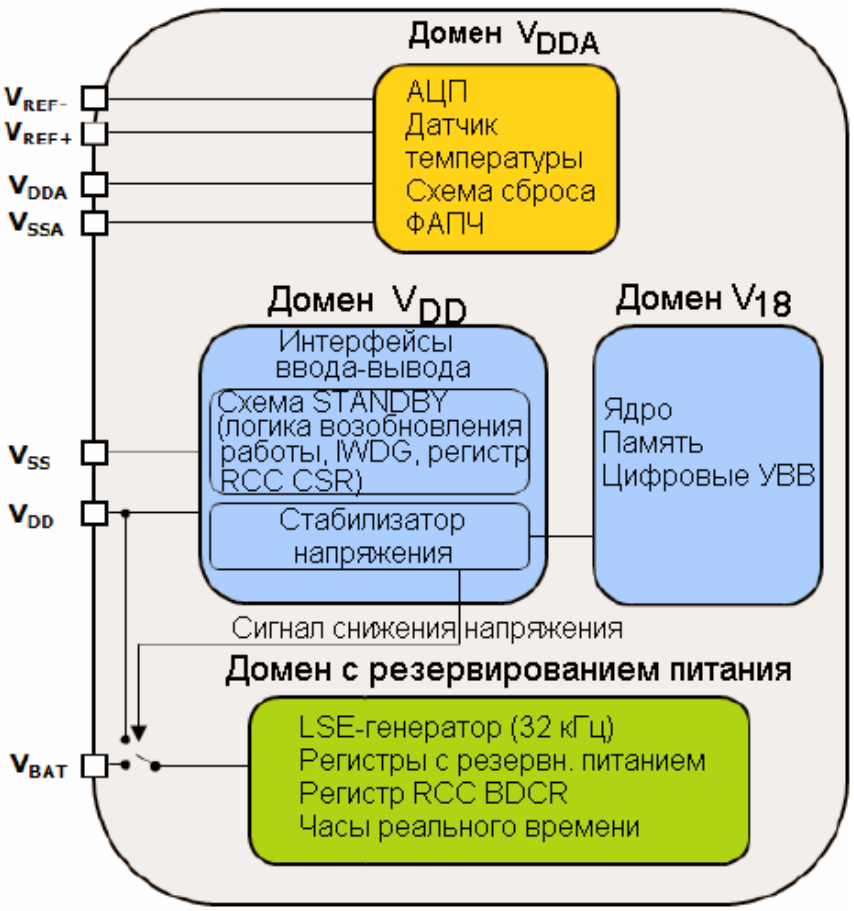
Простейшая схема включения МК STM32 требует от разработчика минимум действий. Микроконтроллеры STM32 имеют собственные RC-генераторы и схему сброса, поэтому, для включения их в работу достаточно подать напряжение питания. В данном разделе будут рассмотрены основные аппаратные узлы МК, от которых зависит их схема включения.

3.1. Типы корпусов

Микроконтроллеры из серии Access line и соответствующим им версии из серии Performance line доступны в корпусах одного типа. Благодаря этому, можно быстро выполнить обновление схемы без необходимости повторного проектирования печатной платы. Все микроконтроллеры STM32 доступны в корпусах LQFP с числом выводов от 48 до 144.

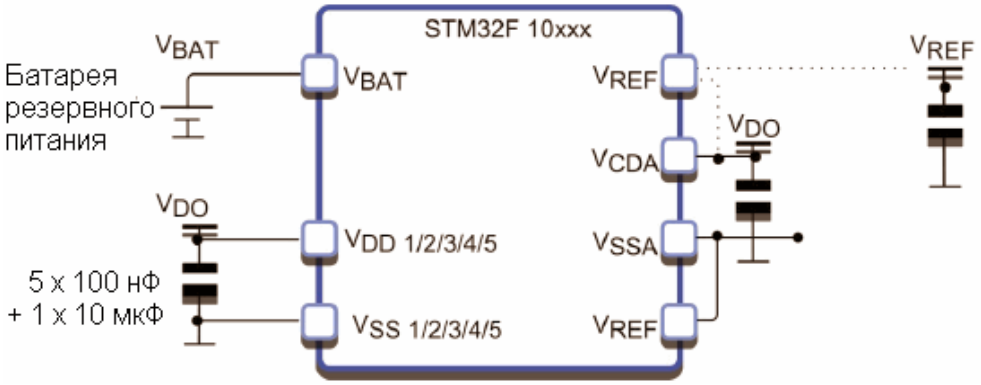
3.2. Напряжение питания

Для работы микроконтроллеров STM32 их необходимо питать одним напряжением в диапазоне от 2.0..3.6В. Для питания ядра Cortex напряжением 1.8В в МК интегрирован стабилизатор напряжения. Однако к STM32 может быть подано еще два опциональных напряжения питания. В домене с отдельным питанием расположены часы реального времени и небольшое число регистров, что позволяет организовать их резервное питание и обеспечить работоспособность даже при нахождении МК в режиме полного отключения (deep power down). Если же в схеме данная функция не нужна, то вывод VBAT необходимо соединить с VDD.



МК STM32 работает от одного источника питания 2.0..3.6В. При необходимости, отдельным резервным питанием может быть запитан специальный домен и, кроме того, отдельное питание предусмотрено для АЦП (только у МК в 144-выводных корпусах)

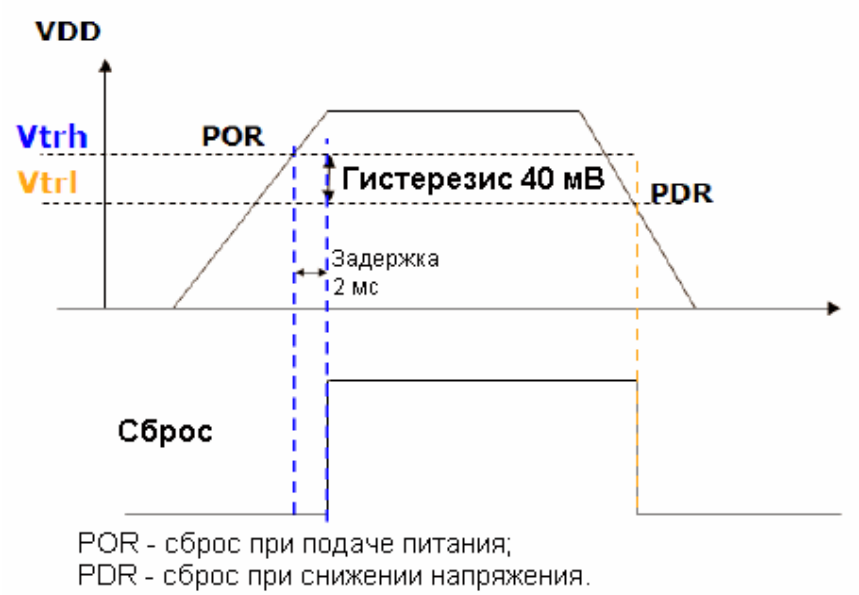
Второе опциональное напряжение питания предназначено для АЦП. Если АЦП задействован, то диапазон основного напряжения питания VDD сужается до 2.4..3.6В. У МК в 100-выводных корпусах имеются дополнительные выводы источника опорного напряжения (ИОН) АЦП, VREF+ и VREF-. Вывод VREF- должен быть соединен с VSSA, а напряжение на VREF+ может варьироваться от 2.4В до VDDA. У МК во всех остальных корпусах ИОН соединен внутренне с выводами питания АЦП. На каждом из входов питания необходимо предусмотреть стабилизационные конденсаторы, как показано ниже.



Благодаря интегрированию схемы сброса и стабилизатора напряжения, МК необходимо дополнить только семью внешними конденсаторами

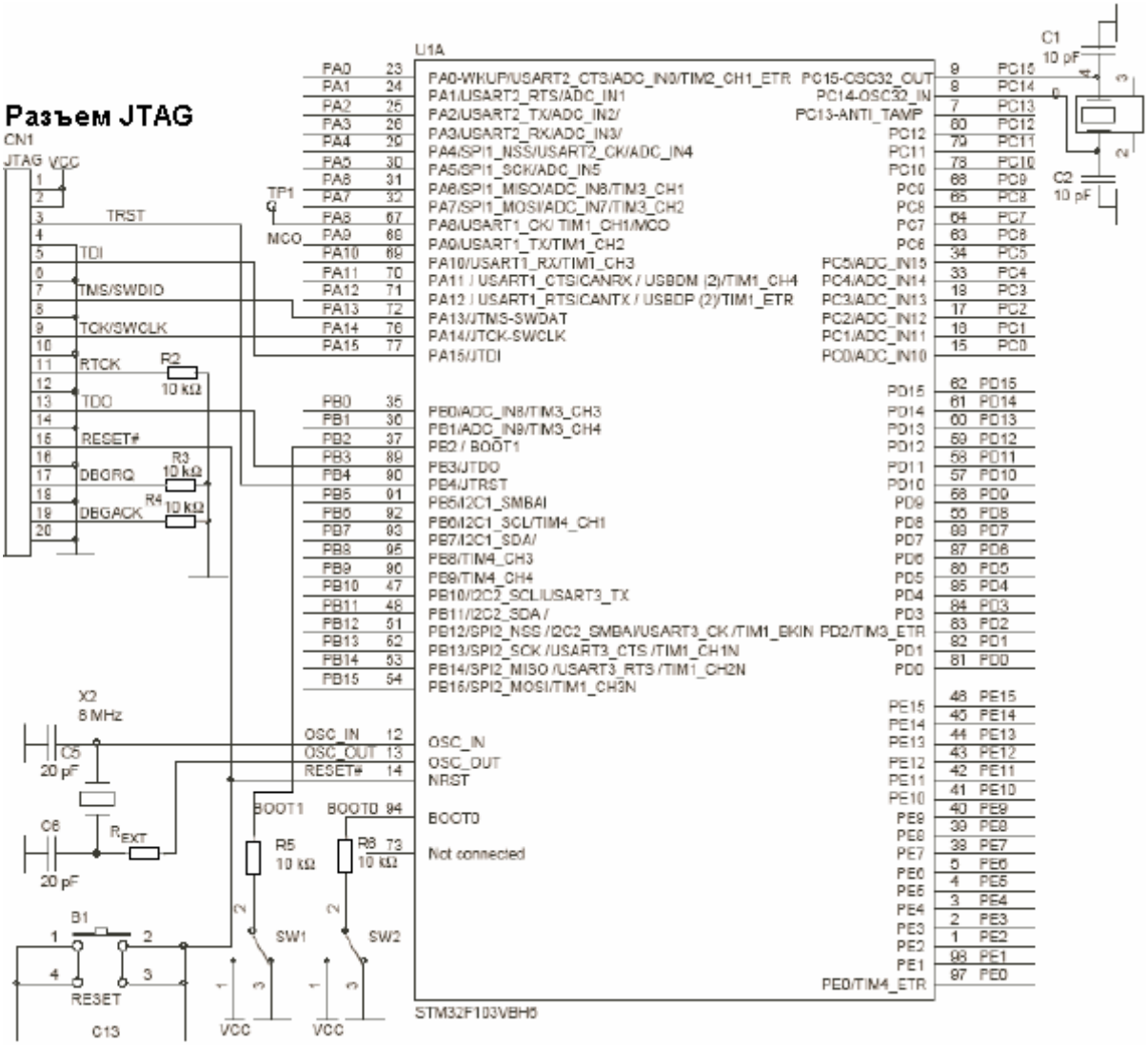
3.3. Схема сброса

В МК STM32 входит схема сброса, которая удерживает его в сброшенном состоянии до тех пор, пока VDD будет ниже 2.0В (гистерезис 40 мВ).



Встроенные схемы сброса при подаче (POR) и снижении (PDR) напряжения питания гарантируют возможность работы МК только при подаче стабильного напряжения питания. Внешняя схема сброса не требуется

3.3.1. Основная схема включения



Несмотря на то, что внешняя схема сброса не нужна в схеме включения STM32, на фазе проектирования может оказаться удобным подключение вывода nRST к обычной кнопке сброса. Вывод nRST также подключается к отладочному порту JTAG. Это позволяет отладочному средству управлять сбросом микроконтроллера. У МК STM32 имеется несколько внутренних источников сброса, которые активизируются в случае обнаружения аварийных режимов работы (будут рассматриваться позже в разделе, посвященном безопасности работы МК).

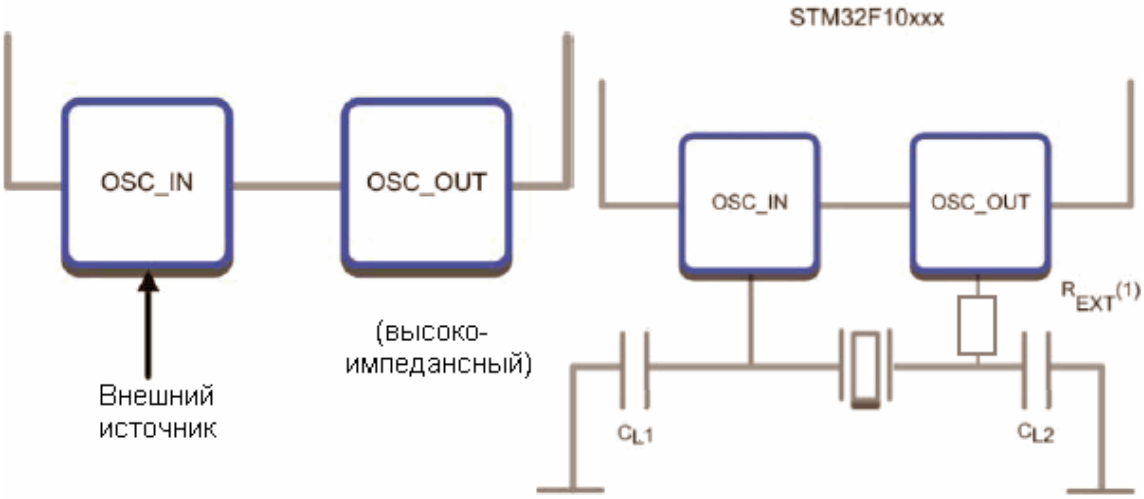
3.4. Генераторы

У МК STM32 имеются внутренние RC-генераторы, способные синхронизировать встроенную схему ФАПЧ. Благодаря их совместной работе, МК могут синхронизироваться частотой до 72 МГц. Внутренние генераторы, по сравнению с кварцевыми, не отличаются такой же высокой точностью или стабильностью, поэтому, во многих применениях может

потребоваться использование как минимум одного внешнего кварцевого генератора.

3.4.1. Внешний высокочастотный генератор

Основной внешний источник синхронизации используется для тактирования процессора и УВВ STM32. Он называется внешним высокочастотным генератором (HSE-генератор). Совместно с генератором может использоваться кварцевый/керамический резонатор или отдельный источник синхронизации. Сигнал внешнего источника синхронизации может иметь прямоугольную, синусоидальную или треугольную форму, но, при этом, заполнение импульсов должно быть 50%-ым, а частота не более 25МГц.



Внешний генератор может работать совместно с кварцевым резонатором или внешним источником синхронизации

Если же используется внешний кварцевый/керамический резонатор, то его частота должна лежать в пределах 4..16 МГц. Чтобы добиться работы МК на его максимальной частоте 72 МГц, необходимо выбрать такую частоту внешней синхронизации, которая бы нацело делила максимальную рабочую частоту. Это связано с тем, что внутренняя схема ФАПЧ умножает частоту HSE-генератора на целое число.

3.4.2. Внешний низкочастотный генератор

МК STM32 могут иметь еще один внешний генератор, который называется внешним низкочастотным генератором (LSE-генератор). Он предназначен для синхронизации часов реального времени и оконного сторожевого таймера. Также как и HSE-, LSE-генератор может работать совместно с кварцевым резонатором или внешним сигналом синхронизации снова-таки прямоугольной, синусоидальной или треугольной формы, и с заполнением импульсов 50%. В каждом из этих случаев частота LSE-генератора должна быть равна 32,768 кГц, что необходимо для точной работы часов реального времени. Часы реального времени также могут синхронизироваться внутренним низкочастотным генератором, однако ввиду его недостаточной точности, обычно для реализации функций ЧРВ используется LSE-генератор.

3.4.3. Выход синхронизации

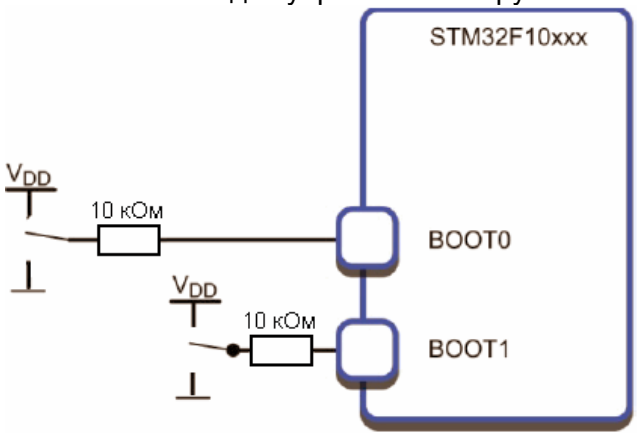
Одна из линий ввода-вывода может быть настроена, как выход синхронизации (MCO). В этом режиме, вывод MCO может генерировать один из четырех внутренних источников синхронизации. Об этом более детально пойдет речь при рассмотрении настроек внутренней системы синхронизации.

3.4.4. Выводы управления загрузкой и внутрисистемное программирование

Микроконтроллер может начать свою работу в одном из трех различных режимов загрузки. Эти режимы выбираются с помощью выводов BOOT0 и BOOT1. От выбранного режима загрузки зависит, какую область карты памяти микроконтроллер будет считать началом памяти. МК может исполнять код программы из Flash памяти, внутреннего статического ОЗУ или системной памяти. Если выбирается загрузка из системной памяти, то STM32 начнет свою работу с выполнения запрограммированной производителем загрузочной программы, которая позволяет пользователю перепрограммировать Flash память внутрисистемно.

3.4.5. Режимы загрузки

Для работы в обычном режиме вывод BOOT0 необходимо соединить с GND. Если же планируется использование других режимов, необходимо предусмотреть джамперы для задания различных состояний на выводах управления загрузкой.



Выводы управления загрузкой позволяют указать, какая область памяти будет использоваться как первые 2 кбайт памяти. В их качестве могут выступать Flash память, встроенная программа загрузчика или первые 2 кбайт статического ОЗУ.

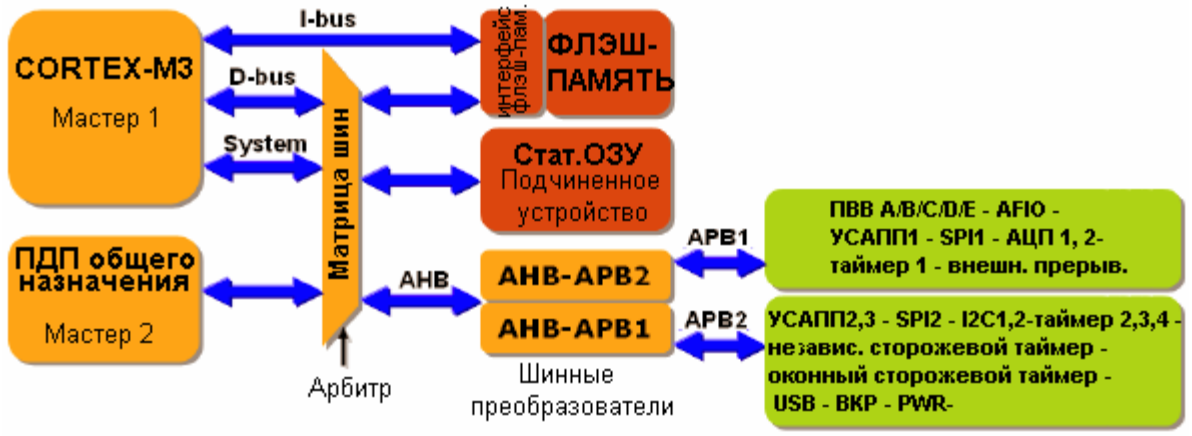
Обычно потребность в этом возникает при обновлении ПО уже на фазе эксплуатации продукции. Программа загрузчика для получения кода программы от ПК по умолчанию использует последовательный интерфейс USART1, поэтому, если планируется ее использование, то в схеме необходимо предусмотрен ИС приемо-передатчика RS232.

3.4.6. Отладочный порт

Завершающим звеном схемы включения является отладочный порт. Он необходим для подключения отладчика к МК STM32. Отладочная система Cortex CoreSight поддерживает два типа подключений: 5-выводной порт JTAG и 2-выводной последовательный порт Cortex. Оба этих порта задействуют для связи с отладчиком линии ввода-вывода общего назначения. После сброса ЦПУ Cortex назначает этим линиям их альтернативные функции, что дает возможность использовать отладочный порт. При необходимости использования линий отладочного интерфейса, как обычных линий ввода-вывода, необходимо соответствующим образом запрограммировать регистры альтернативных функций. 5-проводной интерфейс JTAG выводится на 20-выводной разъем IDC со стандартным для всех JTAG-совместимых инструментальных средств расположением выводов. Последовательный интерфейс использует порт A_13 для последовательной передачи данных и порт A_14 для синхронизации.

4. Архитектура системы микроконтроллеров STM32

МК STM32 выполнены на основе ядра Cortex, которое подключено к Flash памяти по отдельной шине инструкций. Шина данных и системная шина Cortex подключены к матрице высокоскоростных шин AHB. Внутреннее статическое ОЗУ подключено напрямую к матрице шин AHB, с которой также связан блок ПДП. Подключение встроенных УВВ распределено между двумя шинами APB. Каждая из шин связана с матрицей шин AHB посредством шинных преобразователей. Матрица шин AHB синхронизируется той же частотой, что и ядро Cortex. Однако, у шин AHB имеются отдельные предделители и, поэтому, в целях снижения энергопотребления их можно синхронизировать более низкими частотами. Важно обратить внимание, что шина APB2 может работать с максимальным бытродействием 72 МГц, а быстродействие шины APB1 ограничено частотой 36 МГц. В качестве шинных мастеров могут выступать, как ЦПУ Cortex, так и блок ПДП. Благодаря свойственной матрице шин параллелизму, необходимость в арбитраже возникает только в случае попыток одновременного доступа обеих мастеров к статическому ОЗУ, шине APB1 или APB2. Тем не менее, как мы убедимся при изучении раздела посвященному ПДП, шинный арбитр гарантированно предоставляет 2/3 времени доступа для блока ПДП и 1/3 для ЦПУ Cortex.

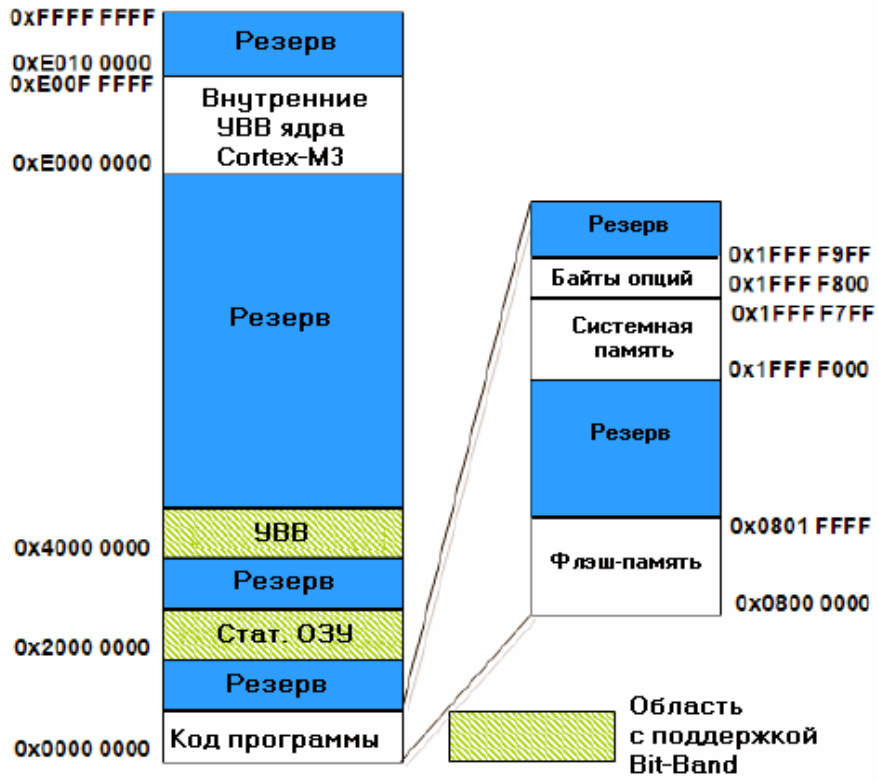


В структуре внутренних шин предусмотрены отдельная шина инструкций и матрица шин, которая

предоставляет несколько каналов передачи данных для ЦПУ Cortex и блока ПДП

4.1 Распределение памяти

Несмотря на то, что у МК STM32 имеется множество внутренних шин, адресное пространство для программиста предлагается как линейное размером 4 Гбайт. Поскольку МК STM32 выполнены на основе Cortex, то у них используется стандартное распределение памяти. Память программ начинается с адреса 0x00000000. Встроенное статическое ОЗУ стартует с адреса 0x20000000. Все ячейки статического ОЗУ расположены в области хранения бит. Регистры УВВ представлены в карте памяти, начиная с адреса 0x40000000, и также расположены в области хранения бит УВВ. Наконец, регистры Cortex находятся в их стандартном месте, начиная с адреса 0xE0000000.



Выводы выбора режима загрузки		Режим загрузки
BOOT1	BOOT0	
x	0	Флэш-память пользователя
0	1	Системная память
1	1	Встроенное статич.ОЗУ

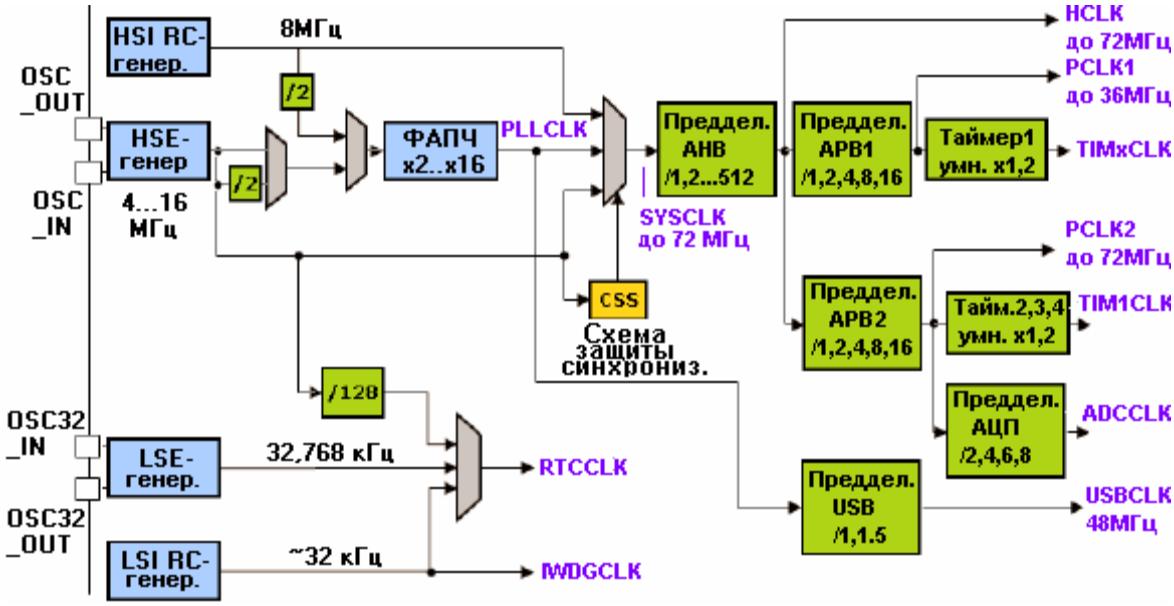
Карта памяти STM32 выполнена по стандарту Cortex. Первые 2 кбайт памяти могут быть связаны с Flash памятью, системной памятью или статическим ОЗУ, в зависимости от состояния выводов управления загрузкой

Область Flash памяти разделена на три секции. Первая - Flash память пользователя - начинается с адреса 0x0000000. Далее следует системная память, которая также называется большим информационным блоком. Она представляет собой Flash память размером 4 кбайт, которая запрограммирована производителем кодом программы загрузчика. Последняя секция, которая стартует с адреса 0x1FFFF800, называется малым информационным блоком. В ней находится группа опциональных байт, с помощью которых можно повлиять на некоторые системные настройки микроконтроллера STM32. Программа загрузчика позволяет посредством интерфейса USART1 загрузить код программы и запрограммировать его во Flash память пользователя. Чтобы перевести МК STM32 в режим загрузчика, нужно на внешних выводах BOOT0 и BOOT1 установить низкий и высокий уровни, соответственно. Если установить именно такие состояния на выводах управления загрузкой, то блок системной памяти начнется с адреса 0x00000000. После сброса, МК STM32, вместо выполнения прикладного кода из Flash памяти пользователя, начнет выполнение программы загрузчика. Чтобы пользователь имел возможность стирать и перепрограммировать Flash память на компьютере необходимо запустить еще одну программу загрузчика, которую можно скачать с сайта компании ST. Программа для ПК также доступна в виде DLL-файла, что позволяет создавать собственное ПО для программирования микроконтроллеров на фазах производства или эксплуатации продукции.

С помощью выводов управления загрузкой адрес 0x00000000 вместо Flash памяти пользователя может быть также связан со статическим ОЗУ. Поскольку загрузка статического ОЗУ осуществляется более быстро, то эта возможность может оказаться полезной на фазе проектирования для исполнения кода программы из статического ОЗУ. Кроме того, появляется возможность сократить частоту перепрограммирования Flash памяти.

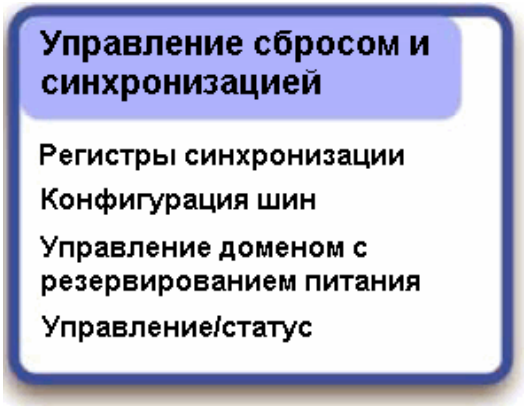
4.2. Работа с максимальным быстродействием

Помимо двух внешних генераторов, у STM32 имеется два внутренних RC-генератора. Сразу после сброса ядро Cortex синхронизируется внутренним высокочастотным генератором, номинальная рабочая частота которого составляет 8 МГц. Второй внутренний генератор - низкочастотный генератор на частоту 32,768 кГц. Данный генератор предназначен для совместной работы с часами реального времени и сторожевым таймером.



У МК STM32 используется достаточно сложная система синхронизации с двумя внешними и двумя внутренними генераторами, а также схемой ФАПЧ. В целях безопасности системы предусмотрена возможность мониторинга внешнего высокочастотного генератора

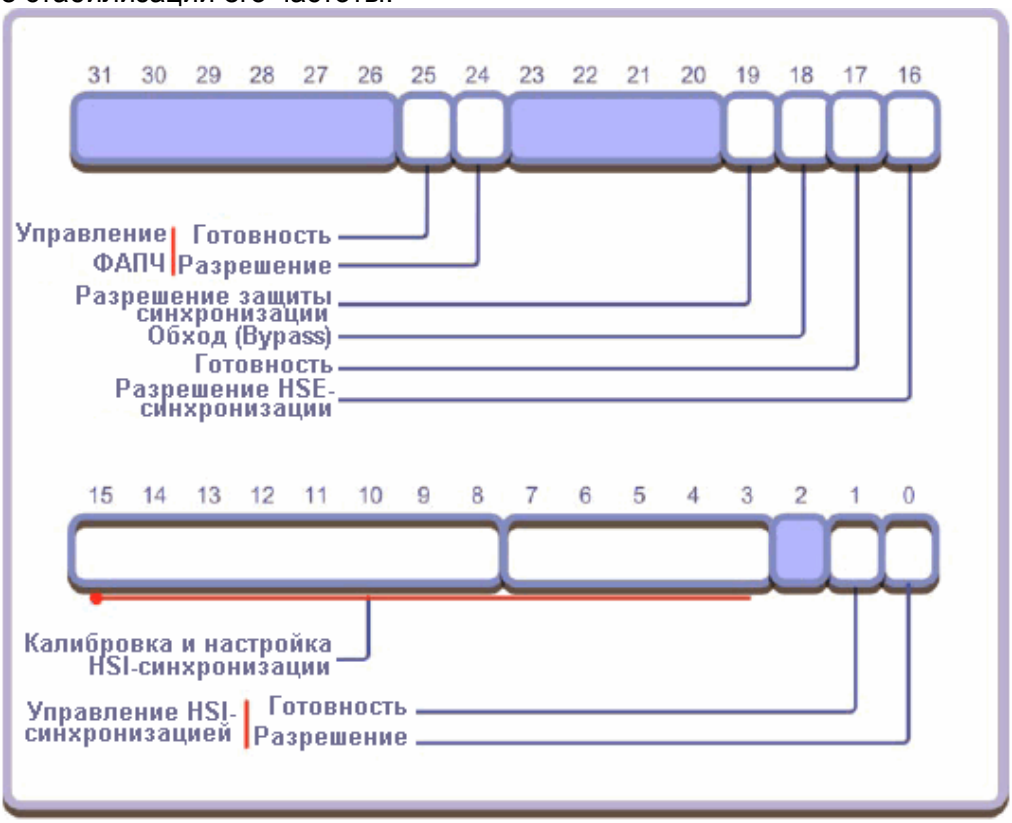
Процессор Cortex поддерживает возможность синхронизации внешним или внутренним высокочастотным генератором или от внутренней схемы ФАПЧ. Источником синхронизации схемы ФАПЧ может служить внутренний или внешний высокочастотный генератор. Таким образом, МК STM32 могут работать на частоте 72 МГц без использования внешнего генератора. Недостатком использования внутреннего генератора на частоту 8МГц является его невысокая точность и стабильность. Его нельзя использовать для синхронизации последовательных интерфейсов или для выполнения точных измерений временных интервалов. Независимо от выбранного генератора, чтобы добиться работы ядра Cortex на максимальной частоте 72 МГц, необходимо использовать ФАПЧ. Все регистры управления настройками генераторов, ФАПЧ и шин находятся в группе управления сбросом и синхронизацией (группа RCC).



Блок управления сбросом и синхронизацией воздействует на систему синхронизации, шинные преобразователи и домен с резервированием питания

4.2.1. Блок фазовой автоподстройки частоты (PLL)

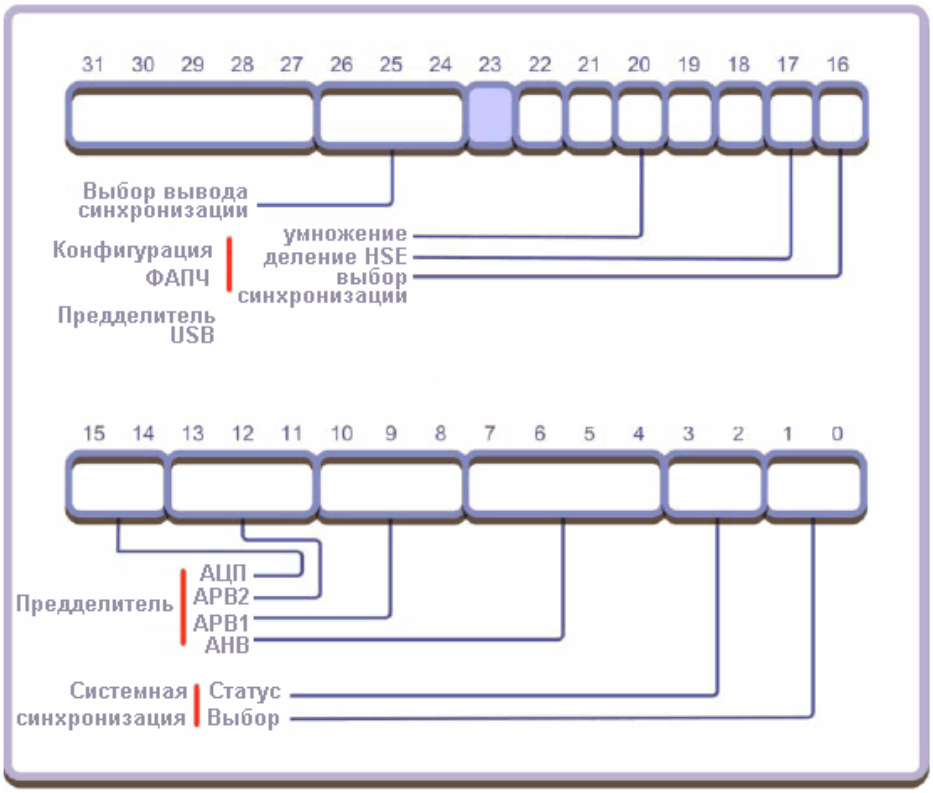
Сразу после сброса источником синхронизации ЦПУ является HSI-генератор. В таком состоянии МК его внутренний генератор отключен. Первым шагом по обеспечению работы STM32 с максимальным быстродействием является включение HSE-генератора и ожидание стабилизации его частоты.



После сброса МК STM32 работает от внутреннего высокочастотного генератора. Внешний генератор необходимо включить

```
RCC->CR |= 0x10000; //включение HSE

// Ожидание стабилизации частоты HSE-генератора
while(!(RCC->CR &0x00020000)) { };
```

Включается внешний генератор с помощью регистра RCC_Control. Бит готовности сигнализирует о стабилизации частоты внешнего генератора. После того, как частота внешнего генератора станет стабильной, его можно выбрать как вход блока ФАПЧ. Выходная частота блока ФАПЧ зависит от заданного целочисленного значения коэффициента умножения частоты, которое хранится в регистре RCC_PLL_configuration. Если используется генератор на частоту 8 МГц, для генерации максимальной тактовой частоты 72 МГц необходимо задать коэффициент умножения 9. После настройки коэффициента умножения частоты, необходимо разрешить работу блока ФАПЧ посредством регистра управления. Как только выходная частота блока ФАПЧ станет стабильной, установится соответствующий флаг и выход ФАПЧ можно выбрать в качестве источника синхронизации ЦПУ Cortex.

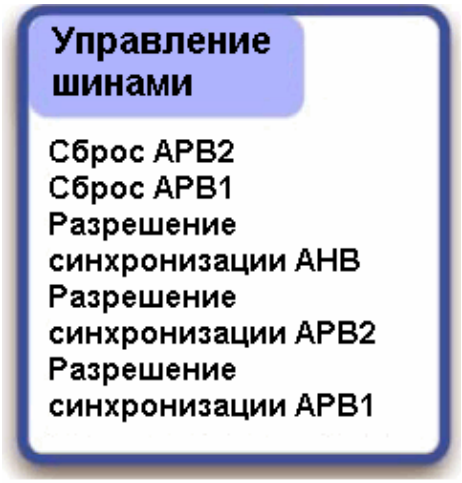


После запуска HSE-генератора его сигнал можно подать на вход схемы ФАПЧ. По завершении переходных процессов внутри схемы ФАПЧ ее сигнал можно использовать для синхронизации системы

```
//Синхронизация HSE-генератором и умножением частоты на 9
RCC->CFGR = 0x001D0000; //Активизация ФАПЧ
RCC->CR |= 0x01000000;

while(!(RCC->CR & 0x02000000))
{
    ; }

//Установка остальных полей управления
RCC->CR |= 0x00000001;

//Установка остальных полей конфигурации
RCC->CFGR |= 0x005D0402;
```

4.2.1.1. Настройка шин

Сразу после выбора ФАПЧ в качестве источника системной синхронизации ЦПУ Cortex будет работать на частоте 72 МГц. Чтобы остальная часть микроконтроллера работала с оптимальным быстродействием, необходимо выполнить настройку шин AHB и APB.



После сброса синхронизация многие из УВВ находятся в сброшенном состоянии и с отключенной синхронизацией. Перед использованием УВВ необходимо разрешить его синхронизацию и вывести из состояния сброса

```
//Разрешение синхронизации шин AHB,APB1 и APB2
RCC->AHBENR    = 0x00000014;
RCC->APB2ENR   = 0x00005E7D;
RCC->APB1ENR   = 0x1AE64807;

//Освобождение линий сброса УВВ на шинах APB1 и APB2
RCC->APB2RSTR= 0x00000000
RCC->APB1RSTR= 0x00000000;
```

4.2.2. Буфер Flash памяти

Если рассмотреть системную архитектуру МК STM32, не трудно заметить, что ядро Cortex-M3 связано с внутренней Flash памятью посредством отдельной шины инструкций I-Bus. Данная шина работает на той же частоте, что и ЦПУ, поэтому, после активизации ФАПЧ ядро будет пытаться работать с ней с максимальным быстродействием (72 МГц). Поскольку, большинство операций ЦПУ Cortex выполняет за один период синхронизации, то доступ к Flash памяти будет осуществляться каждые 1.3 нс. Сразу после запуска МК STM32 синхронизируется внутренним генератором частоты 8МГц, поэтому, проблем с доступом к Flash памяти на этом этапе еще нет. Однако сразу после активизации блока ФАПЧ и выбора его в качестве источника синхронизации время доступа к Flash памяти окажется слишком большим (35 нс), чтобы ЦПУ Cortex могло работать с максимальным быстродействием. Чтобы ЦПУ могло работать на частоте 72 МГц без состояний ожидания в цикле доступа, у Flash памяти предусмотрен буфер упреждающей выборки, состоящий из двух 64-битных буферов. Каждый из этих буферов отвечает за считывание 64-битного слова из Flash памяти и дальнейшую передачу 16- или 32-битных инструкций в ЦПУ Cortex. Данный способ хорошо совместим с инструкциями условного перехода набора инструкций Thumb-2 и предсказанием переходов на конвейере Cortex. В ходе нормального функционирования МК, программисту не следует выполнять каких-либо особых действий с буфером Flash памяти. Однако перед выбором ФАПЧ в качестве основного источника синхронизации необходимо убедиться, что буфер Flash памяти активен. Управление буфером осуществляется через регистр управления доступом к Flash памяти. Помимо активизации буфера, также необходимо задать количество

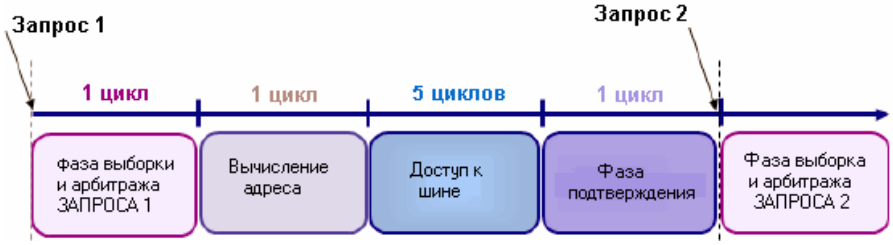
состояний ожидания, которое необходимо буферу предварительной выборки для считывания 8 байт инструкций из Flash памяти. Задержка выбирается следующим образом:

```
0<  SYSCLK  <24МГц  0 состояний ожидания
24< SYSCLK  <48МГц  1 состояние ожидания
48< SYSCLK  <72 МГц 2 состояния ожидания
```

Данные состояния ожидания действуют между буфером предварительной выборки и Flash памятью и не оказывают влияния на ЦПУ Cortex. После выполнения ЦПУ инструкции из первой части буфера, вторая его часть загружается, таким образом, чтобы выполнение кода осуществлялось непрерывно с оптимальным быстродействием.

4.2.3. Прямой доступ к памяти

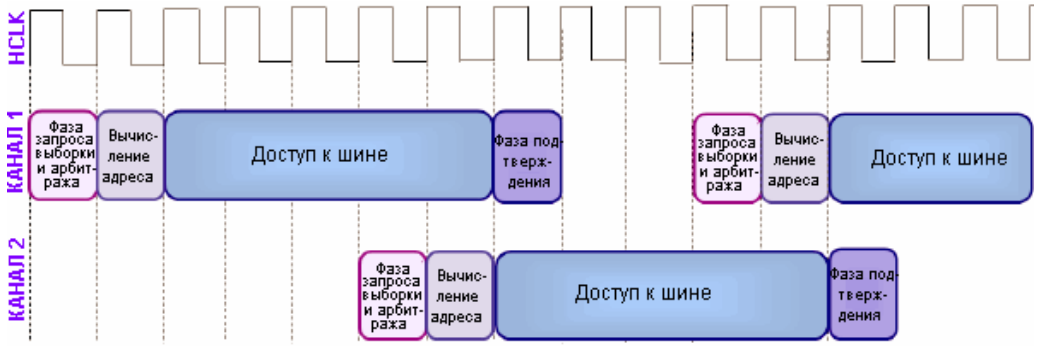
Передача данных между УВВ и внутренним статическим ОЗУ может осуществляться, как при участии ЦПУ Cortex, так и автоматически под управлением встроенного блока ПДП. Блок ПДП микроконтроллеров STM32 имеет семь раздельно настраиваемых каналов, позволяющие автоматически передавать данных из памяти в память, из УВВ в память, из памяти в УВВ и из УВВ в УВВ. Передача память-память выполняется с максимально-возможным для канала ПДП быстродействием. Если же в передаче данных участвует УВВ, то блок ПДП оказывается под его управлением и передача данных будет происходить по запросу УВВ в любом из направлений. Помимо передачи блоков данных, каждый блок ПДП может непрерывно передавать данные в кольцевой буфер. Поскольку встроенные коммуникационные УВВ вообще не оснащены буферами FIFO, то каналы ПДП могут использоваться для передачи данных между УВВ и буферами в статическом ОЗУ. Блок ПДП был специально разработан под особенности МК STM32 и оптимизирован под частую передачу коротких потоков данных, что типично для микроконтроллерных применений.



Каждая передача память-память состоит из четырех фаз. Фаза доступа к шине длится 5 циклов, а все остальные - 1 цикл

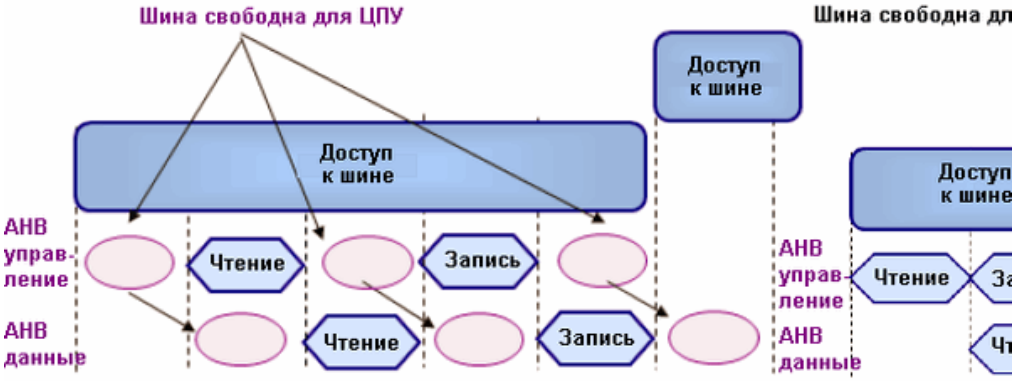
Каждая передача, осуществляемая блоком ПДП, состоит из четырех фаз: фаза выборки и арбитража, фаза вычисления адреса, фаза доступа к шине и фаза подтверждения. Все фазы, кроме фазы доступа к шине, длятся 1 цикл. Фаза доступа к шине (имеет место всякий раз, когда передаются реальные данные) длится 3 цикла и необходима для передачи слова данных. Чтобы блок ПДП и ЦПУ Cortex могли работать совместно, их активность чередуется, т.о. ПДП не блокирует работу ЦПУ и наоборот. Прежде чем перейти к рассмотрению механизма чередования, необходимо упомянуть о влиянии приоритетов на передачу между различными каналами ПДП. Каждому из каналов ПДП

программным способом назначается один из четырех уровней приоритета. На фазе арбитража доступ к шине получает канал с наивысшим уровнем приоритета. Если запрос на передачу отправили два блока ПДП и оба имеют одинаковый уровень приоритета, то доступ к шине получит канал с наименьшим порядковым номером.



Блок ПДП разработан для быстрой передачи небольших потоков данных. Потребность в такой передаче типична для небольших встраиваемых систем. Блок ПДП занимает шину только на фазе доступа к шине

Блок ПДП может выполнять фазу арбитража и вычисления адреса, даже если другой канала ПДП находится на фазе доступа к шине. Как только активный канал закончит передачу данных по внутренней шине, очередной канал ПДП уже будет готов к передаче и незамедлительно приступит к ее осуществлению, когда текущая передача завершится выполнением фазы подтверждения. Таким образом, каналы ПДП позволяют не только передавать данные быстрее, чем ЦПУ. Их работа еще и тщательно чередуется, а шина занимается только на время фактической передачи данных.



На каждой фазе доступа к шине три цикла свободны для ЦПУ. При передачах типа память-память этим гарантируется, что ЦПУ будет выделено минимум 60% шины, даже если ПДП выполняется непрерывно. Помните, что это касается только передачи данных, а выборку инструкций ЦПУ Cortex осуществляет по отдельной шине инструкций I-code

В передачах типа память-память каждый из каналов ПДП будет занимать шину данных только во время фазы доступа к шине, при этом, на передачу каждого слова данных будет затрачиваться 5 циклов. Из них один цикл - для чтения и еще один - для записи, причем данные циклы чередуются холостыми циклами, во время которых шина освобождается для ЦПУ Cortex. Это означает, что блоки ПДП будут использовать

не более 40% от пропускной способности шины данных даже во время непрерывной передачи данных на максимальной скорости. При передачах типа УВВ-УВВ и УВВ-память ситуациия несколько более сложная.

Передачи по шине AHB выполняются за два цикла на частоте синхронизации этой шины, а передачи по шине APB требуют для выполнения 2 циклов на частоте синхронизации этой шины и еще 2 цикла на частоте синхронизации шины AHB. Каждая передача блока ПДП состоит из двух периодов передачи по шине и свободного цикла. Например, передача из модуля SPI в статическое ОЗУ состоит из передачи из модуля SPI, из передачи в статическое ОЗУ и из одного свободного цикла. Следовательно,

```
Передача из SPI в стат. ОЗУ = Передача SPI (APB) + передача стат.
ОЗУ (AHB) + свободный цикл (AHB) = (2 цикла APB + 2 цикла AHB) +
2 цикла AHB + 1 цикл AHB = 2 цикла APB + 5 циклов AHB
```

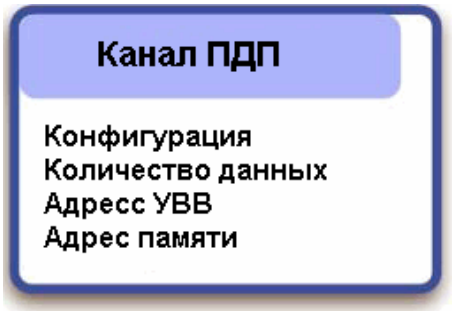
Помните, что все это касается только передачи данных, т.к. все выборку инструкций ЦПУ Cortex осуществляет по отдельной шине I-Bus.

У блока ПДП имеется семь каналов. Каждый канал является полностью ортогональным

Очередной хорошей новостью, касающейся блока ПДП, является то, что он чрезвычайно прост в использовании. Вначале необходимо включить синхронизацию блока ПДП и вывести его из состояния сброса. Это выполняется через регистр разрешения синхронизации AHB в блоке управления сбросом и синхронизацией.

```
RCC->AHBENR |= 0x00000001; // разрешение синхронизации блока ПДП
```

После подачи питания на блок ПДП. каждый из его каналов управляется через четыре регистра. В двух регистрах хранятся адреса источника и получателя (регистр УВВ и ячейка памяти). Данные о размере передачи хранятся в регистре "количества данных", а общие характеристики ПДП-передачи задаются через регистр конфигурации.



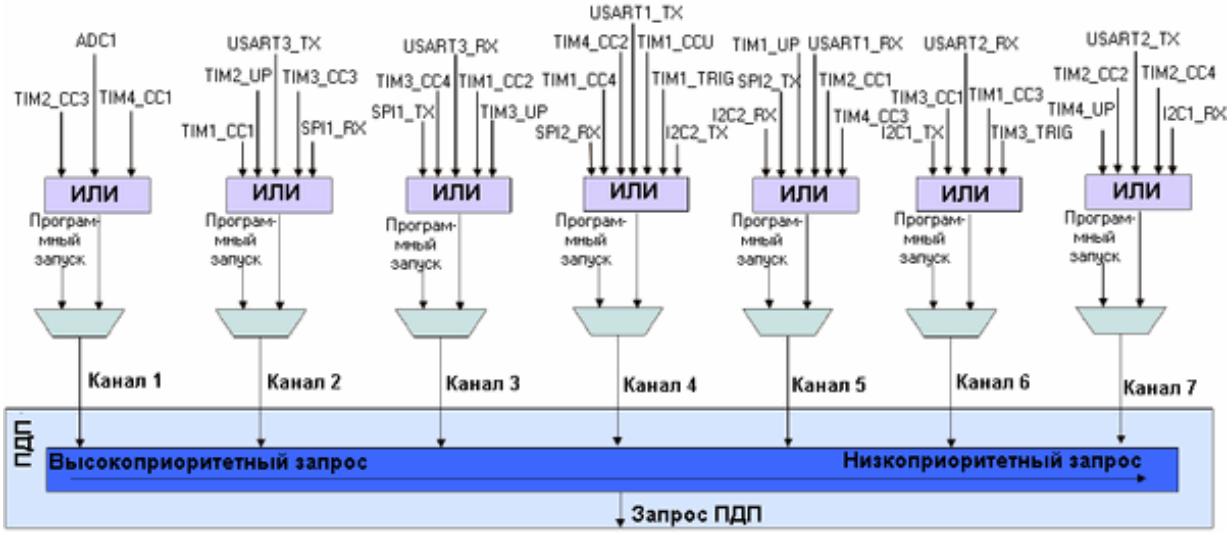
Каждый из каналов ПДП управляется через четыре регистра и поддерживает генерацию трех прерываний: по завершении, половинном завершении и ошибках передачи

Каждому из каналов ПДП можно назначить четыре уровня приоритета: "очень высокий", "высокий", "средний" и "низкий". Размера передаваемого слова задается раздельно для памяти и УВВ. Например, мы можем передать 32-битное слово в канал ПДП (3 цикла) из памяти, а затем передать четыре 8-битных слова в регистр данных УАПП (всего потребуется 35 циклов вместо 64, если бы все данные передавали 8-битными порциями.) Также имеется возможность инкрементирования адресов памяти и УВВ. Потребность в этом может возникнуть, например, при периодической передаче данных из регистра результата АЦП в массив памяти для дальнейшей обработки. Чтобы задать в каком из направлений, память - УВВ или УВВ - память, необходимо передавать данные, предусмотрен бит направления передачи. При передачах типа память-память необходимо установить бит 14 для передачи данных между двумя буферами в статическом ОЗУ на максимально-возможной скорости. Использовать каналы ПДП можно, как в режиме опроса, так и с использованием прерываний по завершению, половинному завершению и при ошибках передачи. Наконец, после завершения настройки ПДП-передачи, необходимо установить бит разрешения работы канала и передача начнется. Передачу память-память можно выполнить с помощью следующего кода программы:

```
DMA_Channel1->CCR = 0x00007AC0;                    //настройка передачи
память-память
DMA_Channel1->CPAR = (unsigned int)src_arry;       //выбор источника и
получателя
DMA_Channel1->CMAR = (unsigned int)arry_dest;
DMA_Channel1->CNDTR = 0x000A;                       //задание размера передачи

TIM2->CR1 = 0x00000001;                             //запуск таймера
DMA_Channel1->CCR |= 0x00000001;                    //запуск ПДП-передачи
while(!(DMA->ISR & 0x0000001))                      //ожидание завершения
передачи
{;}
TIM2->CR1 = 0;                                       //остановка таймера
TIM2->CNT = 0;                                       //сброс счетчика
TIM2->CR1 = 1;                                       //перезапуск таймера
for(index = 0;index <0xA;index++)                    //повторяющиеся операции,
выполняемые ЦПУ
{
arry_dest[index] = arry_src[index]; }
TIM2->CR1 = 0; //остановка таймера }
```

В данном коде программы демонстрируется простая передача память-память. Счет количества циклов выполняется внутренним таймером

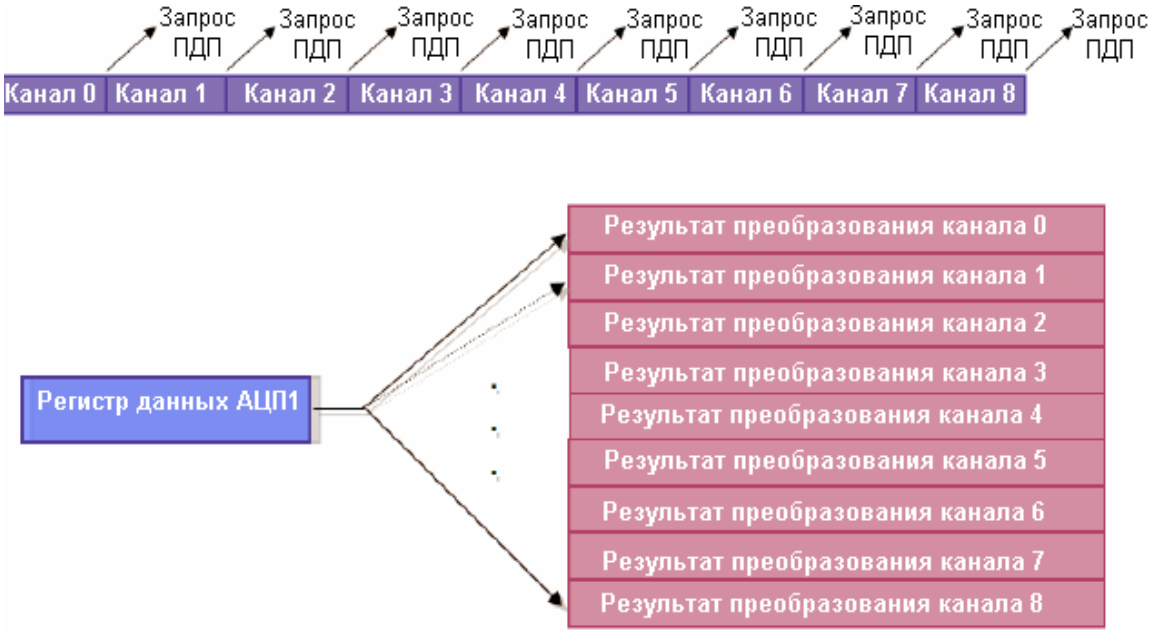
В приведенном выше коде выполняется передача 10 слов данных между двумя массивами в статическом ОЗУ: вначале с использованием ПДП, а затем с использованием только ЦПУ Cortex. В каждом из этих случаев, перед началом передачи запускается таймер и останавливается по завершении передачи. В данном примера блок ПДП выполняет передачу за 220 циклов, ЦПУ - за 536.



Каждое УВВ, которое поддерживает ПДП, связано с конкретным каналом. После разрешения работы, УВВ становится контроллером потока для ПДП-передачи. Благодаря этому, прием и передача данных выполняются без какого-либо участия ЦПУ

Несмотря на то, что передачи типа память-память выгодно использовать для инициализации областей памяти, а также для передачи блоков данных, большую часть времени каналы ПДП будут использоваться для перемещения данных между памятью и различными УВВ. В связи с этим, каждый из каналов ПДП связан с определенной группой УВВ. Вначале, необходимо инициализировать УВВ и разрешить поддержку им ПДП. Затем, нужно настроить соответствующий канал ПДП для передачи данных по запросу поддерживаемого им УВВ. Например, более подробно рассматриваемый далее АЦП по завершении каждого преобразования помещает 10-битный результат в регистр результата преобразования. Если ПДП не использовать, то ЦПУ будет вынужден периодически обрабатывать прерывания по завершению преобразования АЦП. Если же использовать ПДП, то по окончании каждого преобразования АЦП будет генерировать запрос на ПДП-передачу, а блок ПДП передаст результат преобразования АЦП в статическое ОЗУ по инкрементированному адресу. При таком подходе участие ЦПУ потребуется только для обработки подготовленного массива данных после завершения

передачи последней выборки.



УВВ микроконтроллеров STM32 не содержат каких-либо буферов памяти. Использование ПДП в его кольцевом режиме позволяет использовать любой объем памяти в качестве буфера УВВ. Используя прерывания по половинному и полному завершению передачи, можно организовать двойной кольцевой буфер

Чтобы этот процесс был более эффективен, можно активизировать поддержку кольцевого буфера, что позволит АЦП непрерывно записывать данные в этот буфер. Затем, используя прерывания по половинному и полному завершению ПДП передачи, можно создать двойной буфер. После заполнения первой половины буфера генерируется прерывание, которое позволяет перейти к обработке накопленных данных, а, при этом, новые данные будут продолжать накапливаться во второй половине буфера. Аналогичным образом, после заполнения второй части буфера, можно перейти к обработке данных, при этом, ПДП начнет перезаполнение буфера новыми данными. Такой же механизм работы ПДП можно использовать и совместно с другими УВВ. Единственно, важно обратить внимание, что у коммуникационных УВВ реализованы раздельные каналы ПДП приема и передачи. Например, модуль SPI может одновременно передавать данные в обоих направлениях.

5. Устройства ввода-вывода

В данном разделе будет представлена общая информация об устройствах ввода-вывода (УВВ), которые доступны пользователю в различных версиях микроконтроллеров STM32. Для удобства УВВ разделяются на две группы: микроконтроллерные УВВ общего назначения и коммуникационные УВВ. Все УВВ микроконтроллеров STM32 отличаются высокой степенью сложности и очень плотно связаны с блоком ПДП. Каждое УВВ обладает избыточными схемными блоками, которые позволяют минимизировать участие ЦПУ в выполнении каждым конкретным УВВ задач. Иными словами, они облегчают

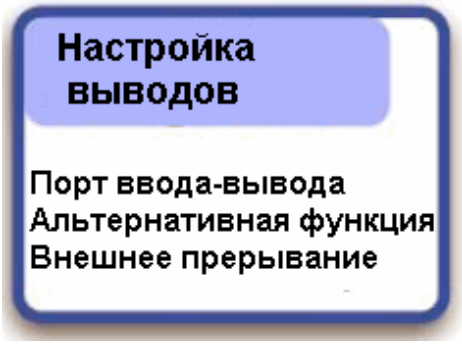
автоматизацию аппаратных ресурсов, снижая нагрузку на ЦПУ по управлению УВВ.

5.1. УВВ общего назначения

УВВ общего назначения микроконтроллеров STM32 состоят из портов ввода-вывода (ПВВ) общего назначения, контроллера внешних прерываний, аналогово-цифровых преобразователей, таймеров общего назначения, расширенного таймера и часов реального времени с энергонезависимыми (за счет резервирования питания) регистрами и входом обнаружения вмешательства.

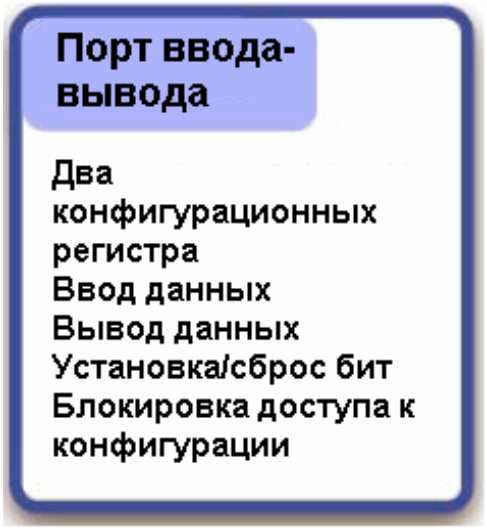
5.1.1. Порты ввода-вывода общего назначения

У МК STM32 предусмотрено до 80 двунаправленных линий ввода-вывода. Все линии ввода-вывода разделены на 5 портов по 16 линий ввода-вывода в каждой.



Каждая цифровая линия ввода-вывода может выполнять функцию линии ввода-вывода общего назначения или альтернативную функцию. Каждый из выводов может выполнять дополнительную функцию одного из 16 входов внешних прерываний

Данные порты называются A..E и совместимы с напряжением 5В. Многие из внешних выводов могут выполнять альтернативную функцию линии ввода-вывода встроенного УВВ, например, модуля USART или I2C. Кроме того, 16 входных линий встроенного блока внешних прерываний могут быть соединены с любыми из линиями портов ввода-вывода.

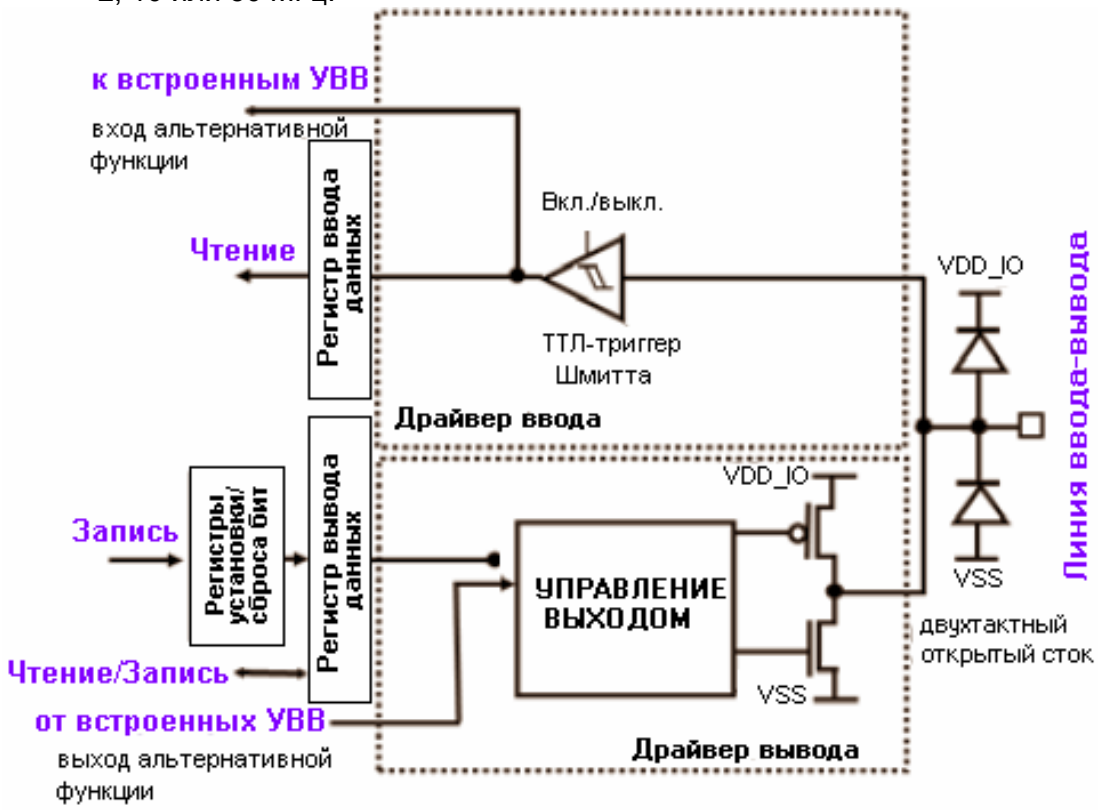


Каждый ПВВ поддерживает возможность раздельной конфигурации линий на ввод или на вывод.

Кроме того, предусмотрены регистры как для записи целых слов, так и для воздействия на отдельные биты. После завершения конфигурации доступ к ней можно заблокировать

У каждого ПВВ имеется два 32-битных конфигурационных регистра. Совместно они образуют 64-битный конфигурационный регистр. Эти 64 бита разделены на 4-битные поля, позволяющие настроить соответствующую им линию ввода-вывода. В свою очередь, 4-битное поле конфигурации состоит из 2-битного поля режима и 2-битного поля конфигурации. Поле режима позволяет указать, в каком направлении работает линия: на ввод или на вывод, а поле конфигурации позволяет настроить характеристики управления:

- если линия настроена как вход, то к ней можно подключить подтягивающий к плюсу или минусу резистор;
- линия, настроенная на вывод, может быть либо двухтактной, либо с открытым стоком. Для линий вывода можно также указать ее максимальное быстродействие: 2, 10 или 50 МГц.



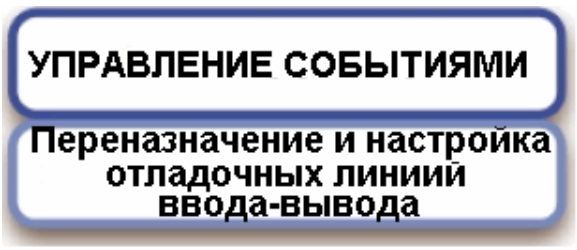
Конфигурация	CNF1	CNF0	MOD1	MOD0
Аналоговый вход	0	0	00	
Плавающий вход (состояние после сброса)	0	1		
Вход с подтягиванием к плюсу	1	0		
Вход с подтягиванием к минусу	1	0		

Двухтактный выход	0	0	00: резерв 01: 10МГц 10: 2МГц 11: 50МГц	
Выход с открытым стоком	0	1		
Двухтактный выход альтернат. функции	1	0		
Выход с открытым стоком альтернат. функции	1	1		

После завершения настройки портов, конфигурационные параметры можно защитить. Для этого необходимо выполнить запись в регистр блокировки конфигурации. В этом регистре для каждого вывода предусмотрен бит блокировки. После его установки блокируется возможность записи в соответствующие поля конфигурации и режима. После установки всех требуемых бит блокировки, необходимо в бит 16 регистра блокировки записать последовательность 1, 0, 1. Этим будет активизирована блокировка. Убедиться в активизации блокировки можно, если сразу после записи выполнить подряд два считывания этого же бита. Считывание значений 0, 1 свидетельствует об успешной активации блокировки. Доступ к линиям ввода-вывода осуществляется через регистры ввода и вывода данных. Для выполнения действий над отдельными битами можно использовать либо поддерживаемый ядром Cortex способ "bit banding" применительно к регистрам ввода и вывода, либо воспользоваться двумя специальными регистрами битовой обработки. Регистр установки/сброса бит - 32-битный регистр. Верхние 16 бит связаны с каждой из линий ПВВ. Запись в них логической 1 приводит к сбросу соответствующей линии ввода-вывода. Идентично этому, запись логической 1 в любой из младших 16 бит приведет к установке соответствующей линии ввода-вывода. Второй регистр битовой обработки - регистр сброса бит. Этот регистр 16-битный. Запись в его биты логических 1 приводит к сбросу соответствующих линий ввода-вывода. Сочетание регистров ПВВ, способа "bit banding" и регистров битовой обработки предоставляют пользователю отличные возможности управления всеми линиями ввода-вывода МК STM32 и могут чрезвычайно эффективно использоваться в применениях с интенсивным использованием линий ввода-вывода.

5.1.1. Альтернативные функции

Регистры альтернативных функций управляют переключением линии ввода-вывода между ПВВ и одним из встроенных УВВ. Чтобы схемотехническое проектирование было более гибким, линии ввода-вывода УВВ могут быть связаны с одним из нескольких выводов МК.



Опоздавшее высокоприоритетное прерывание отложит выполнение низкоприоритетного

прерывания, не выполняя при этом каких-либо дополнительных операций над стеком

Альтернативные функции у МК STM32 управляются через регистр переназначения и отладки. Каждое из цифровых УВВ (USART,CAN, таймеры, I2C и SPI) имеет одно- или двухбитное поле, которое позволяет назначить работу с различной комбинацией выводов. После выбора альтернативных функций выводов, необходимо в конфигурационных регистрах ПВВ переключить назначение вывода с линии ввода-вывода на альтернативную функцию. Регистр переназначения также управляет конфигурацией выводов отладочного JTAG-порта. Сразу после сброса, порт JTAG активизируется с отключенной функцией трассировки данных. JTAG можно переключить в режим двухпроводного отладочного интерфейса, а неиспользуемые выводы использовать в качестве линий ввода-вывода общего назначения.

5.1.2. Сигнализация событий

Процессор Cortex имеет возможность генерации импульса сигнализации событий, предназначенный для возобновления работы отдельного микроконтроллера, находящегося в экономичном режиме работы. Обычно, импульс сигнализации событий подается на вход возобновления работы второго МК STM32. Этот импульс генерируется при выполнении команды SEV из набора инструкций Thumb-2. Чтобы связать импульс сигнализации с конкретным выводом ПВВ, предусмотрен регистр управления событиями. Регистр управления событиями содержит поля для выбора порта и его линии ввода-вывода. После выбора линии ввода-вывода, необходимо завершить настройку установкой бита разрешения сигнализации событий.

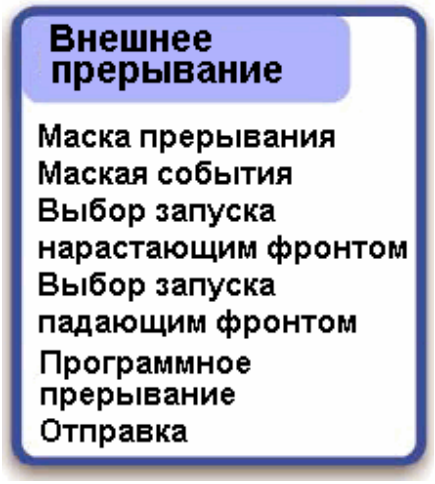
5.1.2. Внешние прерывания

Блок внешних прерываний имеет 19 линий прерываний, которые связываются с векторами прерываний посредством NVIC. Из них 16 линий связаны с линиями ПВВ и могут генерировать прерывание по нарастающему или подающему фронту, или же по обоим фронтам. Оставшиеся три линии связаны с линией прерывания ЧРВ, линией возобновления работы порта USB и выходом сигнализации блока контроля напряжения питания. NVIC предоставляет отдельные векторы прерываний для линий внешних прерываний (EXTI) 0-4, ЧРВ, блока контроля напряжения питания и блока USB. Остальные линии EXTI разделены на две группы, линии 5-9 и линии 10-15, которые связаны с двумя дополнительными векторами прерываний. Блок внешних прерываний играет важную роль для управления энергопотреблением МК STM32. Данный блок является асинхронным и, поэтому, может использоваться для возобновления работы микроконтроллера, находящегося в режиме STOP, когда оба основных генератора отключены. EXTI может генерировать прерывание, как для выхода из состояния Wait в режиме прерываний, так и для выхода из состояния Wait в режиме событий.

МК STM32 имеют 16 линий внешних прерываний, которые можно подключить к любой из линий ввода-вывода

Каждую из 16 линий EXTI можно связать с соответствующей линией ввода-вывода любого из портов. Для этого предусмотрены четыре конфигурационных регистра. Данные регистры разделены на четырехбитные поля, связанные с каждой линией EXTI. С помощью данного поля каждую линию EXTI можно связать с любым из пяти ПВВ, например, линию EXTI0 можно связать с линией 0 порта A, B, C, D или E. Такой подход позволяет использовать любой из выводов МК в качестве линии прерывания. Функцию EXTI можно также использовать в связке с альтернативной функцией, активизированной на внешнем выводе.

```
//Назначение функций внешних прерываний линиям ПВВ
AFIO->EXTICR[0]     = 0x00000000;
//Разрешение источников внешних прерываний
EXTI->IMR           = 0x00000001;
//Разрешение возобновления при внешних событиях
EXTI->EMR           = 0x00000000;
//Выбор падающего фронта в качестве источника запуска
EXTI->FTSR          = 0x00000001;
//Выбор нарастающего фронта в качестве источника запуска
EXTI->RTSR          = 0x00000000;
//Разрешение источников прерываний в NVIC
NVIC->Enable[0]     = 0x00000040;
NVIC->Enable[1]     = 0x00000000;
```

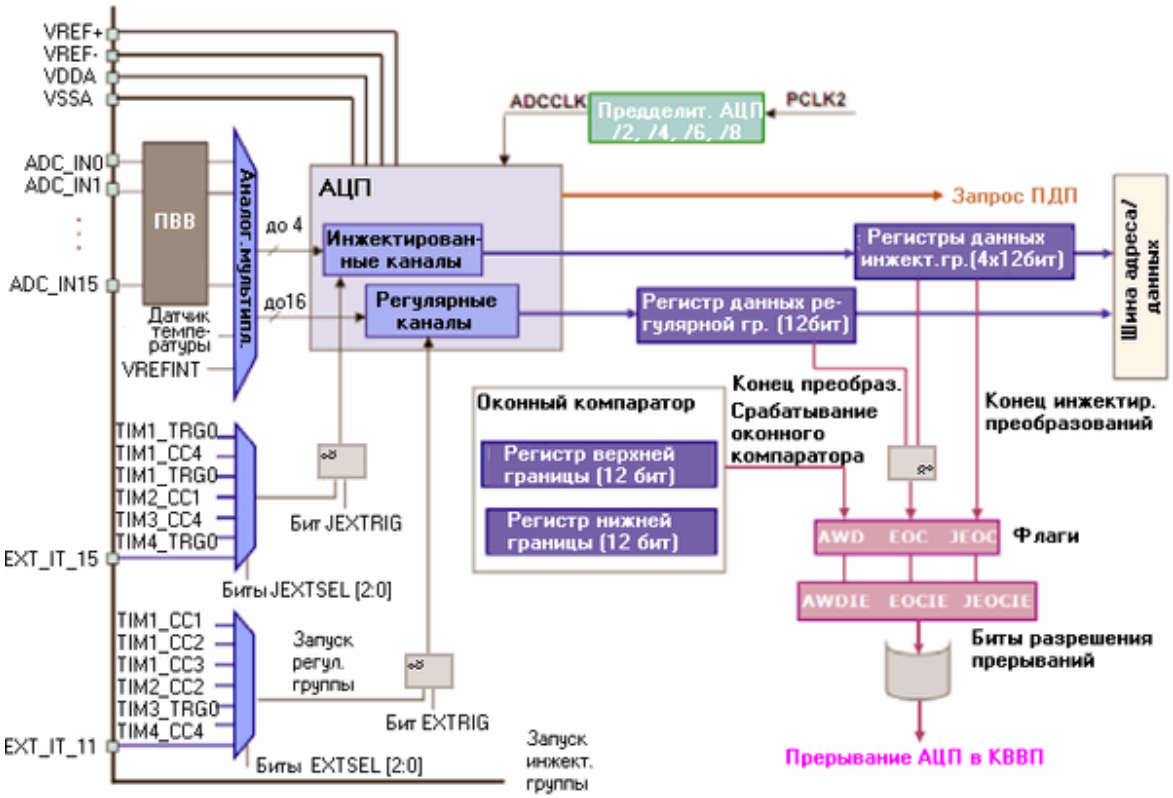


У МК STM32 имеется 16 линий прерывания, которые можно подключить к любой линии ввода-вывода. После подключения, выводы внешних прерываний могут генерировать прерывание по падающему и/или нарастающему фронту

После установки регистров конфигурации EXTI, каждое внешнее прерывание можно настроить на генерацию прерывания по нарастающему или падающему фронтам. Также предусмотрена возможность принудительной генерации прерывания EXTI путем записи в соответствующие биты регистра программного прерывания.

5.1.3. АЦП

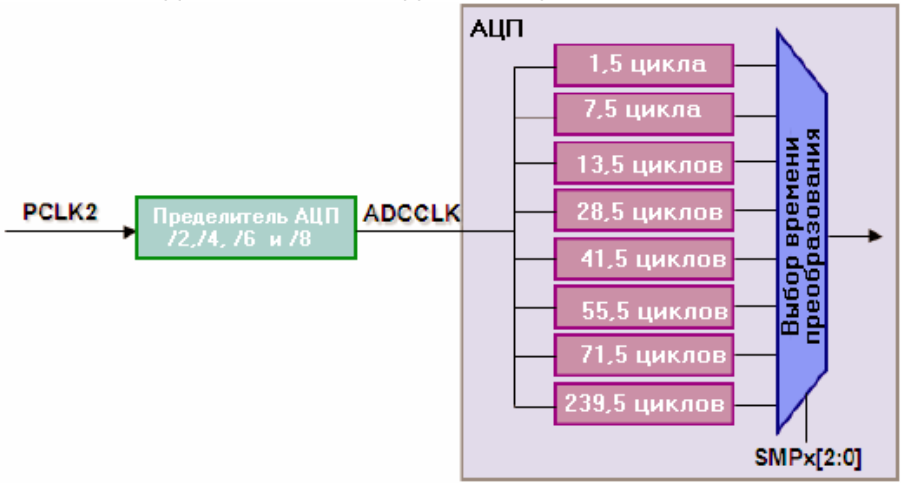
В зависимости от модели, в микроконтроллеры STM32 может быть встроено один или два аналогово-цифровых преобразователя. АЦП питаются отдельным напряжением, которое в зависимости типа корпуса может находиться в пределах 2.4..3.6В. Источник опорного напряжения (ИОН) АЦП соединен либо внутренне с напряжением питания АЦП, либо со специальными внешними выводами. АЦП характеризуется 12-битной разрешающей способностью и частотой преобразования 1МГц. У него имеется до 18 мультиплексированных каналов, 16 из которых можно использовать для измерения внешних сигналов. Оставшиеся два канала связаны со встроенным датчиком температуры и внутренним ИОН.



В МК STM32 интегрирован высококачественный 12-битный АЦП на частоту преобразования 1 МГц со встроенными ИОН и датчиком температуры

5.1.3.1. Время преобразования и группы преобразования

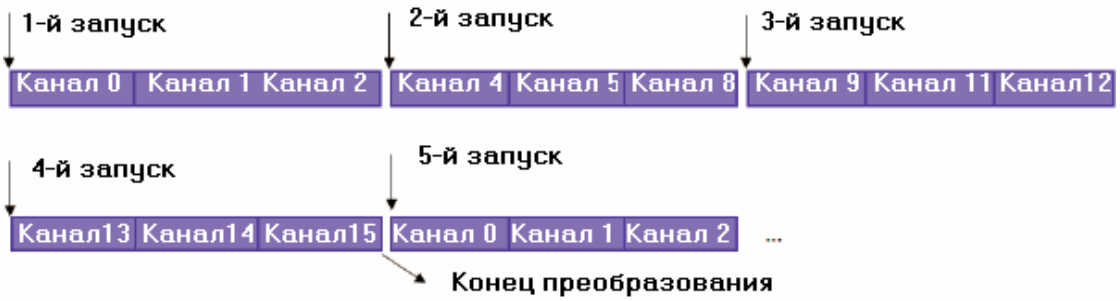
АЦП поддерживает возможность раздельного программирования времени преобразования в каждом из каналов. Всего предусмотрена возможность выбора 8 дискретных значений из диапазона от 1.5 до 239.5 циклов.



Частота преобразования задается индивидуально для каждого из каналов АЦП

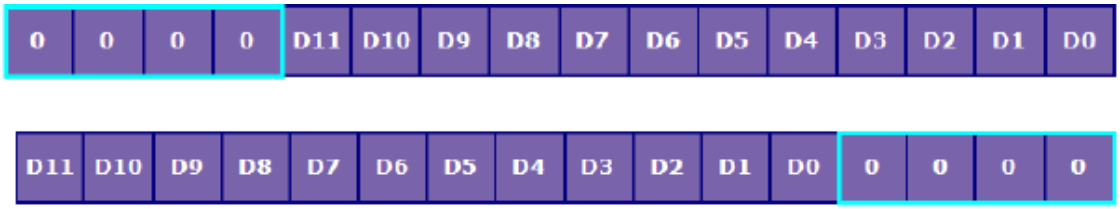
Каждый АЦП поддерживает два базовых режима преобразования: регулярный и

инжектированный. В режиме регулярных преобразований задается канал или группа каналов, которые в дальнейшем преобразовываются поочередно. Число каналов в группе регулярных преобразований конфигурируется пользователем (до 16 каналов). Кроме того, можно задавать порядок преобразования каналов, а один и тот же канал канал в цикле преобразования может быть оцифрован несколько раз. Группа регулярных преобразований запускается программно или аппаратно различными сигналами таймеров или по 1-ой линии внешних прерываний (EXTI 1). Сразу после запуска, преобразования в регулярной группе выполняются непрерывно. Альтернативно, группа может работать в режиме с остановкой преобразования, когда по завершении оцифровки выбранных каналов преобразование приостанавливается вплоть до следующего запуска группы регулярных преобразований.



Последовательность преобразования в группе регулярных преобразований может быть непрерывной (циклической) или периодической, когда после каждого запуска преобразований выполняется преобразование выбранных каналов

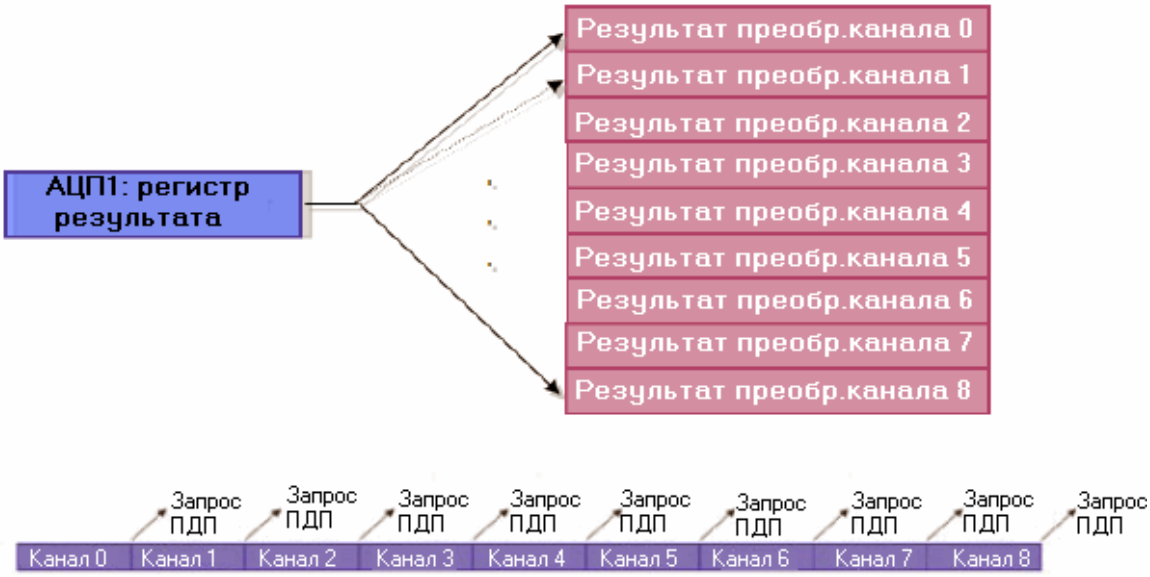
Каждый раз, когда завершается оцифровка в группе регулярных преобразований, результат преобразования помещается в единственный регистр результата преобразования и генерируется прерывание. 12-битный результат хранится в 16-битном регистре с левым или правым выравниванием бит.



12-битный результат хранится в 16-битном регистре результата с левым или правым выравниванием бит

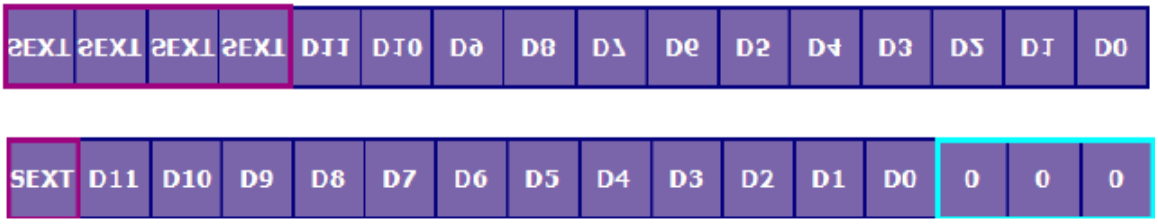
У АЦП1 имеется специальный канал ПДП, который может использоваться для передачи каждого результата преобразования в организованный в памяти буфер данных. Использование данного метода позволяет сократить частоту генерации прерываний до одного по завершении каждого цикла оцифровки группы регулярных преобразований. Более продвинутым методом является использование двойного буфера, который позволяет выполнять обработку накопленных данных из одной части буфера и, при этом, продолжать записывать результаты преобразований во вторую часть буфера, а по завершении цикла преобразования наоборот - обрабатывать данные из второй части, а помещать новые данные в первую. Для реализации данного метода необходимо

использовать прерывания по половинному и полному завершению ПДП, а также режим кольцевого буфера в блоке ПДП.



АЦП1 имеет поддержку ПДП для автоматической передачи в созданный пользователем в статическом ОЗУ буфер данных

Другая группа преобразований называется инжектированной. В последовательность преобразований инжектированной группы может входить до четырех каналов, а запуск преобразований выполняется программно или аппаратно. Сразу после запуска этой группы, приостанавливается оцифровка в группе регулярных преобразований, выполняется собственная последовательность преобразований, а затем возобновляется очередность преобразований в регулярной группе. Также как и в случае регулярной группы, может быть заданы любая последовательность каналов и многократное преобразование одного и того же канала в одной последовательности преобразований. Однако, в отличие от регулярной группы, у каждого инжектированного преобразования имеется свой собственный регистр результата и регистр смещения.

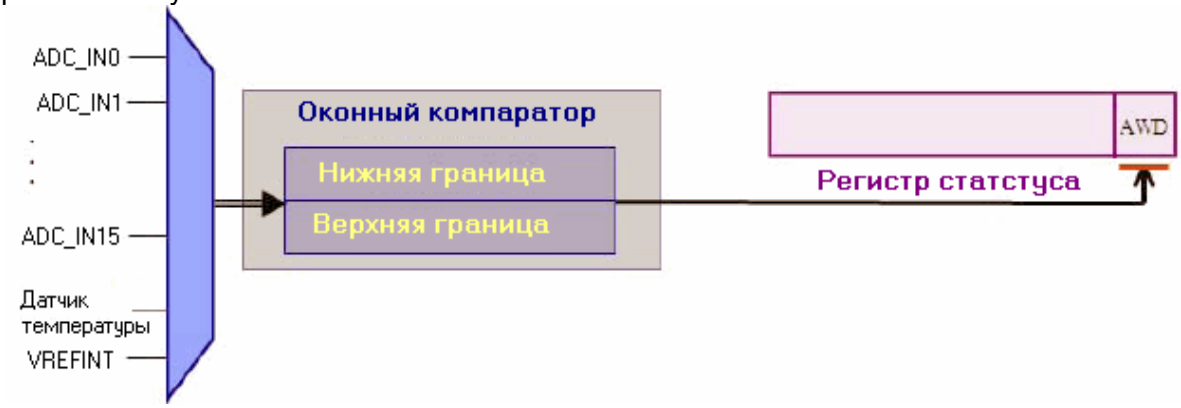


Регистры результата преобразования инжектированной группы дополнены битом знака и могут быть с правым или левым выравниванием

В регистр смещения можно запрограммировать 16-битное значение, которое автоматически вычитается из результата преобразования АЦП. Если результирующее значение - отрицательное, то регистр результата инжектированной группы дополняется знаком. Также как и в регулярной группе, результат может храниться с правым или левым выравниванием

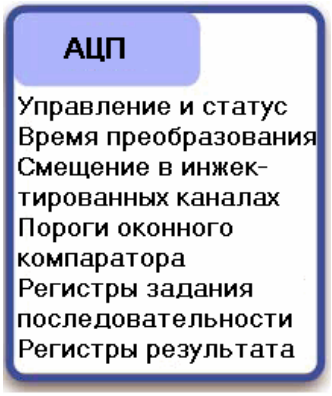
5.1.3.2. Функция оконного компаратора

Помимо двух режимов преобразования, АЦП поддерживают функцию оконного компаратора, которая заключается в генерации прерывания при выходе результата преобразования за пределы заданных пользователем нижней и верхней границ (условия снижения и превышения напряжения, соответственно). Оконный компаратор может использоваться для мониторинга выбранного регулярного или инжектированного канала, или же всех регулярных или инжектированных каналов. Помимо мониторинга напряжения, функция оконного компаратора может использоваться в качестве детектора пересечения нуля.



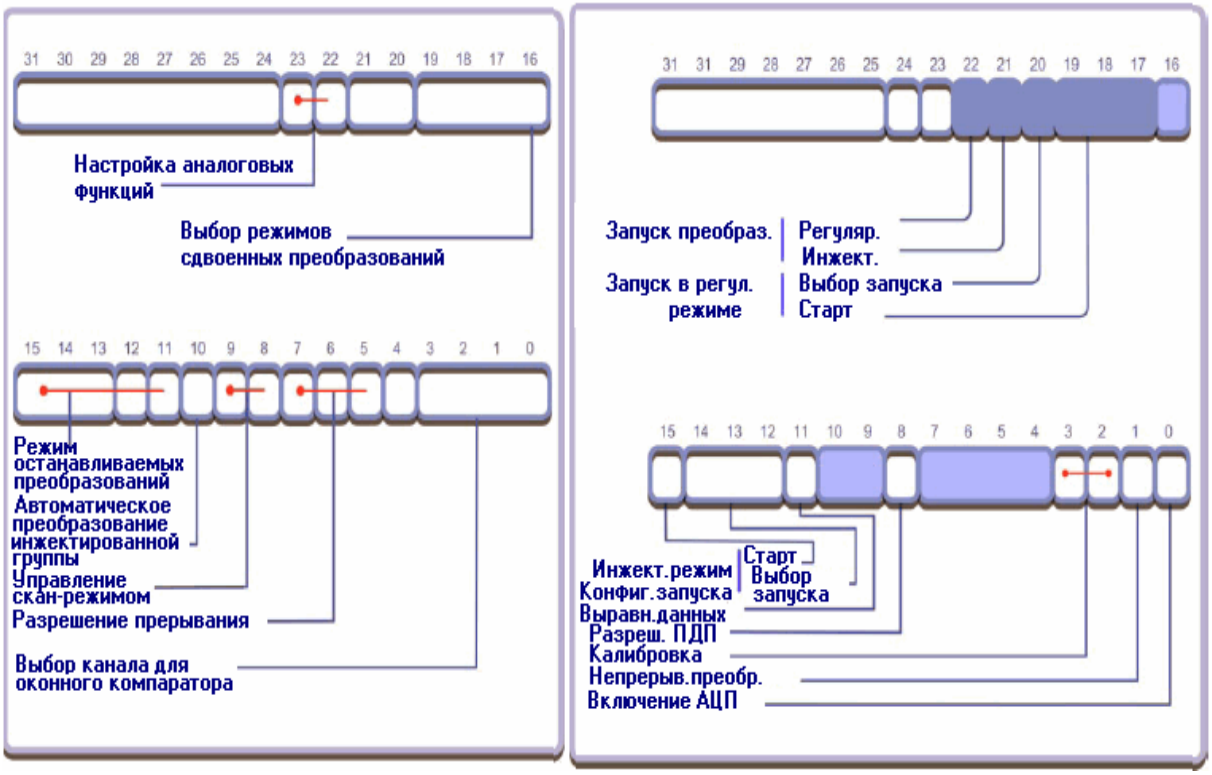
Функция оконного компаратора предназначена для мониторинга одного или всех каналов на предмет выхода за пределы заданной пользователем верхней и нижней границ

5.1.3.3. Базовая конфигурация АЦП



Регистры АЦП разделены на шесть групп. Конфигурация работы АЦП осуществляется через регистры управления и статуса

АЦП имеет блоки регистров для настройки: индивидуального времени преобразования, регулярных и инжектированных последовательностей преобразований, значений смещения для инжектированной группы и пороговых значений оконного компаратора. Вся настройка АЦП выполняется через регистры управления и статуса.



Режим работы АЦП задается через два регистра управления. Ниже показан пример конфигурации преобразования одного канала с генерацией прерывания

```
ADC1->CR2    = 0x005E7003;    //Включение АЦП и разрешение непрерывного
преобразования
ADC1->SQR1   = 0x0000;        //установка длины последовательности равной 1
ADC1->SQR2   = 0x0000;        //выбор преобразования канала 0
ADC1->SQR3   = 0x0001;
ADC1->CR2   |= 0x005E7003;    //перезапись бита разрешения работы
ADC1->CR1    = 0x000100;      //запуск преобразования регулярных каналов,
                              //разрешение прерывания АЦП
NVIC->Enable[0] = 0x00040000; //разрешение прерывания АЦП
NVIC->Enable[1] = 0x00000000;
```

В процедуре обработки прерывания результат преобразования считывается из регистра результата и выводится на линии порта ввода-вывода.

```
void ADC_IRQHandler (void) {
   GPIOB->ODR = ADC1->DR<<5; // копирование результата АЦП в ПВВ }
```

Вместо процедуры обработки прерывания, для передачи результата АЦП в ПВВ можно использовать канал ПДП.

```
DMA_Channel1->CCR    = 0x00003A28;        //кольцевой режим,
                                          //инкрементирование УВВ и памяти
отключено

//Задание адреса назначения - регистр данных ПВВ
DMA_Channel1->CPAR   = (unsigned int) 0x4001244C;
```

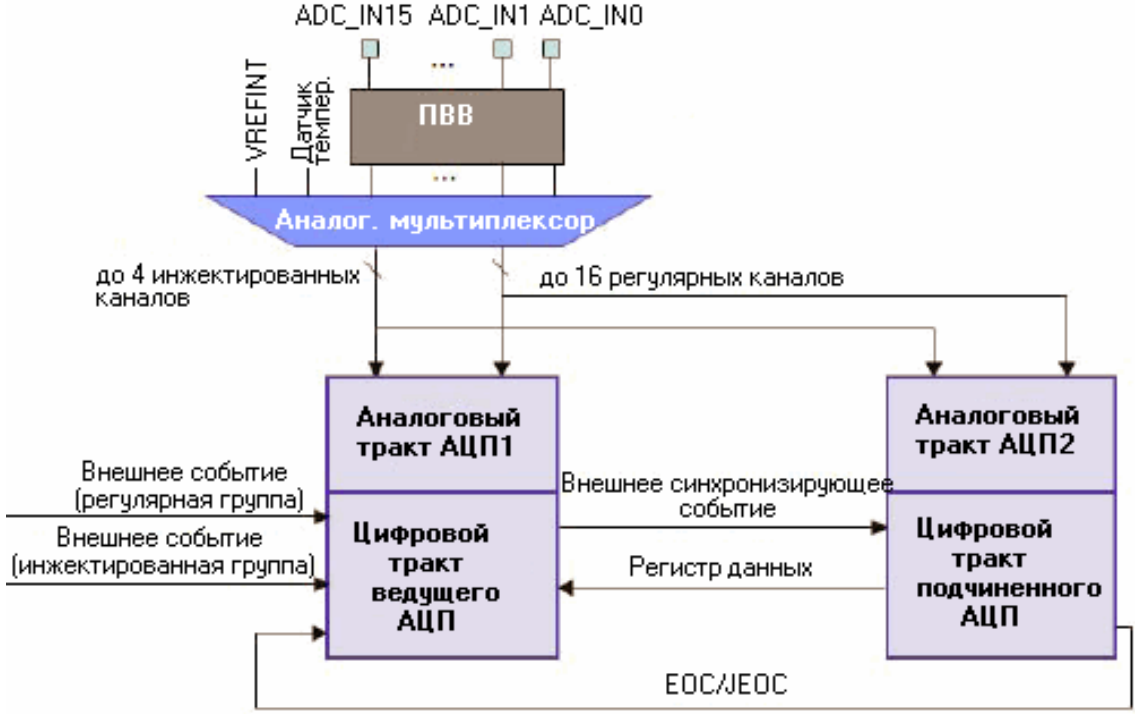
```
//Загрузка адреса источника в регистр памяти
DMA_Channel1->CMAR   = (unsigned int) 0x40010C0C;

DMA_Channel1->CNDTR  = 0x1;                //количество слов для передачи
DMA_Channel1->CCR |= 0x00000001;           //Разрешение ПДП передачи
```

У АЦП должна быть включена поддержка ПДП.

```
ADC1->CR2 |= 0x0100;
```

5.1.3.4. Режимы сдвоенных преобразований

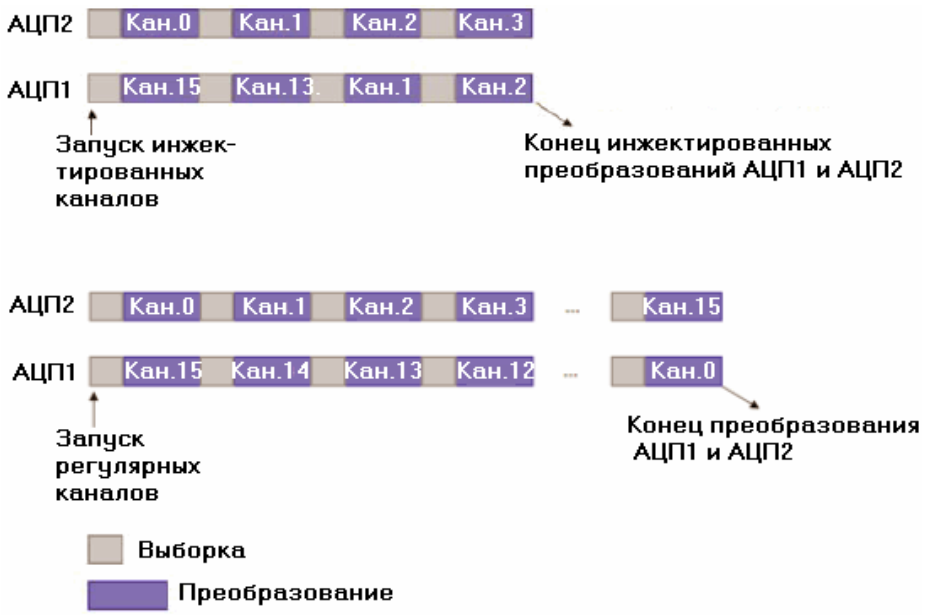
АЦП микроконтроллеров STM32, как для недорогих МК общего назначениия, является чрезвычайно многофункциональным. Не следует жалеть времени на изучение всех возможностей АЦП, т.к. после соответствующей настройки он может аппаратно выполнять те же действия, выполнения которых при использовании обычного модуля АЦП можно добиться только написанием дополнительного кода программы. Мало того, в семействе STM32 имеются МК с двумя интегрированными АЦП, которые поддерживают дополнительные режимы сдвоенных преобразований.



Режимы сдвоенных преобразований позволяют синхронизировать работу двух встроенных АЦП, предоставляя восемь дополнительных режимов

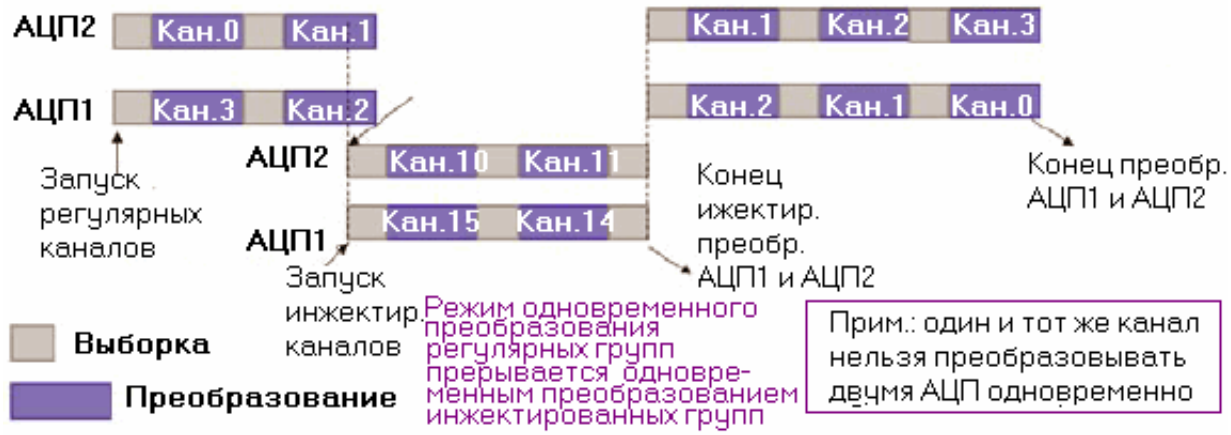
В режимах сдвоенных преобразований АЦП2 подчинен АЦП1, обеспечивая поддержку восьми дополнительных режимов преобразования.

5.1.3.4.1. Режимы одновременного преобразования инжектированных групп и одновременного преобразования регулярных групп



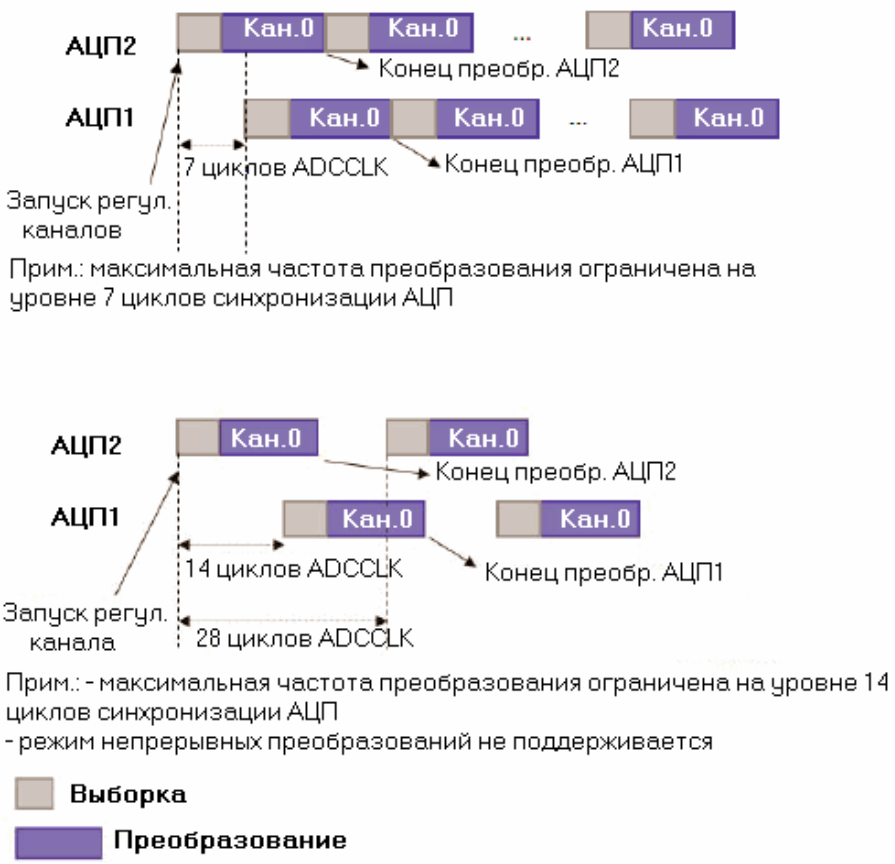
В первых двух режимах сдвоенных преобразований синхронизируется преобразование регулярной и инжектированной групп преобразования двух АЦП. Этот режим полезен при необходимости одновременного измерения двух физический величин, например, ток и напряжение.

5.1.3.5. Комбинированный режим одновременного преобразования регулярных/инжектированных групп



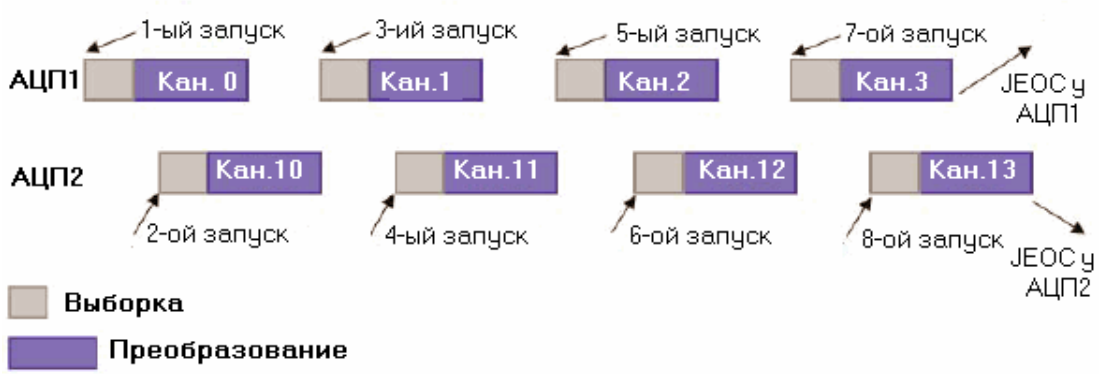
Данный режим предоставляет возможность синхронизировать преобразование, как регулярных, так и инжектированных групп обоих АЦП.

5.1.3.6. Режимы быстрых и медленных преобразований со смещением во времени



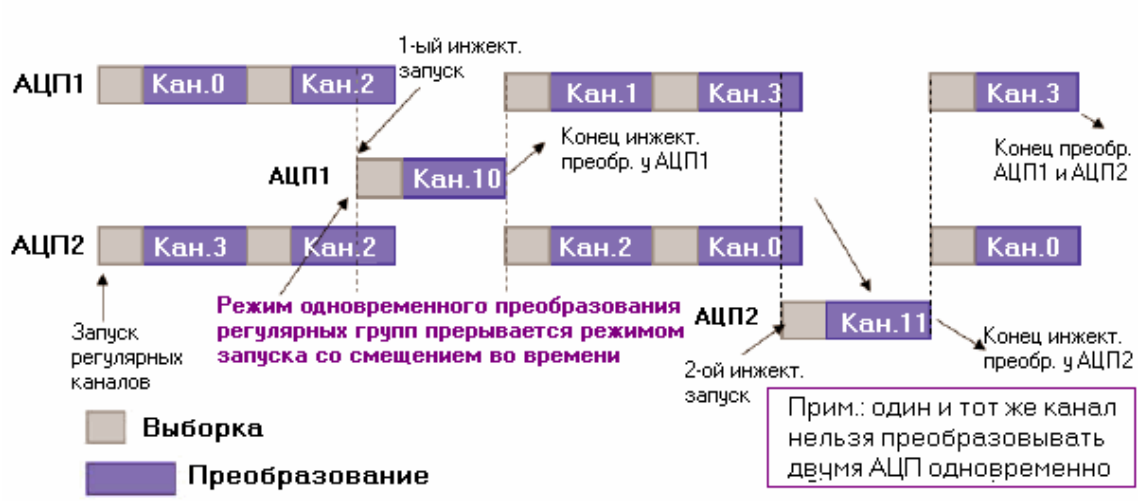
В режимах быстрых и медленных преобразований со смещением во времени синхронизируется работа регулярных групп преобразования обоих АЦП, однако в отличие от режима непрерывных преобразований, здесь перед запуском преобразования АЦП1 вводится небольшая задержка. В режиме быстрого преобразования со смещением во времени эта задержка равна семи циклам синхронизации АЦП и исчисляется по отношению к моменту запуска преобразования АЦП2. В режиме медленных преобразований со смещением во времени задержка равна 14 циклам синхронизации АЦП. Каждый из этих режимов может использоваться для увеличения частоты дискретизации за счет комбинирования работы двух преобразователей.

5.1.3.7. Режим поочередного запуска



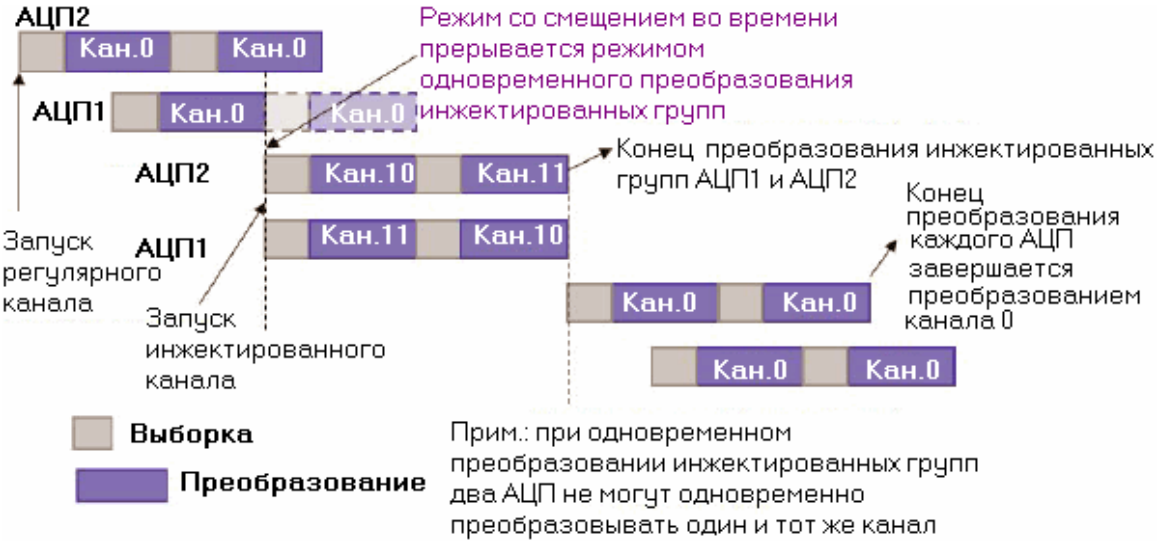
В режиме поочередного запуска аппаратный запуск АЦП1 вначале приведет к запуску преобразований инжектированной группы АЦП1, а затем - к запуску преобразований инжектированной группы АЦП2.

5.1.3.8. Комбинирование режима одновременного преобразования регулярной группы и режима поочередного запуска



Режим поочередного запуска можно скомбинировать с режимом одновременного преобразования регулярных групп. Вследствие этого, преобразования регулярных групп обоих АЦП будут выполняться синхронно, а инжектированных групп - поочередно.

5.1.3.9. Комбинирование режима одновременного преобразования инжектированной группы и режима со смещением во времени



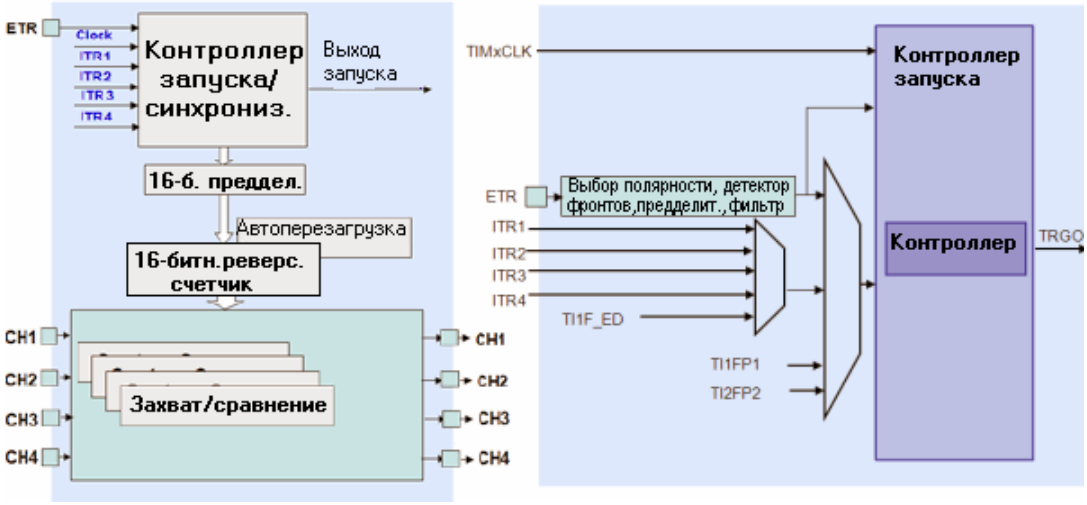
В последнем режиме преобразования преобразование регулярных групп обоих АЦП выполнятся со смещением во времени, а инжектированных - одновременно.

5.1.4. Таймеры общего назначения и многофункциональные таймеры

У МК STM32 имеется четыре блока таймеров. Таймер 1 - расширенный таймер, предназначенный для управления электродвигателем. Остальные таймеры являются таймерами общего назначения. Все таймеры выполнены по общей архитектуре, а расширенный таймер отличается лишь добавлением специальных аппаратных блоков. В данном разделе мы вначале рассмотрим таймеры общего назначения, а затем перейдем к изучению особенностей расширенного таймера.

5.1.4.1. Таймеры общего назначения

Все блоки таймеров выполнены на основе 16-битного перезагружаемого счетчика, который синхронизируется с выхода 16-битного предделителя. Перезагружаемое значение хранится в отдельном регистре. Счет может быть прямой, обратный или двунаправленный (сначала прямой до определенного значения, а затем обратный). Вход синхронизации счетчика можно связать с одним из восьми различных источников. В их число входят: специальный сигнал синхронизации, производный от сигнала главной системной синхронизации; выходной сигнал синхронизации одного из других таймеров или внешний сигнал синхронизации, связанный с выводами захвата/сравнения.

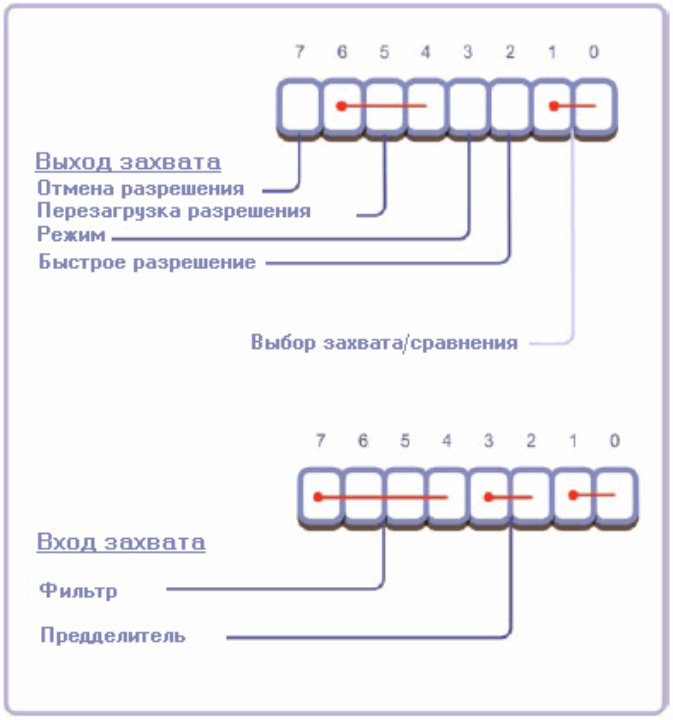


Каждый из четырех таймеров МК STM32 содержит 16-битный счетчик, 16-битный предделитель частоты и 4-канальный блок захвата/сравнения. Их можно синхронизировать системной синхронизацией, внешними сигналами или другими таймерами

Помимо составляющего основу таймера счетчика, в каждый блок таймера также входит четырехканальный блок захвата/сравнения. Данный блок выполняет, как стандартные функции захвата и сравнения, так и ряд специальных функций. Каждый из таймеров может генерировать прерывания и поддерживает ПДП.

5.1.4.1.1. Блок захвата/сравнения

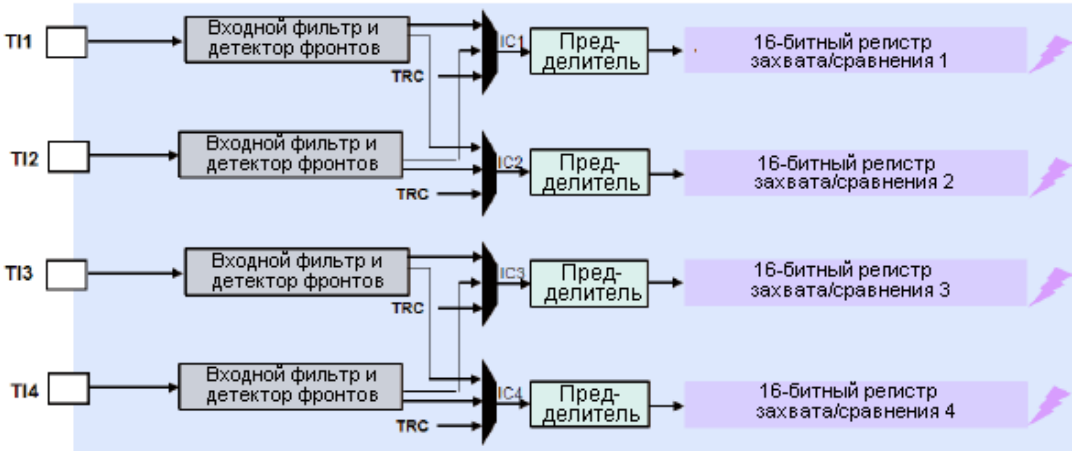
Каждый канал захвата/сравнения управляется через один регистр. Данный регистр имеет несколько функций, которые зависят от установок бит выбора. В режиме захвата, данный блок выполняет фильтрацию на входах, поддерживает специальный режим измерения внешнего ШИМ-сигнала, а также имеет входы для подключения внешнего энкодера. В режиме сравнения, блок выполняет стандартные функции сравнения, генерации ШИМ-сигналов, а также поддерживает опциональную функцию одновибратора.



У каждого канала захвата/сравнения имеется один регистр для задания режима работы

5.1.4.1.2. Блок захвата

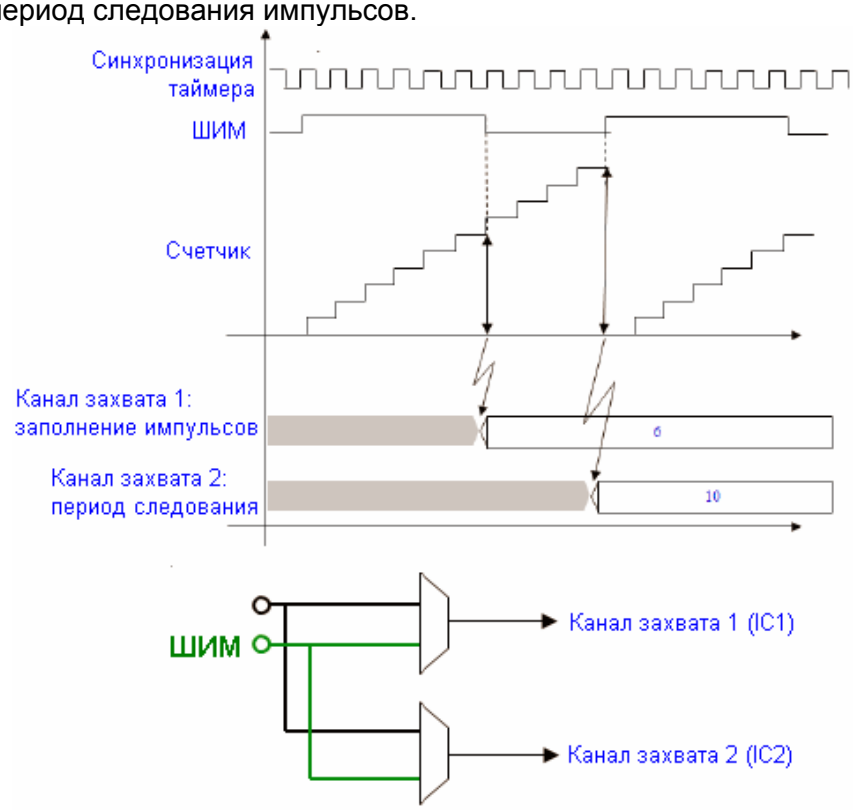
Базовый блок захвата имеет 4 канала, подключенных к конфигурируемым детекторам фронтов. При обнаружении нарастающего или падающего фронта, текущее значение счетчика записывается в 16-битный регистр захвата/сравнения. Когда возникает событие захвата, счетчик таймера может быть сброшен или приостановлен. Кроме того, одновременно с этим может быть запущено прерывание или ПДП-передача.



Каждый из четырех блоков захвата имеют входной фильтр и детектор фронтов. При возникновении события захвата может быть запущено прерывание или ПДП-передача

5.1.4.1.3. Режим измерения параметров ШИМ-сигнала

Блок захвата имеет возможность использования двух каналов захвата для автоматического измерения параметров внешнего ШИМ-сигнала, в т.ч. заполнение импульсов и период следования импульсов.



В режиме измерения параметров ШИМ-сигнала два канала могут использоваться для автоматического измерения периода и заполнения импульсов ШИМ-сигнала

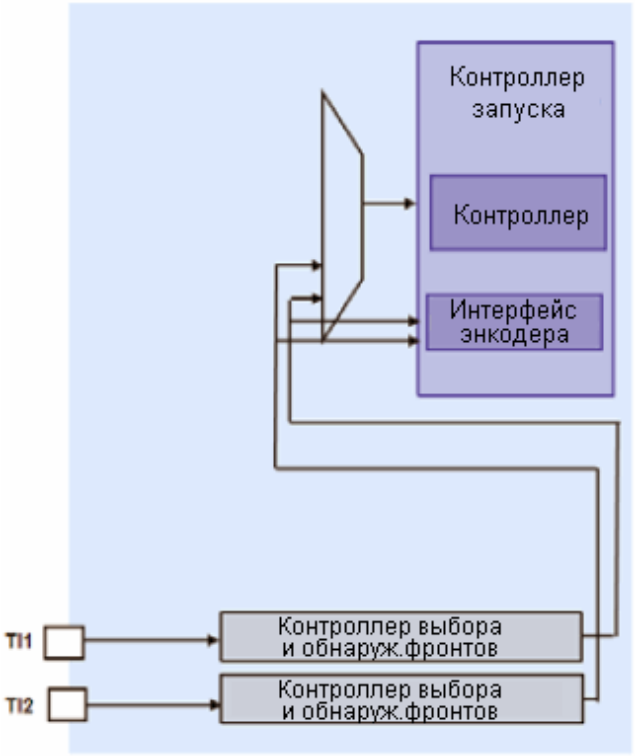
```
M3->CR1           = 0x00000000;     //по умолчанию
TIM3->PSC         = 0x000000FF;     //установка макс. коэф. предделителя
TIM3->ARR         = 0x00000FFF;     //установка макс. перезагружаемого знач.
TIM3->CCMR1       = 0x00000001;     //Вход IC1 связываем с TI1
TIM3->CCER       |= 0x00000000;     //IC1 реагирует на нарастающий фронт
TIM3->CCMR1      |= 0x00000200;     //Вход IC2 связываем с TI1
TIM3->CCER       |= 0x00000020;     //IC2 реагирует на падающий фронт
TIM3->SMCR        = 0x00000054;     //Выбор TI1FP1 в качестве входа, запуск
по нарастающему фронту
                                       //сброс счетчика
TIM3->CCER       |= 0x00000011;     //разрешение каналов захвата
TIM3->CR1         = 0x00000001;     //разрешение таймера
```

В режиме измерения параметров ШИМ-сигнала входной сигнал соединен с двумя каналами захвата. В начале цикла ШИМ основной счетчик сбрасывается через второй канал захвата (нарастающий фронт ШИМ-сигнала) и начинает прямой счет. При возникновении падающего фронта ШИМ-сигнала срабатывает первый канал захвата, что приводит к фиксации значения заполнения импульсов. При обнаружении следующего

нарастающего фронта вторым каналом захвата в начале следующего цикла ШИМ сбрасывается таймер и фиксируется значение периода ШИМ-сигнала.

5.1.4.1.4. Интерфейс энкодера

Блок захвата всех таймеров также поддерживает возможность непосредственного подключения к внешнему энкодеру, который обычно используется для контроля угловых перемещений и частоты вращения электродвигателей.

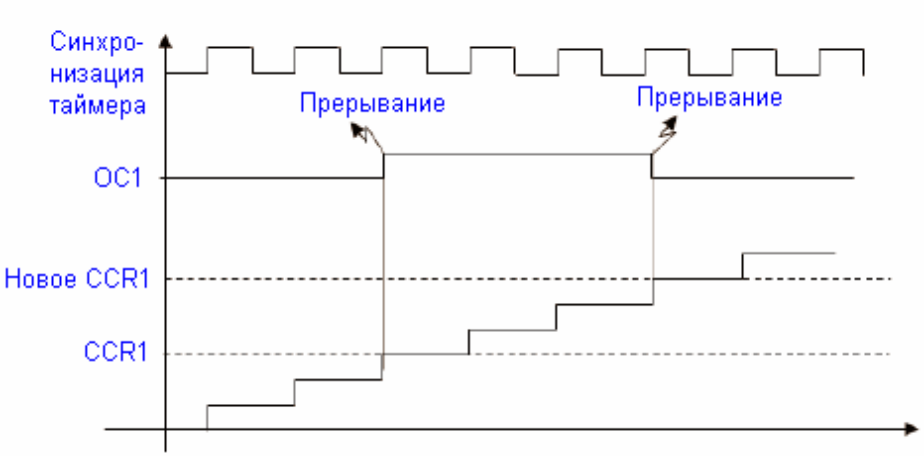


Каждый таймер имеет возможность подключения к линейному или поворотному энкодеру для контроля положения, скорости и направления

В данной конфигурации выводы захвата выступают в роли входов синхронизации счетчика. Состояние счетчика в дальнейшем используется для определения положения. Для контроля скорости необходимо использовать еще один таймер. Он необходим для измерения интервала времени между импульсами сигнала энкодера.

5.1.4.1.5. Режим сравнения

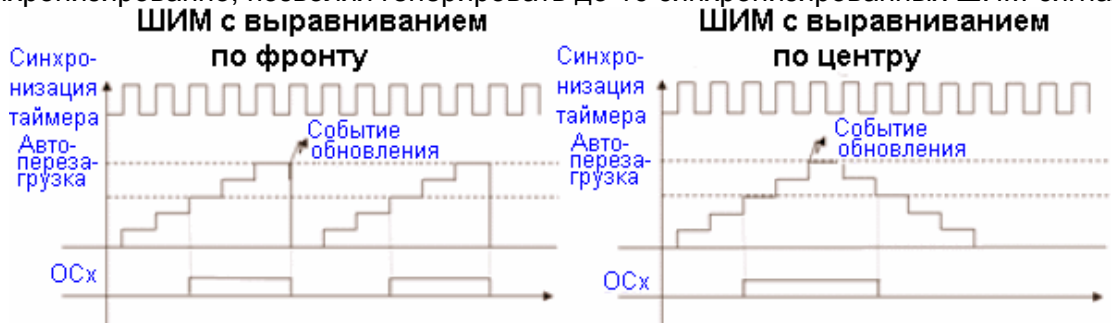
Каждый таймер микроконтроллеров STM32 имеет 4 канала схем сравнения. В базовом режиме сравнения генерируется событие захвата, если обнаруживается совпадение состояния счетчика с 16-битным значением, хранящимся в регистре захвата/сравнения. Данное событие может использоваться для изменения состояния связанного с каналом захвата/сравнения вывода, генерации сброса таймера, прерывания или ПДП-передачи.



В режиме сравнения, каждый канал может использоваться для генерации прерывания или изменения состояния CAP/COM при совпадении содержимого регистра сравнения с текущим содержимым счетчика

5.1.4.1.6. Режим широтно-импульсной модуляции

Помимо базового режима сравнения, каждый таймер поддерживает специальный режим генерации ШИМ-сигналов. В этом режиме период ШИМ задается с помощью регистра автоматической перезагрузки таймера. Значение заполнения импульсов задается через регистр захвата/сравнения канала. Таким образом, каждый таймер может генерировать до четырех независимых ШИМ-сигналов. Позже мы увидим, что таймеры могут работать и синхронизированно, позволяя генерировать до 16 синхронизированных ШИМ-сигналов.



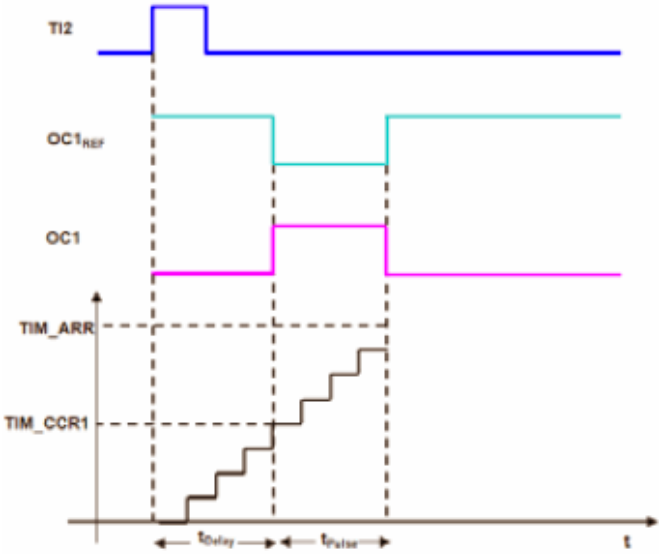
Каждый таймер поддерживает специальный режим ШИМ, в котором можно генерировать ШИМ-сигналы с выравниванием по фронту или центру

В каждом канале можно генерировать ШИМ-сигнал с выравниванием по фронту или по центру. В режиме с выравниванием по фронту, падающий фронт импульса всегда совпадает с моментом перезагрузки таймера. Изменение значения в регистре захвата/сравнения позволяет легко управлять моментом возникновения нарастающего фронта ШИМ-сигнала. В режиме с выравниванием по центру, таймер конфигурируется как реверсивный счетчик, который сначала считает в прямом направлении, а затем - в обратном. Когда будет выявлено совпадение счетчика с регистром захвата/сравнения канала, инвертируется состояние выходного сигнала канала.

```
TIM2->CR1       = 0x00000000;       //по умолчанию
TIM2->PSC       = 0x000000FF;       //установка макс. знач. предделителя
TIM2->ARR       = 0x00000FFF;       //установка макс. перезагружаемого знач.
TIM2->CCMR1     = 0x00000068;       //Устанавливаем режим ШИМ
TIM2->CCR1      = 0x000000FF;       //Задаем стартовое значение ШИМ
TIM2->CCER      = 0x00000101;       //разрешаем выхода канала 1
TIM2->DIER      = 0x00000000;       //разрешаем обновление прерывания
TIM2->EGR       = 0x00000001;       //разрешаем обновление
TIM2->CR1       = 0x00000001;       //разрешаем работу таймера
```

5.1.4.1.7. Режим одновибратора

Как в базовом режиме сравнения, так и в режиме ШИМ блоки таймеров непрерывно генерируют прямоугольные импульсы. Однако, при необходимости, в каждом из таймеров можно задействовать опциональный режим одновибратора для генерации одиночного импульса. Данный режим, по сути, является особой версией режима ШИМ, в котором генерация ШИМ-сигнала инициируется внешним сигналом запуска (внешний вывод или выход запуска любого другого таймера) и длится один цикл. В результате генерируется один импульс с программируемой задержкой и длительностью.

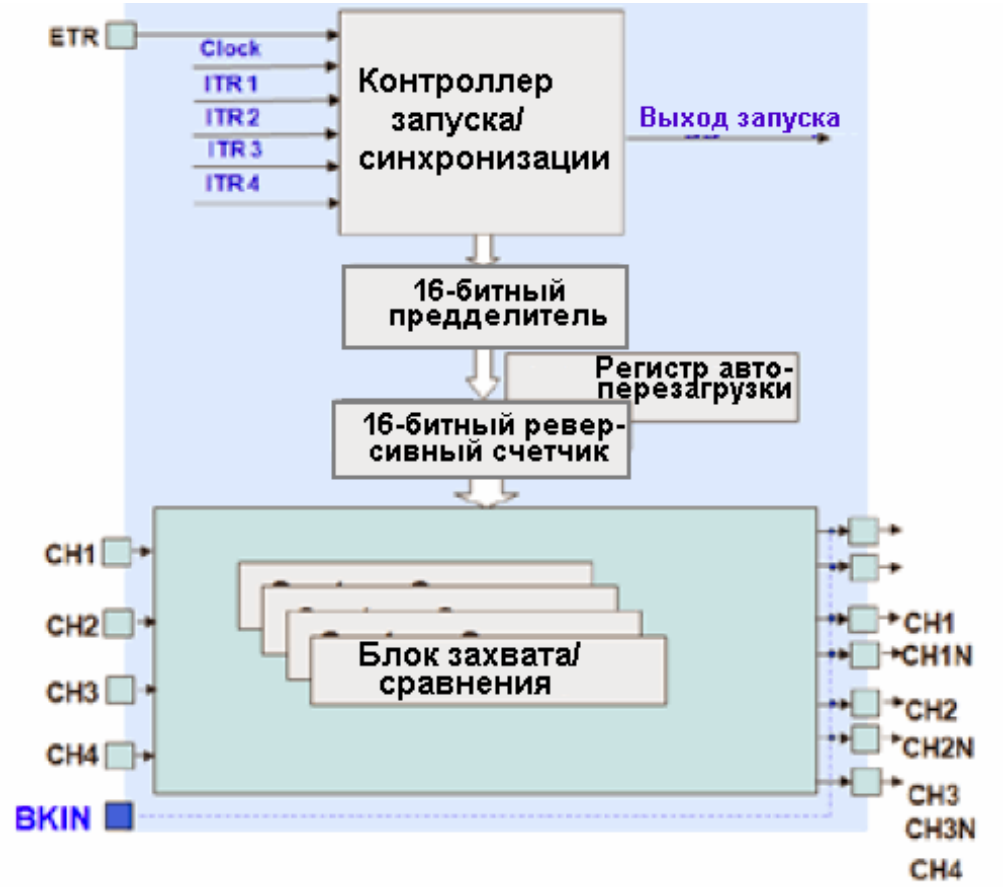


В режиме одновибратора можно сгенерировать одиночный импульс с программируемой задержкой и длительностью

5.1.4.2. Расширенный таймер

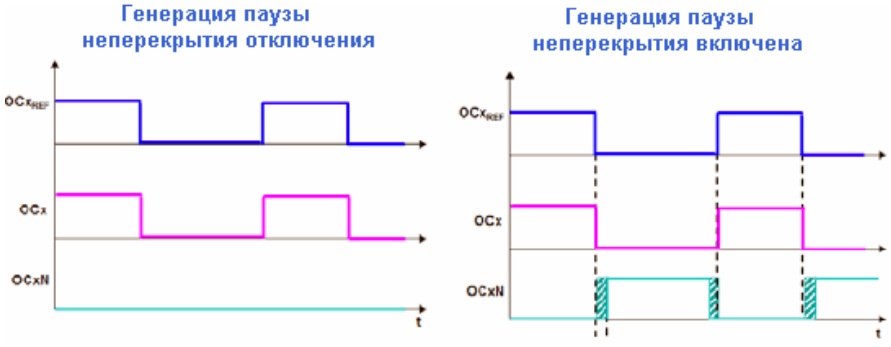
Расширенным таймером является блок таймера 1. Этот таймер дополнен рядом аппаратных узлов, предназначенных для управления электродвигателем.
В каждом из трех каналов этого таймера предусмотрены два противофазных выхода.

Таким образом, он представляет собой 6-канальный ШИМ-блок. Поскольку данный блок предназначен для управления трехфазным электродвигателем, у него имеется возможность программирования в каждом канале паузы неперекрытия и общий вход экстренного отключения. Кроме того, в дополнение к интерфейсу энкодера здесь предусмотрен интерфейс подключения датчиков Холла.



Расширенный таймер выполнен на основе той же структуры, что и таймеры общего назначения. У трех каналов сравнения предусмотрены противофазные выходы. Также имеется дополнительный вход экстренного отключения и интерфейс датчиков Холла

В каждом из каналов предусмотрена возможность программирования паузы неперекрытия противофазных выходов, которая позволяет избежать протекания сквозного тока через полумост силового коммутатора.



У расширенного таймера предусмотрена возможность программирования паузы неперекрытия

противофазных выходов, что важно для управления электродвигателями

5.1.4.2.1. Функция экстренного отключения

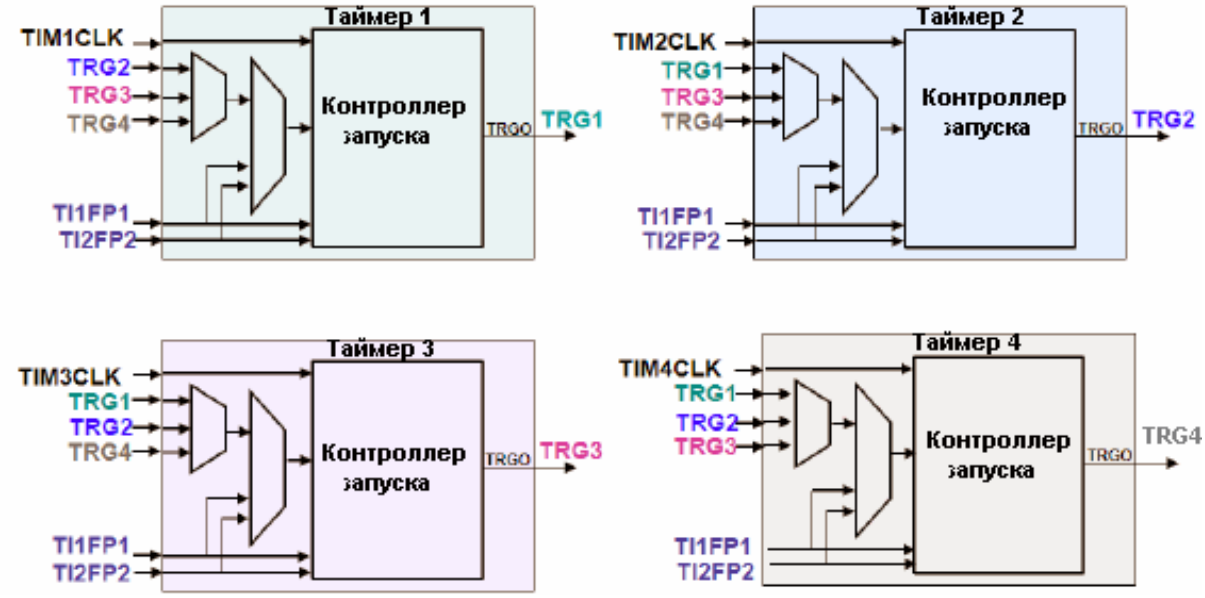
Расширенный таймер имеет возможность перевести свои ШИМ-выходы и их противофазные выходы в заранее запрограммированное состояние по сигналу на входе экстренного отключения. Источником данного сигнала может быть специальный внешний вывод экстренного отключения или система защиты синхронизации, которая контролирует работу внешнего высокочастотного генератора. После активизации, функция экстренного отключения работает полностью автономно и гарантирует перевод ШИМ-выходов в безопасное состояние в случае отказа системы синхронизации МК STM32 или в случае повреждения внешней схемы.

5.1.4.2.2. Интерфейс датчика Холла

Каждый из таймеров, в т.ч. и расширенный, разработан с учетом простоты подключения к датчику Холла, предназначенного для измерения угловой частоты вращения электродвигателя. Первые три вывода захвата каждого таймера можно связать с каналом 1 через логический элемент "исключающее ИЛИ". В этом случае, по мере вращения двигателя и прохождения возле каждого датчика, в канале будет генерироваться событие захвата. Это приведет к копированию текущего состояния таймера в регистр захвата канала, а также к сбросу таймера. Таким образом, значение счетчика, которое окажется в регистре захвата, можно пересчитать в частоту вращения электродвигателя.

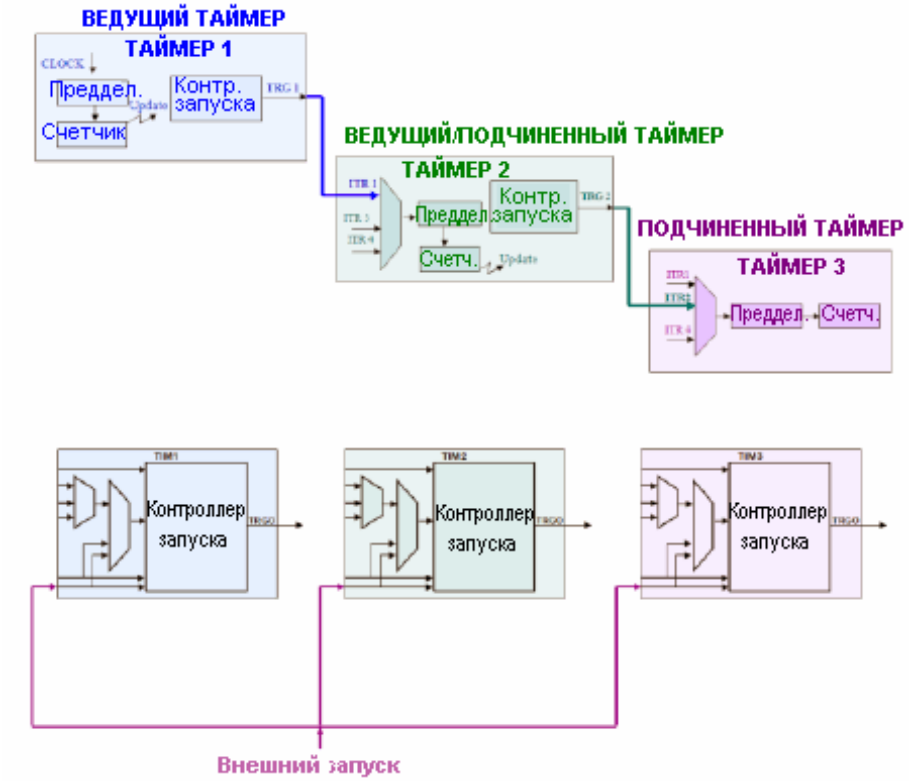
5.1.4.3. Синхронизированная работа таймеров

И хотя каждый из блоков таймеров полностью независим друг от друга, у них предусмотрена возможность синхронизированной друг с другом работы. Это позволяет создавать более сложные массивы таймеров, которые работают полностью автономно и не требуют написания какого-либо кода программы для выполнения сложных временных функций.



У каждого таймера имеются входы запуска от других трех таймеров, а также внешние входы, связанные с выводами блоков захвата/сравнения

У каждого из блоков таймеров имеется выход запуска, который соединен с входами других трех таймеров. Дополнительно, выводы входов захвата таймеров 1 и 2 (TI1FP1 и TI2FP2) соединены с контроллером запуска каждого блока таймеров. Синхронизация таймеров возможна в нескольких различных режимах. Ниже показаны примеры нескольких типичных конфигураций.



Работу всех таймеров можно организовать каскадно с широкими возможностями конфигурации

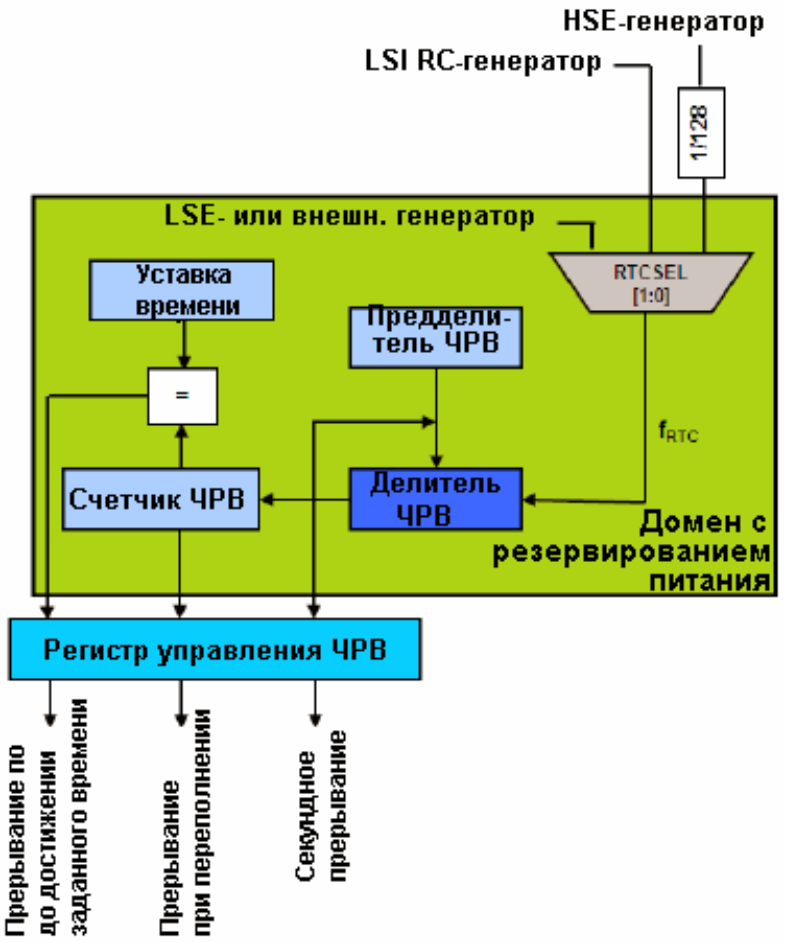
для создания сложных массивов таймеров

Один таймер является ведущим, а другие два - подчиненными. Ведущий таймер генерирует синхронизацию для двух подчиненных таймеров, образуя один большой таймер. Альтернативно, он может генерировать временную задержку запуска двух подчиненных таймеров. Аналогичным образом, активностью каждого таймера может управлять внешний сигнал запуска.

5.1.5. Часы реального времени и регистры с резервированием питания

У МК STM32 имеется два домена питания: основной домен питания системы и УВВ, и домен с резервированием питания. В последнем домене расположены 16-битные регистры, часы реального времени и независимый сторожевой таймер.
Расположенные в этом домене регистры - это обычные 10 ячеек памяти, которые можно использовать для хранения критической информации, когда МК STM32 находится в дежурном режиме работы и основное питание отключено. В экономичных режимах работы МК остаются в работе часы реального времени и независимый сторожевой таймер и, поэтому, их можно использовать для возобновления нормальной работы системы STM32 или для выполнения сброса всего МК.

Входящие в МК STM32 часы реального времени представляют собой 32-битный счетчик, оптимизированный под счет секунд при синхронизации частотой 32.768 кГц. Во время настройки системы синхронизации, в качестве источника синхронизации часов реального времени можно выбрать внутренний низкочастотный генератор, внешний низкочастотный генератор или внешний высокочастотный генератор. После выбора источник синхронизации соединяется с часами реального времени через делитель частоты с фиксированным коэффициентом деления (128). Для организации точного счета секунд далее может быть задействован еще один предделитель ЧРВ. Счетчик ЧРВ может генерировать три прерывания: при инкрементировании секунд, переполнении счетчика и срабатывании функции сигнализации. Последнее прерывание генерируется в случае, когда текущее значение счетчика ЧРВ достигнет значения, хранящегося в регистре уставки времени.



Часы реального времени могут синхронизироваться внутренним или внешним низкочастотным генератором. С их помощью можно получить счетчик секунд с возможностью сигнализации о достижении заданного времени

ЧРВ расположены в домене с резервированием питания, который получает питание от напряжения VBAT, а прерывание, сигнализирующее о достижении заданного времени, связано с 17-ой линией внешних прерываний. Это означает, что после перевода основного домена питания STM32 в экономичный режим работы. ЧРВ останутся в работе. Кроме того, воздействуя на линию внешнего прерывания, ЧРВ могут сигнализировать NVIC Cortex о необходимости возобновления работы основного домена питания STM32. Данная конфигурация очень важна для маломощных устройств, которые в целях сбережения энергии батарейного источника большую часть времени проводят в дежурном режиме, но при этом должны иметь возможность, при необходимости, возобновить активную работу.

5.1.6. Регистры с резервированием питания и вход вмешательства

В домене с резервированием питанием также имеется десять 16-битных регистров,

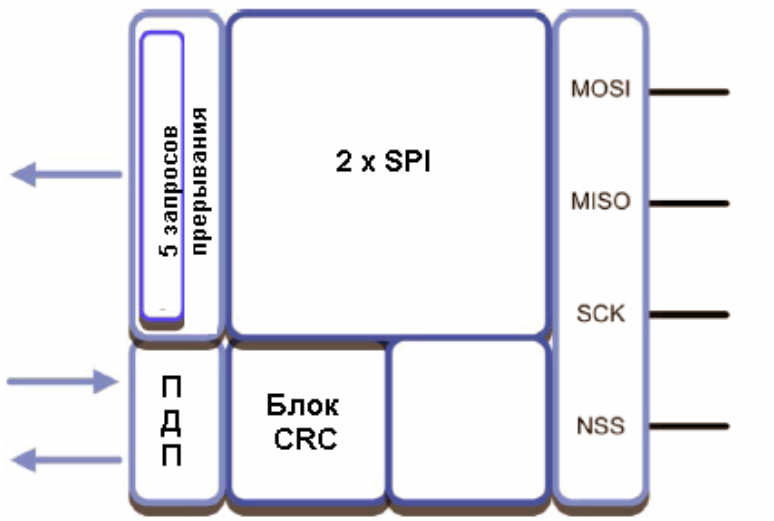
которые функционируют как энергонезависимое статическое ОЗУ. Хранящиеся в них данные можно стереть путем выполнения записи в соответствующий регистр управления. Этот же регистр управляет активностью внешнего входа вмешательства. Состояние данного вывода (высокое или низкое) можно настроить во время запуска. В ходе нормальной работы изменение логического уровня на этом входе приведет к запуску события обнаружения вмешательства, что вызовет очистку регистров с резервированием питания. Также можно активировать прерывание, которое позволяет МК выполнить защитные действия при обнаружении вмешательства.

5.2. Коммуникационные УВВ

В домене с резервированием питанием также имеется десять 16-битных регистров, которые функционируют как энергонезависимое статическое ОЗУ. Хранящиеся в них данные можно стереть путем выполнения записи в соответствующий регистр управления. Этот же регистр управляет активностью внешнего входа вмешательства. Состояние данного вывода (высокое или низкое) можно настроить во время запуска. В ходе нормальной работы изменение логического уровня на этом входе приведет к запуску события обнаружения вмешательства, что вызовет очистку регистров с резервированием питания. Также можно активировать прерывание, которое позволяет МК выполнить защитные действия при обнаружении вмешательства.

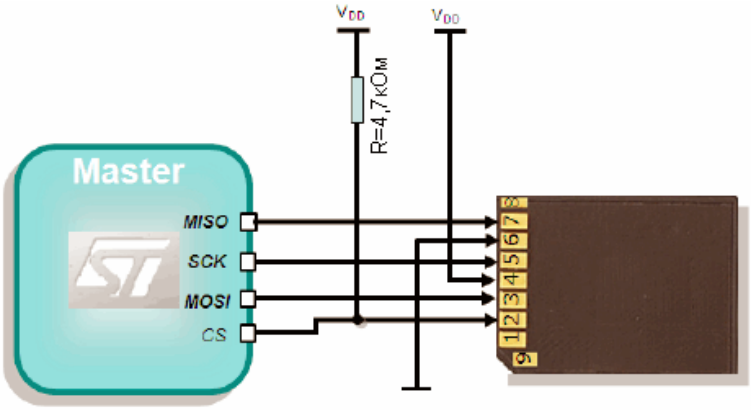
5.2.1. Интерфейс SPI

Для организации быстродействующей связи с интегральными схемами у МК STM32 имеется два модуля SPI, предназначенных для полнодуплексной передачи данных на частоте до 18МГц. Важно обратить внимание, что один модуль SPI подключен высокоскоростной шине УВВ APB2, которая может работать на частоте до 72МГц. Второй модуль связан с более низкоскоростной шиной APB1, максимальное быстродействие которой ограничивается частотой 36 МГц. У каждого из модулей SPI предусмотрена возможность программирования полярности и фазы синхронизации и формата передаваемых данных (8 или 16 бит, передача первым старшего или младшего значащего разряда). Кроме того, каждый модуль SPI может работать в ведущем или подчиненном режиме, что позволяет ему связаться с любой другой ИС, оснащенной интерфейсом SPI.



Каждый модуль SPI может работать в ведущем или подчиненном режиме на частоте до 18МГц. Для повышения эффективности передачи данных можно использовать два канала ПДП

Для передачи данных на больших скоростях у каждого модуля SPI предусмотрено два канала ПДП: один для передачи данных и один для копирования принятых данных в память. С помощью ПДП можно добиться автономной двунаправленной передачи высокоскоростных потоков данных. В дополнение к стандартным возможностям интерфейса SPI, модуль SPI микроконтроллеров STM32 содержит два аппаратных блока CRC. Один блок CRC используется для передачи данных, а другой - для приема. Оба блока могут генерировать и проверять коды CRC8 и CRC16. Данная возможность на практике необходима при использовании одного из модулей SPI для подключения к карте MMC/SD.



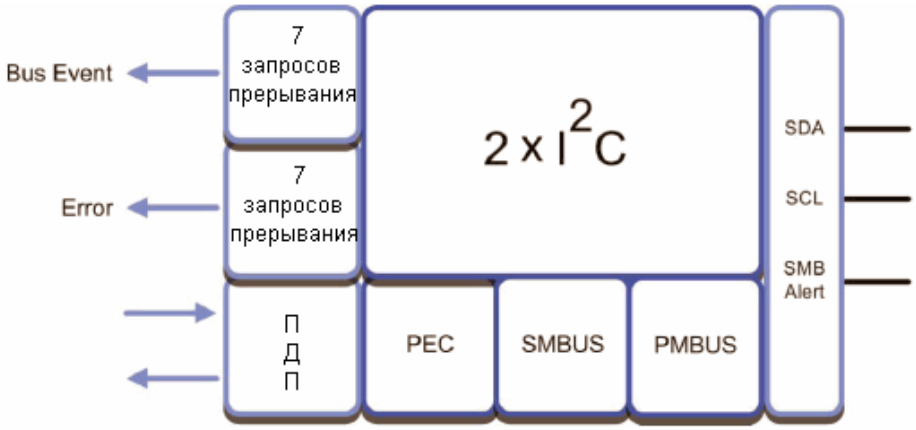
Модуль SPI содержит аппаратный блок CRC, необходимый для подключения к картам Flash памяти типа MMC и SD

5.2.2. Модуль I2C

Для связи с интегральными схемами у МК STM32 имеется еще один специальный интерфейс - I2C. Интерфейс I2C может работать в ведущем или подчиненном режиме и поддерживает возможность арбитра шины, что необходимо в мультимастерных системах. Интерфейс I2C поддерживает оба скоростных режима шины: стандартный со скоростями

до 100 кГц и быстродействующий со скоростями до 400 кГц.

Модуль I2C использует 7- и 10-битные режимы адресации. Модуль полностью реализует протокол передачи данных по шине и требует для управления только необходимой протоколу передачи информации. Модуль I2C может генерировать два прерывания: одно при обнаружении ошибок и другое для управления адресом связи и передаваемыми данными. Кроме того, блок ПДП предоставляет два канала, которые можно использовать для чтения из буфера передачи и записи данных в этот буфер. Таким образом, сразу после задания адреса и подлежащих передаче данных можно начать двунаправленную передачу данных полностью под аппаратным управлением.



Два модуля I2C дополнены возможностями, которые делают их совместимыми с шинами SMBus и PMBus. В них входит аппаратный блок коррекции ошибок

Благодаря поддержке всех перечисленных возможностей, модули I2C являются скоростными и эффективными шинными интерфейсами. Однако у них реализован ряд дополнительных возможностей, позволяющих расширить базовые функции шины I2C. В модуль I2C микроконтроллеров STM32 входит блок аппаратной проверки ошибок в пакете (блок PEC). После активизации, блок PEC будет генерировать байт с CRC-кодом для проверки ошибок.

Данный байт автоматически помещается в конец передаваемого потока данных. Блок PEC также имеет возможность проверки принятых данных на соответствие переданному байту защиты от ошибок.

Данный байт автоматически помещается в конец передаваемого потока данных. Блок PEC также имеет возможность проверки принятых данных на соответствие переданному

байту защиты от ошибок.

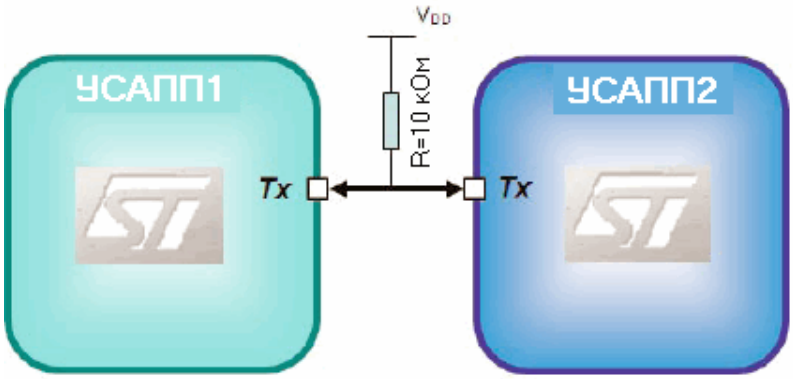
Модуль I2C МК STM32 также поддерживает два дополнительных коммуникационных протокола: SMBus и PMBus. Протокол SMBus был предложен Intel в 1995 году для использования внутри ПК и серверов. Протоколом SMBus оговариваются требования к канальному слою, в т.ч. использование PEC и передача стандартизованных конфигурационных данных между BIOS компьютера и ИС различных производителей. Работая в режиме SMBus, модуль I2C, помимо PEC, поддерживает ряд других возможностей протокола SMBus. К их числу относятся протокол разрешения адреса SMN, протокол уведомления host-устройства и работа с сигналом SMBALERT. Протокол PMBus является разновидностью SMBus и предназначен для работы в системах электропитания. PMBus позволяет конфигурировать, программировать и контролировать в реальном времени системы электропитания.

5.2.3. Модуль USART

Несмотря на то, что порты последовательной связи уже практически не используются в ПК, они все еще остаются популярными во многих встраиваемых применениях для организации простого интерфейса последовательной связи. Его высокая популярность обусловлена свойственной ему надежностью работы и простотой использования. В МК STM32 интегрируется до 3 модулей USART, каждый из которых поддерживает несколько расширенных режимов работы, позволяющие использовать МК в самых современных коммуникационных применениях. Все три USART способны передавать данные на скорости до 4.5 Мбит/сек. Каждый из них также полностью программируется, в т.ч. размер передаваемых данных (8 или 9 бит), передача бита паритета или стоп-бита, а также скорость передачи. Один USART подключен к шине APB2, которая способна синхронизироваться частотой до 72 МГц. а остальные связаны с 36-мегагерцевой шиной APB1.

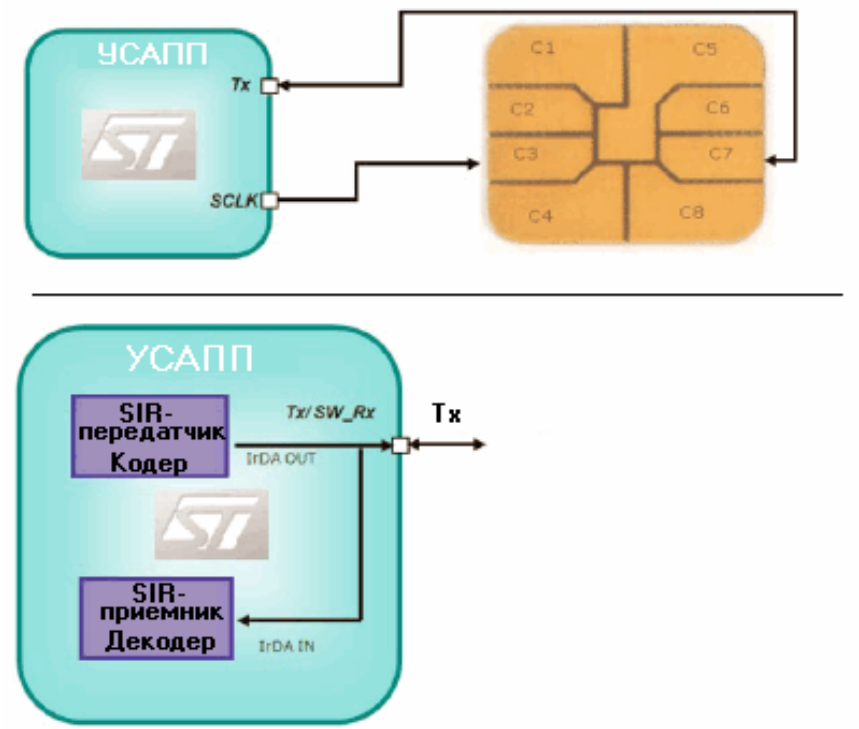
USART можно использовать для асинхронной связи с УАПП и модемами, а также с интерфейсами LIN, IrDA и смарт-картами

Каждый USART имеет собственный генератор скорости связи с возможностями дробного деления частоты. В отличие от обычных делителей частоты, такой генератор позволяет получить стандартные скорости связи при любой частоте синхронизации шины. Также как и остальные модули последовательных интерфейсов, каждый модуль USART оснащен двумя каналами ПДП для двунаправленной связи с буфером данных. В конфигурации УАПП, модуль USART может работать в нескольких режимах работы. USART имеет возможность работы с однопроводной полудуплексной линией, используя для этого только вывод Tx. Для связи с модемами, а также для аппаратного управления передачей потока данных, у каждого USART предусмотрены дополнительные линии управления CTS и RTS.



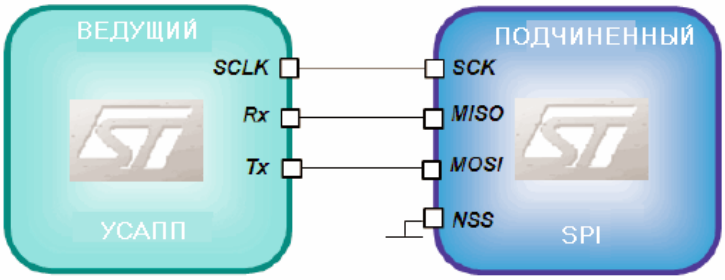
Модули USART поддерживают возможность работы с однопроводной полудуплексной линией связи

Каждый из USART также может использоваться для подключения к шине LIN. Данная шина используется в автомобильном транспорте для низкоскоростной связи с кластерными микроконтроллерами. Все USART поддерживают возможности кодера/декодера инфракрасной последовательной связи (SIR). Эти возможности отвечают стандарту IrDA по инфракрасной передаче данных на скоростях до 115200 бит/сек с использованием полудуплексной NRZ-модуляции и экономичной работой при синхронизации USART частотами в пределах 1.4..2.2МГц. У каждого USART поддерживается дополнительный режим смарт-карты, в котором активизируется поддержка стандарта ISO 7618-3.



USART могут использоваться в качестве интерфейса смарт-карта или IrDA

Помимо работы в роли высокоскоростного интерфейса УАПП, каждый USART может быть переведен в режим синхронной связи. Его можно использовать для связи с внешними УВВ, оснащенными SPI-совместимым интерфейсом, по 3-проводной линии. Работая в данном режиме, USART работает в роли ведущего шины SPI и поддерживает возможность программирования полярности и фазы синхронизации. Благодаря этому, возможна связь с практически любой подчиненной SPI ИС.



В синхронном режиме USART можно использовать в роли дополнительных ведущих интерфейсов SPI

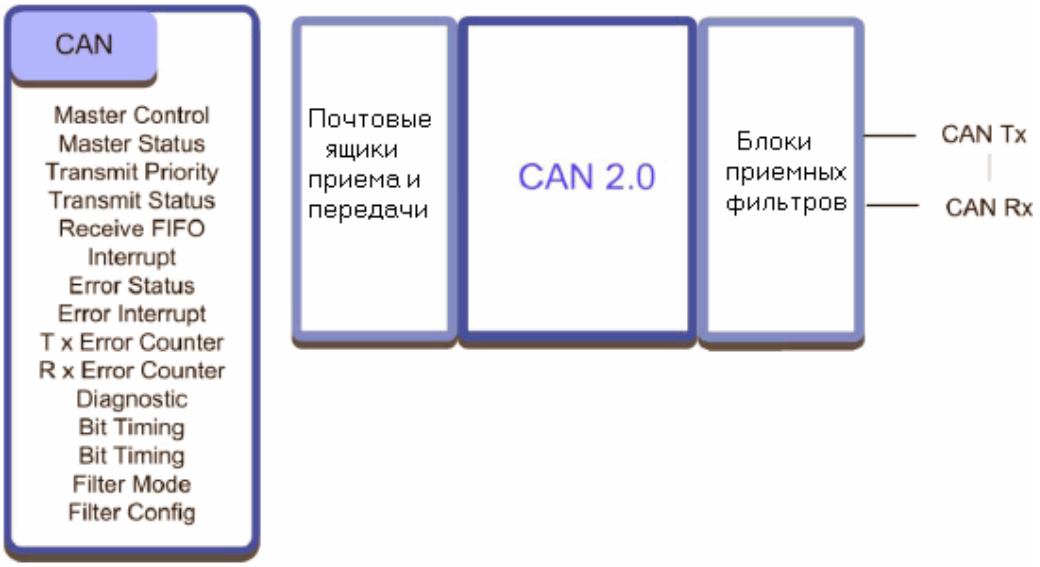
5.3. Модули CAN и USB

Два оставшихся коммуникационных модуля микроконтроллеров STM32 - CAN-контроллер и интерфейс полноскоростного устройства USB. Оба этих протокола связи достаточно сложны, поэтому, если вы впервые сталкиваетесь с ними, то дополнительно необходимо ознакомится с отдельными руководствами по интерфейсам

CAN и USB. Оба модуля, USB и CAN, требуют выделения сравнительно большого объема статического ОЗУ для организации буферов сообщений, поэтому, у МК STM32 имеется дополнительная область статического ОЗУ размером 512 байт, которая может использоваться модулями CAN и USB. Доступ к этой памяти имеют только названные модули. Кроме того, данную память можно назначить на работу только с одним из этих модулей. Это означает, что модули CAN и USB не могут работать одновременно. Если же в применении использование обеих интерфейсов обязательно, то их работу нужно чередовать.

5.3.1. CAN-контроллер

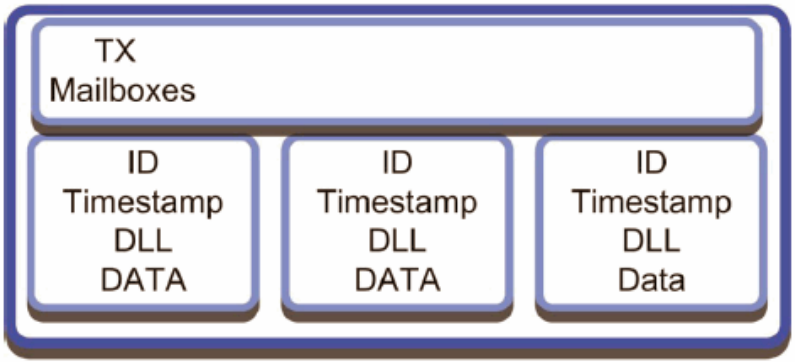
Входящий в МК STM32 CAN-контроллер является полнофункциональным CAN-узлом, отвечающий требованиям к активным и пассивным устройствам CAB 2.0A и 2.0B и поддерживающий передачу данных на скорости не более 1 Мбит/сек. CAN-контроллер оснащен также дополнительными возможностями для организации детерминистической передачи данных по специальному CAN-протоколу передачи в реальном времени TTCAN. После активизации функции TTCAN будет поддерживаться автоматическая повторная передача сообщений и автоматическая вставка в CAN-пакет двух дополнительных байт с зафиксированным моментом времени передачи сообщения. Все эти возможности необходимы в системах управления через CAN-интерфейс в масштабе реального времени.



Модуль CAN отвечает стандарту CAN 2.0B и поддерживает протокол TTCAN

Полное наименование CAN-контроллера - модуль bxCAN, где bx указывает на поддержку модулем дополнительных возможностей. Обычный модуль CAN использует один буфер приема и передачи, а у расширенного модуля CAN используется несколько буферов приема и передачи. Модуль bxCAN является гибридом двух архитектур модулей CAN. У него имеется три почтовых ящика для передаваемых сообщений и два почтовых ящика для принимаемых сообщений. Каждый из принимающих почтовых ящиков имеет буфер FIFO для помещения в него трех сообщений. Данная архитектура является

компромиссной с точки зрения производительности передачи данных и занимаемого места в кристалле ИС.



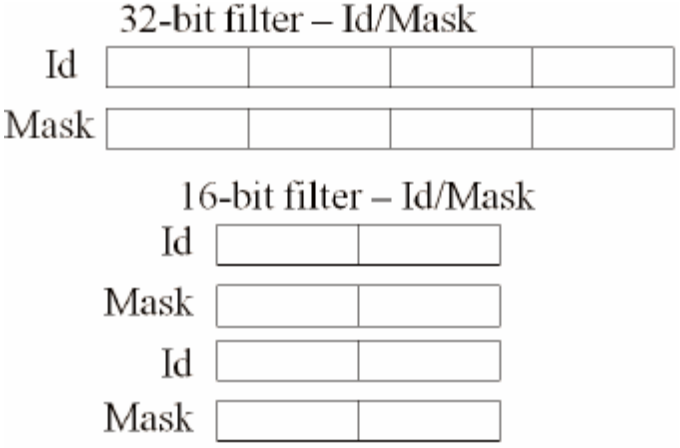
Модуль CAN оснащен тремя почтовыми ящиками для передачи сообщений и имеет возможность автоматической вставки в сообщение текущего времени по протоколу TTCAN

Следующая важная функция CAN-контроллера - фильтрация получаемых сообщений. Поскольку CAN является широковещательной шиной, каждое переданное сообщение принимается всеми узлами шины. В CAN-шине любой разумной степени сложности передается достаточно большое число сообщений. Задачей каждого подключенного к CAN-узлу ЦПУ является реагирование на CAN-сообщения. Таким образом, чтобы избавить CAN-контроллер от проблемы приема в буфер нежелательных сообщений, необходима их фильтрация. У CAN-контроллера микроконтроллеров STM32 имеется 14 банков фильтров, которые можно использовать для блокировки всех CAN-сообщений, кроме избранных сообщений или групп сообщений.

14 фильтров сообщений поддерживают две конфигурации, которые можно использовать для фильтрации индивидуальных сообщений

Каждый банк фильтров состоит из двух 32-битных регистров и может работать в одном из четырех режимов. При использовании базового метода в каждый регистр банка фильтров записывается идентификатор сообщения. После поступления сообщения проверяется его идентификатор и, исходя из этого, принимается решение о приеме или отклонении сообщения. Данный режим поддерживает две конфигурации. В первой конфигурации регистры банков фильтров являются 3-битными и могут использоваться для фильтрации 11- и 29-битных полей идентификаторов сообщения, а также бит RTR и

IDE в 16-битном режиме.



Одни и те же банки фильтров могут использоваться для фильтрации группы сообщений

Во второй конфигурации, в первый 32-битный регистр записывается идентификатор сообщения, во второй - маска сообщения. Регистр маски маркирует биты регистра идентификатора, как 'важный' или 'неважный'. Благодаря этому, появляется возможность принимать группу сообщений с помощью одного банка фильтров. Если принимающие фильтры пропускают сообщение, то вместе с ним принимающий буфер FIFO будет записан указатель на определивший совпадение фильтр. Это позволит прикладной программе ускорить идентификацию сообщения без необходимости считывания и дешифрации идентификатора пакета сообщения.

Все CAN-контроллеры поддерживают два режима работы: нормальный режим для приема и передачи пакетов сообщений и режим инициализации для задания параметров связи. Как уже говорилось, МК STM32 могут работать в экономичном режиме SLEEP. В этом режиме синхронизация модуля bxCAN отключена, однако доступ к регистрам почтовых ящиков остается возможным. Модуль bxCAN имеет возможность активизации работы при обнаружении активности на шине CAN. Его работу можно также реактивировать прикладной программой. Работая в нормальном режиме, поддерживаются два дополнительных подрежима. Первый подрежим - режим SILENT. В нём CAN-контроллер может принимать сообщения, но не может передавать и не генерирует бит ошибок в посылке и подтверждения сообщения. Данный режим рассчитан на CAN-шины с пассивным мониторингом. Второй подрежим - режим LOOPBACK. В этом режиме, передаваемые сообщения сразу же принимаются в приемный буфер. Он необходим для реализации диагностических функций и также полезен на фазе отладки кода программы. Оба рассмотренных режима можно комбинировать. Они идеальны для выполнения функций самотестирования при подключении к работающей шине.

5.3.2. Модуль интерфейса USB

Модуль интерфейса USB, входящий в состав некоторых МК STM32, является

полноскоростным (12 Мбит/сек) и предназначен для взаимодействия с хост-интерфейсом USB, как например, используемым в ПК. Данный модуль полностью реализует физический уровень и уровень передачи данных, в т.ч. проверка ошибок в пакетах и повторная передача. Интерфейс USB-устройства также поддерживает возможности приостановки и возобновления, необходимые для снижения потребляемой мощности. Разрабатывая устройство с интерфейсом USB, необходимо хорошо разбираться в технических характеристиках интерфейса USB и его прикладных классах. На сайте компании ST доступен полный набор разработчика USB устройств. В него входит программный стек для инициализации интерфейса USB и реализации наиболее популярных USB-классов, в т.ч. HID, MASS STORAGE, AUDIO и LEGACY COMMUNICATIONS PORT. Использование данного стека или любого другого аналогичного программного стека других компаний позволит существенно ускорить процесс проектирования.

Модуль USB отвечает требованиям к USB-устройству версии 2.0

Интерфейс USB поддерживает до 8 конечных точек, которые пользователь может настроить как конечная точка управления, прерывания, потока или изохронная. Для организации буфера конечной точки может использоваться статическое ОЗУ размером 512 байт, использование которого совмещено еще и с CAN-контроллером. По завершении инициализации модуля USB, прикладная программа разделяет данное статическое ОЗУ на последовательность буферов.

Конфигурация статического ОЗУ в качестве буферов конечной точки выполняется с помощью таблицы определений буферов, которая хранится в основном статическом ОЗУ. В ней для каждой конечной точки указывается начальный адрес в статическом ОЗУ и значение, указывающее на ее размер. Каждая активная конечная точка управления, прерывания и потока связана с буфером пакета конечной точки, а изохронная конечная точка - с двойным буфером. Благодаря этому, данные можно принимать в один буфер и, при этом, выполнять обработку данных из другого. После получения очередного пакета, новые данные поступают во второй буфер, а обрабатываются данные из первого буфера. Метод двойного буфера позволяет передавать реально-временные потоки данных, как например, аудиопоток.

Для хранения пакетов данных используется 512-байтное статическое ОЗУ. Во время инициализации, эта область памяти разделяется на несколько буферов для каждой активной конечной точки. Изохронные конечные точки используют специальный

двойной буфер, который позволяет во время считывания данных из одной части буфера продолжать прием в другую часть.

6. Экономичные режимы работы

МК STM32, будучи высококачественными микроконтроллерами, в дополнение к работе в обычном режиме RUN поддерживают возможность работы в нескольких экономичных режимах. Правильное использование режимов SLEEP, STOP и STANDBY делает возможным реализацию практичных решений с батарейным питанием. МК STM32 объединяет два противоречивых свойства: малое энергопотребление и высокую производительность. При знакомстве с ядром Cortex уже говорилось, что после перехода в экономичный режим работы ЦПУ и УВВ приостанавливают свою работу и потребляют минимальную мощность. При переходе процессора Cortex в экономичный режим работы, он может сигнализировать об этом еще одному внешнему микроконтроллеру генерацией сигнала SLEEPDEEP. Для перехода в экономичный режим работы необходимо выполнить инструкцию WFI или WFE. После выполнения такой инструкции будет введен экономичный режим работы, указанный в регистре управления энергопотреблением. Далее будет дан обзор каждого экономичного режима работы и их сравнение по показателям потребляемого тока и задержки возобновления работы.

6.1. Режим RUN

В режиме RUN микроконтроллер STM32 исполняет код программы, поэтому, энергопотребление максимально.

В данном разделе будут рассмотрены различные способы снижения результирующего энергопотребления в ходе выполнения кода программы. Важно запомнить, что все данные возможности можно использовать динамически. Это означает, что имеется возможность выполнять код программы в маломощной, низкопроизводительной конфигурации, а затем, в ответ на прерывание или программное событие переключится к более мощной и высокопроизводительной конфигурации.

При обычном использовании, процессор Cortex и большинство УВВ STM32 могут работать на частоте 72 МГц. Работая с максимальным быстродействием, МК потребляет ток более 30 мА. Первым способом его снижения является отключение синхронизации всех неиспользуемых УВВ. Это позволит вычесть энергопотребление всех неиспользуемых частей микроконтроллера. Включение и отключение синхронизации УВВ можно осуществлять динамически через модуль управления сбросом и синхронизации (RCC).

Еще большего снижения энергопотребления можно добиться снижением частоты системной синхронизации. Если работа на высокой частоте не обязательна, блок ФАПЧ

можно отключить и МК STM32 будет синхронизироваться непосредственно с выхода HSE-генератора. Дальнейшего снижения потребления можно достигнуть переходом с использования HSE- на HSI-генератор. Однако, по сравнению с HSE-, HSI-генератор обладает существенным недостатком - он не столь точен. Еще некоторую часть энергопотребления можно исключить отключением LSI-генератора, если не используется оконный сторожевой таймер и часы реального времени.

6.1.1. Буфер предварительной выборки и режим полуцикла

При работе напрямую от HSE-генератора на максимальной частоте 8МГц, также можно отключить буфер предварительной выборки Flash памяти и активировать режим полуцикла. Вследствие этого, вводятся дополнительные состояния ожидания, но при этом, снижается потребляемый ток в режиме RUN.

При работе с максимальным быстродействием потребляемый ток составляет в районе 34 мА, а при работе на частоте 8 МГц (9.6 DMIPS) он становится менее 1 мА

APB1	APB2	УВВ	Частота	Предварительная выборка	Режим Полуцикл	WFI	Генератор	Типичное потребление при 25°C [мА]
DIV4	DIV2	Все вкл.	72 МГц	Вкл.	Откл.	Откл.	HSE	33.15
DIV8	DIV8	USART	72 МГц	Вкл.	Откл.	Откл.	HSE	27.75
DIV8	DIV8	USART	72 МГц	Вкл.	Откл.	Откл.	HSE	23.65
DIV4	DIV2	USART	8 МГц	Вкл.	Откл.	Откл.	HSE	8.65
DIV4	DIV2	USART	8 МГц	Откл.	Откл.	Откл.	HSE	8.48
DIV4	DIV2	USART	8 МГц	Откл.	Откл.	Вкл.	HSE	1.68
DIV4	DIV2	USART	8 МГц	Откл.	Откл.	Вкл.	HSI	0.9

6.2. Экономичные режимы работы

Тщательно сконфигурировав работу МК в режиме RUN, можно снизить потребляемый ток до приблизительно 8.5 мА. Однако, чтобы реализовать по-настоящему маломощное устройство, необходимо использовать экономичные режимы работы МК STM32.

6.2.1. Режим SLEEP

Первая ступень экономичной работы - режим SLEEP. По умолчанию, после выполнения

процессором Cortex инструкции WFE или WFI, отключается внутренняя синхронизация и прекращается выполнение кода программы. В режиме SLEEP, остальная часть МК STM32 продолжает работу. Выход из режима SLEEP происходит, когда УВВ генерирует прерывание. Если МК STM32 синхронизируется блоком ФАПЧ и HSE-генератором частотой 72 МГц и использует все УВВ, то при переходе в режим SLEEP потребляемый ток снизится до приблизительно 14.4мА. Тем не менее, если выполнить специальную подготовку МК STM32 к переходу в экономичный режим, отключив синхронизацию всех УВВ, кроме тех, что используются для возобновления работы процессора Cortex, и переключившись на синхронизацию от HSI-генератора (частоту которого можно снизить до 1 МГц и даже менее), можно добиться снижения потребляемого тока до приблизительно 0.5 мА.

Потребляемый ток в режиме SLEEP можно снизить до 0.14 мА:

Условия	fHCLK	Все УВВ на шине APB включены	Все УВВ отключены	Ед. изм.
Синхронизация от HSE, для снижения частоты используется предделитель AHB	72 МГц	14.4	5.5	мА
	48 МГц	9.9	3.9	
	36 МГц	7.6	3.1	
	24 МГц	5.3	2.3	
	16 МГц	3.8	1.8	
	8 МГц	2.1	1.2	
	4 МГц	1.6	1.1	
	2 МГц	1.3	1	
	1 МГц	1.11	0.98	
	500 кГц	1.04	0.96	
	125 кГц	0.98	0.95	
Синхронизация от внутреннего НЧ RC генератора (HSI), для снижения частоты используется предделитель AHB	64 МГц	12.3	4.4	
	48 МГц	9.3	3.3	
	36 МГц	7	2.5	
	24 МГц	4.8	1.8	
	16 МГц	3.2	1.2	
	8 МГц	1.6	0.6	

	4 МГц	1	0.5	
	2 МГц	0.72	0.47	
	1 МГц	0.56	0.44	
	500 кГц	0.49	0.42	
	125 кГц	0.43	0.41	

В маломощных применениях, чтобы добиться минимального энергопотребления, в режим SLEEP необходимо переходить настолько часто, насколько это возможно. На результирующий уровень энергопотребления также влияет задержка, которая необходима микроконтроллеру STM32 для выхода из экономичного режима работы и восстановления исполнения кода программы. Не представленных ниже рисунках будет показано, какие задержки необходимы ЦПУ Cortex, который синхронизируется от HSI RC-генератора, для возобновления нормальной работы.

6.2.2. Режим STOP

Микроконтроллер можно настроить на переход в экономичный режим STOP. Для этого необходимо установить бит SLEEPDEEP в регистре управления энергопотреблением ядра Cortex и сбросить бит Power Down Deep Sleep (PDDS) в регистре управления энергопотреблением МК STM32.

После завершения конфигурации режима STOP, выполнение инструкции WFI или WFE приведет к остановке процессора Cortex и отключению HSI- и HSE-генераторов. Флэш-память, статическое ОЗУ и УВВ остаются запитанными, поэтому, состояние МК STM32 сберегается. Также как и в случае с режимом SLEEP, выход из режима STOP возможен путем генерации прерывания УВВ. Однако в режиме STOP синхронизация всех УВВ отключена, за исключением контроллера внешних прерываний. Таким образом, выход из режима STOP возможен при изменении состояния любой из линии ввода-вывода.

Кроме того, у контроллера внешних прерываний имеется одна линия, которая может, как запрашивать, так и генерировать прерывание по достижении заданного времени часами реального времени. Поскольку у часов реального времени имеется отдельный генератор (LSI или LSE), то они могут использоваться для генерации периодических прерываний для вывода МК STM32 из режима STOP.

После перехода МК STM32 в режим STOP его потребляемый ток снижается с миллиампер, потребляемых в режиме RUN, до приблизительно 24 мкА. Дальнейшего снижения энергопотребления можно добиться переводом внутреннего генератора в специальный экономичный режим работы. Для этого необходимо установить бит LPDS в регистре управления энергопотреблением МК STM32. Если при переходе в режим STOP данный бит был установлен, то потребляемый ток снизится до 14 мкА. Если используются часы реального времени, то потребляемый ток увеличится на 1.4 мкА.

Условия	VDD/VBAT=2.4В	VDD/VBAT=3.3В	Ед.изм.
Стабилизатор в режиме Run, низкочастотный и высокочастотный внутренние RC генераторы, а также высокочастотный генератор отключены (работа без независимого сторожевого таймера)	NA	24	мкА
Стабилизатор в экономичном режиме, низкочастотный и высокочастотный внутренние RC генераторы, а также высокочастотный генератор отключены (работа без независимого сторожевого таймера)	NA	14	

Обозначение	Параметр	Условия измерения	Значение	Ед. изм.
tWUSTOP	Выход из режима STOP (стабилизатор в режиме RUN)	Возобновление по HSI RC генератору	3.52	мкс
	Выход из режима STOP (стабилизатор в режиме RUN + WFI)		5.42	
	Выход из режима STOP (стабилизатор в экономичном режиме + WFE)		5.32	
	Выход из режима STOP (стабилизатор в экономичном режиме + WFI)		7.21	

Задержка возобновления при выходе из режима STOP в худшем случае составит 5.5 мкс, если стабилизатор оставался в полностью активном состоянии, и 7.3 мкс, если стабилизатор переводился в экономичный режим работы.

6.3. Режим STANDBY

МК STM32 можно настроить на работу в режиме STANDBY, если установить бит SLEEPDEEP в регистре управления энергопотреблением ядра Cortex и установить бит Power Down Deep Sleep в одноименном регистре МК STM32. После этого, выполнение инструкции WFI или WFE приведет к переводу МК STM32 в режим с наименьшим энергопотреблением. В режиме STANDBY МК STM32 абсолютно полностью бездействует. Отключены внутренний стабилизатор напряжения и HSE- и HSI-генераторы. В этом режиме МК STM32 потребляет ток всего лишь 2 мкА.

В режиме STANDBY потребляемый ток равен 2 мкА, а задержка возобновления составляет 50 мкс

Условия измерения	VDD/VBAT=2.4В	VDD/VBAT=3.3В	Ед.изм.
НЧ внутренний генератор и независимый сторожевой таймер отключены, НЧ генератор и часы реального времени отключены	NA	2	мкА
НЧ генератор и часы реального времени включены	1.08	1.4	

Обозначение	Параметр	Условия измерения	Значение	Единица измерения
tWUSTDBY	Задержка возобновления для режима STANDBY	Возобновление с синхронизацией HSI RC-генератором	50	мкс

Выход из режима STANDBY возможен по прерыванию часов реального времени (достижение заданного времени) точно также как и при выходе из режима STOP. Кроме того, возобновление возможно через внешний вывод сброса МК STM32 или с помощью независимого сторожевого таймера. Выход из режима STANDBY также возможен по нарастающему фронту на линии 0 порта A. Данный вывод можно настроить, как вывод возобновления WKUP путем установки бита EWUP в регистре управления энергопотреблением и статуса. Поскольку режим STANDBY самый маломощный, то и выход из него осуществляется дольше всего: задержка возобновления исполнения инструкций составляет около 50 мкс. После перехода в режим STANDBY содержимое статического ОЗУ, регистров ядра Cortex и МК STM32 теряется. Выход из режима STANDBY практически идентичен программному сбросу.

6.4. Потребляемый ток области с резервированием питания

В область с резервированием питания входят ОЗУ и часы реального времени. Подача питания на них сохраняется во всех экономичных режимах работы. Их потребляемый ток ток при напряжении питания 3.3В составляет около 1.4 мкА.

6.5. Возможность отладки в экономичных режимах

Отладка традиционных микроконтроллерных систем, в которых используются экономичные режимы работы, может оказаться затрудненной. Это связано с тем, что после перехода в экономичный режим микроконтроллер перестает взаимодействовать с отладчиком. Вследствие этого, отладчик выдает сообщения об ошибке или перестает работать. Во избежание этого, у МК STM32 предусмотрена возможность сохранения в работе HSI-генератора в качестве источника синхронизации встроенной отладочной системы CoreSight даже после перевода МК в экономичный режим работы. Расширенные возможности отладки МК STM32 настраиваются через регистр DBG_MCU.

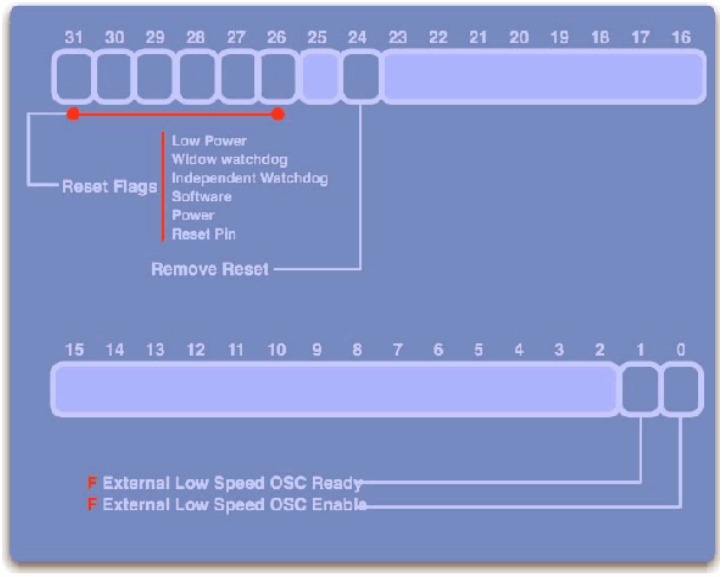
7. Возможности по обеспечению безопасной работы

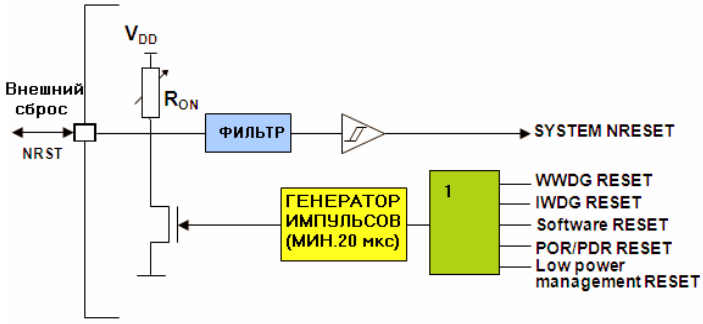
Микроконтроллеры STM32 поддерживают ряд возможностей, направленных на выявление некорректного выполнения кода программы и неправильного поведения микроконтроллера в целом. Чтобы исключить возможность работы микроконтроллера от ненадежного источника питания, у МК STM32 предусмотрена встроенная схема, которая переводит его в состояние сброса, если напряжение VDD будет ниже минимально-допустимого значения. Кроме того, в МК интегрирована программируемая схема контроля напряжения, которая еще раньше позволяет выявить проблемы с питанием. После выявления нарушения питания эта схема генерирует прерывание, позволяющее перевести ИС в безопасное состояние. В структуре системы синхронизации МК предусмотрены элементы контроля HSE-генератора. В случае выявления нарушений в его работе, МК автоматически переключится на работу от HSI-генератора. Корректность выполнения программы можно контролировать с помощью двух встроенных сторожевых таймеров. Один из них - оконный сторожевой таймер, который необходимо обновлять с определенной частотой. Другой - независимый сторожевой таймер, который синхронизируется отдельным генератором, несвязанным с основной системной синхронизацией. Кроме того, встроенная Flash память поддерживает возможность хранения данных в течение 30 лет при температуре 85╟C, что является лучшим в своем классе показателем хранения данных для микроконтроллера общего назначения. Перечисленные возможности по обеспечению безопасной работы МК неприемлемы для использования в оборудовании, к которому предъявляются максимально-высокие требования безопасности (в таком оборудовании элементы, отвечающие за контроль программного обеспечения, как например, сторожевой таймер должны быть отдельными внешними устройствами). Тем не менее, микроконтроллеры STM32 позволяют разрабатывать надежные самокорректирующие системы с использованием способов, применяющихся в критичных к безопасности авиационных и автомобильных системах, но

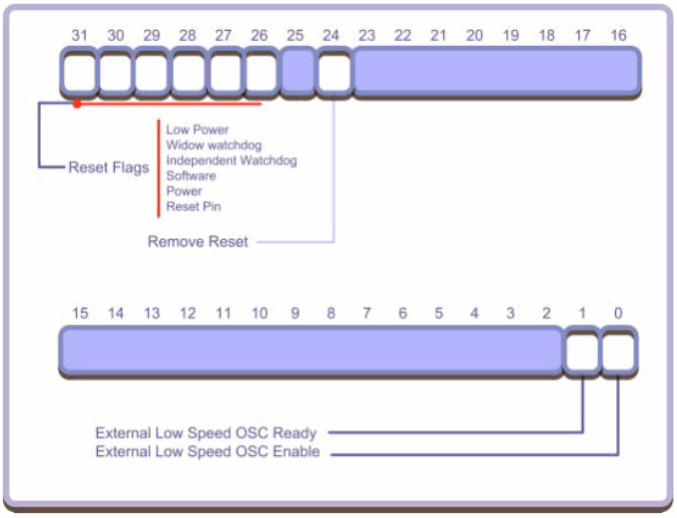
с использованием чрезвычайно простых схемных решений. Благодаря этому, возможно создание гораздо более надежной и качественной недорогой и простой электронной техники.

7.1. Управление сбросом

У МК STM32, помимо внешней линии сброса, имеется множество источников сброса. Сброс МК STM32 может быть выполнен встроенными сторожевыми таймерами, программно через NVIC, встроенными схемами сброса при подаче/отключении и снижении ниже допустимого уровня напряжения питания. В случае генерации сброса устанавливаются соответствующие флаги в регистре управления и статуса RCC, т.о. опросом этих флагов можно определить причину, вызвавшую сброс микроконтроллера. Состояние данных флагов сохраняется до следующего сброса при подаче питания или до записи лог. 1 в бит стирания причины сброса.



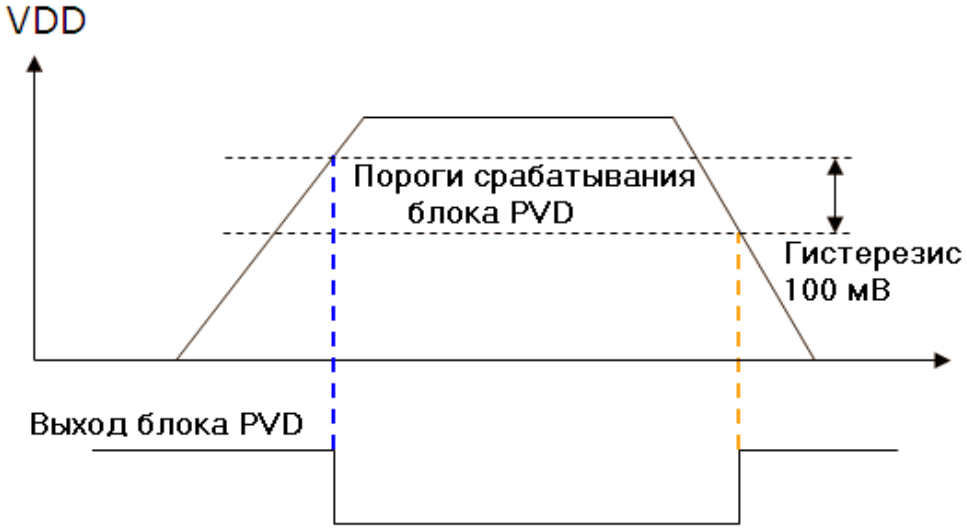




Микроконтроллер STM32 имеет несколько источников сброса. Определить источник сброса можно с помощью регистра управления и статуса RCC

7.2. Контроль напряжения питания

В состав микроконтроллеров STM32 входит специальный блок для мониторинга питания. Он называется блоком контроля напряжения питания (блок PVD). Блок PVD поддерживает возможность программирования порога срабатывания в диапазоне от 2.2 до 2.9В с шагом 0.1В. Этот порог срабатывания задается в регистре управления энергопотреблением.

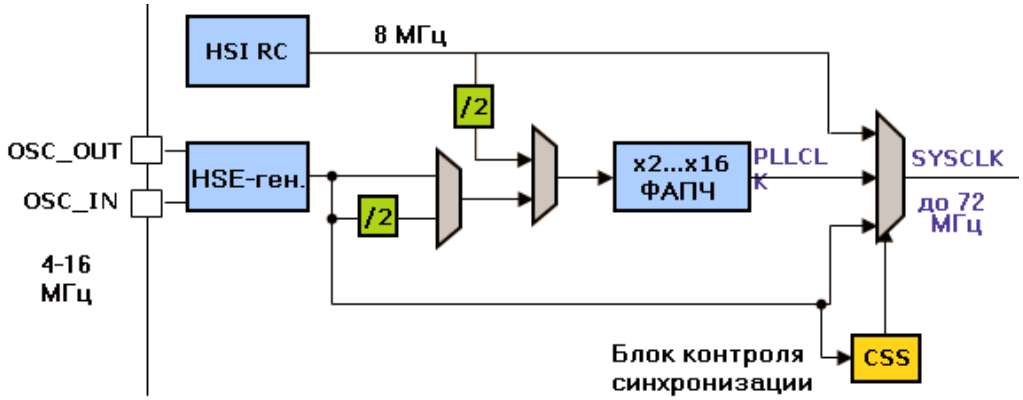


В микроконтроллер входит специальная схема контроля напряжения питания, которая может генерировать прерывание, если напряжение питания снижается ниже заданного порога

Выход блока PVD связан с 16-ой линией блока внешних прерываний. Поскольку линии внешних прерываний могут реагировать на нарастающий, падающий или нарастающий и падающий фронты, то блок PVD можно использовать для генерации прерывания при выполнении условий превышения или понижения напряжения.

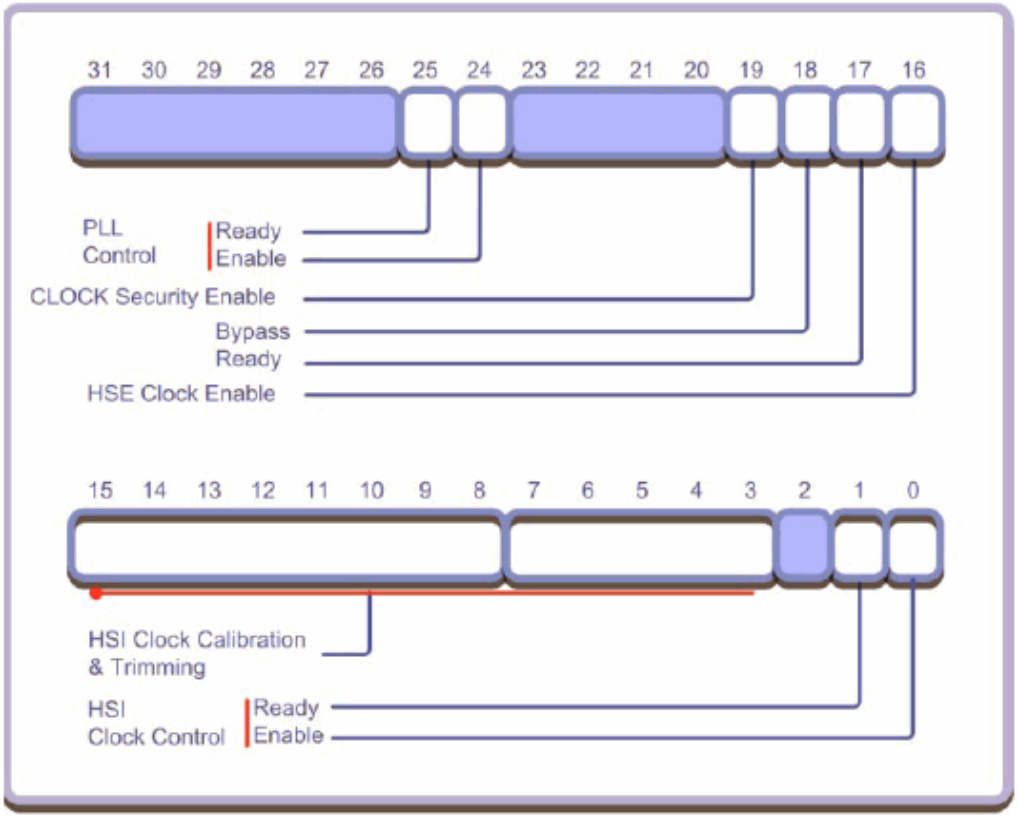
7.3. Защищенная система синхронизации

В большинстве применений МК STM32 в качестве основной системной синхронизации процессора Cortex и УВВ STM32 выступает внешний кварцевый резонатор, подключенный к выводам HSE-генератора. В структуре системы синхронизации предусмотрен специальный блок CSS, который контролирует внешний сигнал. В случае если этот блок обнаружит отказ кварцевого резонатора, МК переключится на аварийную синхронизацию от внутреннего генератора частоты 8 МГц.



В случае отказа внешнего генератора, блок CSS генерирует прерывание и переключается на работу от RC-генератора

Для активизации блока CSS необходимо установить соответствующий бит в регистре управления RCC.



Для включения контроля синхронизации необходимо установить бит разрешения работы CSS в регистре управления RCC

У блока CSS имеется линия прерывания, которая связана с прерыванием по экстренному отключению расширенного таймера 1, в свою очередь соединенной с линией немаскируемого прерывания NVIC Cortex. Этим гарантируется незамедлительный переход ШИМ-выходов таймера в предварительно-запрограммированное безопасное состояние в случае отказа основного генератора.

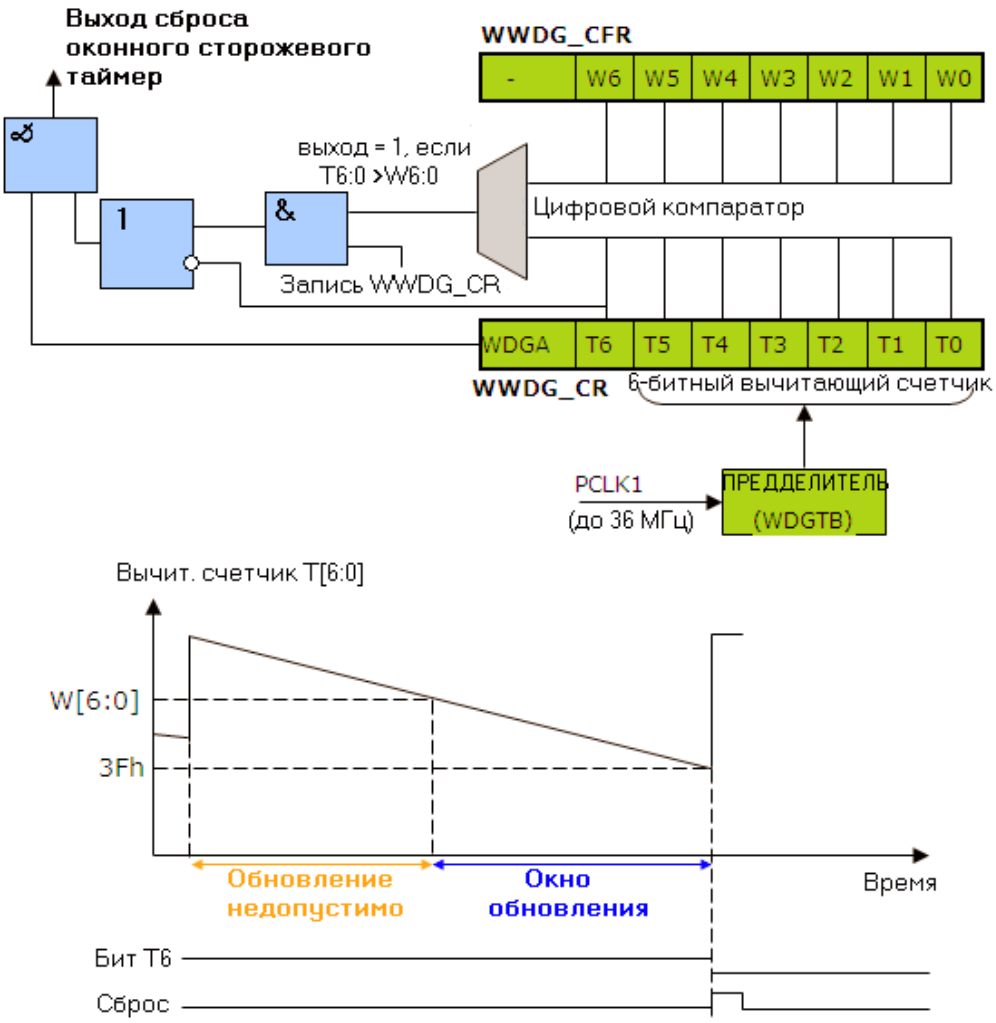
Таким образом, если процессор Cortex теряет управление над ШИМ-выходами, их работа и связанной с ними внешней схемы блокируется. Даная функция особенно важно в устройствах управления электродвигателями.

7.4. Сторожевые таймеры

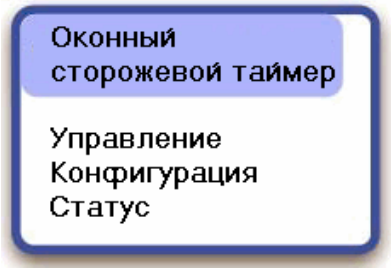
В МК STM32 входят два отдельных сторожевых таймера. Независимый сторожевой таймер полностью отделен от основной системы МК STM32. Он расположен в домене с резервированием питания и синхронизируется встроенным низкочастотным генератором (LSI). Оконный сторожевой таймер является частью основной системы МК STM32 и связан с сигналом синхронизации первой шины УВВ. Оба сторожевых таймера поддерживают возможность раздельного включения/отключения и могут использоваться одновременно.

МК STM32 содержат два сторожевых таймера, один из которых синхронизируется отдельным генератором

7.4.1. Оконный сторожевой таймер



Оконный сторожевой таймер является расширенной версией традиционного встраиваемого сторожевого таймера. После активизации, сторожевой таймер начинает счет в обратном направлении и генерирует сброс при изменении состояния счетчика с 0x40 на 0x3F, т.е. когда сбрасывается бит T6.



Кроме того, в конфигурационном регистре оконного сторожевого таймера предусмотрена возможность задания верхней границы счета. Если во время программного обновления содержимого счетчика сторожевого таймера, его фактическое значение окажется больше заданного, тоже генерируется сброс. Следовательно, сторожевой таймер выделяет программе строго ограниченное время на обновление содержимого счетчика, что позволяет быть уверенным не только в факте выполнения

кода программы и во временных характеристиках его выполнения.

Оконный сторожевой таймер представляет собой 6-битный вычитающий счетчик, который синхронизируется сигналом PCLK1 через 12-битный предделитель (делит частоту PCLK1 на 4096). У предделителя имеется 2 дополнительных бита, которые может запрограммировать пользователь для дальнейшего деления частоты на 1, 2, 4 или 8. Эти биты находятся в 6 и 7 разрядах регистра управления.

Таким образом, период срабатывания оконного сторожевого таймера определяется по выражению:

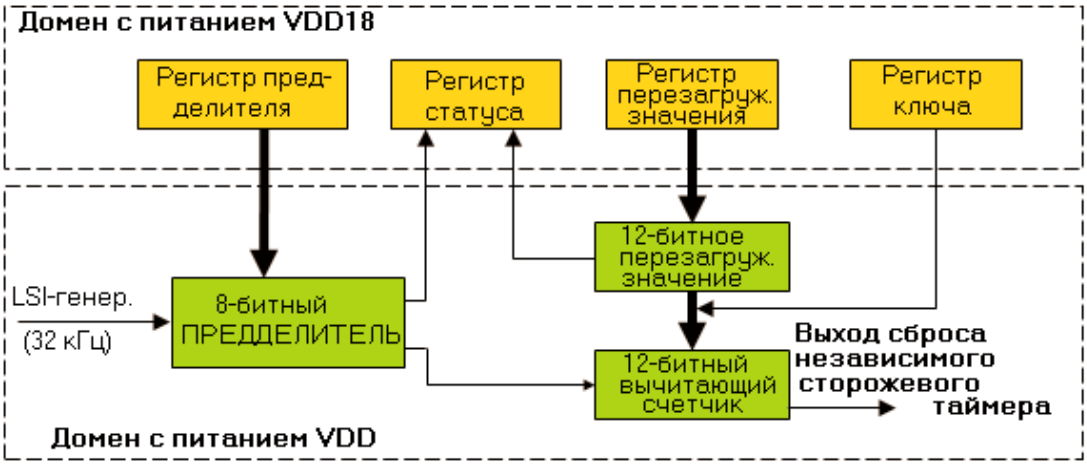
```
Twwdg = Tpclk1 x 4096 x 2^(WDGTB) x (перезагружаемое значение + 1)
```

Если Pclk1 имеет максимальное значение 36 МГц, минимальный период срабатывания сторожевого таймера составит 910 мкс, а максимальный - 58.25 мс.

По завершении настройки оконного сторожевого таймера, его работу можно разрешить установкой соответствующего бита в регистре управления. После разрешения, остановить работу сторожевого таймера можно только сбросом - программно это сделать невозможно.

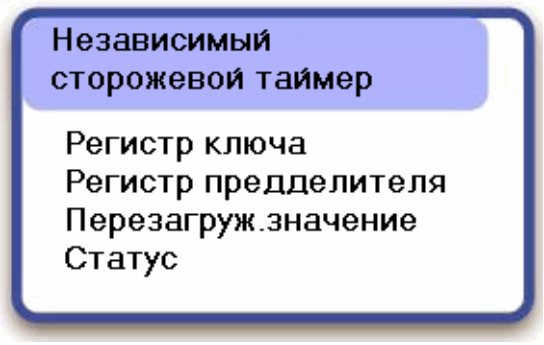
7.4.2. Независимый сторожевой таймер

Несмотря на то, что независимый сторожевой таймер является частью одного и того же кристалла МК STM32, он синхронизируется собственным генератором, который не связан с основной синхронизацией МК STM32. Независимый сторожевой таймер расположен в домене с питанием VDD, которое остается активным в режимах STOP и STANDBY.



Независимый сторожевой таймер представляет собой 12-битный вычитающий счетчик, который генерирует сигнал сброса при переходе через нулевое значение. Счетчик синхронизируется от отдельного внутреннего низкочастотного генератора через 8-битный

предделитель. Номинальная частота LSI-генератора равна 32.768 кГц, но на практике она может варьироваться в пределах 30..60 кГц. Для инициализации сторожевого таймера вначале необходимо настроить регистр предделителя, исходя из того, что коэффициент деления частоты LSI-генератора, который может лежать в пределах 4..256, равен 2 в степени числа, записываемого в регистр предделиетеля значения. Минимальный период срабатывания независимого сторожевого таймера составляет 0.1мс, а максимальный - свыше 26 секунд. Для задания периода срабатывания необходимо запрограммировать регистр перезагрузки.



Независимый сторожевой таймер - вычитающий счетчик с собственным генератором. Он расположен в домене с резервированием питания и. поэтому, остается активным в режимах Stop и Standby

Для автоматической или программной конфигурации независимого сторожевого таймера может быть задействован небольшой информационный блок во Flash памяти, состоящей из нескольких опциональных байт. Для запуска независимого сторожевого таймера при программном управлении необходимо записать 0xCCCC в регистр ключа. После этого, счет начнется в обратном направлении со значения 0xFFF. Для обновления сторожевого таймера в регистр ключа необходимо записать 0xAAAA. Это приведет к записи заданного перезагружаемого значения в регистр вычитающего счетчика.

Обычно, отлаживать программу микроконтроллеру с активным сторожевым таймером очень сложно. Если ЦПУ окажется остановленным, то и обновлять сторожевой таймер будет некому. В итоге, произойдет его срабатывание и будет выполнен сброс микроконтроллера, который нарушит отладочную сессию. Во избежание этого, разработчики обычно на время отладки отключают сторожевой таймер. Но такой подход, в свою очередь, затрудняет тестирование и испытание функции обновления сторожевого таймера. У МК STM32 в регистре MCUDBG предусмотрена возможность настройки отключения независимого и отдельного сторожевых таймеров на время приостановки ЦПУ Cortex-M3, выполненной встроенной отладочной системой CoreSight. Благодаря этому, появляется возможность пошагового выполнения кода программы даже с активным сторожевым таймером.

7.5. Особенности УВВ

Встроенные УВВ тоже имеют ряд возможностей, направленных на обеспечение безопасной работы МК STM32. Их полное описание можно найти в разделе, посвященному соответствующему УВВ, а здесь остановимся на следующем:

7.5.1. Блокировка конфигурации ПВВ

Во время инициализации портов ввода-вывода каждая их линия настраивается на ввод или на вывод. По завершении настройки, конфигурационные регистры ПВВ можно заблокировать. Это поможет предотвратить возможность дальнейших непредсказуемых изменений настроек портов. Блокировку у каждого порта можно выполнить побитно.

7.5.2. Оконный компаратор

У каждого модуля АЦП имеется оконный компаратор, который генерирует прерывание при выходе контролируемого напряжения за заданные верхнюю и нижнюю границы.

7.5.3. Вход экстренного отключения

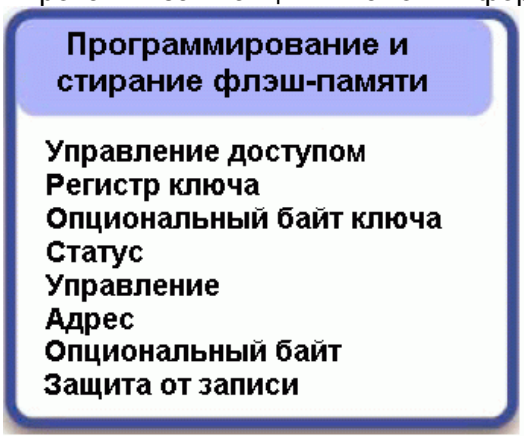
В устройствах управления электродвигателями может использоваться вход экстренного отключения. В случае воздействия на этот вход или в случае отказа основной синхронизации МК, происходит автоматический перевод противофазных ШИМ-выходов в заранее заданное безопасное состояние.

8. Модуль Flash памяти

Встроенная в МК STM32 Flash память состоит из трех областей. Первая из них - основная Flash память, предназначенная для хранения кода программы. Данная память является 64-битной. Это необходимо для повышения быстродействия считывания инструкций в буфер предварительной выборки. Для выполнения программирования и стирания данная Flash память разделена на 4 тыс. страниц. Память характеризуется износостойкостью 10 тысяч циклов перезаписи и 30-летним хранением данных при температуре 85°C. Износостойкость Flash памяти большинства других микроконтроллеров приводится для 25°C. Это означает, что МК STM32 оснащены превосходной Flash памятью. Помимо основной памяти программ, также имеется две области меньшего размера: большой информационный блок и малый информационный блок. Большой информационный блок занимает 2 кбайт Flash памяти и предназначен для хранения запрограммированной производителем программы загрузчика, которая использует для передачи кода программы последовательный интерфейс USART 1. Малый информационный блок состоит из шести конфигурационных байт. Они предназначены для определения свойств сброса МК STM32 и управления защитой памяти.

8.1. Защита и программирование Flash памяти

Встроенную Flash память можно обновить под управлением встроенный программы загрузчика, с использованием отладочных средств с интерфейсом JTAG или внутрисистемно с помощью специального набора регистров, называемых контроллером программирования и стирания Flash памяти (FPEC-контроллер). FPEC-контроллер также используется для программирования байт опций в малом информационном блоке.



FPEC-контроллер предназначен для внутрисистемного программирования Flash памяти. Флэш-память можно также защитить от считывания отладочными средствами и от записи

8.2. Операции стирания и записи

Сразу после сброса, регистры FPEC-контроллера защищены от записи. Чтобы их разблокировать, необходимо выполнить запись специальной последовательности чисел в регистр ключа. Запись 0x45670123, а затем 0xCDEF89AB приведет к разблокировке FPEC-контроллера. Если этого не сделать или сделать с ошибкой, FPEC-контроллер останется заблокированным вплоть до следующего сброса. Сразу после разблокировки FPEC-контроллера, появляется возможность стирания и записи основной Flash памяти. В пределах основного блока Flash памяти имеется возможность стирания всей памяти или выбранных 4 тыс. страниц. Для выполнения массового стирания достаточно установить биты массового стирания и старта в регистре управления. Если бит занятости BSY в этом же регистре будет сброшенным, то каждая ячейка основной Flash памяти будет переведена к состоянию 0xFFFF. Страничное стирание выполняется также просто. Вначале необходимо запрограммировать начальный адрес страницы Flash памяти в регистр адреса, а затем установить биты стирания страницы и старта в регистре управления. Опять-таки, если бит занятости BSY будет сброшенным, то страница будет стерта. Запись новых данных в ячейки Flash памяти можно выполнять только после их стирания. Для выполнения записи необходимо установить бит программирования в регистре управления, а затем выполнить запись полуслова по требуемому адресу. Если адресованная ячейка Flash памяти будет стертой и не защищена от записи, FPEC-контроллер запишет в нее новое значение.

8.3. Байты опций

Малый информационный блок содержит восемь программируемых пользователем байт опций. Четыре байта из них предназначены для управления защитой от записи основной Flash памяти. Пятый байт предназначен для установки защиты от чтения, которая предотвращает доступ к областям памяти, когда МК находится в отладочном режиме. Шестой байт необходим для конфигурации сброса и экономичной работы. Последние два байта - обычные ячейки Flash памяти, которые могут использоваться по усмотрению пользователя. Прежде чем выполнить запись байт опций, необходимо разблокировать FPEC-контроллер по описанной выше методике. После этого, необходио разблокировать байты опций записью тех же двух ключей в регистр ключа байт опций. Байты опций используют различные процедуры программирования и стирания основной Flash памяти. Для стирания малого информационного блока необходимо установить бит OPTER в регистре управления, а затем бит STRT. Сигнализирует о завершении стирания малого информационного блока переход в сброшенное состояние бита занятости BSY. Чтобы выполнить программирование байта опций, необходимо установить бит OPTPG в регистре управления Flash памятью и записать полуслово в байт опций. Каждый байт опций хранится в виде полуслова. В младшем байте этого полуслова хранится значение байта опций, а в старшем - его двоичное дополнение. Двоичное дополнение высчитывается автоматически FPEC-контроллером сразу после записи в младший байт полуслова.

8.3.1. Защита от записи

Разрешение защиты от записи Flash памяти осуществляется на постраничной основе. Каждый бит защиты от записи отвечает за разрешение защиты от записи соответствующей ему страницы Flash памяти. Защита от записи отключается стиранием малого информационного блока.

8.3.2. Защита от чтения

После установки защиты от чтения, блокируется возможность чтения из Flash памяти при переходе МК в отладочный режим. Доступ к статическому ОЗУ остается разрешенным, поэтому, код программы можно загружать и исполнять в этой области памяти. Исполняемая в статическом ОЗУ программа имеет возможность отключить защиту от чтения, однако одновременном с этим будет выполнено стирание всей внутренней Flash памяти. Это необходимо для исключения возможности хищения прошивки МК. После активизации защиты от чтения также включается и защита от записи Flash памяти. Это необходимо для исключения возможности несанкционированных изменений в код программы и в частности в таблицу векторов. Защита Flash памяти будет активной, если байт защиты от чтения и его двоичное дополнение будут установлены равными 0xFF. Для снятия защиты необходимо записать полуслово, состоящее из 0xFA и его двоичного дополнения, в байт защиты от чтения.

8.3.3. Конфигурационный байт

Конфигурационный байт содержит три активных бита. Два из них управляют механизмом перехода МК STM32 в режимы STANDBY и STOP. Для каждого из этих режимов можно активизировать генерацию сброса при входе в режим. Это приведет перенастройке цифровых линий ввода-вывода на ввод, что снизит общее энергопотребление МК STM32. Также будут отключены ФАПЧ и внешний генератор, а МК перейдет на синхронизацию от внутреннего высокочастотного RC-генератора. Последний бит конфигурационного байта управляет активностью независимого сторожевого таймера. У него предусмотрено два режима работы: аппаратный, в котором он включается в работу сразу после сброса процессора, и программный, в котором запуск осуществляет программно.

9. Инструментальные средства для проектирования

Интеграция ядер ARM7 и ARM9 в стандартные микроконтроллеры привела к настоящему взрыву в предложениях инструментальных средств для этих ЦПУ. Выпуск инструментальных средств для ARM микроконтроллеров осуществляют все ведущие разработчики компиляторов, в т.ч. GCC, Greenhills, Keil, IAR и Tasking. С появлением процессора Cortex к данным инструментальным средствам была добавлена поддержка набора инструкций Thumb-2. Поэтому, если вы уже используете любые другие ARM-микроконтроллеры, то у вас есть хорошие шансы сгенерировать код программы для МК STM32 с помощью имеющихся инструментальных средств. В худшем случае может потребоваться обращение к поставщику используемых инструментальных средств с запросом обновлений.

В случае, если вы впервые используете в своем проекте ARM-микроконтроллер, появляется возможность выбора инструментальных средств от наиболее предпочтительного для вас производителя. Но, поскольку в наши дни очень трудно найти плохие инструментальные средства, далее перейдем к обсуждению двух компиляторов. Первый компилятор "GCC" или "GNU". Он представляет собой инструментальное средство с открытым исходным кодом, поэтому, распространяется и используется бесплатно. Компилятор GCC, в целях снижения стоимости средств для проектирования и оценочных наборов, встраивается во многие коммерческие интегрированные среды для проектирования и отладчики. Несмотря на то, GCC компилятор является надежным и стабильным компилятором, наш опыт говорит о том, что генерируемый им код не столь эффективен, как при использовании коммерческих компиляторов. Кроме того, в случае возникновения проблем с его использованием, не к кому обратится за технической поддержкой, что может замедлить проектирование. Среди коммерческих компиляторов можно выделить ARM RealView, разработанный компанией ARM для использования с ее ЦПУ. Компилятор RealView доступен как часть набора инструментальных средств ARM RealView. Этот набор ориентирован на разработчиков систем на кристалле и не совсем подходит для микроконтроллерных проектов. Тем не менее, начиная с января 2006 года компилятор RealView интегрируется в состав микроконтроллерного набора для проектирования компании Keil (MDK-ARM). Из наименования MDK-ARM следует, что

данный набор разработан специально для работы с ARM-микроконтроллерами. Набор MDK прост в использовании (весь проект можно сконфигурировать выбором около 4 опций) и представляет собой цепочку тесно-взаимосвязанных инструментов от одного производителя.

Если вы обосновываете выбор использования компилятора GCC или коммерческого компилятора, в первую очередь необходимо руководствоваться бюджетом проекта. Бюджет простого проекта вряд ли позволит оправдать приобретение коммерческих инструментальных средств. Однако, если вы планируете массово использовать ARM-микроконтроллеры, то затраты на дорогостоящие инструментальные средства окупятся за счет ускорения проектирования и за счет генерации более компактного кода программы. При выборе компилятора также необходимо руководствоваться уровнем своей квалификации. Если вы опытный разработчик, то, скорее всего, разработка целого проекта с использованием компилятора GCC вам окажется под силу. Однако, при недостаточных навыках, в т.ч. программирования на Си, могут возникнуть серьезные проблемы.

9.1. Оценочные средства

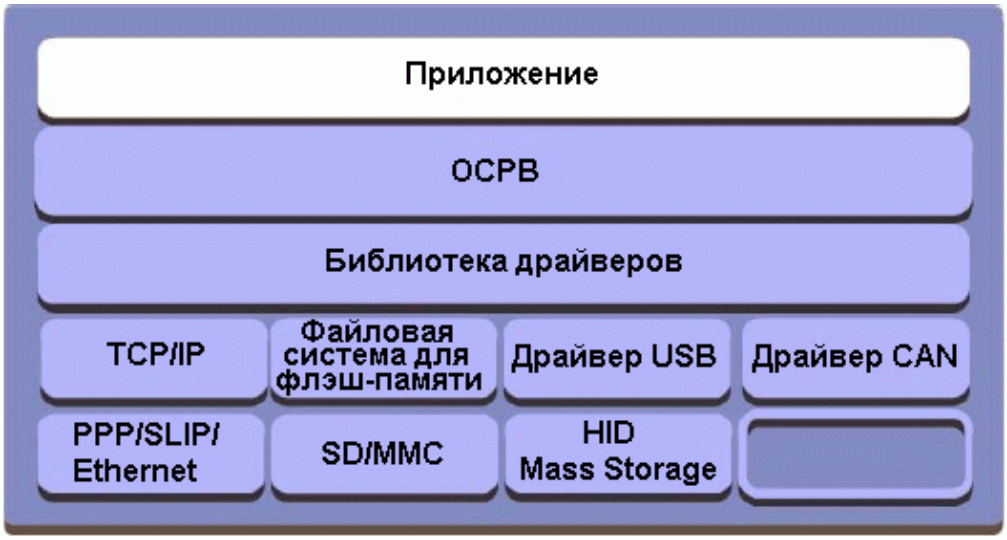
Большинство разработчиков компиляторов также предлагают оценочные или стартовые наборы. Обычно они состоят из печатной платы и сокращенной или ограниченной по времени версии набора инструментов. На вэб-сайте ST можно найти актуальный на данный момент список выпускаемых оценочных наборов. Одним из лучших оценочным средством является STM32 Performance Stick компании Hitex. При цене всего лишь около 50 Евро, Performance Stick представляет собой завершенное оценочное средство для МК STM32. Он рассчитан на подключение к ПК через USB-кабель и делает возможной разработку и отладку неограниченного по размеру кода программы в интегрированной среде для проектирования HiTOP и с использованием компиляторов GCC или Tasking. Помимо МК STM32, на плате Performance Stick установлен еще один микроконтроллер STR750. Он предназначен для измерения с помощью встроенных в него АЦП и таймеров потребляемого МК STM32 тока и задержек реагирования на прерывания. Данная информация передается в специальную программу на ПК для визуализации. Программа визуализации позволяет вручную оценивать различные возможности МК STM32 и сравнить некоторые характеристики МК с приводимыми в документации данными, например, потребляемый ток, задержка возобновления работы и др.

Performance Stick компании Hitex - весьма недорогое оценочное средство для МК STM32.

Performance Stick выполнено на основе отладчика HiTOP и компилятора GCC и, поэтому, не накладывает каких-либо ограничений на разрабатываемый код программы. Если требуется разработка продукции с нуля, то совместно с теми же интегрированной средой для проектирования и компилятором может быть использован JTAG-отладчик Tantino компании Hitex

9.2. Библиотеки и протокольные стеки

Чтобы помочь разработчику в ускорении разработки кода программы, компания ST разработала библиотеку программ для МК STM32, которые можно свободно скачать с её вэб-сайта. Библиотека программ поддерживает функции драйверов низкого уровня всех встроенных УВВ. Таким образом, пользователю предоставляется некоторое количество базовых составных блоков, из которых он может начать создание собственного проекта. Наиболее сложным УВВ среди всех существующих разновидностей МК STM32 является контроллер USB-устройства. Чтобы облегчить реализацию наиболее распространенных USB-классов, компания ST также предлагает бесплатный набор для разработки USB-устройств. Этот набор, также как и библиотеку программ можно скачать с вэб-сайта ST. В комплект набора для разработки USB-устройств входят USB-библиотека и демонстрационные программы для классов HID, Mass Storage, Audio и Device Field Upgrade.



В связи с возрастающей сложностью встраиваемых в микроконтроллеры УВВ важно, чтобы выбранный набор инструментальных средств для проектирования дополнялся широким ассортиментом протокольных стеков и примерами программ

По мере появления новых МК STM32, их будут оснащать все более и более сложными УВВ (MAC-контроллер Ethernet, интерфейс TFT-дисплея и др.). Такой рост сложности делает просто невозможным самостоятельное написание всего кода программы. Поэтому, еще на фазе выбора инструментальных средств необходимо оценить доступность протокольных стеков, как например, TCP/IP, и различное прикладное ПО, в т.ч. графические интерфейсы пользователя, которое может потребоваться в последующих проектах. Идеально, чтобы они были доступны от одного и того же поставщика и были интегрированы в выбранный набор инструментальных средств.

9.3. Операционные системы реального времени

Если вы прежде работали с 8- или 16-битными микроконтроллерами, то, скорее всего, еще не используете RTOS. Мы уже могли убедиться, что процессор Cortex-M3 обладает существенно большей вычислительной мощностью по сравнению с другими сопоставимыми по стоимости микроконтроллерами и разработан с учетом работы под управлением занимающими небольшое место в памяти RTOS. Таким образом, если вы прежде не использовали RTOS, то при освоении МК STM32 важно уделить ее изучению особое внимание. Использование RTOS даст вам преимущества более абстрактной разработки кода программы, более широких возможностей по повторному использованию программ, более простого управления проектом и более широких возможностей отладки. Использование RTOS также позволяет структурировать вашу программу. Это означает, что вначале составляется план программы и только после этого начинается написание текста программы. Больше всего существует RTOS для ЦПУ ARM и Cortex, чем для большинства других встраиваемых ЦПУ. Многие поставщики компиляторов предлагают свою собственную RTOS, однако наибольшую популярность среди операционных систем с открытым исходным кодом имеет "FreeRTOS". Ее можно скачать с сайта

www.freertos.org. Коммерческая версия FreeRTOS называется "SafeRTOS". Она протестирована на соответствие стандарту безопасности IEC 61508 и доступна на том же сайте.